SKRIPSI

PENGAMANAN PESAN RAHASIA MENGGUNAKAN ALGORITMA KRIPTOGRAFI ELGAMAL ATAS GRUP PERGANDAAN Zp*

Sebagai salah satu syarat untuk memperoleh derajat S-1
Program Studi Matematika pada Jurusan Matematika

Disusun Oleh :
MUHAMAD ZAKI RIYANTO
02 / 156792 / PA / 08944

DEPARTEMEN PENDIDIKAN NASIONAL
UNIVERSITAS GADJAH MADA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
YOGYAKARTA
2007

HALAMAN PENGESAHAN

SKRIPSI

PENGAMANAN PESAN RAHASIA MENGGUNAKAN ALGORITMA KRIPTOGRAFI ELGAMAL ATAS GRUP PERGANDAAN Zp*

MUHAMAD ZAKI RIYANTO

02 / 156792 / PA / 08944

Dinyatakan lulus ujian skripsi oleh tim penguji
pada tanggal 25 September 2007

Tim Penguji

Dosen Pembimbing	Ketua Tim Penguji
Dra. Diah Junia Eksi Palupi, MS	**Dr. Ari Suparwanto, M.Si**
Penguji I	**Penguji II**
Dr. Budi Surodjo, M.Si	**Drs. Retantyo Wardoyo, M.Sc., Ph.D**

HALAMAN MOTO

"Hai orang-orang beriman, apabila kamu mengadakan pembicaraan rahasia, janganlah kamu membicarakan tentang membuat dosa, permusuhan dan berbuat durhaka kepada Rasul. Dan bicarakanlah tentang membuat kebajikan dan taqwa. Dan bertaqwalah kepada Allah yang kepada-Nya kamu akan dikembalikan".

-QS. Al-Mujaadilah : 9

"Apakah mereka mengira, bahwa Kami tidak mendengar rahasia dan bisikan-bisikan mereka? Sebenarnya (Kami mendengar), dan utusan-utusan (malaikat-malaikat) Kami selalu mencatat di sisi mereka".

-QS. Az-Zukhruf : 80

"Barang siapa ingin meraih dunia maka harus dengan ilmu, barang siapa ingin meraih akherat maka harus dengan ilmu, barang siapa ingin meraih kedua-duanya maka harus dengan ilmu".

-Al-Hadits

"Tak ada satupun sistem pengamanan yang ada sekarang, apakah kunci stir ataupun ruangan baja yang benar-benar aman. Yang terbaik yang bisa kita lakukan adalah untuk membuatnya sesulit mungkin bagi seseorang untuk merusak alat keamanan atau masuk ke dalamnya...".

-Bill Gates, *The Road Ahead*

"Kita semua mempunyai hak untuk menyimpan rahasia", katanya. "Suatu hari nanti saya akan memastikan hal itu terjadi".

-Tankado, Dan Brown - *Digital Fortress*

HALAMAN PERSEMBAHAN

Skripsi ini penulis persembahkan untuk seluruh orang-orang yang telah membantu dan memberikan inspirasi kepada penulis :

- Ayahanda dan Ibunda tercinta, yang telah membesarkan, senantiasa membimbing dan mendo'akanku dengan penuh kasih sayang dan kesabaran.
- Guru-guruku dan sahabatku di dalam menuntut ilmu.
- Pecinta matematika dan pemerhati perkembangan ilmu kriptologi di seluruh Indonesia.

KATA PENGANTAR

Alhamdulillah, puji syukur tak henti-hentinya penulis panjatkan ke hadirat Allah S.W.T. atas segala limpahan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penulisan skripsi yang berjudul "*Pengamanan Pesan Rahasia Menggunakan Algoritma Kriptografi ElGamal Atas Grup Pergandaan Zp**" ini.

Penulis menyadari sepenuhnya bahwa dalam penyusunan skripsi ini tidak terlepas dari dukungan, dorongan, kerjasama maupun bimbingan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Ibu Dra. Diah Junia Eksi Palupi, MS selaku dosen pembimbing skripsi yang telah bersedia meluangkan pikiran dan waktu hingga akhirnya penulis dapat menyelesaikan skripsi ini.
2. Bapak Dr. Ari Suparwanto, M.Si, Bapak Dr. Budi Surodjo, M.Si, dan Bapak Drs. Retantyo Wardoyo, M.Sc., Ph.D, yang telah bersedia menjadi dosen penguji skripsi. Terima kasih atas saran dan masukannya.
3. Bapak Sutopo, M.Si selaku dosen wali akademik atas segala pengarahan selama penulis belajar di Fakultas MIPA UGM.
4. Dosen di Fakultas MIPA UGM yang telah membagi ilmunya kepada penulis.
5. Ayahanda dan Ibunda tersayang, serta ketiga adikku tercinta yang telah memberikan dorongan semangat, do'a, dan motivasi tiada henti.
6. Peminat skripsi *cryptography* & *coding*, Ardhi (*Elliptic Curve Cryptography*), Ariswan (*RSA*), Ikhwanudin (*Extended Hill Cipher*), Kristiyono-dkk (*XTR*), Rudi (*Secret Sharing Scheme*), dan Tommy (*Reed-Solomon Codes*).

7. Sahabat-sahabatku Akhid, Dede, Diki, Eris, Fira Zakia, Fitria, Ikhwanudin, Indra, Lalu Tamrin, Nanang, Tia, Triastuti dan semua teman-teman matematika angkatan 2002 yang tidak dapat penulis sebutkan satu persatu.
8. Semua pihak yang turut membantu hingga selesainya skripsi ini yang tidak dapat penulis sebutkan satu persatu, terima kasih.

Penulis sangat menyadari bahwa dalam skripsi ini masih banyak sekali kekurangan dan kesalahan. Oleh karena itu penulis mengharapkan kritik dan saran yang membangun untuk menyempurnakan skripsi ini. Kritik dan saran tersebut dapat disampaikan melalui e-mail di zaki@mail.ugm.ac.id atau di website penulis di http://zaki.math.web.id. Salinan skripsi ini juga dapat diperoleh di website penulis. Akhir kata, penulis berharap semoga skripsi ini dapat memberikan sesuatu yang bermanfaat bagi semua pihak yang membacanya.

Yogyakarta, September 2007

Penulis

DAFTAR ISI

Halaman Judul……..……..……..…….. i

Halaman Pengesahan…….……..... ii

Halaman Moto……..... iii

Halaman Persembahan……..……..... iv

Kata Pengantar…….....…..... v

Daftar Isi…...……...……..... viii

Intisari…..……....……....……. x

Daftar Gambar…….....……..... xi

Daftar Tabel.....…….....……..... xii

Daftar Algoritma..... xiii

Arti Lambang……...…….…… xiv

Bab I. PENDAHULUAN

1.1. Latar Belakang...……..... 1

1.2. Perumusan Masalah…..…….……..... 2

1.3. Batasan Masalah..…….…….……..... 2

1.4. Maksud dan Tujuan..…….…….…….…… 3

1.5. Tinjauan Pustaka…….…….……..... 3

1.6. Metodologi Penelitian…….…….……..... 4

1.7. Sistematika Penulisan…….…….……..... 4

Bab II. LANDASAN TEORI

2.1. Kriptografi.....…….…….……..... 7

2.1.1. Sejarah Kriptografi..... 9

2.1.2. Algoritma Kriptografi..... 11

2.1.2.1. Algoritma Simetris..... 11

2.1.2.2. Algoritma Asimetris..... 12

2.1.3. Sistem Kriptografi..... 14

2.2. Bilangan Bulat…...…….……..... 15

2.2.1. Divisibility..... 16

2.2.2. Algoritma Pembagian pada Bilangan Bulat..... 17

2.2.3. Representasi Bilangan Bulat..... 19
2.2.4. Pembagi Persekutuan Terbesar..... 24
2.2.5. Algoritma Euclide..... 28
2.2.6. Algoritma Euclide yang Diperluas..... 30
2.2.7. Faktorisasi ke Bilangan Prima..... 33
2.3. Dasar Struktur Aljabar..... 37
2.3.1. Partisi dan Relasi Ekuivalensi..... 37
2.3.2. Grup..... 38
2.3.3. Homomorfisma Grup..... 41
2.3.4. Gelanggang dan Lapangan..... 43
Bab III. PERSAMAAN KONGRUEN DAN HIMPUNAN BILANGAN BULAT MODULO
3.1. Persamaan Kongruen..... 48
3.2. Gelanggang Bilangan Bulat Modulo..... 51
3.3. Pembagian pada Gelanggang Bilangan Bulat Modulo..... 54
3.4. Grup Pergandaan Bilangan Bulat Modulo..... 58
3.5. Order Elemen-Elemen Grup..... 62
3.6. Teorema Fermat..... 64
3.7. Metode Fast Exponentiation..... 66
3.8. Penghitungan Order Elemen Grup..... 68
3.9. Polinomial..... 71
3.10. Polinomial atas Lapangan..... 72
3.11. Grup Unit atas Lapangan Berhingga..... 74
3.12. Struktur Grup Pergandaan Bilangan Bulat Modulo Prima..... 76
Bab IV. TES KEPRIMAAN
4.1. Tes Fermat..... 78
4.2. Bilangan Carmichael..... 80
4.3. Tes Miller-Rabbin..... 81
Bab V. MASALAH LOGARITMA DISKRET DAN ALGORITMA ELGAMAL
5.1. Masalah Logaritma Diskret..... 84

5.1.1. Masalah Logaritma Diskret pada Grup Pergandaan Bilangan Bulat Modulo Prima..... 85
5.2. Algoritma ElGamal..... 86
5.2.1. Pembentukan Kunci..... 87
5.2.2. Enkripsi..... 92
5.2.3. Dekripsi..... 97
Bab VI. IMPLEMENTASI DAN UJI COBA
6.1. Sarana Implementasi..... 101
6.2. Implementasi Algoritma ElGamal..... 103
6.2.1. Deklarasi Nama Program, Unit, Variabel-Variabel dan Tipe Data..... 103
6.2.2. Beberapa Fungsi dan Prosedur..... 103
6.2.3. Program Utama..... 108
6.3. Uji Coba Program..... 110
6.3.1. Bahan Pengujian..... 110
6.3.2. Pengujian Program..... 111
6.3.2.1. Pengujian Proses Pembentukan Kunci..... 111
6.3.2.2. Pengujian Proses Enkripsi..... 113
6.3.2.3. Pengujian Proses Dekripsi..... 115
Bab VII. PENUTUP
7.1. Kesimpulan…… 118
7.2. Saran……..…… 120
Daftar Pustaka…….....….... 121
LAMPIRAN
Lampiran 1. Kode Program ……..... 122
Lampiran 2. Tabel Kode ASCII..... 139
Lampiran 3. Data Pribadi Penulis..... 141

INTISARI

PENGAMANAN PESAN RAHASIA MENGGUNAKAN ALGORITMA KRIPTOGRAFI ELGAMAL ATAS GRUP PERGANDAAN Zp*

Oleh :

MUHAMAD ZAKI RIYANTO
02 / 156792 / PA / 08944

Algoritma ElGamal merupakan algoritma kriptografi asimetris yang menggunakan dua jenis kunci, yaitu kunci publik dan kunci rahasia. Tingkat keamanan algoritma ini didasarkan atas masalah logaritma diskret pada grup pergandaan bilangan bulat modulo prima, $\mathbb{Z}_p^* = \{1, 2, ..., p-1\}$, dengan p adalah bilangan prima. Sehingga apabila digunakan bilangan prima dan logaritma diskret yang besar, maka upaya untuk menyelesaikan masalah logaritma diskret ini menjadi sia-sia dan dirasakan tidak sesuai dengan isi informasi yang ingin diperoleh.

Algoritma ElGamal mempunyai kunci publik berupa tiga pasang bilangan dan kunci rahasia berupa satu bilangan. Algoritma ini melakukan proses enkripsi dan dekripsi pada blok-blok plainteks dan dihasilkan blok-blok cipherteks yang masing-masing terdiri dari dua pasang bilangan.

Pada skripsi ini pembahasan difokuskan pada algoritma ElGamal yang digunakan dalam proses enkripsi dan dekripsi, beserta konsep-konsep matematis yang melandasinya, yang meliputi teori bilangan dan struktur aljabar. Kemudian dibuat sebuah program pengamanan pesan rahasia yang sederhana berdasarkan algoritma ElGamal.

Kata kunci : algoritma, asimetris, cipher blok, ElGamal, kriptografi, kunci publik, masalah logaritma diskret

DAFTAR GAMBAR

Gambar 2.1.	Skema algoritma simetris.....	12
Gambar 2.2.	Skema algoritma asimetris.....	13
Gambar 2.3.	Hubungan antara G, G/H dan $\phi[G]$	43
Gambar 3.1.	Hubungan antara $\mathbb{Z}$, $\mathbb{Z}/m.\mathbb{Z}$ dan $\mathbb{Z}_m$	53
Gambar 5.1.	Sistem kriptografi ElGamal pada $\mathbb{Z}_p$ *.....	87
Gambar 6.1.	Tampilan program menu utama.....	110
Gambar 6.2.	Tampilan program pada menu pembentukan kunci.....	111
Gambar 6.3.	Tampilan program untuk membentuk kunci otomatis.....	112
Gambar 6.4.	Tampilan program untuk membentuk kunci pilihan.....	113
Gambar 6.5.	Tampilan program proses enkripsi.....	114
Gambar 6.6.	Tampilan program menampilkan hasil cipherteks.....	114
Gambar 6.7.	Tampilan program menampilkan proses dekripsi.....	115
Gambar 6.8.	Tampilan program pada menu program tambahan.....	116

DAFTAR TABEL

Tabel 2.1	Perhitungan gcd(100,35) menggunakan algoritma Euclide.....	30
Tabel 2.2	Perhitungan *x* dan *y* menggunakan Teorema 2.2.6.1.....	32
Tabel 2.3	Perhitungan menggunakan algoritma Euclide yang diperluas.....	33
Tabel 3.1	Beberapa nilai Euler φ*-function*.....	59
Tabel 3.2	Perhitungan $5^{596} \bmod 1234$	68
Tabel 5.1	Perhitungan $\alpha^2 \bmod 2579$ dan $\alpha^{1289} \bmod 2579$	90
Tabel 5.2	Konversi karakter pesan ke kode ASCII.....	94
Tabel 5.3	Proses enkripsi.....	95
Tabel 5.4	Proses dekripsi.....	99
Tabel 6.1	Spesifikasi perangkat keras.....	101
Tabel 6.2	Spesifikasi perangkat lunak.....	102

DAFTAR ALGORITMA

Algoritma 2.1	Representasi Bilangan Bulat.....	23
Algoritma 2.2	Algoritma Euclide.....	29
Algoritma 2.3	Algoritma Euclide yang Diperluas.....	32
Algoritma 2.4	Tes Keprimaan Biasa.....	37
Algoritma 3.1	Mencari Invers Pergandaan Modulo.....	57
Algoritma 3.2	Metode *Fast Exponentiation*.....	67
Algoritma 4.1	Tes Fermat.....	79
Algoritma 4.2	Tes Miller-Rabbin.....	82
Algoritma 5.1	Tes Bilangan Prima Aman.....	88
Algoritma 5.2	Tes Elemen Primitif.....	89
Algoritma 5.3	Algoritma Pembentukan Kunci.....	91
Algoritma 5.4	Algoritma Enkripsi.....	93
Algoritma 5.5	Algoritma Dekripsi.....	98

ARTI LAMBANG

$x \in X$	: x anggota X
$x \notin X$	: x bukan anggota X
$A \subseteq X$	: A subset (himpunan bagian) atau sama dengan X
$A \cup B$	: gabungan himpunan A dengan himpunan B
$A \cap B$	: irisan himpunan A dengan himpunan B
$\bigcup_{i=1}^{n} A_i$	: gabungan himpunan-himpunan $A_1 \cup A_2 \cup ... \cup A_n$
$\lvert A \rvert$	: kardinal (banyaknya elemen) himpunan A
$\varnothing$	: himpunan kosong
$\mathbb{Z}$	: himpunan semua bilangan bulat
$\mathbb{R}$	: himpunan semua bilangan real
$\mathbb{N}$	: himpunan semua bilangan asli
$\Rightarrow$	: maka
$\Leftrightarrow$	: jika dan hanya jika
$\rightarrow$	: menuju
$\square$	: akhir suatu bukti
$\sum_{i=1}^{n} a_i$	: penjumlahan $a_1 + a_2 + ... + a_n$
$\prod_{i=1}^{n} a_i$	: perkalian $a_1.a_2...a_n$
$n!$	: n faktorial
C_r^n	: r-kombinasi dari n unsur yang berbeda
$x \leftarrow a$	: nilai a dimasukkan ke x

BAB I

PENDAHULUAN

1.1. Latar Belakang

Kemajuan dan perkembangan teknologi informasi dewasa ini telah berpengaruh pada hampir semua aspek kehidupan manusia, tak terkecuali dalam hal berkomunikasi. Dengan adanya internet, komunikasi jarak jauh dapat dilakukan dengan cepat dan murah. Namun di sisi lain, ternyata internet tidak terlalu aman karena merupakan media komunikasi umum yang dapat digunakan oleh siapapun sehingga sangat rawan terhadap penyadapan informasi oleh pihak-pihak yang tidak berhak mengetahui informasi tersebut. Oleh karena penggunaan internet yang sangat luas seperti pada bisnis, perdagangan, bank, industri dan pemerintahan yang umumnya mengandung informasi yang bersifat rahasia maka keamanan informasi menjadi faktor utama yang harus dipenuhi. Berbagai hal telah dilakukan untuk mendapatkan jaminan keamanan informasi rahasia ini. Salah satu cara yang digunakan adalah dengan menyandikan isi informasi menjadi suatu kode-kode yang tidak dimengerti sehingga apabila disadap maka akan kesulitan untuk mengetahui isi informasi yang sebenarnya.

Metode penyandian yang pertama kali dibuat masih menggunakan metode algoritma rahasia. Metode ini menumpukan keamanannya pada kerahasian algoritma yang digunakan. Namun metode ini tidak efisien saat digunakan untuk berkomunikasi dengan banyak orang. Oleh karena itu seseorang harus membuat algoritma baru apabila akan bertukar informasi rahasia dengan orang lain.

Karena penggunaannya yang tidak efisien maka algoritma rahasia mulai ditinggalkan dan dikenalkan suatu metode baru yang disebut dengan algoritma kunci. Metode ini tidak menumpukan keamanan pada algoritmanya, tetapi pada kerahasian kunci yang digunakan pada proses penyandian. Algoritmanya dapat diketahui, digunakan dan dipelajari oleh siapapun. Metode algoritma kunci mempunyai tingkat efisiensi dan keamanan yang lebih baik dibandingkan dengan algoritma rahasia. Sampai sekarang algoritma kunci masih digunakan secara luas di internet dan terus dikembangkan untuk mendapatkan keamanan yang lebih baik.

Algoritma ElGamal merupakan salah satu dari algoritma kunci. Algoritma ini dikembangkan pertama kali oleh Taher ElGamal pada tahun 1985. Sampai saat ini, algoritma ElGamal masih dipercaya sebagai metode penyandian, seperti aplikasi PGP dan GnuPG yang dapat digunakan untuk pengamanan e-mail dan tanda tangan digital. Pada tahun 1994 pemerintah Amerika Serikat mengadopsi *Digital Signature Standard*, sebuah mekanisme penyandian yang berdasar pada algoritma ElGamal.

1.2. Perumusan Masalah

Masalah yang dibahas pada skripsi ini adalah konsep-konsep matematis yang melandasi pembentukan algoritma ElGamal, proses penyandian serta implementasi algoritma ElGamal dalam bentuk sebuah program komputer yang sederhana.

1.3. Batasan Masalah

Pada skripsi ini, pembahasan algoritma ElGamal meliputi konsep matematis yang melandasinya dan proses penyandiannya. Serta mengenai pembuatan sebuah

program komputer yang digunakan untuk menyandikan suatu pesan. Program ini merupakan implementasi algoritma ElGamal dan dibuat menggunakan bahasa Pascal. Pada skripsi ini tidak membahas mengenai sulitnya dan cara-cara untuk memecahkan mekanisme penyandian.

1.4. Maksud dan Tujuan

Selain untuk memenuhi syarat kelulusan program Strata-1 (S1) program studi Matematika Universitas Gadjah Mada, penyusunan skripsi ini bertujuan untuk mempelajari konsep matematis yang melandasi pembentukan algoritma ElGamal dan penggunaannya. Sedangkan pembuatan program komputer hanya ditujukan sebagai contoh semata agar mempermudah pemahaman.

1.5. Tinjauan Pustaka

Algoritma ElGamal banyak dibahas pada buku-buku kriptografi, tetapi masih sedikit yang membahas secara mendetail tentang konsep-konsep matematisnya. Stinson (1995) telah menjelaskan secara umum tentang algoritma ElGamal beserta sistem pendukungnya. Buchmann (2000) secara khusus menitikberatkan pada pemahaman konsep dasar matematis dari algoritma ElGamal, seperti teori bilangan bulat, persamaan kongruen, dan struktur aljabar abstrak yang meliputi grup, homomorfisma dan gelanggang. Pembahasan aljabar abstrak yang lebih terperinci diberikan oleh Fraleigh (2000), namun tidak ada pembahasan yang mengaitkan secara langsung dengan algoritma ElGamal. Sedangkan implementasi algoritma ElGamal diberikan oleh Menezes, Oorschot dan Vanstone (1996), termasuk

penjelasan beberapa algoritma yang dapat digunakan untuk membuat program komputer.

1.6. Metodologi Penelitian

Metode yang digunakan dalam pembuatan skripsi ini adalah dengan terlebih dahulu melakukan studi literatur mengenai algoritma ElGamal pada beberapa buku, paper, maupun situs internet yang berhubungan dengan algoritma ElGamal. Kemudian penulis mengambil beberapa materi yang menjelaskan mengenai algoritma ElGamal dan membahasnya. Langkah terakhir adalah melakukan perancangan dan menerapkan algoritma tersebut menggunakan bahasa Pascal untuk membuat sebuah program komputer yang digunakan untuk menyandikan pesan.

1.7. Sistematika Penulisan

Dalam skripsi ini pembahasan materi disusun menjadi tujuh bab. Materi tersebut disusun dengan sistematika berikut ini.

BAB I PENDAHULUAN

Pada bab ini dibahas mengenai latar belakang, perumusan masalah, batasan masalah, maksud dan tujuan penulisan skripsi, tinjauan pustaka, metode penulisan, serta sistematika penulisan skripsi.

BAB II LANDASAN TEORI

Pada bab ini dibahas mengenai tiga landasan teori yang harus dipahami sebelum membahas bagian inti dari skripsi ini, yaitu mengenai kriptografi,

bilangan bulat dan struktur aljabar. Pada bagian kriptografi akan diberikan definisi kriptogafi, algoritma kriptografi dan sistem kriptografi. Pada bagian bilangan bulat akan dibahas mengenai beberapa sifat bilangan bulat seperti *divisibility*, algoritma pembagian pada bilangan bulat, representasi bilangan bulat, pembagi persekutuan terbesar, algoritma Euclide, serta faktorisasi ke bilangan prima. Sedangkan pada pembahasan mengenai struktur aljabar akan dibahas mengenai grup, grup siklik, partisi dan relasi ekuivalensi, homomorfisma, grup fakor, gelanggang, dan lapangan.

BAB III PERSAMAAN KONGRUEN DAN HIMPUNAN BILANGAN BULAT MODULO

Pada bab ini dibahas mengenai konsep-konsep dasar matematika yang secara khusus mendasari pembentukan algoritma ElGamal yang meliputi persamaan kongruen, himpunan bilangan bulat modulo, gelanggang bilangan bulat modulo, grup pergandaan bilangan bulat modulo, Euler φ*-function*, teorema Fermat, metode *fast exponentiation*, grup unit atas lapangan berhingga dan elemen primitif.

BAB IV TES KEPRIMAAN

Pada bab ini dibahas mengenai dua tes keprimaan (*primality test*), yaitu tes Fermat dan tes Miller-Rabbin. Tes keprimaan merupakan suatu algoritma yang digunakan untuk mengecek apakah suatu bilangan bulat positif ganjil merupakan bilangan prima atau bukan.

BAB V MASALAH LOGARITMA DISKRET DAN ALGORITMA ELGAMAL

Pada bab ini dibahas dua hal yang menyangkut algoritma ElGamal, yaitu masalah logaritma diskret yang mendasari pembentukan algoritma ElGamal dan proses penyandian menggunakan algoritma ElGamal. Pada penjelasan mengenai proses penyandian dijelaskan tiga hal yaitu proses pembentukan kunci, proses enkripsi dan proses dekripsi. Serta diberikan contoh kasus penggunaannya.

BAB VI IMPLEMENTASI DAN UJI COBA

Bab ini membahas mengenai langkah- langkah pembuatan program komputer yang digunakan untuk menyandikan suatu pesan menggunakan algoritma ElGamal. Serta pembahasan hasil uji coba program tersebut.

BAB VII PENUTUP

Bab ini berisi kesimpulan dan saran-saran yang dapat diambil berdasarkan materi-materi yang telah dibahas pada bab-bab sebelumnya.

BAB II

LANDASAN TEORI

Pada bab ini dibahas konsep dasar yang berhubungan dengan kriptografi seperti definisi kriptografi, algoritma kriptografi, sistem kriptografi, serta jenis-jenis kriptografi, bilangan bulat, algoritma pembagian, pembagi persekutuan terbesar, grup, gelanggang, lapangan, dan sebagainya.

2.1. Kriptografi

Kriptografi (*cryptography*) berasal dari bahasa Yunani, terdiri dari dua suku kata yaitu *kripto* dan *graphia*. *Kripto* artinya menyembunyikan, sedangkan *graphia* artinya tulisan. Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data (Menezes, Oorschot and Vanstone, 1996). Tetapi tidak semua aspek keamanan informasi dapat diselesaikan dengan kriptografi. Kriptografi dapat pula diartikan sebagai ilmu atau seni untuk menjaga keamanan pesan. Ketika suatu pesan dikirim dari suatu tempat ke tempat lain, isi pesan tersebut mungkin dapat disadap oleh pihak lain yang tidak berhak untuk mengetahui isi pesan tersebut. Untuk menjaga pesan, maka pesan tersebut dapat diubah menjadi suatu kode yang tidak dapat dimengerti oleh pihak lain.

Enkripsi adalah sebuah proses penyandian yang melakukan perubahan sebuah kode (pesan) dari yang bisa dimengerti (*plainteks*) menjadi sebuah kode yang tidak bisa dimengerti (*cipherteks*). Sedangkan proses kebalikannya untuk mengubah

cipherteks menjadi plainteks disebut *dekripsi*. Proses enkripsi dan dekripsi memerlukan suatu mekanisme dan kunci tertentu.

Kriptoanalisis (*cryptanalysis*) adalah kebalikan dari kriptografi, yaitu suatu ilmu untuk memecahkan mekanisme kriptografi dengan cara mendapatkan kunci dari cipherteks yang digunakan untuk mendapatkan plainteks. *Kriptologi* (*cryptology*) adalah ilmu yang mencakup kriptografi dan kriptoanalisis.

Ada empat tujuan mendasar dari kriptografi yang juga merupakan aspek keamanan informasi, yaitu

1. *Kerahasiaan*, adalah aspek yang berhubungan dengan penjagaan isi informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka informasi yang telah dienkripsi.
2. *Integritas data*, adalah aspek yang berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubsitusian data lain kedalam data yang sebenarnya.
3. *Autentikasi*, adalah aspek yang berhubungan dengan identifikasi atau pengenalan, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan harus diautentikasi keaslian, isi datanya, waktu pengiriman, dan lain-lain.
4. *Non-repudiation* (menolak penyangkalan), adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman suatu informasi oleh yang

mengirimkan, atau harus dapat membuktikan bahwa suatu pesan berasal dari seseorang, apabila ia menyangkal mengirim informasi tersebut.

(Menezes, Oorschot and Vanstone, 1996).

2.1.1. Sejarah Kriptografi

Kriptografi sudah digunakan sekitar 40 abad yang lalu oleh orang-orang Mesir untuk mengirim pesan ke pasukan yang berada di medan perang dan agar pesan tersebut tidak terbaca oleh pihak musuh walaupun pembawa pesan tersebut tertangkap oleh musuh. Sekitar 400 SM, kriptografi digunakan oleh bangsa Spartan dalam bentuk sepotong papirus atau perkamen yang dibungkus dengan batang kayu.

Pada zaman Romawi kuno, ketika Julius Caesar ingin mengirimkan pesan rahasia pada seorang Jendral di medan perang. Pesan tersebut harus dikirimkan melalui seorang prajurit, tetapi karena pesan tersebut mengandung rahasia, Julius Caesar tidak ingin pesan tersebut terbuka di tengah jalan. Di sini Julius Caesar memikirkan bagaimana mengatasinya yaitu dengan mengacak isi pesan tersebut menjadi suatu pesan yang tidak dapat dipahami oleh siapapun kecuali hanya dapat dipahami oleh Jendralnya saja. Tentu sang Jendral telah diberi tahu sebelumnya bagaimana cara membaca pesan yang teracak tersebut, karena telah mengetahui kuncinya.

Pada perang dunia kedua, Jerman menggunakan mesin enigma atau juga disebut dengan mesin rotor yang digunakan Hitler untuk mengirim pesan kepada tentaranya di medan perang. Jerman sangat percaya bahwa pesan yang dienkripsi menggunakan enigma tidak dapat dipecahkan. Tapi anggapan itu keliru, setelah

bertahun-tahun sekutu mempelajarinya dan berhasil memecahkan kode-kode tersebut. Setelah Jerman mengetahui bahwa enigma dapat dipecahkan, maka enigma mengalami beberapa kali perubahan. Enigma yang digunakan Jerman dapat mengenkripsi suatu pesan sehingga mempunyai 15×10^{18} kemungkinan untuk dapat mendekripsi pesan.

Perkembangan komputer dan sistem komunikasi pada tahun 60-an berdampak pada permintaan dari pihak-pihak tertentu sebagai sarana untuk melindungi informasi dalam bentuk digital dan untuk menyediakan layanan keamanan. Dimulai dari usaha Feistel dari IBM di awal tahun 70-an dan mencapai puncaknya pada 1977 dengan pengangkatan DES (*Data Encryption Standard*) sebagai standar pemrosesan informasi federal Amerika Serikat untuk mengenkripsi informasi yang tidak belum diklasifikasi. DES merupakan mekanisme kriptografi yang paling dikenal sepanjang sejarah.

Pengembangan paling mengejutkan dalam sejarah kriptografi terjadi pada 1976 saat Diffie dan Hellman mempublikasikan *"New Directions in Cryptography"*. Tulisan ini memperkenalkan konsep revolusioner kriptografi kunci publik dan juga memberikan metode baru untuk pertukaran kunci, keamanan yang berdasar pada kekuatan masalah logaritma diskret. Meskipun Diffie dan Hellman tidak memiliki realisasi praktis pada ide enkripsi kunci publik saat itu, idenya sangat jelas dan menumbuhkan ketertarikan yang luas pada komunitas kriptografi. Pada 1978 Rivest, Shamir dan Adleman menemukan rancangan enkripsi kunci publik yang sekarang disebut RSA. Rancangan RSA berdasar pada masalah faktorisasi bilangan yang sulit, dan menggiatkan kembali usaha untuk menemukan metode yang lebih efisien untuk

pemfaktoran. Tahun 80-an terjadi peningkatan luas di area ini, sistem RSA masih aman. Sistem lain yang merupakan rancangan kunci publik ditemukan oleh Taher ElGamal pada tahun 1985. Rancangan ini berdasar pada masalah logaritma diskret.

Salah satu kontribusi penting dari kriptografi kunci publik adalah tanda tangan digital. Pada 1991 standar internasional pertama untuk tanda tangan digital diadopsi. Standar ini berdasar pada rancangan kunci publik RSA. Pada 1994 pemerintah Amerika Serikat mengadopsi *Digital Signature Standard*, sebuah mekanisme kriptografi yang berdasar pada algoritma ElGamal.

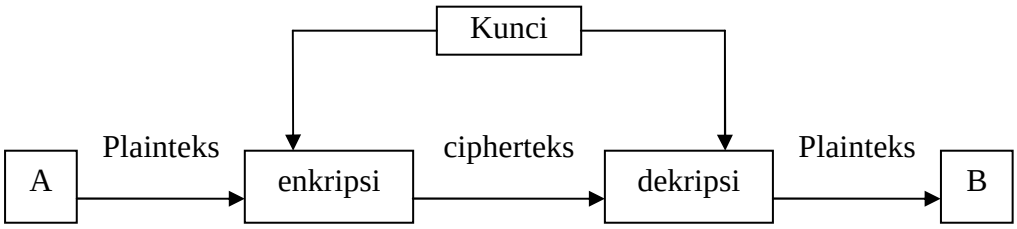
2.1.2. Algoritma Kriptografi

Algoritma kriptografi atau sering disebut dengan *cipher* adalah suatu fungsi matematis yang digunakan untuk melakukan enkripsi dan dekripsi (Schneier, 1996). Ada dua macam algoritma kriptografi, yaitu *algoritma simetris* (*symmetric algorithms*) dan *algoritma asimetris* (*asymmetric algorithms*).

2.1.2.1. Algoritma Simetris

Algoritma simetris adalah algoritma kriptografi yang menggunakan kunci enkripsi yang sama dengan kunci dekripsinya. Algoritma ini mengharuskan pengirim dan penerima menyetujui suatu kunci tertentu sebelum mereka saling berkomunikasi. Keamanan algoritma simetris tergantung pada kunci, membocorkan kunci berarti bahwa orang lain dapat mengenkripsi dan mendekripsi pesan. Agar komunikasi tetap aman, kunci harus tetap dirahasiakan. Algoritma simetris sering juga disebut dengan *algoritma kunci rahasia*, *algoritma kunci tunggal*, atau *algoritma satu kunci*.

Sifat kunci yang seperti ini membuat pengirim harus selalu memastikan bahwa jalur yang digunakan dalam pendistribusian kunci adalah jalur yang aman atau memastikan bahwa seseorang yang ditunjuk membawa kunci untuk dipertukarkan adalah orang yang dapat dipercaya. Masalahnya akan menjadi rumit apabila komunikasi dilakukan secara bersama-sama oleh sebanyak *n* pengguna dan setiap dua pihak yang melakukan pertukaran kunci, maka akan terdapat sebanyak $C_2^n = \frac{n!}{(n-2)!.2!} = \frac{n.(n-1)}{2}$ kunci rahasia yang harus dipertukarkan secara aman.



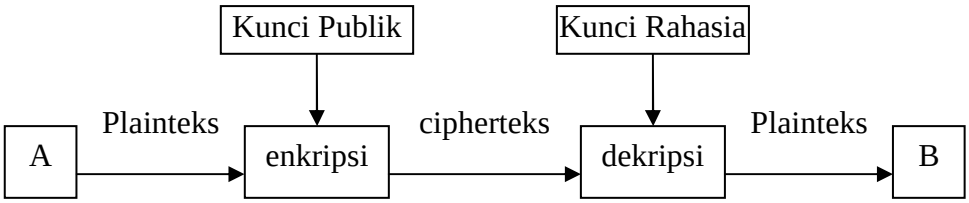
Gambar 2.1. Skema algoritma simetris

Contoh dari algoritma kriptografi simetris adalah Cipher Permutasi, Cipher Substitusi, Cipher Hill, OTP, RC6, Twofish, Magenta, FEAL, SAFER, LOKI, CAST, Rijndael (AES), Blowfish, GOST, A5, Kasumi, DES dan IDEA.

2.1.2.2. Algoritma Asimetris

Algoritma asimetris, sering juga disebut dengan *algoritma kunci publik*, menggunakan dua jenis kunci, yaitu *kunci publik* (*public key*) dan *kunci rahasia* (*secret key*). Kunci publik merupakan kunci yang digunakan untuk mengenkripsi pesan. Sedangkan kunci rahasia digunakan untuk mendekripsi pesan.

Kunci publik bersifat umum, artinya kunci ini tidak dirahasiakan sehingga dapat dilihat oleh siapa saja. Sedangkan kunci rahasia adalah kunci yang dirahasiakan dan hanya orang-orang tertentu saja yang boleh mengetahuinya. Keuntungan utama dari algoritma ini adalah memberikan jaminan keamanan kepada siapa saja yang melakukan pertukaran informasi meskipun di antara mereka tidak ada kesepakatan mengenai keamanan pesan terlebih dahulu maupun saling tidak mengenal satu sama lainnya.



Gambar 2.2. Skema algoritma asimetris

Algoritma asimetris pertama kali dipublikasikan oleh Diffie dan Hellman pada tahun 1976 dalam papernya yang berjudul *"New Directions in Cryptography"*. Menurut Diffie dan Hellman, ada beberapa syarat yang perlu diperhatikan pada algoritma asimetris, yaitu:

1. Penerima B membuat pasangan kunci, yaitu kunci publik k_{pB} dan kunci rahasia k_{rB}.
2. Pengirim A dengan kunci publik B dan pesan *x*, pesan dienkripsi dan diperoleh cipherteks $c = e_{k_{pB}}(x)$.
3. Penerima B untuk mendekripsi cipherteks menggunakan kunci privat B untuk mendapatkan kembali pesan aslinya

$$d_{k_{rB}}\left[e_{k_{pB}}(x)\right] = d_{k_{rB}}(c) = x\,.$$

4. Dengan mengetahui kunci publik k_{pB}, bagi penyerang akan kesulitan dalam melakukan untuk mendapatkan kunci rahasia.
5. Dengan mengetahui kunci publik k_{pB} dan cipherteks *c*, bagi penyerang akan mengalami kesulitan untuk mengetahui pesan *x*.

Contoh dari algoritma asimetris adalah RSA, ElGamal, McEliece, LUC dan DSA (*Digital Signature Algorithm*).

Dalam melakukan proses enkripsi, sering digunakan plainteks berupa data ataupun pesan yang besar, sehingga membutuhkan waktu yang lama apabila dilakukan proses sekaligus pada plainteks tersebut. Oleh karena itu, plainteks dapat dipotong-potong menjadi beberapa blok-blok yang sama panjang. Kemudian dari blok-blok yang diperoleh tersebut dilakukan proses enkripsi, dan hasil cipherteksnya dapat didekripsi dan digabungkan kembali menjadi plainteks. Algoritma kriptografi yang menggunakan mekanisme seperti ini disebut dengan cipher blok (*block cipher*).

2.1.3. Sistem Kriptografi

Definisi 2.1.3.1. (Stinson, 1995) *Sistem kriptografi* (*cryptosystem*) adalah suatu 5-*tuple* ($\mathcal{P}$, $\mathcal{C}$, $\mathcal{K}$, $\mathcal{E}$, $\mathcal{D}$) yang memenuhi kondisi sebagai berikut :

1. $\mathcal{P}$ adalah himpunan plainteks,
2. $\mathcal{C}$ adalah himpunan cipherteks,
3. $\mathcal{K}$ atau *ruang kunci* (*keyspace*), adalah himpunan kunci,

4. $\mathcal{E}$ adalah himpunan fungsi enkripsi $e_k : \mathcal{P} \to \mathcal{C}$,

5. $\mathcal{D}$ adalah himpunan fungsi dekripsi $d_k : \mathcal{C} \to \mathcal{P}$,

6. Untuk setiap $k \in \mathcal{K}$ terdapat $\varepsilon_k \in \mathcal{E}$ dan $d_k \in \mathcal{D}$. Setiap $e_k : \mathcal{P} \to \mathcal{C}$ dan $d_k : \mathcal{C} \to \mathcal{P}$ merupakan fungsi sedemikian hingga $d_k\left(\varepsilon_k(x)\right) = x$, untuk setiap plainteks $x \in \mathcal{P}$.

Suatu sistem kriptografi terdiri dari sebuah algoritma, seluruh kemungkinan plainteks, cipherteks dan kunci-kuncinya. Sistem kriptografi merupakan suatu fasilitas untuk mengkonversikan plainteks menjadi cipherteks, dan sebaliknya.

Setelah mengetahui konsep kriptografi, algoritma kriptografi serta jenis-jenisnya, berikut ini dibahas mengenai bilangan bulat dan hasil yang dapat diperoleh dari bilangan bulat yang digunakan sebagai landasan untuk membahas konsep matematis pada algoritma ElGamal.

2.2. Bilangan Bulat

Himpunan semua bilangan bulat yang dinotasikan dengan $\mathbb{Z}$ adalah himpunan $\{..., -3, -2, -1, 0, 1, 2, 3, ...\}$. Himpunan ini berperan sangat penting dalam kriptografi karena banyak algoritma kriptografi yang menggunakan sifat-sifat himpunan semua bilangan bulat dalam melakukan prosesnya. Pada himpunan ini berlaku sifat assosiatif, komutatif dan distributif terhadap operasi penjumlahan dan pergandaan biasa.

2.2.1. Divisibility

Definisi 2.2.1.1. (Buchmann, 2000) Diberikan $a, n \in \mathbb{Z}$. Bilangan bulat *a* dikatakan *membagi* (*divides*) *n* jika terdapat $b \in \mathbb{Z}$ sedemikian hingga $n = a.b$. Jika *a* membagi *n*, maka *a* disebut *pembagi* (*divisior*) *n*, dan *n* disebut *kelipatan* (*multiple*) *a*. Bilangan bulat *a* yang membagi *n* ditulis $a \mid n$.

Contoh 2.2.1.2.

$5 \mid 30$ dan $7 \mid 42$.

Teorema 2.2.1.3. (Buchmann, 2000) Diberikan $a, b, c \in \mathbb{Z}$.

1. Jika $a \mid b$ dan $b \mid c$, maka $a \mid c$.
2. Jika $a \mid b$, maka $a.c \mid b.c$ untuk setiap $c \in \mathbb{Z}$.
3. Jika $c \mid a$ dan $c \mid b$, maka $c \mid (d.a + e.b)$ untuk setiap $d, e \in \mathbb{Z}$.
4. Jika $a \mid b$ dan $b \neq 0$, maka $|a| \leq |b|$.
5. Jika $a \mid b$ dan $b \mid a$, maka $|a| = |b|$.

Bukti:

1. Jika $a \mid b$ dan $b \mid c$, maka terdapat $p, q \in \mathbb{Z}$ sedemikian hingga $b = a.p$ dan $c = b.q$. Akibatnya $c = b.q = (a.p).q = a.(p.q)$. Karena $p.q \in \mathbb{Z}$, diperoleh $a \mid c$.
2. Jika $a \mid b$, maka terdapat $p \in \mathbb{Z}$ sedemikian hingga $b = a.p$. Akibatnya, untuk sebarang $c \in \mathbb{Z}$ diperoleh $b.c = (a.p).c = p.(a.c)$. Terbukti bahwa $a.c \mid b.c$, untuk setiap $c \in \mathbb{Z}$.

3. Jika $c|a$ dan $c|b$, maka terdapat $p,q\in\mathbb{Z}$ sedemikian hingga $a=c.p$ dan $b=c.q$. Akibatnya, untuk sebarang $d,e\in\mathbb{Z}$ diperoleh $d.a+e.b=d.c.p+e.c.q=c.(d.p+e.q)$, dengan kata lain $c|(d.a+e.b)$.
4. Jika $a|b$ dan $b\neq 0$, maka terdapat $p\in\mathbb{Z}, p\neq 0$ sedemikian hingga $b=a.p$. Akibatnya $|b|=|a.p|\geq|a|$.
5. Diketahui $a|b$ dan $b|a$. Jika a = 0, maka b = 0, dan sebaliknya. Jika $a\neq 0$ dan $b\neq 0$, menggunakan hasil (4) diperoleh bahwa $|a|\leq|b|$ dan $|a|\geq|b|$, akibatnya $|a|=|b|$. □

2.2.2. Algoritma Pembagian pada Bilangan Bulat

Berikut ini diberikan sebuah teorema yang disebut dengan *algoritma pembagian pada bilangan bulat*, seperti dijelaskan pada Teorema 2.2.2.3.

Definisi 2.2.2.1. (Buchmann, 2000) Untuk setiap bilangan real $\alpha\in\mathbb{R}$ didefinisikan

$$\lfloor\alpha\rfloor=\max\{z\in\mathbb{Z}:z\leq\alpha\}.$$

Dengan demikian, $\lfloor\alpha\rfloor$ merupakan bilangan bulat terbesar yang lebih kecil atau sama dengan α.

Contoh 2.2.2.2.

1. $\lfloor 13,75\rfloor=13$.
2. $\lfloor -5,42\rfloor=-6$.

Teorema 2.2.2.3. (Buchmann, 2000) Jika a dan b bilangan bulat dengan $b>0$, maka terdapat dengan tunggal bilangan bulat q dan r sedemikian hingga $a=b.q+r$ dengan $0\le r<b$, yaitu $q=\left\lfloor \frac{a}{b} \right\rfloor$ dan $r=a-b.q$.

Bukti:

Diambil sebarang bilangan bulat a dan b dengan $b>0$, akan ditunjukkan bahwa terdapat $q=\left\lfloor \frac{a}{b} \right\rfloor \in \mathbb{Z}$ dan $r\in\mathbb{Z}$ sedemikian hingga $a=b.q+r$ dengan $0\le r<b$. Karena $a,b\in\mathbb{Z}$ dan $b>0$, menggunakan Definisi 2.2.2.1 diperoleh bilangan $q=\left\lfloor \frac{a}{b} \right\rfloor \in \mathbb{Z}$ sehingga diperoleh $a\ge b.q$. Akibatnya terdapat $r\in\mathbb{Z}, r\ge 0$ sehingga $a=b.q+r$. Jika b pembagi dari a, maka $a = b.q$ sehingga diperoleh $r=0$. Jika b bukan pembagi dari a, maka $a=q.b+r$ dengan hasil bagi $q=\left\lfloor \frac{a}{b} \right\rfloor \in \mathbb{Z}$, dan $r\in\mathbb{Z}$ adalah sisa a dibagi b. Jika diambil $r = b$, maka $a=b.(q+1)$ sehingga $q=\left\lfloor \left(\frac{a}{b}\right)-1 \right\rfloor$, akibatnya terjadi kontradiksi dengan yang diketahui yaitu $q=\left\lfloor \frac{a}{b} \right\rfloor$. Selanjutnya, dari hasil terakhir dan karena $b>0$, maka $0\le r<b$. Untuk membuktikan ketunggalannya, misalkan terdapat $q_1,q_2,r_1,r_2\in\mathbb{Z}$ sedemikian hingga $a=q_1.b+r_1$ dan $a=b.q_2+r_2$. Akibatnya diperoleh $(b.q_1+r_1)-(b.q_2+r_2)=0$ atau $b.(q_1-q_2)+(r_1-r_2)=0$. Karena $q_1=\left\lfloor \frac{a}{b} \right\rfloor$ dan $q_2=\left\lfloor \frac{a}{b} \right\rfloor$, maka $q_1=q_2$, sehingga

diperoleh $q_1 - q_2 = 0$. Akibatnya $r_1 - r_2 = 0$, sehingga diperoleh $r_1 = r_2$. Terbukti bahwa *q* dan *r* tunggal. Dengan demikian teorema terbukti. □

Pada teorema 2.2.2.3, bilangan bulat *q* disebut dengan *hasil bagi* (*quotient*) dan *r* disebut *sisa* (*remainder*) dari pembagian *a* dengan *b*, ditulis $r = a \bmod b$.

Contoh 2.2.2.4.

Diberikan bilangan bulat 25 dan 70. Menggunakan Definisi 2.2.2.1 diperoleh bilangan bulat $\left\lfloor \frac{70}{25} \right\rfloor = \lfloor 2,8 \rfloor = 2$. Menggunakan Teorema 2.2.2.3 terdapat dengan tunggal bilangan bulat *q* dan *r* sedemikian hingga $70 = 25.q + r$, dengan $0 \le r < 25$ yaitu $q = 2$ dan $r = 20$. Dapat dilihat bahwa $70 = 25.2 + 20$, dengan $0 \le 20 < 25$. Bilangan bulat 20 merupakan sisa pembagian, ditulis $20 = 70 \bmod 25$.

2.2.3. Representasi Bilangan Bulat

Bilangan bulat merupakan bilangan yang ditulis dengan ekspansi desimal, sedangkan pada komputer yang digunakan adalah ekspansi biner. Secara umum, bilangan bulat dapat direpresentasikan menggunakan ekspansi *b-adic* yang akan dijelaskan pada Definisi 2.2.3.3.

Teorema 2.2.3.1 di bawah ini dapat digunakan sebagai algoritma untuk merepresentasikan sebarang bilangan bulat positif *n* ke dalam suatu ekspansi *b-adic* yang diinginkan.

Teorema 2.2.3.1. (Rosen, 1992) Diberikan bilangan bulat positif *b* dengan $b > 1$. Untuk setiap bilangan bulat positif *a* dapat disajikan secara tunggal ke dalam bentuk ekspansi

$$a = r_k.b^k + r_{k-1}.b^{k-1} + ... + r_1.b + r_0,$$

dengan *k* adalah bilangan bulat nonnegatif, r_j adalah bilangan bulat dengan $0 \le r_j < b$ untuk $j = 0,1,...,k$ dan $r_k \neq 0$.

Bukti:

Menggunakan algoritma pembagian, langkah pertama, *a* dibagi dengan *b*, diperoleh

$$a = b.q_0 + r_0, \ 0 \le r_0 < b \tag{2.1}$$

Jika $q_0 \neq 0$, maka q_0 dibagi dengan *b*, diperoleh

$$q_0 = b.q_1 + r_1, \ 0 \le r_1 < b. \tag{2.2}$$

Selanjutnya, jika proses ini diteruskan, maka diperoleh

$$\begin{aligned}
q_1 &= b.q_2 + r_2, \ 0 \le r_2 < b \\
q_2 &= b.q_3 + r_3, \ 0 \le r_3 < b. \\
&\vdots \\
q_{k-2} &= b.q_{k-1} + r_{k-1}, \ 0 \le r_{k-1} < b \\
q_{k-1} &= b.0 + r_k, \ 0 \le r_k < b.
\end{aligned}$$

Pada langkah terakhir dari proses perhitungan, terlihat bahwa sisa terkahir yang diperoleh adalah 0. Jelas bahwa

$$a > q_0 > q_1 > q_2 > ... \ge 0.$$

Pada persamaan (2.1) diketahui

$$a = b.q_0 + r_0.$$

Menggunakan persamaan (2.2) diperoleh

$$a = b.(b.q_1 + r_1) + r_0 = b^2.q_1 + r_1.b + r_0 .$$

Selanjutnya, menggunakan substitusi untuk $q_1, q_2, ..., q_{k-1}$, diperoleh

$$\begin{aligned}
a &= b^3.q_2 + r_2.b^2 + r_1.b + r_0, \\
&\vdots \\
a &= b^{k-1}.q_{k-2} + r_{k-2}.b^{k-2} + ... + r_1.b + r_0, \\
a &= b^k.q_{k-1} + r_{k-1}.b^{k-1} + ... + r_1.b + r_0 \\
&= r_k.b^k + r_{k-1}.b^{k-1} + ... + r_1.b + r_0,
\end{aligned}$$

dengan $0 \le r_j < b$ untuk $j = 0,1,...,k$ dan $r_k \neq 0$, sebab $r_k = q_{k-1}$ adalah sisa terakhir yang tidak nol. Dengan demikian terbukti bahwa *a* dapat disajikan ke dalam bentuk ekspansi

$$a = r_k.b^k + r_{k-1}.b^{k-1} + ... + r_1.b + r_0 .$$

Selanjutnya, untuk membuktikan ketunggalannya, diasumsikan terdapat dua bentuk ekspansi dari *a*, yaitu

$$a = r_k.b^k + r_{k-1}.b^{k-1} + ... + r_1.b + r_0 \tag{2.3}$$

dan

$$a = c_k.b^k + c_{k-1}.b^{k-1} + ... + c_1.b + c_0 \tag{2.4}$$

dengan $0 \le r_k < b$ dan $0 \le c_k < b$. Selanjutnya, dari persamaan (2.3) dan (2.4) diperoleh

$$(r_k - c_k).b^k + (r_{k-1} - c_{k-1}).b^{k-1} + ... + (r_1 - c_1).b + (r_0 - c_0) = 0 . \tag{2.5}$$

Jika persamaan (2.3) dan (2.4) berbeda, maka terdapat bilangan bulat terkecil *j*, $0 \le j \le k$, sedemikian hingga $r_j \neq c_j$. Berarti dari persamaan (2.5) diperoleh bentuk

$$(r_k - c_k).b^k + (r_{k-1} - c_{k-1}).b^{k-1} + ... + (r_{j+1} - c_{j+1}).b^{j+1} + (r_j - c_j).b^j = 0$$

atau

$$b^j.\left[\left(r_k - c_k\right).b^{k-j} + ... + \left(r_{j+1} - c_{j+1}\right).b + \left(r_j - c_j\right)\right] = 0,$$

diperoleh

$$\left(r_k - c_k\right).b^{k-j} + ... + \left(r_{j+1} - c_{j+1}\right).b + \left(r_j - c_j\right) = 0,$$

atau

$$\begin{aligned} r_j - c_j &= \left(c_k - r_k\right).b^{k-j} + ... + \left(c_{j+1} - r_{j+1}\right).b \\ &= b.\left[\left(c_k - r_k\right).b^{k-j-1} + ... + \left(c_{j+1} - r_{j+1}\right)\right]. \end{aligned}$$

Dari sini, diperoleh $b \mid (r_j - c_j)$.

Karena $0 \le r_j < b$ dan $0 \le c_j < b$, maka $-b < r_j - c_j < b$. Selanjutnya, karena $b \mid (r_j - c_j)$ dan $-b < r_j - c_j < b$, akibatnya $r_j = c_j$. Kontradiksi dengan asumsi bahwa kedua ekspansi berbeda, yang benar adalah kedua bentuk ekspansi adalah sama. Dengan kata lain terbukti bahwa ekspansi dari *a* adalah tunggal. □

Pada Teorema 2.2.3.1 diperoleh suatu barisan $\left(r_k, r_{k-1}, ..., r_1, r_0\right)$, yaitu barisan yang elemennya diperoleh dari bentuk ekspansi suatu bilangan bulat.

Definisi 2.2.3.2. (Buchmann, 2000) Barisan $\left(r_k, r_{k-1}, ..., r_1, r_0\right)$ dari Teorema 2.2.3.1 disebut dengan *ekspansi b-adic* dari bilangan bulat *a*. Elemen-elemennya disebut *digits*. Bilangan bulat *b* pada Teorema 2.2.3.1 disebut dengan *basis*. Jika *b* = 2, barisannya disebut *ekspansi biner*. Jika *b* = 16, barisannya disebut *ekspansi*

heksadesimal. Barisan $(r_k, r_{k-1}, ..., r_1, r_0)$ dengan basis *b* ditulis dengan $(r_k, r_{k-1}, ..., r_1, r_0)_b$ atau $(r_k r_{k-1} ... r_1 r_0)_b$.

Contoh 2.2.3.3.

Ekspansi dengan basis 7 dari 135 adalah $135 = 2.7^2 + 3.7 + 6 = (236)_7$. Ekspansi biner $(10010011)_2 = 1.2^7 + 1.2^4 + 1.2^1 + 1 = 147$.

Berikut ini diberikan sebuah algoritma yang dapat digunakan untuk merepresentasikan suatu bilangan bulat ke dalam ekspansi yang diinginkan. Algoritma ini didasarkan pada Teorema 2.2.3.1.

Algoritma 2.1 : Representasi Bilangan Bulat (Menezes, Oorschot and Vanstone, 1996)

Input : Bilangan bulat *a* dan *b*, dengan $a \geq 0, b \geq 2$.

Output : Ekspansi *b-adic* $a = (r_k r_{k-1} ... r_1 r_0)_b$, $k \geq 0$ dan $r_k \neq 0$ jika $k \geq 1$.

Langkah :

1. $i \leftarrow 0, x \leftarrow a, q \leftarrow \left\lfloor \frac{x}{b} \right\rfloor, r_i \leftarrow (x - q.b)$.

2. *While* $q > 0$, lakukan langkah berikut:

   2.1. $i \leftarrow i + 1, x \leftarrow q, q \leftarrow \left\lfloor \frac{x}{b} \right\rfloor, r_i \leftarrow (x - q.b)$.

3. *Output* $((r_i r_{i-1} ... r_1 r_0))$.

2.2.4. Pembagi Persekutuan Terbesar

Berikut ini dijelaskan pengertian dan sifat-sifat suatu bilangan yang disebut dengan pembagi persekutuan terbesar.

Definisi 2.2.4.1. (Buchmann, 2000) *Pembagi persekutuan* dari bilangan bulat $a_1, a_2, ..., a_k$ adalah suatu bilangan bulat yang membagi $a_1, a_2, ..., a_k$.

Contoh 2.2.4.2.

Diberikan $30, 50 \in \mathbb{Z}$, maka $5, 10 \in \mathbb{Z}$ adalah pembagi persekutuan dari 30 dan 50, sebab 5 dan 10 membagi 30 dan 50.

Definisi 2.2.4.3. (Buchmann, 2000) Diberikan $a_1, a_2, ..., a_k \in \mathbb{Z}$. Suatu bilangan bulat nonnegatif *d* disebut *pembagi persekutuan terbesar* (*greatest common divisior*) dari $a_1, a_2, ..., a_k$ jika

1. Bilangan bulat *d* merupakan pembagi persekutan dari $a_1, a_2, ..., a_k$, yaitu *d* membagi $a_1, a_2, ..., a_k$,
2. Untuk sebarang bilangan bulat *c*, jika *c* membagi $a_1, a_2, ..., a_k$, maka *c* membagi *d*.

Bilangan bulat *d* tersebut dinotasikan dengan $d = \gcd(a_1, a_2, ..., a_k)$.

Dengan kata lain, pembagi persekutuan terbesar adalah nilai maksimum dari semua pembagi persekutuan, yaitu

$$\gcd(a_1, a_2, ..., a_k) = \max\{n \in \mathbb{Z} : n \mid a_1 \text{ dan } n \mid a_2 \text{ dan} ... \text{dan } n \mid a_k\}.$$

Contoh 2.2.4.4.

Diberikan $50, 75 \in \mathbb{Z}$, maka

$$\begin{aligned} \gcd(50,75) &= \max\{n \in \mathbb{Z} : n \mid 50 \text{ dan } n \mid 75\} \\ &= \max\{-25,-5,-1,1,5,25\} \\ &= 25. \end{aligned}$$

Selanjutnya, dijelaskan sebuah cara untuk merepresentasikan pembagi persekutuan terbesar bilangan bulat. Diberikan notasi sebagai berikut,

$$a_1.\mathbb{Z} + a_2.\mathbb{Z} + ... + a_k.\mathbb{Z} = \{a_1.z_1 + a_2.z_2 + ... + a_k.z_k : z_i \in \mathbb{Z}, 1 \le i \le k\}$$

yaitu himpunan semua *kombinasi linear* bilangan bulat dari $a_i, 1 \le i \le k$.

Contoh 2.2.4.5.

Himpunan semua kombinasi linear dari 4 dan 5 adalah

$$4.\mathbb{Z} + 5.\mathbb{Z} = \{4.z_1 + 5.z_2 : z_1, z_2 \in \mathbb{Z}\}.$$

Teorema 2.2.4.6. (Buchmann, 2000) Himpunan semua kombinasi linear dari bilangan bulat *a* dan *b* adalah himpunan semua bilangan bulat kelipatan $\gcd(a,b)$, yaitu

$$a.\mathbb{Z} + b.\mathbb{Z} = \gcd(a,b).\mathbb{Z}. \qquad (2.6)$$

Bukti:

Untuk $a = b = 0$, persamaan (2.6) benar. Diambil *a* atau *b* tidak nol, dibentuk himpunan $I = a.\mathbb{Z} + b.\mathbb{Z}$. Diambil $g \in I$, yaitu bilangan bulat positif terkecil dalam *I*.

Diklaim bahwa $I = g.\mathbb{Z}$. Diambil sebuah elemen tak nol $c \in I$. Akan ditunjukkan bahwa $c = q.g$ untuk suatu $q \in \mathbb{Z}$. Menggunakan algoritma pembagian, terdapat $q, r \in \mathbb{Z}$ dengan $c = q.g + r$ dan $0 \le r < g$. Akibatnya, $r = c - q.g \in I$. Akan tetapi, karena g adalah bilangan bulat positif terkecil dalam *I*, maka $r = 0$ dan $c = q.g$. Selanjutnya, akan ditunjukkan bahwa $g = \gcd(a,b)$. Karena $a, b \in I$, maka g adalah pembagi persekutuan dari *a* dan *b*. Karena $g \in I$, maka terdapat $x, y \in \mathbb{Z}$ dengan $g = x.a + y.b$. Oleh karena itu, jika *d* adalah pembagi persekutuan dari *a* dan *b*, maka *d* juga merupakan pembagi persekutuan dari *g*. Menggunakan Teorema 2.2.1.3 diperoleh bahwa $|d| \le g$. Terbukti bahwa $g = \gcd(a,b)$. □

Akibat 2.2.4.7. (Buchmann, 2000) Untuk setiap $a, b, n \in \mathbb{Z}$ persamaan $a.x + b.y = n$ mempunyai penyelesaian yaitu bilangan bulat *x* dan *y* jika dan hanya jika $\gcd(a,b)$ membagi *n*.

Bukti:

$\Rightarrow$ Misalkan terdapat bilangan bulat x dan y yang memenuhi $a.x + b.y = n$, maka $n \in a.\mathbb{Z} + b.\mathbb{Z}$. Menggunakan Teorema 2.2.4.6 diperoleh bahwa $n \in \gcd(a,b).\mathbb{Z}$, misalkan $n = \gcd(a,b).z'$ untuk suatu $z' \in \mathbb{Z}$. Dari sini diperoleh bahwa *n* adalah kelipatan dari $\gcd(a,b)$. Dengan kata lain, $\gcd(a,b)$ membagi *n*.

$\Leftarrow$ Misalkan *n* adalah kelipatan dari $\gcd(a,b)$, maka $n \in \gcd(a,b).\mathbb{Z}$. Menggunakan Teorema 2.2.4.6 diperoleh bahwa $n \in a.\mathbb{Z} + b.\mathbb{Z}$. Akibatnya terdapat $x, y \in \mathbb{Z}$ sedemikian hingga $n = a.x + b.y$. □

Akibat 2.2.4.8. (Buchmann, 2000) Diberikan $a,b\in\mathbb{Z}$, maka terdapat $x,y\in\mathbb{Z}$ dengan $a.x+b.y=\gcd(a,b)$.

Bukti:

Karena $\gcd(a,b)$ membagi dirinya sendiri, menggunakan Akibat 2.2.4.7, Akibat 2.2.4.8 terbukti. □

Definisi 2.2.4.9. [(Stinson, 1995), (Buchmann, 2000)] Diberikan $a,b\in\mathbb{Z}$. Jika $\gcd(a,b)=1$, maka *a* dikatakan *relatif prima* dengan *b*. Bilangan bulat $a_1,a_2,...,a_n$ dikatakan *saling relatif prima* jika $\gcd(a_1,a_2,...,a_n)=1$.

Contoh 2.2.4.10.

Karena $\gcd(17,30)=1$, maka 17 relatif prima dengan 30.

Teorema 2.2.4.11. (Fraleigh, 2000) Jika bilangan bulat *a* dan *b* relatif prima dan $a\mid b.m$, maka $a\mid m$.

Bukti:

Diketahui *a* dan *b* relatif prima, yaitu $\gcd(a,b)=1$ dan $a\mid b.m$. Menggunakan Akibat 2.2.4.8 maka terdapat $x,y\in\mathbb{Z}$ sedemikian hingga $a.x+b.y=1$. Selanjutnya, kedua ruas dikalikan dengan $m\in\mathbb{Z}$, diperoleh $a.x.m+b.y.m=m$. Karena $a\mid a.x.m$ dan $a\mid b.y.m$, maka $a\mid m$. □

2.2.5. Algoritma Euclide

Berikut ini diberikan sebuah algoritma yang dapat digunakan untuk menghitung nilai pembagi persekutuan terbesar dari dua bilangan bulat dengan sangat efisien. Algoritma ini didasarkan pada teorema di bawah ini.

Teorema 2.2.5.1. (Buchmann, 2000) Diberikan $a,b \in \mathbb{Z}$.

1. Jika $b = 0$, maka $\gcd(a,b) = |a|$.
2. Jika $b \neq 0$, maka $\gcd(a,b) = \gcd(b, a \bmod b)$.

Bukti:

1. Dengan sendirinya langsung terbukti, sebab $\gcd(a,b) = \gcd(a,0) = |a|$.
2. Misalkan $d = \gcd(a,b)$ dan $r = a \bmod b$. Menurut Teorema 2.2.2.3, terdapat $q \in \mathbb{Z}$ dengan $a = q.b + r$. Karena $r = a - b.q$ maka $d \mid r$. Akan ditunjukkan bahwa $d = \gcd(b,r)$. Diambil sebarang bilangan bulat t sedemikian hingga $t \mid b$ dan $t \mid r$, yaitu terdapat $n,m \in \mathbb{Z}$ sedemikian hingga $b = n.t$ dan $r = m.t$. Diperoleh bahwa $a = n.t.q + m.t = t.(n.q + m)$ atau $t \mid a$. Diketahui $d = \gcd(a,b)$, karena $t \mid a$ dan $t \mid b$ maka $t \leq d$ dan $t \mid d$. Terbukti bahwa $d = \gcd(a,b) = \gcd(b, a \bmod b)$. □

Misal diberikan bilangan bulat positif r_0 dan r_1, dengan $r_0 \geq r_1$. Selanjutnya dihitung menggunakan algoritma pembagian sebagai berikut.

$$
\begin{aligned}
r_0 &= q_1.r_1 + r_2, && 0 < r_2 < r_1 \\
r_1 &= q_2.r_2 + r_3, && 0 < r_3 < r_2 \\
&\vdots \\
r_{n-2} &= q_{n-1}.r_{n-1} + r_n, && 0 < r_n < r_{n-1} \\
r_{n-1} &= q_n.r_n .
\end{aligned}
$$

Menggunakan Teorema 2.2.5.1, dapat ditunjukkan bahwa $\gcd(r_0, r_1) = \gcd(r_1, r_2) = ... = \gcd(r_{n-1}, r_n) = \gcd(r_n, 0) = r_n$. Jika diberikan bilangan bulat *a* dan *b* dengan $|a| \geq |b|$, maka dengan menentukan $r_0 = |a|$ dan $r_1 = |b|$, Teorema 2.2.5.1 dapat disajikan sebagai algoritma yang dapat digunakan untuk menghitung nilai $\gcd(a,b)$. Algoritma ini disebut dengan algoritma *Euclide*.

Algoritma 2.2 : Algoritma Euclide (Menezes, Oorschot and Vanstone, 1996)

Input : Bilangan bulat nonnegatif *a* dan *b*, $a \geq b$.

Output : $\gcd(a,b)$.

Langkah :

1. *While* $b \neq 0$ *do* :

   1.1 *Set* $r \leftarrow a \bmod b$, $a \leftarrow b$, $b \leftarrow r$.

2. *Output* (*a*).

Contoh 2.2.5.2. (Buchmann, 2000)

Akan dihitung nilai gcd(100,35). Menggunakan algoritma Euclide diperoleh:

Langkah 1 : $\gcd(100,35) = \gcd(35, 100 \bmod 35) = \gcd(35,30)$.

Langkah 2 : $\gcd(35,30) = \gcd(30, 35 \bmod 30) = \gcd(30,5)$.

Langkah 3 : $\gcd(30,5) = \gcd(5, 30 \bmod 5) = \gcd(5,0)$.

Langkah 4 : $\gcd(5,0) = 5$.

Jadi, gcd(100,35) = gcd(35,30) = gcd(30,5) = gcd(5,0) = 5.

Tabel 2.1. Perhitungan gcd(100,35) menggunakan algoritma Euclide

k	0	1	2	3	4
a_k	100	35	30	5	0
q_k		2	1	6	

2.2.6. Algoritma Euclide yang Diperluas

Dengan algoritma Euclide dapat dihitung nilai pembagi persekutuan terbesar dari bilangan bulat *a* dan *b*. Menurut Akibat 2.2.4.8, terdapat bilangan bulat *x* dan *y* dengan $\gcd(a,b) = a.x + b.y$. Selanjutnya, algoritma Euclide dapat diperluas sedemikian hingga dapat digunakan untuk menghitung nilai *x* dan *y* tersebut. Pada pembahasan tentang algoritma Euclide diketahui bahwa diperoleh barisan sisa yaitu $r_0, r_1, ..., r_n$ dan barisan hasil bagi yaitu $q_1, q_2, ..., q_n$.

Selanjutnya, dikontruksi dua barisan (x_k) dan (y_k) yang diperoleh dari barisan sisa dan hasil bagi sedemikian hingga pada iterasi terakhir diperoleh $x = (-1)^n .x_n$ dan $y = (-1)^n .y_n$.

Pertama, ditentukan nilai awal yaitu $x_0 = 1$, $x_1 = 0$, $y_0 = 0$ dan $y_1 = 1$. Selanjutnya, diberikan persamaan $x_{k+1} = q_k .x_k + x_{k-1}$ dan $y_{k+1} = q_k .y_k + y_{k-1}$, $0 \le k \le n$.

Teorema 2.2.6.1. (Buchmann, 2000) Jika $x_0 = 1,\ x_1 = 0,\ y_0 = 0,\ y_1 = 1$ dengan $x_{k+1} = q_k.x_k + x_{k-1}$ dan $y_{k+1} = q_k.y_k + y_{k-1}$, maka

$$r_k = (-1)^k.x_k.a + (-1)^{k+1}.y_k.b \text{, untuk } 0 \le k \le n+1.$$

Bukti:

Akan dibuktikan menggunakan induksi.

Untuk $k = 0$ diperoleh $r_0 = a = (1).a - (0).b = x_0.a - y_0.b.$ Selanjutnya,

$$r_1 = b = -(0).a + (1).b = -x_1.a + y_1.b\,.$$

Misalkan pernyataan benar untuk $k \le n$, maka pernyataan benar untuk $k = n$ yaitu

$$r_n = (-1)^n.x_n.a + (-1)^{n+1}.y_n.b\,.$$

Akan dibuktikan bahwa pernyataan benar untuk $k = n + 1$, yaitu

$$r_{n+1} = (-1)^{n+1}.x_{n+1}.a + (-1)^{n+2}.y_{n+1}.b\,.$$

Dari pembahasan tentang algoritma Euclide, diketahui bahwa $r_{n+1} = r_{n-1} - q_n.r_n$. Oleh karena itu

$$\begin{aligned}
r_{n+1} &= r_{n-1} - q_n.r_n \\
&= \left[(-1)^{n-1}.x_{n-1}.a + (-1)^n.y_{n-1}.b\right] - q_n.\left[(-1)^n.x_n.a + (-1)^{n+1}.y_n.b\right] \\
&= \left[(-1)^{n-1}.x_{n-1} - q_n.(-1)^n.x_n\right].a + \left[(-1)^n.y_{n-1} - q_n.(-1)^{n+1}.y_n\right].b \\
&= \left[(-1)^{n+1}.x_{n-1} + q_n.(-1)^{n+1}.x_n\right].a + \left[(-1)^{n+2}.y_{n-1} + q_n.(-1)^{n+2}.y_n\right].b \\
&= (-1)^{n+1}.\left[x_{n-1} + q_n.x_n\right].a + (-1)^{n+2}.\left[y_{n-1} + q_n.y_n\right].b \\
&= (-1)^{n+1}.x_{n+1}.a + (-1)^{n+2}.y_{n+1}.b.
\end{aligned}$$

Dengan demikian Teorema 2.2.6.1 terbukti. □

Contoh 2.2.6.2. (Buchmann, 2000)

Seperti pada Contoh 2.2.5.2, akan dihitung nilai gcd(100,35). Yaitu $\gcd(100,35) = (-1).100 + 3.35 = 5$, diperoleh hasil yang sama pada Contoh 2.2.5.2.

Tabel 2.2. Perhitungan *x* dan *y* menggunakan Teorema 2.2.6.1

k	0	1	2	3	4
a_k	100	35	30	5	0
q_k		2	1	6	
x_k	1	0	1	1	7
y_k	0	1	2	3	20

Algoritma 2.3 : Algoritma Euclide yang Diperluas (Menezes, Oorschot and Vanstone, 1996)

Input : $a, b \in \mathbb{Z}$, $a \geq b$.

Output : $d = \gcd(a,b)$ dan $x, y \in \mathbb{Z}$ yang memenuhi $a.x + b.y = d$.

Langkah :

1. Jika $b = 0$, maka *set* $d \leftarrow a,\ x \leftarrow 1,\ y \leftarrow 0$, *output (d, x, y).*
2. *Set* $x_2 \leftarrow 1,\ x_1 \leftarrow 0,\ y_2 \leftarrow 0,\ y_1 \leftarrow 1$.
3. *While* $b > 0$:

   3.1 $q \leftarrow \left\lfloor \frac{a}{b} \right\rfloor,\ r \leftarrow a - q.b,\ x \leftarrow x_2 - q.x_1,\ y \leftarrow y_2 - q.y_1$.

   3.2 $a \leftarrow b,\ b \leftarrow r,\ x_2 \leftarrow x_1,\ x_1 \leftarrow x,\ y_2 \leftarrow y_1,\ y_1 \leftarrow y$.
4. *Set* $d \leftarrow a,\ x \leftarrow x_2,\ y \leftarrow y_2,$ *output (d, x, y).*

Contoh 2.2.6.3. (Buchmann, 2000)

Seperti pada Contoh 2.2.5.2 diperoleh $\gcd(100,35) = (-1).100 + 3.35 = 5$, bilangan bulat x = (-1) dan y = 3. Menggunakan Algoritma 2.3, diperoleh hasil perhitungan seperti pada tabel di bawah ini.

Tabel 2.3. Perhitungan menggunakan algoritma Euclide yang diperluas

k	0	1	2	3	4
a_k	100	35	30	5	0
q_k		2	1	6	
x_k	1	0	1	-1	7
y_k	0	1	-2	3	20

Dan ternyata diperoleh hasil yang sama seperti pada Contoh 2.2.6.2.

2.2.7. Faktorisasi ke Bilangan Prima

Selanjutnya, dijelaskan pengertian dan sifat-sifat suatu bilangan yang disebut dengan bilangan prima, yaitu bilangan yang hanya dapat dibagi oleh 1 dan bilangan itu sendiri. Bilangan prima memainkan peran yang penting pada beberapa algoritma kriptografi kunci publik, seperti algoritma ElGamal dan RSA.

Definisi 2.2.7.1. (Buchmann, 2000) Suatu bilangan bulat $p > 1$ disebut *prima* jika p hanya mempunyai tepat dua bilangan pembagi positif yaitu 1 dan p. Jika tidak, p disebut *komposit*. Jika p bilangan prima yang membagi suatu bilangan bulat a, maka p disebut *pembagi prima* dari a.

Contoh 2.2.7.2.

Bilangan bulat 2, 3, 5, 7 dan 11 adalah bilangan prima. Sedangkan 4, 6, 8, 9 dan 10 adalah bilangan komposit.

Teorema 2.2.7.3. (Buchmann, 2000) Setiap bilangan bulat $a > 1$ mempunyai bilangan pembagi prima.

Bukti:

Diketahui bilangan bulat *a* mempunyai suatu pembagi yang lebih besar dari 1, yaitu *a* sendiri. Pandang semua pembagi *a* yang lebih besar dari 1, misalkan *p* adalah yang terkecil. Maka *p* pastilah prima, sebab jika tidak, maka *p* akan mempunyai suatu pembagi *b* dengan $1 < b < p \le a$. Timbul kontradiksi dengan asumsi bahwa *p* adalah pembagi terkecil dari *a* yang lebih besar dari 1. □

Contoh 2.2.7.4.

1. 100 mempunyai pembagi prima yaitu 2 dan 5.
2. 23 mempunyai pembagi prima yaitu 23 sendiri.

Lemma 2.2.7.5. (Buchmann, 2000) Jika suatu bilangan prima membagi hasil perkalian dari dua bilangan bulat, maka bilangan prima tersebut membagi paling sedikit satu faktornya.

Bukti:

Diberikan $a, b \in \mathbb{Z}$ dan bilangan prima *p* yang membagi *a.b* tetapi tidak membagi *a*. Karena *p* bilangan prima, maka $\gcd(a, p) = 1$. Menurut Akibat 2.2.4.8, terdapat

$x, y \in \mathbb{Z}$ sedemikian hingga $1 = a.x + p.y$. Akibatnya $b = a.b.x + p.b.y$, sehingga p membagi $a.b.x$ dan $p.b.y$. Menggunakan Teorema 2.2.1.3 diperoleh bahwa p merupakan pembagi dari b. □

Akibat 2.2.7.6. (Buchmann, 2000) Jika suatu bilangan prima p membagi $\prod_{i=1}^{k} q_i$ dengan $q_1, q_2, ..., q_k$ adalah bilangan-bilangan prima, maka p sama dengan salah satu dari $q_1, q_2, ..., q_k$.

Bukti:

Untuk k = 1, p jelas merupakan pembagi dari q_1 yaitu $p = q_1$. Untuk $k > 1$, p membagi $q_1.(q_2.q_3...q_k)$. Menggunakan Lemma 2.2.7.5 diperoleh bahwa p membagi q_1 atau $q_2.q_3...q_k$. Jika $p = q_1$ maka bukti selesai. Jika $p \neq q_1$, maka p membagi $q_2.(q_3...q_k)$. Sehingga p membagi q_2 atau $q_3...q_k$. Jika $p = q_2$ maka bukti selesai. Jika $p \neq q_2$, maka p membagi $q_3.(q_4...q_k)$, dan seterusnya. Dengan demikian terdapat q_i, $1 \leq i \leq k$ sedemikian hingga $p = q_i$. □

Teorema 2.2.7.7. (Buchmann, 2000) Setiap bilangan bulat $a > 1$ dapat disajikan sebagai hasil kali dari sejumlah bilangan prima berhingga secara tunggal.

Bukti:

Akan dibuktikan menggunakan induksi. Diberikan sebarang bilangan bulat $a > 1$. Untuk a = 2, maka jelas a merupakan hasil kali dari bilangan prima. Untuk $a > 2$, diasumsikan benar untuk $a - 1$ dan untuk setiap m dengan $2 \leq m \leq a - 1$. Akan

ditunjukkan bahwa *a* merupakan hasil kali dari sejumlah bilangan prima. Jika *a* merupakan bilangan prima, maka bukti selesai. Jika *a* merupakan bilangan komposit, maka *a* dapat dinyatakan sebagai $a = m_1.m_2...m_k$ dengan $m_i \in \mathbb{N}$ dan $1 < m_i < a$, $1 \le i \le k$. Menurut asumsi yang diambil di atas, maka m_i adalah hasil kali dari sejumlah bilangan prima. Akibatnya, $a = m_1.m_2...m_k$ juga merupakan hasil kali dari sejumlah bilangan prima.

Untuk membuktikan ketunggalannya, misalkan $a = p_1.p_2...p_r$ dan $a = q_1.q_2...q_s$, dengan $p_1, p_2, ..., p_r, q_1, q_2, ..., q_s$ adalah bilangan-bilangan prima. Akan ditunjukkan bahwa penyajian bilangan bulat *a* adalah tunggal, yaitu $r = s$. Diasumsikan benar untuk setiap m dengan $2 \le m \le a-1$. Karena $a = p_1.p_2...p_r = q_1.q_2...q_s$, maka p_1 membagi $q_1.q_2...q_s$. Menggunakan Akibat 2.2.7.6, diperoleh bahwa p_1 adalah salah satu dari $q_1, q_2, ..., q_s$. Tanpa mengurangi keumuman, diambil $p_1 = q_1$. Berdasarkan asumsi induksi, faktorisasi prima dari $\frac{a}{p_1} = \frac{a}{q_1}$ adalah tunggal. Sehingga diperoleh bahwa $r = s$. Terbukti bahwa penyajian bilangan bulat *a* adalah tunggal. □

Untuk mengecek apakah suatu bilangan bulat ganjil $a > 1$ adalah bilangan prima, dilakukan suatu *tes keprimaan* (*primality test*), yaitu suatu algoritma untuk membuktikan bahwa suatu bilangan bulat positif ganjil adalah bilangan prima atau komposit. Berikut ini diberikan sebuah tes keprimaan yang didasarkan pada Definisi 2.2.7.1.

Algoritma 2.4 : Tes Keprimaan Biasa

Input : Bilangan bulat ganjil $a > 1$.

Output : Pernyataan ”prima” atau ”komposit”.

Langkah :

1. *Set* $b \leftarrow 1$.
2. *Repeat* :

   2.1. $b \leftarrow b + 1$.

   2.2. $c \leftarrow a \bmod b$.
3. *Until* $c = 0$.
4. Jika $a = b$, maka *output*(”prima”).
5. Jika $a \neq b$, maka *output* (”komposit”).

2.3. Dasar Struktur Aljabar

Selanjutnya, pada subbab ini dijelaskan beberapa konsep dasar struktur aljabar seperti relasi ekuivalensi, grup, grup siklik, grup faktor, homomorfisma, gelanggang dan lapangan. Konsep ini penting, karena pada pembahasan selanjutnya mengenai algoritma ElGamal, perhitungan-perhitungannya dilakukan di dalam suatu struktur aljabar.

2.3.1. Partisi dan Relasi Ekuivalensi

Berikut ini dijelaskan tentang partisi, relasi ekuivalensi dan klas ekuivalensi pada suatu himpunan.

Definisi 2.3.1.1. (Fraleigh, 2000) Suatu *partisi* pada himpunan tak kosong *S* adalah suatu dekomposisi *S* ke dalam subset-subset yang saling asing sedemikian hingga setiap elemen dari *S* berada pada tepat satu subset. Subset yang demikian ini dinamakan *cell*.

Definisi 2.3.1.2. (Fraleigh, 2000) Diberikan himpunan tak kosong *S* dan ~ adalah relasi antar elemen-elemen *S*. Relasi ~ disebut *relasi ekuivalensi* jika memenuhi sifat-sifat berikut. Untuk setiap $a, b, c \in S$

1. Refleksif, yaitu $a \sim a$,
2. Simetris, yaitu jika $a \sim b$, maka $b \sim a$,
3. Transitif, yaitu jika $a \sim b$ dan $b \sim c$, maka $a \sim c$.

Dengan adanya suatu relasi ekuivalensi pada *S*, maka dapat ditentukan suatu partisi pada *S*. Partisi ini mendekomposisi *S* menjadi *cell-cell*. *Cell* yang memuat $a \in S$ dilambangkan dengan $\bar{a} = \{x \in S : x \sim a\}$. *Cell* seperti ini disebut dengan *klas ekuivalensi* yang memuat *a*.

2.3.2. Grup

Berikut ini dijelaskan mengenai suatu struktur aljabar yang disebut dengan grup. Grup merupakan suatu himpunan tak kosong yang dilengkapi dengan operasi biner dan memenuhi beberapa sifat, seperti dijelaskan berikut ini.

Definisi 2.3.2.1. (Fraleigh, 2000) Diberikan sebarang himpunan tidak kosong *G* dan operasi biner "*" pada *G*, maka *G* disebut *grup* terhadap operasi biner "*" dan ditulis $(G,*)$ jika dipenuhi

1. Operasi biner "*" pada *G* bersifat assosiatif,
2. Terdapat dengan tunggal elemen identitas yaitu $e \in G$ sedemikian hingga untuk setiap $a \in G$ berlaku $e * a = a * e = a$,
3. Untuk setiap $a \in G$ terdapat elemen inversnya, yaitu $a^{-1} \in G$ sedemikian hingga berlaku $a * a^{-1} = a^{-1} * a = e$.

Suatu grup $(G,*)$ disebut *Abelian* jika operasi binernya bersifat komutatif. Selanjutnya, grup $(G,*)$ dapat dituliskan dengan *G* apabila operasi binernya telah diketahui.

Definisi 2.3.2.2. (Fraleigh, 2000) Diberikan grup *G* dan subset tak kosong $H \subseteq G$. Subset *H* disebut *subgrup G* jika terhadap operasi biner yang sama pada *G*, maka *H* membentuk grup, ditulis $H < G$.

Selanjutnya diberikan beberapa definisi dan teorema yang menjelaskan sifat-sifat grup dan elemen grup. Seperti subgrup, order, grup siklik, pembangun, koset dan subgrup normal. Diberikan grup *G* dan subset tak kosong $H \subseteq G$.

Teorema 2.3.2.3. (Fraleigh, 2000) Subset tak kosong *H* merupakan subgrup *G* jika dan hanya $a * b^{-1} \in H$, untuk setiap $a, b \in H$.

Definisi 2.3.2.4. (Fraleigh, 2000) Jika *G* mempunyai banyak elemen yang berhingga, maka *G* disebut *grup berhingga* (*finite group*) dan banyaknya elemen *G* disebut *order G*, ditulis $|G|$.

Definisi 2.3.2.5. (Fraleigh, 2000) Diberikan *H* subgrup *G* dan $a \in G$. Didefinisikan himpunan $Ha = \{h * a : h \in H\}$ dan $aH = \{a * h : h \in H\}$, maka *Ha* disebut dengan *koset kanan* dan *aH* disebut dengan *koset kiri*. Jika *aH = Ha*, maka *H* disebut *subgrup normal* dan ditulis $H \lhd G$.

Definisi 2.3.2.6. (Fraleigh, 2000) Jika terdapat $a \in G$ sedemikian hingga untuk setiap $x \in G$, $x = a^k = \underbrace{a * a * ... * a}_{k \text{ faktor}}$, untuk suatu $k \in \mathbb{Z}$, maka *G* disebut *grup siklik* yang dibangun oleh *a*. Selanjutnya, *a* disebut *pembangun G* dan *k* disebut dengan *eksponen*, ditulis $G = \{a^n : n \in \mathbb{Z}\} = \langle a \rangle$.

Berikut ini diberikan sebuah teorema yang menyatakan bahwa order dari subgrup pasti membagi order grup. Teorema 2.3.2.7 di bawah ini disebut dengan *teorema Lagrange*.

Teorema 2.3.2.7. (Fraleigh, 2000) Jika $|G| = n$ dan *H* subgrup *G* dengan $|H| = m$, maka $m \mid n$.

2.3.3. Homomorfisma Grup

Selanjutnya diberikan pengertian tentang homomorfisma grup, yaitu suatu pemetaan dari suatu grup ke grup yang lain dan bersifat mengawetkan operasi.

Definisi 2.3.3.1. (Fraleigh, 2000) Diberikan grup $(G,*)$ dan $(G',*')$. Suatu pemetaan $\phi : G \to G'$ disebut *homomorfisma grup* jika untuk setiap $a,b \in G$,

$$\phi(a*b) = \phi(a) *' \phi(b).$$

Selanjutnya, jika ϕ bersifat injektif, maka ϕ disebut *monomorfisma grup*. Jika ϕ bersifat surjektif, maka ϕ disebut *epimorfisma grup*. Jika ϕ bersifat bijektif, maka ϕ disebut *isomorfisma grup*. Jika terdapat isomorfisma dari G ke G', maka G dikatakan *isomorfis* dengan G', ditulis $G \cong G'$.

Selanjutnya, dengan memahami sifat isomorfisma, dapat disimpulkan bahwa jika $G \cong G'$, maka G dan G' mempunyai struktur yang identik. Dengan demikian, untuk menyelidiki G' cukup dengan menyelidiki G, dan juga sebaliknya.

Definisi 2.3.3.2. (Fraleigh, 2000) Diberikan $\phi : G \to G'$, dan diberikan $A \subseteq G$ dan $B \subseteq G'$. *Peta* A adalah himpunan $\phi[A] = \{\phi(a) : a \in A\}$. *Range* ϕ adalah himpunan $\phi[G]$. *Prapeta* yaitu $\phi^{-1}[B] = \{x \in G : \phi(x) \in B\}$. Jika e' adalah elemen identitas grup G', maka $\phi^{-1}[\{e'\}] = \{x \in G : \phi(x) = e'\}$ disebut *kernel* ϕ, ditulis $\ker(\phi)$.

Dapat dilihat bahwa homomorfisma ϕ akan bersifat injektif apabila $\ker(\phi) = \{e\}$, dengan *e* adalah elemen identitas grup *G*, dan akan bersifat surjektif apabila $\phi[G] = G'$.

Berikut ini diberikan pengertian tentang suatu grup yang disebut dengan grup faktor. Selanjutnya, pada grup ini dapat dibentuk suatu isomorfisma dengan peta isomorfismanya.

Definisi 2.3.3.3. (Fraleigh, 2000) Diberikan *H* subgrup normal dari grup $(G,*)$. Didefinisikan himpunan $G/H = \{aH : a \in G\}$ dan operasi biner "$*$" pada G/H sebagai berikut. Untuk sebarang $aH, bH \in G/H$,

$$aH * bH = (a*b)H .$$

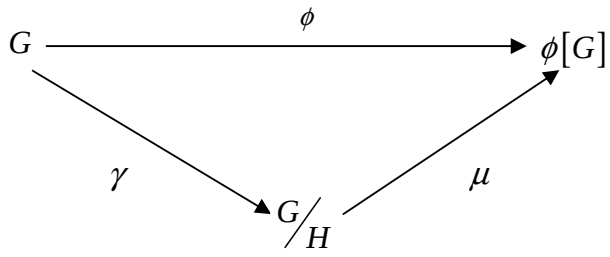
Himpunan G/H yang dilengkapi dengan operasi biner "$*$" akan membentuk suatu grup yang disebut dengan *grup faktor* dari *G modulo H*.

Selanjutnya, diberikan sebuah teorema yang menjelaskan hubungan antara suatu grup, grup faktor dan peta homomorfismanya. Teorema ini disebut dengan *toerema fundamental homomorfisma*. Untuk lebih jelasnya, diberikan pada Teorema 2.3.3.4 di bawah ini.

Teorema 2.3.3.4. (Fraleigh, 2000) Jika diberikan homomorfisma grup $\phi : G \to G'$ dengan $\ker(\phi) = H$, maka $\phi[G]$ merupakan subgrup dan dapat dibentuk suatu

isomorfisma $\mu : G/H \to \phi[G]$ dengan aturan untuk sebarang $aH \in G/H$, maka $\mu[aH] = \phi(a)$, dengan $a \in G$. Jika $\gamma : G \to G/H$ dengan aturan $\gamma(a) = aH$ adalah homomorfisma, maka $\phi(a) = \mu(\gamma(a))$, $a \in G$.

Di bawah ini diberikan ilustrasi yang menunjukkan hubungan antara *G*, G/H dan $\phi[G]$ seperti dijelaskan pada teorema fundamental homomorfisma.



Gambar 2.3. Hubungan antara *G*, G/H dan $\phi[G]$

Karena terdapat isomorfisma antara grup faktor G/H dan $\phi[G]$, maka $G/H \cong \phi[G]$.

2.3.4. Gelanggang dan Lapangan

Berikut ini diperkenalkan suatu struktur aljabar yang lain, yaitu gelanggang dan lapangan. Serta diberikan beberapa definisi yang berhubungan dengan gelanggang dan lapangan.

Definisi 2.3.4.1. (Fraleigh, 2000) Suatu *gelanggang* (*ring*) $(R,+,\cdot)$ adalah himpunan R tak kosong yang dilengkapi dengan dua operasi biner yaitu operasi penjumlahan "+" dan operasi pergandaan "$\cdot$" yang memenuhi

1) $(R,+)$ merupakan grup Abelian,
2) Operasi pergandaan bersifat assosiatif,
3) Untuk setiap $a,b,c \in R$ berlaku sifat distributif kiri, yaitu $a.(b+c) = a.b + a.c$ dan sifat distributif kanan yaitu $(a+b).c = a.c + b.c$.

Gelanggang $(R,+,\cdot)$ dapat dituliskan dengan R apabila operasi binernya diketahui.

Jelas bahwa pada gelanggang R memuat elemen identitas terhadap operasi penjumlahan yaitu $0 \in R$ sedemikian hingga $a+0=0+a=a,$ untuk setiap $a \in R$.

Definisi 2.3.4.2. (Fraleigh, 2000) Diberikan gelanggang R dan $S \subseteq R$, $S \neq \varnothing$. Subset S disebut *gelanggang bagian* (*subring*) R jika S merupakan gelanggang terhadap operasi biner yang sama pada R.

Diberikan pengertian tentang gelanggang komutatif, yaitu gelanggang yang operasi pergandaannya bersifat komutatif. Serta pengertian tentang uniti, unit dan gelanggang pembagi.

Definisi 2.3.4.3. (Fraleigh, 2000) Suatu gelanggang R yang operasi pergandaannya bersifat komutatif disebut *gelanggang komutatif*. Suatu gelanggang yang mempunyai

elemen identitas terhadap pergandaan disebut *gelanggang dengan uniti*, elemen identitas terhadap pergandaan yaitu $1 \in R$ disebut dengan *uniti*.

Definisi 2.3.4.4. (Fraleigh, 2000) Diberikan gelanggang *R* dengan uniti $1 \neq 0$. Suatu elemen $u \in R$ disebut *unit* jika *u* mempunyai invers terhadap operasi pergandaan. Jika untuk setiap elemen tak nol di *R* adalah unit, maka *R* disebut *gelanggang pembagi* (*division ring*).

Definisi 2.3.4.5. (Fraleigh, 2000) Diberikan gelanggang – gelanggang $(R_1,+,\cdot),(R_2,+,\cdot),...,(R_n,+,\cdot)$. Dibentuk $R=\prod_{i=1}^{n} R_i$, yaitu $R=\{(r_1,r_2,...,r_n): r_i \in R_i\}$. Didefinisikan operasi "+" dan "." pada *R* sebagai berikut, untuk setiap $(r_1,r_2,...,r_n),(s_1,s_2,...,s_n) \in R$,

$$(r_1,r_2,...,r_n)+(s_1,s_2,...,s_n)=(r_1+s_1,r_2+s_2,...,r_n+s_n), \text{ dan}$$

$$(r_1,r_2,...,r_n)\cdot(s_1,s_2,...,s_n)=(r_1\cdot s_1,r_2\cdot s_2,...,r_n\cdot s_n).$$

Dapat ditunjukkan bahwa $(R,+,\cdot)$ merupakan gelanggang.

Selanjutnya, diberikan pengertian tentang suatu gelanggang yang dibentuk dari suatu himpunan yang elemen-elemennya merupakan polinomial, gelanggang ini disebut dengan gelanggang polinomial. Lebih jelasnya diberikan pada definisi berikut ini.

Definisi 2.3.4.6. (Fraleigh, 2000) Diberikan gelanggang *R* dan suatu simbol *x* yang disebut *indeterminit*. Suatu *polinomial* $f(x)$ dengan koefisien dalam *R* adalah

$$f(x) = \sum_{i=0}^{\infty} a_i.x^i = a_0 + a_1.x + ... + a_n.x^n + ...$$

dengan $a_i \in R$, dan $a_i \neq 0$ untuk nilai-nilai *i* yang banyaknya berhingga, sedangkan yang lainnya semuanya nol. Elemen $a_i \in R$ disebut *koefisien-koefisien* dari $f(x)$.

Teorema 2.3.4.7. (Fraleigh, 2000) Diberikan $R[x]$ yaitu himpunan semua polinomial dalam *x* dengan koefisien dalam gelanggang *R*. Himpunan $R[x]$ merupakan gelanggang terhadap operasi biner penjumlahan polinomial dan pergandaan polinomial. Untuk setiap $f(x), g(x) \in R[x]$, yaitu $f(x) = a_0 + a_1.x + ... + a_n.x^n + ...$ dan $g(x) = b_0 + b_1.x + ... + b_n.x^n + ...$, maka

$f(x) + g(x) = c_0 + c_1.x + ... + c_n.x^n + ...$ dengan $c_n = a_n + b_n$, dan

$f(x).g(x) = d_0 + d_1.x + ... + d_n.x^n + ...$ dengan $d_n = \sum_{i=0}^{n} a_i.b_{n-i}$.

Jika *R* adalah gelanggang komutatif, maka $R[x]$ merupakan gelanggang komutatif. Gelanggang $R[x]$ seperti ini disebut dengan *gelanggang polinomial* atas *R*.

Selanjutnya, diberikan konsep pembagi nol, daerah integral dan homomorfisma gelanggang dan lapangan. Sama halnya seperti pada homomorfisma grup, pada homomorfisma gelanggang ini juga merupakan pemetaan antar gelanggang dan bersifat mengawetkan operasi.

Definisi 2.3.4.8. (Fraleigh, 2000) Diberikan gelanggang komutatif R dan $a \in R$, $a \neq 0$. Elemen a disebut *pembagi nol* jika terdapat $b \in R$, $b \neq 0$ sedemikian hingga $a.b = 0$. Suatu gelanggang R disebut *daerah integral* jika operasi pergandaannya bersifat komutatif, memuat uniti dan tidak memuat pembagi nol. Jadi, untuk suatu daerah integral R, jika $a.b = 0$, maka $a = 0$ atau $b = 0$, $a, b \in R$.

Definisi 2.3.4.9. (Fraleigh, 2000) Diberikan gelanggang $(R, +, \cdot)$ dan $(R', +', \cdot')$. Suatu pemetaan $\phi : R \to R'$ disebut *homomorfisma gelanggang* jika untuk sebarang $a, b \in R$

1. $\phi(a + b) = \phi(a) +' \phi(b)$,
2. $\phi(a.b) = \phi(a) .' \phi(b)$.

Selanjutnya, jika ϕ bersifat injektif, maka ϕ disebut *monomorfisma gelanggang*, jika ϕ bersifat surjektif, maka ϕ disebut *epimorfisma gelanggang* dan jika ϕ bersifat bijektif, maka ϕ disebut *isomorfisma gelanggang*.

Definisi 2.3.4.10. (Fraleigh, 2000) Suatu gelanggang pembagi yang bersifat komutatif disebut dengan *lapangan* (*field*). Jika suatu lapangan F memuat elemen sebanyak berhingga, maka F disebut *lapangan berhingga* (*finite field*).

Definisi 2.3.4.11. (Fraleigh, 2000) Diberikan lapangan F. Subset tak kosong $S \subseteq F$ disebut *lapangan bagian* (*subfield*) jika S merupakan lapangan terhadap operasi biner yang sama pada F.

BAB III

PERSAMAAN KONGRUEN DAN HIMPUNAN BILANGAN BULAT MODULO

Pada bab ini dibahas mengenai persamaan kongruen, himpunan bilangan bulat modulo, gelanggang bilangan bulat modulo, *residue class ring* dan suatu grup berorder prima. Juga dibahas beberapa algoritma untuk suatu grup Abelian berhingga. Pembahasan ini penting karena mendasari beberapa perhitungan pada algoritma ElGamal.

3.1. Persamaan Kongruen

Definisi 3.1.1. (Buchmann, 2000) Diberikan bilangan bulat *a* dan *b*, dan bilangan bulat positif *m*. Bilangan bulat *a* dikatakan *kongruen b modulo m* jika $m \mid (b-a)$, ditulis $a \equiv b \pmod{m}$. Selanjutnya, bilangan bulat *m* disebut *modulus*, dan persamaan $\equiv$ disebut *persamaan kongruen modulo m*.

Contoh 3.1.2.

1) $24 \equiv 9 \pmod{5}$, sebab 24 – 9 = (3).(5).

2) $-11 \equiv 17 \pmod{7}$, sebab – 11 – 17 = (– 4).(7).

Berikut ini diberikan beberapa teorema yang menjelaskan sifat-sifat persamaan kongruen.

Teorema 3.1.3. (Buchmann, 2000) Diberikan persamaan kongruen $a \equiv b \pmod{m}$ dan $c \equiv d \pmod{m}$, maka berlaku $-a \equiv -b \pmod{m}$, $a + c \equiv b + d \pmod{m}$, dan $a.c \equiv b.d \pmod{m}$.

Bukti:

Karena m membagi $(a - b)$, maka m juga membagi $(-a + b)$, akibatnya terbukti $-a \equiv -b \pmod{m}$. Karena m membagi $(a - b)$ dan $(c - d)$, maka m juga membagi $(a - b + c - d) = (a + c) - (b + d)$, dengan kata lain terbukti $a + c \equiv b + d \pmod{m}$. Untuk membuktikan $a.c \equiv b.d \pmod{m}$, misalkan $a = b + l.m$ dan $c = d + k.m$, maka $a.c = (b + l.m).(d + k.m) = b.d + m.(l.d + k.b + l.k.m)$ atau dengan kata lain m membagi $(a.c - b.d)$, terbukti bahwa $a.c \equiv b.d \pmod{m}$. □

Teorema 3.1.4. (Stinson, 1995) Diberikan sebarang $a, b \in \mathbb{Z}$, $m \in \mathbb{Z}$ positif, $r_1 = a \bmod m$ dan $r_2 = b \bmod m$, maka $a \equiv b \pmod{m}$ jika dan hanya jika $r_1 = r_2$.

Bukti:

$\Rightarrow$ Diketahui $a \equiv b \pmod{m}$ dengan $m \in \mathbb{Z}$ positif, akan ditunjukkan bahwa $r_1 = r_2$. Karena $a \equiv b \pmod{m}$ maka menggunakan algoritma pembagian pada bilangan bulat (Teorema 2.2.2.3) terdapat $q_1, q_2 \in \mathbb{Z}$ dan $r_1, r_2 \in \mathbb{Z}$ yang tunggal sedemikian hingga

$$a = q_1.m + r_1,\ 0 \le r_1 < m$$

dan

$$b = q_2.m + r_2,\ 0 \le r_2 < m.$$

Dari Definisi 3.1.1, karena $a \equiv b \pmod{m}$, maka *m* membagi (*a* – *b*). Berarti terdapat bilangan bulat *k* sedemikian hingga *a* – *b* = *k*.*m* + 0. Sehingga diperoleh bahwa

$$a - b = k.m + 0$$

$$\Leftrightarrow (q_1.m + r_1) - (q_2.m + r_2) = k.m + 0$$

$$\Leftrightarrow (q_1 - q_2).m + (r_1 - r_2) = k.m + 0,$$

dapat dilihat bahwa $r_1 - r_2 = 0$ atau $r_1 = r_2$.

$\Leftarrow$ Diketahui $r_1 = r_2$, akan ditunjukkan bahwa $a \equiv b \pmod{m}$, maka

$$r_1 = r_2$$

$$\Leftrightarrow r_1 - r_2 = 0$$

$$\Leftrightarrow a - b \equiv 0 \pmod{m}$$

$$\Leftrightarrow a = b + k.m, k \in \mathbb{Z}$$

$$\Leftrightarrow a \equiv b \pmod{m}.$$

Dengan demikian, teorema ini terbukti. □

Dapat ditunjukkan bahwa persamaan kongruen adalah suatu relasi ekuivalensi pada bilangan bulat, akibatnya dapat dibentuk klas ekuivalensi yang memuat $a \in \mathbb{Z}$. Lebih jelasnya diberikan pada definisi berikut ini.

Definisi 3.1.5. (Buchmann, 2000) Diberikan relasi ekuivalensi persamaan kongruen modulo *m* pada himpunan bilangan bulat $\mathbb{Z}$. Klas ekuivalensi yang memuat $a \in \mathbb{Z}$ adalah

$$\begin{aligned}\bar{a} &= \{x \in \mathbb{Z} : x \equiv a \pmod m\} \\ &= \{a + k.m : k \in \mathbb{Z}\} \\ &= a + m.\mathbb{Z}.\end{aligned}$$

Klas ekuivalensi seperti ini disebut dengan *residue class a mod m.*

Contoh 3.1.6.

Residue class 2 *mod* 4 adalah

$\bar{2} = \{2 + 4.k : k \in \mathbb{Z}\} = \{2, 2 \pm 4, 2 \pm (2).4, 2 \pm (3).4, ...\} = \{2, -2, 6, -6, 10, -10, 12, ...\}$.

3.2. Gelanggang Bilangan Bulat Modulo

Diberikan bilangan bulat $m > 1$, didefinisikan himpunan $\mathbb{Z}_m = \{x \bmod m : x \in \mathbb{Z}\}$, maka $\mathbb{Z}_m$ merupakan himpunan sisa pembagian semua bilangan bulat dengan *m*. Selanjutnya, elemen-elemen himpunan $\mathbb{Z}_m$ dapat dipandang sebagai klas-klas saling asing yang menyatakan himpunan bilangan bulat yang mempunyai sisa yang sama apabila dibagi dengan *m*, yaitu $\mathbb{Z}_m = \{\bar{0}, \bar{1}, \bar{2}, ..., \overline{m-1}\}$. Jika $\bar{a} \in \mathbb{Z}_m$, maka $\bar{a} = \{x \in \mathbb{Z} : x \bmod m = a\}$. Untuk mempersingkat penulisan, himpunan ini ditulis $\mathbb{Z}_m = \{0, 1, 2, ..., m-1\}$. Himpunan $\mathbb{Z}_m$ seperti ini disebut dengan *himpunan bilangan bulat modulo m.*

Pada himpunan $\mathbb{Z}_m$ berlaku operasi penjumlahan modulo dan pergandaan modulo *m*, yaitu untuk setiap $a, b \in \mathbb{Z}_m$ maka $a + b = (a + b) \bmod m$ dan

$a.b = (a.b) \bmod m$. Jelas bahwa kedua operasi tersebut merupakan operasi biner pada $\mathbb{Z}_m$. Selanjutnya, himpunan $\mathbb{Z}_m$ yang dilengkapi dengan operasi penjumlahan modulo dapat membentuk grup $(\mathbb{Z}_m, +)$ dengan $0 \in \mathbb{Z}_m$ adalah elemen identitas $\mathbb{Z}_m$.

Diberikan grup $(\mathbb{Z}, +)$ dengan "+" adalah operasi penjumlahan biasa pada himpunan bulat. Didefinisikan pemetaan $\phi : \mathbb{Z} \to \mathbb{Z}_m$ dengan aturan bahwa untuk setiap $x \in \mathbb{Z}$,

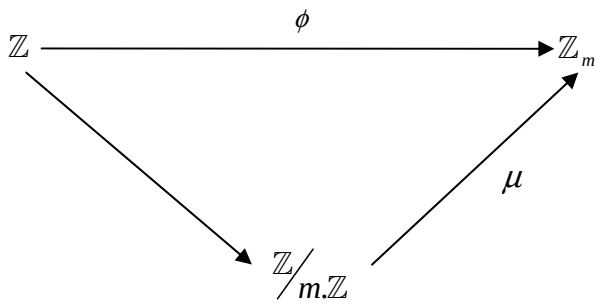
$$\phi(x) = x \bmod m .$$

Dapat ditunjukkan bahwa ϕ merupakan homomorfisma grup dengan $\phi[\mathbb{Z}] = \mathbb{Z}_m$ dan $\ker(\phi) = \{x \in \mathbb{Z} : \phi(x) = 0\} = \{m.z : z \in \mathbb{Z}\} = m.\mathbb{Z}$.

Selanjutnya, menggunakan Teorema 2.3.3.4 (teorema fundamental homomorfisma) dapat dibentuk grup faktor $\left(\mathbb{Z}/_{m.\mathbb{Z}}, +\right)$ dengan $\mathbb{Z}/_{m.\mathbb{Z}} = \{a + m.\mathbb{Z} : a \in \mathbb{Z}\}$. Dengan memperhatikan Definisi 3.1.5, dapat dilihat bahwa $\mathbb{Z}/_{m.\mathbb{Z}}$ merupakan himpunan semua *residue class mod m*. Definisi operasi penjumlahan pada $\mathbb{Z}/_{m.\mathbb{Z}}$ adalah sebagai berikut, untuk sebarang $\bar{a}, \bar{b} \in \mathbb{Z}/_{m.\mathbb{Z}}$ maka $\bar{a} + \bar{b} = \overline{a + b}$. Lebih lanjut, dapat dibentuk suatu isomorfisma $\mu : \mathbb{Z}/_{m.\mathbb{Z}} \to \mathbb{Z}_m$ dengan aturan untuk sebarang $\bar{a} \in \mathbb{Z}/_{m.\mathbb{Z}}$,

$$\mu(\bar{a}) = \mu(a + m.\mathbb{Z}) = \phi(a) .$$

Dengan demikian dapat dikatakan bahwa grup $\left(\mathbb{Z}/_{m.\mathbb{Z}}, +\right)$ isomorfis dengan $(\mathbb{Z}_m, +)$.



Gambar 3.1. Hubungan antara $\mathbb{Z}$, $\mathbb{Z}/m.\mathbb{Z}$ dan $\mathbb{Z}_m$

Selanjutnya, pada himpunan $\mathbb{Z}_m$ atau $\mathbb{Z}/m.\mathbb{Z}$ dapat dibentuk gelanggang dengan operasi biner pergandaan sebagai berikut. Untuk sebarang $\bar{a},\bar{b} \in \mathbb{Z}/m.\mathbb{Z}$, $\bar{a}.\bar{b} = \overline{a.b}$. Diketahui bahwa $(\mathbb{Z},+,\cdot)$ adalah gelanggang komutatif dengan uniti 1.

Teorema 3.2.1. (Buchmann, 2000) Jika *m* adalah bilangan bulat dengan $m>1$, maka $(\mathbb{Z}_m,+,\cdot)$ adalah gelanggang komutatif dengan uniti $1 \in \mathbb{Z}_m$. Selanjutnya, gelanggang $\mathbb{Z}_m$ seperti ini disebut dengan *gelanggang bilangan bulat modulo m*.

Teorema 3.2.2. (Buchmann, 2000) Jika *m* adalah bilangan bulat dengan $m>1$, maka $(\mathbb{Z}/m.\mathbb{Z},+,\cdot)$ adalah gelanggang komutatif dengan uniti $\bar{1} = 1 + m.\mathbb{Z}$. Selanjutnya, gelanggang seperti ini disebut dengan *residue class ring modulo m*.

Bukti:

Diketahui bahwa $(\mathbb{Z}/m.\mathbb{Z},+)$ adalah grup. Diambil sebarang $\bar{a},\bar{b},\bar{c} \in \mathbb{Z}/m.\mathbb{Z}$. Karena $\bar{a}+\bar{b} = \overline{a+b} = \overline{b+a} = \bar{b}+\bar{a}$, maka $(\mathbb{Z}/m.\mathbb{Z},+)$ Abelian. Selanjutnya, karena

$\bar{a}.\left(\bar{b}.\bar{c}\right)=\bar{a}.\left(\overline{b.c}\right)=\overline{a.(b.c)}=\overline{(a.b).c}=\left(\overline{a.b}\right).\bar{c}=\left(\bar{a}.\bar{b}\right).\bar{c}$, diperoleh bahwa operasi pergandaan bersifat assosiatif. Dapat ditunjukkan bahwa memenuhi distributif kiri dan kanan, yaitu $\bar{a}.(\bar{b}+\bar{c})=\bar{a}.\left(\overline{b+c}\right)=\overline{a.(b+c)}=\overline{a.b+a.c}=\overline{a.b}+\overline{a.c}=\bar{a}.\bar{b}+\bar{a}.\bar{b}$ dan $(\bar{a}+\bar{b}).\bar{c}=\left(\overline{a+b}\right).\bar{c}=\overline{(a+b).c}=\overline{a.c+b.c}=\overline{a.c}+\overline{b.c}=\bar{a}.\bar{c}+\bar{b}.\bar{c}$. Dapat ditunjukkan operasi pergandaan bersifat komutatif, yaitu $\bar{a}.\bar{b}=\overline{a.b}=\overline{b.a}=\bar{b}.\bar{a}$. Selanjutnya, terdapat $\bar{1}\in\mathbb{Z}/_{m.\mathbb{Z}}$ sedemikian hingga $\bar{a}.\bar{1}=\overline{a.1}=\bar{a}$ dan $\bar{1}.\bar{a}=\overline{1.a}=\bar{a}$. Dengan demikian terbukti bahwa $\left(\mathbb{Z}/_{m.\mathbb{Z}},+,\cdot\right)$ adalah gelanggang komutatif dengan uniti. □

3.3. Pembagian pada Gelanggang Bilangan Bulat Modulo

Pada pembahasan mengenai sifat *divisibility* pada bilangan bulat (Definisi 2.2.1.1), sifat ini dapat diterapkan pada elemen-elemen gelanggang. Sifat tersebut dijelaskan pada Definisi 3.3.1 di bawah ini.

Definisi 3.3.1. (Buchmann, 2000) Diberikan gelanggang $(R,+,\cdot)$ dan $a,n\in R$. Elemen *a* dikatakan membagi *n* jika terdapat $b\in R$ sehingga $n=a.b$. Elemen *a* dengan sifat seperti ini disebut *pembagi* (*divisior*) *n* dan *n* disebut *kelipatan* (*multiple*) *a*. Elemen *a* yang membagi *n* dituliskan dengan $a|n$.

Selanjutnya, akan diselidiki elemen-elemen dari $\mathbb{Z}_m$ mana saja yang merupakan unit, yaitu mempunyai invers terhadap operasi pergandaan. Perlu

diperhatikan bahwa $a \in \mathbb{Z}_m$ merupakan unit dalam $\mathbb{Z}_m$ sama halnya dengan mengatakan bahwa persamaan kongruen

$$a.x \equiv 1 \pmod{m} \tag{3.1}$$

mempunyai penyelesaian untuk $a, x \in \mathbb{Z}_m$.

Untuk selanjutnya, pernyataan invers yang dimaksud adalah invers terhadap operasi pergandaan.

Teorema 3.3.2. (Buchmann, 2000) Elemen $a \in \mathbb{Z}_m$ merupakan unit jika dan hanya jika $\gcd(a,m) = 1$. Jika $\gcd(a,m) = 1$, maka invers *a* tunggal.

Bukti:

$\Rightarrow$ Misalkan $g = \gcd(a,m)$ dan $x \in \mathbb{Z}_m$ menjadi solusi penyelesaian dari persamaan kongruen (3.1). Karena $g = \gcd(a,m)$, maka $g \mid m$ dan $g \mid a$. Menurut Definisi 3.1.1, maka $g \mid (a.x - 1)$. Oleh karena itu, $g \mid 1$. Karena pembagi dari 1 adalah 1 sendiri, maka diperoleh $\gcd(a,m) = 1$.

$\Leftarrow$ Diberikan $\gcd(a,m) = 1$, menggunakan Akibat 2.2.4.8 maka terdapat bilangan bulat *x* dan *y* sedemikian hingga *a.x* + *m.y* = 1. Menggunakan Akibat 2.2.4.7 diperoleh bahwa *x* adalah solusi penyelesaian dari persamaan kongruen (3.1) sehingga *x* adalah invers dari $a \in \mathbb{Z}_m$.

Untuk membuktikan sifat ketunggalan *x* sebagai invers dari *a* digunakan langkah sebagai berikut. Diambil $v \in \mathbb{Z}_m$ sebagai invers yang lain dari *a*, maka $a.x \equiv a.v \pmod{m}$. Akibatnya $m \mid a.(x - v)$. Karena $\gcd(a,m) = 1$, maka *m* adalah

pembagi dari $x - v$. Hal ini membuktikan bahwa $x \equiv v \pmod{m}$. Dengan kata lain terbukti bahwa invers dari *a* adalah tunggal. □

Selanjutnya, dapat dilihat bahwa suatu $a \in \mathbb{Z}_m$, $a \neq 0$ dapat merupakan pembagi nol atau juga merupakan unit.

Contoh 3.3.3.

Diberikan $m = 8$, $a \in \mathbb{Z}_8$ merupakan unit dalam $\mathbb{Z}_8$ jika $\gcd(a,8) = 1$. Dari sini diperoleh bahwa elemen-elemen $\mathbb{Z}_8$ yang mempunyai invers adalah 1, 3, 5 dan 7, masing-masing inversnya adalah 1, 3, 5 dan 7.

Dari Teorema 3.3.2 dapat dilihat bahwa persamaan kongruen (3.1) dapat diselesaikan menggunakan algoritma Euclide yang diperluas, sebab akan diperoleh dua hal sekaligus yaitu gcd(*a*,*m*) serta bilangan bulat *x* dan *y* sedemikian hingga $\gcd(a, m) = a.x + m.y$.

Menurut Teorema 3.3.2, persamaan kongruen (3.1) akan mempunyai penyelesaian jika gcd(*a*,*m*) = 1. Apabila diperoleh gcd(*a*,*m*) = 1, maka invers $a \in \mathbb{Z}_m$ juga dapat diperoleh, yaitu $a^{-1} \bmod m = x$.

Berikut ini diberikan algoritma yang dapat digunakan untuk menghitung invers pergandaan menggunakan algoritma Euclide yang diperluas pada himpunan bilangan bulat modulo *m*, dengan *m* bilangan bulat positif.

Algoritma 3.1 : Mencari Invers Pergandaan Modulo (Menezes, Oorschot and Vanstone, 1996)

Input : $a \in \mathbb{Z}_m$, m bilangan bulat positif.

Output : $a^{-1} \bmod m$.

Langkah :

1. Tentukan $x, y \in \mathbb{Z}$ yang memenuhi $a.x + m.y = d$ dengan $d = \gcd(a, m)$ menggunakan algoritma Euclide yang diperluas (Algoritma 2.3).
2. Jika $d > 1$, maka $a^{-1} \bmod m$ tidak ada. Jika tidak, *output* (x).

Contoh 3.3.4.

Akan dihitung $35^{-1} \bmod 100$. Dari Contoh 2.2.6.3 diperoleh gcd(100,35) = 5, x = -1 dan y = 3. Karena d = gcd(100,35) = 5 > 1, maka $35^{-1} \bmod 100$ tidak ada.

Teorema 3.3.5. (Buchmann, 2000) Gelanggang $(\mathbb{Z}_m, +, \cdot)$ merupakan lapangan berhingga jika dan hanya jika m adalah bilangan prima.

Bukti:

$\Rightarrow$ Andaikan m bukan bilangan prima, maka $m = a.b$ dengan $1 < a, b < m - 1$. Karena $\mathbb{Z}_m$ merupakan lapangan, maka setiap elemen tak nolnya pasti mempunyai invers. Misalkan c adalah invers dari b, berarti $b.c \equiv 1 (\bmod m)$ dan $a.b.c \equiv a (\bmod m)$. Karena $m = a.b$, maka $a.b \equiv 0 (\bmod m)$, akibatnya $0 \equiv a (\bmod m)$. Timbul kontradiksi dengan pengandaian di atas, yang benar m merupakan bilangan prima.

$\Leftarrow$ Diketahui *m* adalah bilangan prima. Karena $\mathbb{Z}_m$ merupakan gelanggang, maka akan dibuktikan bahwa setiap elemen tak nol mempunyai invers. Karena *m* adalah bilangan prima, maka $\gcd(m,a)=1$, untuk $0<a<m$. Akibatnya terdapat bilangan bulat *x* dan *y* sedemikian hingga $x.m+y.a=1$ yang berarti $y.a \equiv 1 \pmod{m}$, diperoleh $y \equiv a^{-1} \pmod{m}$. Jadi terbukti bahwa setiap elemen tak nolnya mempunyai invers. Dengan kata lain, $\mathbb{Z}_m$ adalah lapangan berhingga. □

Oleh karena itu *residue class ring modulo m* $\left(\mathbb{Z}/_{m.\mathbb{Z}},+,\cdot\right)$ juga merupakan lapangan berhingga jika dan hanya jika *m* adalah bilangan prima. Untuk selanjutnya, lapangan seperti ini disebut dengan *residue class field modulo m*.

3.4. Grup Pergandaan Bilangan Bulat Modulo

Teorema 3.4.1. (Buchmann, 2000) Himpunan semua unit dalam $\mathbb{Z}_m$ yang dilengkapi dengan operasi pergandaan membentuk suatu grup Abelian berhingga.

Bukti:

Karena himpunan ini merupakan himpunan semua unit dalam $\mathbb{Z}_m$, maka dengan sendirinya teorema ini terbukti. □

Untuk selanjutnya, grup seperti dijelaskan pada Teorema 3.4.1 disebut dengan *grup pergandaan bilangan bulat modulo m* dan dinotasikan dengan $\mathbb{Z}_m{}^*$. Dengan demikian dapat ditulis bahwa $\mathbb{Z}_m{}^* = \{a \in \mathbb{Z}_m : \gcd(a,m)=1\}$.

Contoh 3.4.2.

Diberikan gelanggang $\mathbb{Z}_5$. Karena 5 adalah bilangan prima, maka $\mathbb{Z}_5$ adalah lapangan. Selanjutnya, menggunakan sifat lapangan diperoleh bahwa setiap elemen dalam $\mathbb{Z}_5$ kecuali nol pasti mempunyai invers. Dengan kata lain, setiap elemen $\mathbb{Z}_5$ kecuali nol merupakan unit. Jadi, $\mathbb{Z}_5^* = \{1,2,3,4\}$.

Selanjutnya, dapat dibentuk grup $\mathbb{Z}/_{m.\mathbb{Z}}^*$ yaitu grup pergandaan yang elemennya merupakan semua unit dalam gelanggang $\left(\mathbb{Z}/_{m.\mathbb{Z}}, +, \cdot\right)$. Grup $\mathbb{Z}/_{m.\mathbb{Z}}^*$ seperti ini disebut dengan *multipilcative group of residues modulo m*.

Berikut ini dijelaskan suatu fungsi yang menyatakan order dari grup pergandaan bilangan bulat modulo *m*, $\mathbb{Z}_m^*$. Didefinisikan fungsi $\varphi : \mathbb{N} \to \mathbb{N}$ dengan

$$\varphi(m) = \begin{cases} 1 & , m = 1 \\ \text{order}\, \mathbb{Z}_m^* & , m > 1 \end{cases}$$

Fungsi φ seperti ini disebut dengan *Euler φ-function*.

Dengan demikian, $\varphi(m)$ adalah banyaknya $a \in \{1,2,...,m\}$ dengan $\gcd(a,m) = 1$.

Contoh 3.4.3. (Buchmann, 2000)

Tabel 3.1. Beberapa nilai Euler *φ-function*

m	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\phi(m)$	1	1	2	2	4	2	6	4	6	4	10	4	12	6	8

Dari Tabel 3.1, diperoleh bahwa $\varphi(1)=1$, $\varphi(2)=1$, $\varphi(3)=2$, dan seterusnya. Artinya, order dari $\mathbb{Z}_2{}^*$ adalah 1, order dari $\mathbb{Z}_3{}^*$ adalah 2, dan seterusnya.

Teorema 3.4.4. (Buchmann, 2000) Jika p adalah bilangan prima, maka $\varphi(p)=p-1$.

Bukti:

Diberikan sebarang bilangan prima p. Menggunakan Teorema 3.3.6 maka $\left(\mathbb{Z}_p,+,\cdot\right)$ adalah lapangan dengan order p. Selanjutnya, dapat dibentuk grup $\mathbb{Z}_p{}^*$ yang ordernya dinotasikan dengan $\varphi(p)$. Karena $\mathbb{Z}_p$ adalah lapangan, maka setiap elemen tak nolnya pasti mempunyai invers, dengan kata lain ada sebanyak $p-1$ elemen yang mempunyai invers, yang semuanya menjadi anggota $\mathbb{Z}_p{}^*$. Dengan demikian diperoleh bahwa order $\mathbb{Z}_p{}^*$ adalah $p-1$, terbukti bahwa $\varphi(p)=p-1$. □

Pada Teorema 3.4.4 diketahui bahwa jika p adalah bilangan prima, maka $\varphi(p)=p-1$. Selanjutnya, di bawah ini diberikan sebuah teorema tentang Euler φ-*function.*

Teorema 3.4.5. (Buchmann, 2000) Diberikan bilangan bulat positif m dan $d_1,d_2,...,d_n$ adalah semua pembagi positif m yang berbeda, maka

$$\sum_{i=1}^{n}\varphi(d_i)=m.$$

Bukti:

Karena himpunan semua pembagi positif dari m adalah $\left\{\frac{m}{d_i} : i = 1, 2, ..., n\right\}$, maka diperoleh bahwa

$$\sum_{i=1}^{n} \varphi(d_i) = \sum_{i=1}^{n} \varphi\left(\frac{m}{d_i}\right).$$

Selanjutnya, untuk setiap i, $1 \le i \le n$ maka $\varphi\left(\frac{m}{d_i}\right)$ adalah banyaknya bilangan bulat a di dalam himpunan $\left\{1, 2, ..., \frac{m}{d_i}\right\}$ dengan $\gcd\left(a, \frac{m}{d_i}\right) = 1$. Oleh karena itu $\varphi\left(\frac{m}{d_i}\right)$ merupakan banyaknya bilangan bulat b di dalam himpunan $\{1, 2, ..., m\}$ dengan $\gcd(b, m) = d_i$. Oleh karena itu

$$\sum_{i=1}^{n} \varphi(d_i) = \sum_{i=1}^{n} \left|\{b : 1 \le b \le m \text{ dan } \gcd(b, m) = d_i\}\right|.$$

Karena

$$\{1, 2, ..., m\} = \bigcup_{i=1}^{n} \{b : 1 \le b \le m \text{ dan } \gcd(b, m) = d_i\}$$

maka berakibat bahwa

$$\begin{aligned} \sum_{i=1}^{n} \varphi(d_i) &= \sum_{i=1}^{n} \left|\{b : 1 \le b \le m \text{ dan } \gcd(b, m) = d_i\}\right| \\ &= m. \end{aligned}$$

Dengan demikian teorema terbukti. □

Contoh 3.4.6.

Diberikan m = 10. Bilangan bulat positif pembagi 10 adalah 1, 2, 5 dan 10. Menggunakan Teorema 3.4.5 diperoleh

$$\sum_{i=1}^{4} \varphi(d_i) = \varphi(1) + \varphi(2) + \varphi(5) + \varphi(10)$$
$$= 1 + 1 + 4 + 4$$
$$= 10.$$

3.5. Order Elemen-Elemen Grup

Selanjutnya diperkenalkan konsep order elemen-elemen grup. Diberikan grup *G* terhadap operasi pergandaan "." dengan elemen identitas 1.

Definisi 3.5.1. (Buchmann, 2000) Diberikan sebarang $g \in G$. Jika terdapat bilangan bulat positif *k* sedemikian hingga berlaku $g^k = \underbrace{g.g...g}_{k \text{ faktor}} = 1$, maka bilangan bulat positif terkecil seperti ini disebut *order g*. Jika tidak, maka order *g* adalah tak hingga (*infinite*).

Oleh karena itu, untuk sebarang $g \in G$, order *g* adalah order subgrup yang dibangun oleh *g*.

Teorema 3.5.2. (Buchmann, 2000) Diberikan sebarang $g \in G$ dan $u \in \mathbb{Z}$ positif, maka $g^u = 1$ jika dan hanya jika *u* dapat dibagi dengan order *g*.

Bukti:

Misalkan order dari *g* adalah *k*.

$\Leftarrow$ Jika $u = h.k$, maka

$$g^u = g^{h.k} = \left(g^k\right)^h = 1^h = 1.$$

$\Rightarrow$ Diketahui $g^u = 1$ dengan $u = q.k + r$, $0 \le r < k$. Diperoleh

$$g^r = g^{u-q.k} = g^u \left(g^k\right)^{-q} = 1.$$

Karena *k* adalah bilangan bulat positif terkecil sehingga $g^k = 1$ dan karena $0 \le r < k$, diperoleh $r = 0$, sehingga $u = q.k$, untuk suatu $q \in \mathbb{Z}$. □

Akibat 3.5.3. (Buchmann, 2000) Diberikan $g \in G$ dan $k, l \in \mathbb{Z}$, maka $g^l = g^k$ jika dan hanya jika $l \equiv k (\mathrm{mod}\, \mathrm{order}\, g)$.

Bukti:

Ditentukan $u = l - k$. Menggunakan Teorema 3.5.2 maka $g^u = g^{l-k} = 1$ jika dan hanya jika $\mathrm{order}\, g \mid (l-k)$. Terbukti bahwa $l \equiv k (\mathrm{mod}\, \mathrm{order}\, g)$. □

Teorema 3.5.4. (Buchmann, 2000) Jika $g \in G$ mempunyai order *h* dan jika $n \in \mathbb{Z}$, maka $\mathrm{order}\, g^n = \dfrac{h}{\gcd(h,n)}$.

Bukti:

Misalkan $g = \gcd(h,n)$ dan $m = \mathrm{order}\, g^n$. Akan dibuktikan bahwa $m = \dfrac{h}{g}$. Oleh karena

$$\left(g^n\right)^{\frac{h}{g}} = \left(g^h\right)^{\frac{n}{g}} = 1^{\frac{n}{g}} = 1,$$

menggunakan Teorema 3.5.2 diperoleh bahwa $m \left| \left(\dfrac{h}{g} \right) \right.$. Misalkan untuk suatu $k \in \mathbb{Z}$ berlaku

$$1 = \left(g^n\right)^k = g^{n.k}.$$

Berdasarkan Teorema 3.5.2 berakibat bahwa $h \mid n.k$, sehingga diperoleh bahwa $\left(\frac{h}{g}\right) \Big| k$. Karena $m = \text{order}\, g^n$, maka $\left(\frac{h}{g}\right) \Big| m$, di lain pihak diketahui $m \Big| \left(\frac{h}{g}\right)$. Akibatnya diperoleh $m = \frac{h}{g}$, dengan kata lain $\text{order}\, g^n = \frac{h}{\gcd(h,n)}$. □

Teorema 3.5.5. (Buchmann, 2000) Jika *G* adalah grup siklik dan berhingga, maka *G* mempunyai pembangun sebanyak $\varphi(|G|)$ dan order setiap pembangunnya sama dengan order $|G|$.

Bukti:

Diberikan $g \in G$ yang mempunyai order *s*, maka *g* membangun suatu subgrup $\langle g \rangle$ dengan $|\langle g \rangle| = s$. Selanjutnya, suatu elemen *G* yang membangun *G* mempunyai order |*G*|. Akan diselidiki elemen-elemen *G* yang mempunyai order |*G*|. Misalkan *g* adalah pembangun *G*, maka $G = \{g^k : 0 \le k \le |G|\}$. Menggunakan Teorema 3.5.4, elemen dari himpunan ini mempunyai order |*G*| jika dan hanya jika $\gcd(k, |G|) = 1$. Hal ini menunjukkan bahwa banyaknya pembangun dari *G* ada sebanyak $\varphi(|G|)$. □

3.6. Teorema Fermat

Berikut ini diberikan sebuah teorema yang cukup terkenal dalam kriptografi, teorema ini disebut dengan teorema Fermat, seperti diberikan pada Teorema 3.6.1 di bawah ini.

Teorema 3.6.1. (Buchmann, 2000) Untuk sebarang $a \in \mathbb{Z}_m^*$ berlaku $a^{\varphi(m)} \equiv 1 \pmod{m}$.

Bukti:

Dibentuk grup $\mathbb{Z}_m^*$ dengan *m* adalah bilangan bulat positif. Diambil sebarang $a \in \mathbb{Z}_m^*$. Karena $\varphi(m)$ adalah order grup $\mathbb{Z}_m^*$, maka menggunakan Definisi 3.5.1 diperoleh bahwa $a^{\varphi(m)} \equiv 1 \pmod{m}$. □

Teorema 3.6.2. (Buchmann, 2000) Jika diberikan grup *G* dan sebarang $g \in G$, maka order g membagi order *G*.

Bukti:

Karena order *g* merupakan order dari subgrup yang dibangun oleh *g*, maka menggunakan Teorema 2.3.2.7 diperoleh bahwa order subgrup yang dibangun oleh *g* membagi order grup *G*. Dengan demikian teorema terbukti. □

Berikut ini diberikan sebuah teorema yang merupakan generalisasi dari teorema Fermat. Teorema ini didasarkan pada Teorema 3.6.2.

Teorema 3.6.3. (Buchmann, 2000) Untuk setiap $g \in G$, $g^{|G|} = 1$.

Bukti:

Berdasarkan Teorema 3.6.2 maka untuk setiap $g \in G$ diperoleh bahwa order $g \,|\, |G|$.

Menggunakan Teorema 3.5.2 diperoleh bahwa $g^{|G|} = 1$. □

Masalah yang kemudian muncul adalah bagaimana jika *a* dan bilangan $\varphi(m)$ pada teorema Fermat merupakan bilangan bulat yang besar, maka perhitungan akan menjadi sulit dan rumit.

Berikut ini dijelaskan suatu metode yang dapat digunakan untuk menghitung secara cepat pangkat bilangan bulat khususnya untuk bilangan bulat yang besar.

3.7. Metode Fast Exponentiation

Diberikan sebarang grup *G*, $g \in G$ dan bilangan bulat positif *z*. Untuk menghitung g^z dilakukan langkah berikut ini.

Dibentuk ekspansi biner dari bilangan bulat *z*, yaitu

$$z = \sum_{i=0}^{k} a_i.2^i .$$

Karena *z* ditulis dalam ekspansi biner, maka $a_i \in \{0,1\}$. Sehingga

$$g^z = g^{\sum_{i=0}^{k} a_i.2^i} = \prod_{i=0}^{k} \left(g^{2^i}\right)^{a_i} = \prod_{1 \le i \le k, a_i = 1} g^{2^i} .$$

Dengan hasil ini diperoleh cara yang cepat untuk menghitung g^z yang disebut dengan metode *fast exponentiation*, yaitu

1) Hitung nilai g^{2^i}, $0 \le i \le k$.
2) Nilai g^z merupakan hasil dari perkalian nilai-nilai g^{2^i}, dengan $a_i = 1$.

   Diperoleh bahwa

$$g^{2^{i+1}} = \left(g^{2^i}\right)^2 .$$

Contoh 3.7.1. (Buchmann, 2000)

Akan dihitung nilai dari $6^{73} \bmod 100$. Pertama, ditentukan ekspansi biner dari 73

$$73 = 1.2^6 + 1.2^3 + 1.2^0 \text{ atau } (1001001)_2 .$$

Selanjutnya dihitung $6^{2^0} = 6$, $6^{2^1} = 36$, $6^{2^2} = 36^2 \equiv 96 (\bmod 100)$, $6^{2^3} \equiv 16 (\bmod 100)$, $6^{2^4} \equiv 16^2 \equiv 56 (\bmod 100)$, $6^{2^5} \equiv 56^2 \equiv 36 (\bmod 100)$, $6^{2^6} \equiv 56^2 \equiv 96 (\bmod 100)$. Sehingga diperoleh

$$\begin{aligned} 6^{73} &\equiv 6.6^{2^3}.6^{2^6} \pmod{100} \\ &\equiv 6.16.96 (\bmod 100) \\ &\equiv 16 (\bmod 100). \end{aligned}$$

Jadi, $6^{73} \bmod 100 = 16$.

Algoritma 3.2 : Metode Fast Exponentiation (Menezes, Oorschot and Vanstone, 1996)

Input : $a \in \mathbb{Z}_m$, $m \in \mathbb{Z}$ positif dan bilangan bulat *k*, $0 \le k < m$ dengan representasi biner dari $k = \sum_{i=0}^{t} k_i.2^i$.

Output : $a^k \bmod m$.

Langkah :

1. *Set* $b \leftarrow 1$. Jika $k = 0$, maka *output*(*b*).
2. *Set* $A \leftarrow a$.
3. Jika $k_0 = 1$, maka *set* $b \leftarrow a$.
4. Untuk *i* dari 1 sampai *t* kerjakan:

4.1 *Set* $A \leftarrow A^2 \bmod n$.

4.2 Jika $k_i = 1$, maka *set* $b \leftarrow A.b \bmod n$.

5. *Output*(*b*).

Berikut ini diberikan sebuah contoh perhitungan pangkat bilangan bulat modulo yang besar menggunakan Algoritma 3.2.

Contoh 3.7.2. (Menezes, Oorschot and Vanstone, 1996)

Dihitung $5^{596} \bmod 1234$ menggunakan Algoritma 3.2.

Tabel 3.2. Perhitungan $5^{596} \bmod 1234$

i	0	1	2	3	4	5	6	7	8	9
k_i	0	0	1	0	1	0	1	0	0	1
A	5	25	625	681	1011	369	421	779	947	925
b	1	1	625	625	67	67	1059	1059	1059	1013

Dari Tabel 3.2 dapat dilihat bahwa $5^{596} \bmod 1234 = 1013$.

3.8. Penghitungan Order Elemen Grup

Dalam kriptografi, sering digunakan suatu elemen grup dengan order yang besar. Pada subbab ini dibahas bagaimana menemukan nilai dari order suatu elemen *g* di dalam grup berhingga *G* atau menunjukkan apakah jika diberikan sebarang bilangan bulat positif, maka bilangan itu merupakan order *g* atau tidak.

Teorema di bawah ini menunjukkan bagaimana menghitung order *g* jika diketahui faktorisasi prima dari order *G*, yaitu

$$|G| = \prod_{i=1}^{n} p_i^{e(p_i)},$$

dengan p_i, $1 \le i \le n$ adalah bilangan prima yang membagi $|G|$.

Jika faktorisasi prima dari $|G|$ tidak diketahui, maka tidak mudah untuk mencari order *g*. Khususnya pada algoritma ElGamal, order grup dan faktorisasi primanya telah diketahui.

Teorema 3.8.1. (Buchmann, 2000) Diketahui grup *G* dan faktorisasi prima $|G| = \prod_{i=1}^{n} p_i^{e(p_i)}$. Jika $g \in G$ dan $f(p_i)$ adalah bilangan bulat terbesar sedemikian hingga $g^{\frac{|G|}{p_i^{f(p_i)}}} = 1$, $1 \le i \le n$, maka

$$\text{order } g = \prod_{i=1}^{n} p_i^{e(p_i)-f(p_i)}.$$

Bukti:

Misalkan faktorisasi prima dari $|G|$ adalah $|G| = \prod_{i=1}^{n} p_i^{e(p_i)}$. Menggunakan Teorema 3.5.2 diperoleh bahwa $g^{\frac{|G|}{p_i^{f(p_i)}}} = 1$ jika dan hanya jika order *g* membagi $\frac{|G|}{p_i^{f(p_i)}}$ atau order *g* membagi $\frac{\prod_{i=1}^{n} p_i^{e(p_i)}}{p_i^{f(p_i)}}$. Akibatnya $\text{order } g$ membagi $\prod_{i=1}^{n} p_i^{e(p_i)-f(p_i)}$. Diperoleh bahwa $\text{order } g = \prod_{i=1}^{n} p_i^{x(p_i)}$ dengan $0 \le x(p_i) \le e(p_i) - f(p_i)$, untuk setiap p_i, $1 \le i \le n$. Karena $f(p_i)$ adalah bilangan bulat terbesar sedemikian hingga $g^{\frac{|G|}{p_i^{f(p_i)}}} = 1$

maka $x(p_i) = e(p_i) - f(p_i)$, untuk setiap p_i, $1 \le i \le n$. Dengan demikian diperoleh bahwa

$$\text{order } g = \prod_{i=1}^{n} p_i^{x(p_i)} = \prod_{i=1}^{n} p_i^{e(p_i)-f(p_i)} .$$

Dengan demikian teorema ini terbukti. □

Contoh 3.8.2. (Buchmann, 2000)

Diberikan grup $G = \mathbb{Z}_{101}^*$. Grup ini mempunyai order $100 = 2^2.5^2$. Diketahui $e(2) = e(5) = 2$. Akan dihitung order dari 2. Pertama, dihitung nilai $f(p)$ menggunakan Teorema 3.8.1, diperoleh

$$2^{2.5^2} \equiv 2^{50} \equiv 100 \pmod{101}.$$

Oleh karena itu, $f(2) = 0$. Selanjutnya,

$$2^{2^2.5} \equiv 2^{20} \equiv 95 \pmod{101}.$$

Oleh karena itu $f(5) = 0$, jadi order dari $2 \in \mathbb{Z}_{101}^*$ adalah 100.

Selanjutnya, Akibat 3.8.3 di bawah ini dapat digunakan untuk menunjukkan bahwa suatu bilangan bulat merupakan order $g \in G$ atau tidak. Hal ini penting karena digunakan untuk menentukan pembangun dari suatu grup siklik.

Akibat 3.8.3. (Buchmann, 2000) Diberikan $n \in \mathbb{N}$. Jika $g^n = 1$ dan $g^{\frac{n}{p}} \neq 1$ untuk setiap *p* yaitu pembagi prima dari *n*, maka *n* adalah order dari *g*.

Bukti:

Diketahui $n \in \mathbb{N}$, $g \in G$ dan $g^n = 1$. Misalkan $n = \prod_{i=1}^{k} p_i^{e(p_i)}$ adalah faktorisasi prima dari *n*. Karena $g^{\frac{n}{p_i}} \neq 1$, untuk setiap *i*, $1 \leq i \leq k$, maka $f(p_i) = 0$. Menggunakan Teorema 3.8.1 diperoleh bahwa *n* adalah order *g*. □

Contoh 3.8.4. (Buchmann, 2000)

Akan dibuktikan apakah 25 merupakan order dari $5 \in \mathbb{Z}_{101}^*$. Diketahui $5^{25} \equiv 1 \pmod{101}$ dan $5^5 \equiv 95 \pmod{101}$. Oleh karena itu, menggunakan Akibat 3.8.3 maka 25 merupakan order dari $5 \in \mathbb{Z}_{101}^*$.

3.9. Polinomial

Pada Bab I telah dijelaskan mengenai definisi gelanggang polinomial. Diberikan gelanggang komutatif *R* dengan uniti $1 \neq 0$ dan polinomial

$$f(x) = a_n.x^n + a_{n-1}.x^{n-1} + ... + a_1.x + a_0,$$

dengan *x* adalah variabel (indeterminit) dan koefisien $a_0, a_1 ..., a_n$ adalah elemen *R*, dengan $a_i = 0$, untuk $i > n$. Himpunan semua polinomial atas *R* dengan variabel *x* dinotasikan dengan $R[x]$.

Misal diberikan $a_n \neq 0$, maka *n* disebut *derajat* dari polinomial $f(x)$, ditulis $\deg(f(x)) = n$. Selanjutnya, a_n disebut *koefisien leading* (*leading coefficient*). Jika setiap koefisien kecuali koefisien *leading* adalah nol, maka polinomial $f(x)$ disebut *monomial*.

Contoh 3.9.1. (Buchmann, 2000)

Diberikan polinomial $(2.x^3 + x + 1),\ x + 2,\ 5 \in \mathbb{Z}[x]$. Dapat dilihat bahwa polinomial $2.x^3 + x + 1$ mempunyai derajat 3, polinomial $x + 2$ mempunyai derajat 1, dan polinomial 5 mempunyai derajat 0.

Jika $r \in R$, maka $f(r) = a_n.r^n + ... + a_0$ adalah *nilai* dari f pada r. Jika $f(r) = 0$, maka r disebut *pembuat nol* (*zero of*) f.

Contoh 3.9.2. (Buchmann, 2000)

Nilai dari polinomial $2.x^3 + x + 1 \in \mathbb{Z}[x]$ pada -1 adalah $2.(-1)^3 + (-1) + 1 = -2$. Polinomial $x^2 + 1 \in \mathbb{Z}_2[x]$ mempunyai pembuat nol yaitu 1.

3.10. Polinomial atas Lapangan

Diberikan lapangan F, maka gelanggang polinomial $F[x]$ tidak memuat pembagi nol.

Lemma 3.10.1. (Buchmann, 2000) Jika $f(x), g(x) \in F[x]$, $f(x), g(x) \neq 0$, maka $\deg(f(x).g(x)) = \deg(f(x)) + \deg(g(x))$.

Seperti pada himpunan bilangan bulat, pada $F[x]$ juga dapat diterapkan tentang konsep algoritma pembagian yang disebut dengan algoritma pembagian polinomial seperti diberikan pada Teorema 3.10.2 di bawah ini.

Teorema 3.10.2. (Buchmann, 2000) Diberikan $f(x), g(x) \in F[x]$ dengan $g(x) \neq 0$, maka terdapat dengan tunggal polinomial $q(x), r(x) \in F[x]$ dengan $f(x) = q(x).g(x) + r(x)$ dan $r(x) = 0$ atau $\deg(r(x)) < \deg(g(x))$.

Pada Teorema 3.10.2 di atas, $q(x)$ disebut *hasil bagi* (*quotient*) dan $r(x)$ disebut *sisa* (*remainder*) dari pembagian $f(x)$ dengan $g(x)$, ditulis $r(x) = f(x) \bmod g(x)$.

Akibat 3.10.3. (Buchmann, 2000) Jika $f(x) \in F[x]$ adalah polinomial tidak nol dan jika *a* adalah pembuat nol $f(x)$, maka $f(x) = (x-a).q(x)$ dengan $q(x) \in F[x]$.

Bukti:

Menggunakan Teorema 3.10.2, terdapat polinomial $q(x), r(x) \in F[x]$ dengan $f(x) = (x-a).q(x) + r(x)$ dan $r(x) = 0$ atau $\deg(r(x)) < 1$. Akibatnya $0 = f(a) = r(x)$ sehingga diperoleh $f(x) = (x-a).q(x)$. □

Contoh 3.10.4. (Buchmann, 2000) Polinomial $x^2 + 1 \in \mathbb{Z}_2[x]$ mempunyai elemen pembuat nol yaitu 1, dan $x^2 + 1 = (x-1)^2$.

Akibat 3.10.5. (Buchmann, 2000) Polinomial $f(x) \in F[x], f(x) \neq 0$ mempunyai paling banyak $\deg(f(x))$ pembuat nol.

Bukti:

Akan dibuktikan menggunakan induksi pada $n = \deg(f(x))$. Untuk $n = 0$ jelas benar berlaku, sebab $f(x) \in F[x]$ dan $f(x) \neq 0$. Selanjutnya, diberikan $n > 0$. Jika $f(x)$ tidak mempunyai pembuat nol, maka jelas pernyataan benar. Jika $f(x)$ mempunyai pembuat nol, misalkan a adalah pembuat nol $f(x)$, menggunakan Akibat 3.10.3 berakibat bahwa $f(x) = (x-a).q(x)$ dan $\deg(q(x)) = n-1$. Menggunakan asumsi induksi, diperoleh bahwa $q(x)$ mempunyai paling banyak $n-1$ pembuat nol. Dengan kata lain, $f(x)$ mempunyai paling banyak n pembuat nol. □

3.11. Grup Unit atas Lapangan Berhingga

Diberikan lapangan berhingga F yang memuat sebanyak q elemen. Didefinsikan suatu grup F^*, yaitu grup yang elemennya adalah semua unit dalam lapangan F dengan operasi biner pergandaan dalam F. Grup F^* mempunyai order $q-1$, sebab setiap elemen tak nol dalam F merupakan unit. Selanjutnya, grup F^* seperti ini disebut dengan *grup unit* (*unit group*).

Teorema 3.11.1. (Buchmann, 2000) Diberikan lapangan berhingga F yang memuat q elemen. Jika d adalah bilangan bulat positif yang membagi $q-1$, maka terdapat $\varphi(d)$ elemen dengan order d dalam grup unit F^*.

Bukti:

Diberikan bilangan bulat d yang membagi $q-1$. Didefinisikan $\psi(d)$ adalah banyaknya elemen berorder d dalam F. Diasumsikan bahwa $\psi(d) > 0$, akan

dibuktikan bahwa $\psi(d) = \varphi(d)$. Diberikan *a* adalah elemen berorder *d* dalam F^*, maka $a^m, 0 \le m < d$ merupakan elemen yang berbeda dan semuanya merupakan pembuat nol dari polinomial $x^d - 1$. Menggunakan Akibat 3.10.5, terdapat paling banyak *d* pembuat nol dari polinomial $x^d - 1$ dalam *F*. Oleh karena itu, polinomial ini mempunyai tepat *d* pembuat nol dan semuanya merupakan elemen yang diperoleh dari pemangkatan *a*. Selanjutnya, setiap elemen dalam *F* dengan order *d* adalah pembuat nol dari $x^d - 1$ dan merupakan hasil pemangkatan dari *a*. Menggunakan Teorema 3.5.4, suatu pangkat a^m mempunyai order *d* jika dan hanya jika $\gcd(d, m) = 1$. Oleh karena $\psi(d) > 0$, maka berakibat $\psi(d) = \varphi(d)$. Selanjutnya, akan dibuktikan bahwa $\psi(d) > 0$. Misalkan $\psi(d) = 0$ untuk suatu pembagi *d* dari $q - 1$, maka untuk semua bilangan bulat positif $d_1, d_2, ..., d_n$ yang membagi $q - 1$

$$q - 1 = \sum_{i=1}^{n} \psi(d_i) < \sum_{i=1}^{n} \varphi(d_i).$$

Kontradiksi dengan Teorema 3.4.5. yang benar adalah $\psi(d) > 0$. □

Contoh 3.11.2. (Buchmann, 2000)

Diberikan lapangan $\mathbb{Z}_{13}$. Grup unit dari $\mathbb{Z}_{13}$ mempunyai order 12. Pada grup ini terdapat satu elemen dengan order 1, satu elemen dengan order 2, dua elemen dengan order 3, dua elemen dengan order 4, dua elemen dengan order 6, dan empat elemen dengan order 12.

Jika *F* adalah lapangan berhingga dengan *q* elemen, maka menggunakan Teorema 3.10.1, *F* memuat $\varphi(q-1)$ elemen yang mempunyai order $q - 1$.

Akibat 3.11.3. (Buchmann, 2000) Jika F adalah lapangan berhingga dengan q elemen, maka grup unit F^* siklik dan mempunyai $\varphi(q-1)$ pembangun.

Bukti:

Diketahui F^* mempunyai order $q-1$. Karena suatu elemen yang mempunyai order $q-1$ merupakan pembangun, menggunakan Teorema 3.11.1 maka F^* memuat $\varphi(q-1)$ pembangun, akibatnya F^* siklik. Dengan demikian teorema ini terbukti. □

3.12. Struktur Grup Pergandaan Bilangan Bulat Modulo Prima

Diberikan grup pergandaan bilangan bulat modulo *p*, $\mathbb{Z}_p^*$ dengan *p* adalah bilangan prima. Karena $\mathbb{Z}_p$ adalah lapangan berhingga, maka $\mathbb{Z}_p^*$ merupakan grup unit, sebab setiap elemen $\mathbb{Z}_p^*$ merupakan unit.

Akibat 3.12.1. (Buchmann, 2000) Grup $\mathbb{Z}_p^*$ siklik dengan order $p-1$.

Bukti:

Diketahui $\mathbb{Z}_p^*$ adalah grup unit. Menggunakan Akibat 3.11.3 diperoleh bahwa $\mathbb{Z}_p^*$ mempunyai elemen pembangun, serta memuat $p-1$ elemen sebab hanya elemen 0 yang bukan merupakan unit. Dengan demikian, terbukti bahwa $\mathbb{Z}_p^*$ adalah grup siklik dengan order $p-1$. □

Berikut ini diberikan konsep tentang suatu elemen yang membangun grup siklik $\mathbb{Z}_p^*$. Elemen ini nantinya berperan sangat penting dalam algoritma ElGamal, karena digunakan sebagai salah satu kuncinya.

Definisi 3.12.2. (Buchmann, 2000) Suatu elemen yang membangun $\mathbb{Z}_p^*$ disebut *elemen primitif* (*primitive root*) *mod p*.

Contoh 3.12.3. (Buchmann, 2000)

Diberikan p = 13. Karena 13 adalah bilangan prima, maka menggunakan Teorema 3.4.4 diperoleh $\varphi(13) = 12 - 1 = 12$. Kemudian dihitung order dari masing-masing elemen $\mathbb{Z}_{13}^*$. Dari sini diperoleh empat elemen yang mempunyai order 12 dan sama dengan order dari $\mathbb{Z}_{13}^*$, yaitu 12. Oleh karena elemen yang mempunyai order 12 adalah pembangun, maka elemen tersebut adalah elemen primitif. Empat elemen tersebut adalah 2, 6, 7 dan 11. Dengan demikian, elemen primitif $\mathbb{Z}_{13}^*$ adalah 2, 6, 7 dan 11.

BAB IV

TES KEPRIMAAN

Salah satu hal yang berperan sangat penting dalam algoritma kriptografi kunci publik adalah kunci, semakin besar kunci maka tingkat keamanan sistem akan semakin tinggi pula. Pada algoritma ElGamal, bilangan prima digunakan sebagai salah satu kuncinya. Untuk mendapatkan bilangan prima yang besar diperlukan suatu pembangun kunci yang juga harus besar. Pada Bab II telah diberikan sebuah algoritma tes keprimaan (Algoritma 2.4), akan tetapi algoritma ini sangat tidak efisien untuk mengecek bilangan yang besar.

Pada bab ini dijelaskan dua buah tes keprimaan yang dapat digunakan untuk bilangan bulat positif ganjil yang besar, yaitu tes *Fermat* dan tes *Miller-Rabbin*.

4.1. Tes Fermat

Tes Fermat didasarkan pada teorema Fermat (Teorema 3.6.1) yang sedikit dirubah seperti diberikan pada Teorema 4.1.1 di bawah ini.

Teorema 4.1.1. (Buchmann, 2000) Jika p adalah bilangan prima, maka $a^{p-1} \equiv 1 \pmod{p}$, untuk sebarang $a \in \mathbb{Z}_p^*$. Untuk selanjutnya, a disebut *dasar* (*base*).

Bukti:

Teorema 4.7.1 mengatakan bahwa untuk sebarang $a \in \mathbb{Z}_m^*$ dan m bilangan bulat, berlaku $a^{\phi(m)} \equiv 1 \pmod{m}$. Dari sini diambil $m = p$, dengan p bilangan prima.

Menggunakan Teorema 3.4.4 diperoleh bahwa $\phi(p) = p-1$. Sehingga diperoleh $a^{p-1} \equiv a^{\phi(m)} \equiv 1 \pmod{p}$. Dengan demikian teorema terbukti. □

Dengan hasil yang diperoleh pada Teorema 4.1.1, dapat dibentuk sebuah algoritma tes keprimaan, sebagai berikut.

Algoritma 4.1 : Tes Fermat (Menezes, Oorschot and Vanstone, 1996)

Input : Sebuah bilangan bulat positif ganjil $p \geq 3$ dan sebuah parameter $t \geq 1$.

Output : Pernyataan "prima" atau "komposit".

Langkah :

1. Untuk *i* dari 1 sampai *t* kerjakan:
   1.1. Diambil sebarang bilangan bulat positif a, $2 \leq a \leq p-1$.
   1.2. Hitung $y \equiv a^{p-1} \pmod{p}$.
   1.3. Jika $y \neq 1$, maka *output* ("komposit").
2. *Output* ("prima").

Namun pada kenyataanya, nilai $y \equiv 1 \pmod{p}$ tidak selamanya menjamin bahwa *p* adalah prima. Berikut diberikan contohnya.

Contoh 4.1.2. (Buchmann, 2000)

Diberikan *p* = 341. Diambil *a* = 2, dengan tes Fermat diperoleh $2^{340} \equiv 1 \pmod{341}$ yang menyatakan bahwa 341 adalah bilangan prima. Namun hal ini tidak benar

karena 341 adalah bilangan komposit, yaitu 341 = (11).(13). Dari sini terbukti, walaupun diperoleh $y = 1$, ternyata p bukan bilangan prima. Untuk menunjukkan bahwa p benar merupakan bilangan komposit dilakukan tes Fermat sekali lagi. Selanjutnya diambil $a = 3$, dengan tes Fermat diperoleh $3^{340} \equiv 56 \pmod{341}$ yang benar menunjukkan bahwa 341 adalah bilangan komposit.

Contoh di atas memperlihatkan bahwa pengambilan dasar $a \in \mathbb{Z}_p^*$ sangat mempengaruhi hasil akhir perhitungan. Ini menjadi salah satu kelemahan dari tes Fermat. Kelemahan lain dari tes Fermat adalah gagal saat harus mendeteksi kekompositan suatu bilangan tertentu yang dinamakan bilangan *Carmichael*.

4.2. Bilangan Carmichael

Bilangan Carmichael adalah bilangan bulat positif komposit yang tidak dapat dibuktikan kekompositannya menggunakan tes Fermat, untuk semua dasar yang diambil.

Definisi 4.2.1. (Buchmann, 2000) Diberikan sebarang bilangan bulat positif komposit n. Jika untuk setiap $a \in \mathbb{Z}$ berlaku $a^{n-1} \equiv 1 \pmod{n}$, maka n disebut *pseudoprime* ke dasar a. Jika n *pseudoprime* ke semua dasar a dengan $\gcd(a, n) = 1$, maka p disebut bilangan *Carmichael*.

Contoh 4.2.3. (Buchmann, 2000)

Bilangan 561 = 3.11.17 merupakan bilangan Carmichael (yang paling kecil).

Ditemukannya bilangan Carmichael ini membuat tes Fermat tidak lagi optimal untuk digunakan sebagai tes keprimaan. Berikut dijelaskan mengenai tes Miller-Rabbin, yaitu sebuah tes keprimaan yang menggantikan tes Fermat.

4.3. Tes Miller-Rabbin

Tes Miller-Rabbin merupakan penyempurnaan dari tes Fermat. Pada tes keprimaan ini, kelemahan yang terdapat dalam tes Fermat telah dapat dihilangkan. Tes Miller-Rabbin didasarkan pada teorema di bawah ini.

Teorema 4.3.1. (Buchmann, 2000) Diberikan sebarang bilangan bulat positif ganjil $p \geq 3$. Diambil bilangan bulat *s* menjadi pangkat terbesar sedemikian hingga 2^s membagi $p - 1$, yaitu

$$s = \max\{ r \in \mathbb{N} : 2^r \text{ membagi } p - 1\}.$$

Kemudian dihitung $d = \frac{(p-1)}{2^s}$. Jika *p* adalah bilangan prima, maka untuk sebarang $a \in \mathbb{Z}_p^*$ berlaku

$$a^d \equiv 1 \pmod{p}$$

atau terdapat $r \in \{0,1,2,...,s-1\}$ sedemikian hingga

$$a^{2^r.d} \equiv p-1 \pmod{p}.$$

Bukti:

Diberikan sebarang bilangan bulat positif ganjil $p \geq 3$ dan $a \in \mathbb{Z}_p^*$. Diambil bilangan bulat $s = \max\{ r \in \mathbb{N} : 2^r \text{ membagi } (p-1)\}$. Selanjutnya, dihitung $d = \frac{(p-1)}{2^s}$.

Misalkan p adalah bilangan prima, maka order dari $\mathbb{Z}_p^*$ adalah $p-1=2^s.d$. Menggunakan Teorema 3.5.4, jika k adalah order dari a^d, maka k merupakan suatu pangkat dari 2. Jika $k=1=2^0$, maka

$$a^d \equiv 1 \pmod{p}.$$

Jika $k>1$, maka $k=2^l$ dengan $1 \le l \le s$. Menggunakan Teorema 3.5.4, $a^{2^{l-1}.d}$ mempunyai order 2. Karena $p-1$ mempunyai order 2, hal ini berakibat

$$a^{2^r d} \equiv p-1 \pmod{p}$$

untuk $r=l-1$ dan $0 \le r \le s$. □

Algoritma 4.2 : Tes Miller-Rabbin (Menezes, Oorschot and Vanstone, 1996)

Input : Sebuah bilangan bulat positif ganjil $p \ge 3$ dan sebuah parameter $t \ge 1$.

Output : Pernyataan "prima" atau "komposit".

Langkah :

1.Tulis $p-1=2^s.r$, dengan r bilangan bulat positif ganjil.

2.Untuk i dari 1 sampai t kerjakan:

2.1. Diambil sebarang bilangan bulat positif a, $2 \le a \le p-1$.

2.2. Hitung $y \equiv a^r \pmod{p}$ menggunakan metode *fast exponentiation*.

2.3. Jika $y \ne 1$ dan $y \ne p-1$ kerjakan:

2.3.1. *Set* $j \leftarrow 1$.

2.3.2. *While* $j \le s-1$ dan $y \ne p-1$ kerjakan:

2.3.2.1. *Set* $y \leftarrow y^2 \bmod p$.

2.3.2.2. Jika $y = 1$, maka *output*(“komposit”).

2.3.2.3. *Set* $j \leftarrow j + 1$.

2.3.3. Jika $y \neq p - 1$, maka *output*(“komposit”).

3. *Output* (“prima”).

Contoh 4.3.2. (Buchmann, 2000)

Telah diketahui bahwa 561 adalah bilangan Carmichael dan tes Fermat gagal untuk membuktikan kekompositannya. Selanjutnya akan digunakan tes Miller-Rabbin untuk membuktikan kekompositan bilangan ini. Diambil a = 2, s = 4, dan r = 35. Dari sini diperoleh $2^{35} \equiv 263 \pmod{561}$, $2^{2.35} \equiv 166 \pmod{561}$, $2^{4.35} \equiv 67 \pmod{561}$, $2^{8.35} \equiv 1 \pmod{561}$. Jadi, terbukti bahwa 561 adalah bilangan komposit.

BAB V

MASALAH LOGARITMA DISKRET

DAN ALGORITMA ELGAMAL

Pada bab-bab sebelumnya, telah dipelajari tentang grup $\mathbb{Z}_p^*$ beserta sifat-sifatnya, dan metode untuk mendapatkan bilangan prima *p*. Pada bab ini dijelaskan salah satu algoritma kriptografi, yaitu algoritma ElGamal. Algoritma ini didasarkan pada masalah logaritma diskret pada $\mathbb{Z}_p^*$. Untuk lebih jelasnya, diberikan pada subbab-subbab berikut ini.

5.1. Masalah Logaritma Diskret

Misalkan *G* adalah grup siklik dengan order *n*, α adalah pembangun *G* dan 1 adalah elemen identitas *G*. Diberikan $\beta \in G$. Permasalahan yang dimunculkan adalah bagaimana menentukan suatu bilangan bulat nonnegatif terkecil *a* sedemikian hingga

$$\beta = \alpha^a .$$

Bilangan bulat *a* seperti ini disebut dengan *logaritma diskret* (*discrete logarithm*) dari β dengan *basis* α. Selanjutnya, masalah bagaimana menentukan bilangan bulat *a* seperti ini disebut dengan *masalah logaritma diskret* (*discrete logarithm problem*). Masalah logaritma diskret ini menjadi sulit apabila digunakan grup dengan order yang besar (Buchmann, 2000).

5.1.1. Masalah Logaritma Diskret pada Grup Pergandaan Bilangan Bulat Modulo Prima

Diberikan bilangan prima *p*, menggunakan Akibat 3.12.1, diperoleh bahwa $\mathbb{Z}_p^*$ adalah grup siklik yang mempunyai order $p-1$. Akibatnya terdapat suatu elemen yang membangun $\mathbb{Z}_p^*$ yang disebut dengan elemen primitif. Misalkan $\alpha \in \mathbb{Z}_p^*$ adalah elemen primitif, maka untuk sebarang $\beta \in \mathbb{Z}_p^*$ terdapat suatu eksponen $a \in \{0,1,...,p-2\}$ sedemikian hingga

$$\beta \equiv \alpha^a \pmod{p}.$$

Eksponen a merupakan logaritma diskret dari β dengan basis α. Untuk menentukan logaritma diskret tersebut bukanlah permasalahan yang mudah, apalagi bila digunakan bilangan prima dan logaritma diskret yang besar (Buchmann, 2000).

Salah satu metode yang dapat digunakan untuk mencari nilai logaritma diskret adalah metode *enumerasi*, yaitu dengan mengecek seluruh kemungkinan, mulai dari 0, 1, 2, dan seterusnya sampai akhirnya ditemukan nilai *a* yang tepat. Metode enumerasi membutuhkan sebanyak $a-1$ proses pergandaan modulo dan sebanyak *a* perbandingan. Apabila digunakan nilai *a* yang lebih besar, maka metode ini membutuhkan proses perhitungan dan waktu yang lebih banyak lagi.

Contoh 5.1.1.1. (Buchmann, 2000)

Akan dihitung nilai logaritma diskret dari 3 dengan basis 5 pada $\mathbb{Z}_{2017}^*$. Menggunakan metode enumerasi diperoleh nilai logaritma diskretnya, yaitu 1030. Metode ini membutuhkan sebanyak 1029 proses pergandaan modulo 2017.

Namun pada penggunaan yang sebenarnya, digunakan nilai logaritma diskret yang besar seperti $a \geq 2^{160}$. Oleh karena itu, dengan menggunakan metode enumerasi dirasakan menjadi sia-sia karena dibutuhkan paling sedikit sebanyak $2^{160} - 1$ proses perhitungan, sehingga dibutuhkan waktu yang sangat lama untuk mencari nilai logaritma diskret tersebut.

Pada skripsi ini tidak dijelaskan bagaimana sulitnya menyelesaikan masalah logaritma diskret ini.

5.2. Algoritma ElGamal

Algoritma ElGamal merupakan algoritma kriptografi asimetris. Pertama kali dipublikasikan oleh Taher ElGamal pada tahun 1985. Algoritma ini didasarkan atas masalah logaritma diskret pada grup $\mathbb{Z}_p^*$.

Algoritma ElGamal terdiri dari tiga proses, yaitu proses pembentukan kunci, proses enkripsi dan proses dekripsi. Algoritma ini merupakan cipher blok, yaitu melakukan proses enkripsi pada blok-blok plainteks dan menghasilkan blok-blok cipherteks yang kemudian dilakukan proses dekripsi, dan hasilnya digabungkan kembali menjadi pesan yang utuh dan dapat dimengerti. Untuk membentuk sistem kriptografi ElGamal, dibutuhkan bilangan prima *p* dan elemen primitif grup $\mathbb{Z}_p^*$.

Untuk lebih jelasnya mengenai algoritma ElGamal, berikut ini diberikan suatu sistem kriptografi ElGamal, yaitu sistem kriptografi yang menggunakan algoritma ElGamal, definisi himpunan-himpunan plainteks, cipherteks dan kunci, serta proses enkripsi dan dekripsi, seperti diberikan pada gambar berikut ini.

Diberikan bilangan prima p dan sebuah elemen primitif $\alpha \in \mathbb{Z}_p^*$. Ditentukan $\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{C} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ dan $a \in \{0,1,..., p-2\}$. Didefinisikan

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta = \alpha^a \bmod p\}.$$

Nilai p, α, dan β dipublikasikan, dan nilai a dirahasiakan.

Untuk $K = (p, \alpha, a, \beta)$, plainteks $m \in \mathbb{Z}_p^*$ dan untuk suatu bilangan acak rahasia $k \in \{0,1,2,..., p-2\}$, didefinisikan

$$e_K(m, k) = (\gamma, \delta)$$

dengan

$$\gamma = \alpha^k \bmod p.$$

dan

$$\delta = \beta^k . m \bmod p.$$

Untuk $\gamma, \delta \in \mathbb{Z}_p^*$, didefinisikan

$$d_K(\gamma, \delta) = \delta . (\gamma^a)^{-1} \bmod p.$$

Gambar 5.1. Sistem kriptografi ElGamal pada $\mathbb{Z}_p^*$ (Stinson, 1995)

5.2.1. Pembentukan Kunci

Proses pertama adalah pembentukan kunci yang terdiri dari kunci rahasia dan kunci publik. Pada proses ini dibutuhkan sebuah bilangan prima p yang digunakan untuk membentuk grup $\mathbb{Z}_p^*$, elemen primitif α dan sebarang $a \in \{0,1,..., p-2\}$.

Kunci publik algoritma ElGamal berupa pasangan 3 bilangan, yaitu (p, α, β), dengan

$$\beta = \alpha^a \bmod p. \qquad (5.1)$$

Sedangkan kunci rahasianya adalah bilangan a tersebut.

Diketahui order dari $\mathbb{Z}_p^*$ adalah $p-1$. Jika digunakan bilangan prima p dengan $p = 2.q+1$ dan q adalah bilangan prima, maka Akibat 3.8.3 dapat digunakan untuk mengecek apakah suatu $\alpha \in \mathbb{Z}_p^*$ merupakan elemen primitif atau tidak. Karena $p-1 = 2.q$, jelas 2 dan q merupakan pembagi prima dari $p-1$, sehingga harus dicek apakah $\alpha^2 \bmod p \neq 1$ dan $\alpha^q \bmod p \neq 1$. Jika keduanya dipenuhi, maka α adalah elemen primitif.

Agar mempermudah dalam menentukan elemen primitif, digunakan bilangan prima p sedemikian hingga $p = 2.q+1$, dengan q adalah bilangan prima. Bilangan prima p seperti ini disebut dengan bilangan *prima aman*. Untuk menentukan apakah suatu bilangan itu prima atau komposit, dapat digunakan tes keprimaan seperti tes keprimaan biasa dan tes Miller-Rabbin. Kerena digunakan bilangan bulat yang besar maka perhitungan pemangkatan modulo dilakukan menggunakan metode *fast exponentiation*.

Algoritma 5.1 : Tes Bilangan Prima Aman

Input : Bilangan prima $p \geq 5$.

Output : Pernyataan “prima aman” atau “bukan prima aman”.

Langkah :

1. Hitung $q = \dfrac{p-1}{2}$.
2. Jika q adalah bilangan prima, maka *output*(”prima aman”).
3. Jika q komposit, maka *output* (”bukan prima aman”).

Contoh 5.2.1.1.

Diberikan bilangan p = 2579, dapat dicek bahwa 2579 adalah bilangan prima. Selanjutnya, dihitung $q = \frac{p-1}{2} = \frac{2579-1}{2} = \frac{2578}{2} = 1289$. Kemudian, dengan melakukan tes keprimaan, diperoleh bahwa 1289 merupakan bilangan prima. Jadi, 2579 adalah bilangan prima aman.

Berikut ini diberikan sebuah algoritma yang dapat digunakan untuk mengetes elemen primitif. Algoritma ini didasarkan pada Akibat 3.8.3.

Algoritma 5.2 : Tes Elemen Primitif

Input : Bilangan prima aman $p \geq 5$ dan $\alpha \in \mathbb{Z}_p^*$.

Output : Pernyataan "α adalah elemen primitif" atau "α bukan elemen primitif".

Langkah :

1. Hitung $q = \frac{p-1}{2}$.
2. Hitung $\alpha^2 \bmod p$ dan $\alpha^q \bmod p$.
3. Jika $\alpha^2 \bmod p = 1$, maka *output*("α bukan elemen primitif").
4. Jika $\alpha^q \bmod p = 1$, maka *output*("α bukan elemen primitif").
5. *Output*("α adalah elemen primitif").

Berikut ini diberikan contoh beberapa elemen primitif dari grup $\mathbb{Z}_{2579}^*$ yang diperoleh menggunakan Algoritma 5.2.

Contoh 5.2.1.2.

Dari Contoh 5.2.1.1, diketahui bahwa p = 2579 merupakan bilangan prima aman. Oleh karena itu, dapat ditentukan bilangan prima $q = \frac{2579-1}{2} = 1289$. Untuk menunjukkan bahwa suatu bilangan bulat α merupakan elemen primitif $\mathbb{Z}_{2579}^*$, harus ditunjukkan bahwa $\alpha^2 \bmod 2579 \neq 1$ dan $\alpha^{1289} \bmod 2579 \neq 1$. Berikut diberikan tabel perhitungan untuk beberapa nilai α yang diberikan.

Tabel 5.1. Perhitungan $\alpha^2 \bmod 2579$ dan $\alpha^{1289} \bmod 2579$

α	2	3	4	5	6	7	8
$\alpha^2 \bmod 2579$	4	9	16	25	36	49	64
$\alpha^{1289} \bmod 2579$	2578	1	1	1	2578	1	2578

Dari Tabel 5.1 diperoleh bahwa 2, 6 dan 8 merupakan elemen primitif $\mathbb{Z}_{2579}^*$, serta 3, 4, 5 dan 7 bukan merupakan elemen primitif $\mathbb{Z}_{2579}^*$.

Karena pada algoritma ElGamal menggunakan bilangan bulat dalam proses perhitungannya, maka pesan harus dikonversi ke dalam suatu bilangan bulat. Untuk mengubah pesan menjadi bilangan bulat, digunakan kode ASCII (*American Standard for Information Interchange*). Kode ASCII merupakan representasi numerik dari karakter-karakter yang digunakan pada komputer, serta mempunyai nilai minimal 0 dan maksimal 255. Oleh karena itu, berdasarkan sistem kriptografi ElGamal di atas maka harus digunakan bilangan prima yang lebih besar dari 255. Kode ASCII berkorespondensi 1-1 dengan karakter pesan.

Berikut ini diberikan suatu algoritma yang dapat digunakan untuk melakukan pembentukan kunci.

Algoritma 5.3 : Algoritma Pembentukan Kunci

Input : Bilangan prima aman $p > 255$ dan elemen primitif $\alpha \in \mathbb{Z}_p^*$.

Output : Kunci publik (p, α, β) dan kunci rahasia *a*.

Langkah :

1. Pilih $a \in \{0,1,..., p-2\}$.
2. Hitung $\beta = \alpha^a \bmod p$.
3. Publikasikan nilai *p*, α dan β, serta rahasiakan nilai *a*.

Pihak yang membuat kunci publik dan kunci rahasia adalah penerima, sedangkan pihak pengirim hanya mengetahui kunci publik yang diberikan oleh penerima, dan kunci publik tersebut digunakan untuk mengenkripsi pesan. Jadi, kentungan menggunakan algoritma kriptografi kunci publik adalah tidak ada permasalahan pada distribusi kunci apabila jumlah pengirim sangat banyak serta tidak ada kepastian keamanan jalur yang digunakan.

Berikut ini diberikan sebuah contoh kasus penggunaan algoritma ElGamal untuk pengamanan suatu pesan rahasia.

Contoh 5.2.1.3.

Misalkan Andi dan Budi saling berkomunikasi. Pada suatu saat nanti, Andi akan mengirimkan pesan rahasia kepada Budi. Oleh karena itu, Budi harus membuat kunci

publik dan kunci rahasia. Berdasarkan Contoh 5.2.1.2, dipilih bilangan prima aman $p = 2579$ dan elemen primitif $\alpha = 2$. Selanjutnya dipilih $a = 765$ dan dihitung

$$\beta = 2^{765} \bmod 2579 = 949.$$

Diperoleh kunci publik $(p, \alpha, \beta) = (2579, 2, 949)$ dan kunci rahasia $a = 765$. Budi memberikan kunci publik $(2579, 2, 949)$ kepada Andi. Kunci rahasia tetap dipegang oleh Budi dan tidak boleh ada yang mengetahui selain dirinya sendiri.

5.2.2. Enkripsi

Pada proses ini pesan dienkripsi menggunakan kunci publik (p, α, β) dan sebarang bilangan acak rahasia $k \in \{0, 1, ..., p-2\}$. Misalkan *m* adalah pesan yang akan dikirim. Selanjutnya, *m* diubah ke dalam blok-blok karakter dan setiap karakter dikonversikan ke dalam kode ASCII, sehingga diperoleh plainteks $m_1, m_2, ..., m_n$ dengan $m_i \in \{1, 2, ..., p-1\}$, $i = 1, 2, ..., n$. Untuk nilai ASCII bilangan 0 digunakan untuk menandai akhir dari suatu teks.

Proses enkripsi pada algoritma ElGamal dilakukan dengan menghitung

$$\gamma = \alpha^k \bmod p \tag{5.2}$$

dan

$$\delta = \beta^k . m \bmod p\,, \tag{5.3}$$

dengan $k \in \{0, 1, ..., p-2\}$ acak. Diperoleh cipherteks (γ, δ).

Bilangan acak *k* ditentukan oleh pihak pengirim dan harus dirahasiakan, jadi hanya pengirim saja yang mengetahuinya, tetapi nilai *k* hanya digunakan saat melakukan enkripsi saja dan tidak perlu disimpan.

Algoritma 5.4 : Algoritma Enkripsi

Input : Pesan yang akan dienkripsi dan kunci publik (p, α, β).

Output : Cipherteks (γ_i, δ_i), $i = 1, 2, ..., n$.

Langkah :

1. Pesan dipotong-potong ke dalam bentuk blok-blok pesan dengan setiap blok adalah satu karakter pesan.
2. Konversikan masing-masing karakter ke dalam kode ASCII, maka diperoleh plainteks sebanyak *n* bilangan, yaitu $m_1, m_2, ..., m_n$.
3. Untuk *i* dari 1 sampai *n* kerjakan :

   3.1. Pilih sebarang bilangan acak rahasia $k_i \in \{0, 1, ..., p-2\}$.

   3.2. Hitung $\gamma_i = \alpha^{k_i} \bmod p$.

   3.3. Hitung $\delta_i = \beta^{k_i} . m_i \bmod p$.
4. Diperoleh cipherteks yaitu (γ_i, δ_i), $i = 1, 2, ..., n$.

Contoh 5.2.2.1.

Dari Contoh 5.2.1.1, Andi memperoleh kunci publik $(p, \alpha, \beta) = (2579, 2, 949)$. Pada suatu hari, Andi akan mengirimkan pesan rahasia ”Temui aku di kampus jam 7 pagi” kepada Budi. Oleh karena sifat pesan yang rahasia, maka pesan tersebut harus dienkripsi, dan Andi akan mengenkripsi menggunakan kunci publik $(2579, 2, 949)$ yang telah diberikan Budi. Selanjutnya, Andi melakukan proses berikut. Pertama, pesan dipotong-potong menjadi blok-blok karakter dan setiap karakter dikonversi ke dalam kode ASCII. Perhatikan tabel di bawah ini.

Tabel 5.2. Konversi karakter pesan ke kode ASCII

i	Karakter	Plainteks m_i	ASCII
1	T	m_1	84
2	e	m_2	101
3	m	m_3	109
4	u	m_4	117
5	i	m_5	105
6	<spasi>	m_6	32
7	a	m_7	97
8	k	m_8	107
9	u	m_9	117
10	<spasi>	m_{10}	32
11	d	m_{11}	100
12	i	m_{12}	105
13	<spasi>	m_{13}	32
14	k	m_{14}	107
15	a	m_{15}	97
16	m	m_{16}	109
17	p	m_{17}	112
18	u	m_{18}	117
19	s	m_{19}	115
20	<spasi>	m_{20}	32
21	j	m_{21}	106
22	a	m_{22}	97
23	m	m_{23}	109

i	Karakter	Plainteks m_i	ASCII
24	<spasi>	m_{24}	32
25	7	m_{25}	55
26	<spasi>	m_{26}	32
27	p	m_{27}	112
28	a	m_{28}	97
29	g	m_{29}	103
30	i	m_{30}	105

Berdasarkan Tabel 5.1, diperoleh bahwa banyaknya karakter pada pesan tersebut adalah n = 30. Proses selanjutnya adalah menentukan bilangan acak rahasia $k_i \in \{0,1,...,2577\}$, $i = 1,2,...,30$. Kemudian dihitung $\gamma_i = 2^{k_i} \bmod 2579$ dan $\delta_i = 949^{k_i}.m_i \bmod 2579$, $i = 1,2,...,30$. Perhatikan tabel di bawah ini.

Tabel 5.3. Proses enkripsi

i	m_i	k_i	$\gamma_i = 2^{k_i} \bmod 2579$	$\delta_i = 949^{k_i}.m_i \bmod 2579$
1	84	1414	716	814
2	101	1527	711	344
3	109	1843	1512	1252
4	117	2175	559	2337
5	105	1553	929	1032
6	32	2210	838	1014
7	97	1404	313	1998
8	107	2183	1259	257
9	117	1091	195	1173
10	32	1606	2183	275

i	m_i	k_i	$\gamma_i = 2^{k_i} \bmod 2579$	$\delta_i = 949^{k_i}.m_i \bmod 2579$
11	100	1664	363	960
12	105	990	875	1089
13	32	1127	2264	1150
14	107	766	1898	342
15	97	2298	520	1516
16	109	146	22	359
17	112	2483	1742	830
18	117	702	1052	1302
19	115	988	2153	2087
20	32	1230	235	965
21	106	2040	2099	143
22	97	2092	974	1480
23	109	1362	2528	55
24	32	1236	2145	2305
25	55	1463	2435	2113
26	32	1012	998	1790
27	112	2385	1497	1207
28	97	2154	329	1233
29	103	183	1516	960
30	105	869	2473	2104

Berdasarkan Tabel 5.3, diperoleh cipherteks (γ_i, δ_i), $i = 1, 2, ..., 30$ sebagai berikut.

(716, 814)	(711, 344)	(1512, 1252)	(559, 2337)	(929, 1032)
(838, 1014)	(313, 1998)	(1259, 257)	(195, 1173)	(2183, 275)
(363, 960)	(875, 1089)	(2264, 1150)	(1898, 342)	(520, 1516)
(22, 359)	(1742, 830)	(1052, 1302)	(2153, 2087)	(235, 965)
(2099, 143)	(974, 1480)	(2528, 55)	(2145, 2305)	(2435, 2113)
(998, 1790)	(1497, 1207)	(329, 1233)	(1516, 960)	(2473, 2104)

Selanjutnya, Andi mengirimkan cipherteks ini kepada Budi.

Salah satu kelebihan algoritma ElGamal adalah bahwa suatu plainteks yang sama akan dienkripsi menjadi cipherteks yang berbeda-beda. Hal ini dikarenakan pemilihan bilangan *k* yang acak. Akan tetapi, walaupun cipherteks yang diperoleh berbeda-beda, tetapi pada proses dekripsi akan diperoleh plainteks yang sama, seperti dijelaskan berikut ini.

5.2.3. Dekripsi

Setelah menerima cipherteks (γ, δ), proses selanjutnya adalah mendekripsi cipherteks menggunakan kunci publik *p* dan kunci rahasia *a*. Dapat ditunjukkan bahwa plainteks *m* dapat diperoleh dari cipherteks menggunakan kunci rahasia *a*.

Teorema 5.2.3.1. (Menezes, Oorschot and Vanstone, 1996) Diberikan (p, α, β) sebagai kunci publik dan *a* sebagai kunci rahasia pada algoritma ElGamal. Jika diberikan cipherteks (γ, δ), maka

$$m = \delta.\left(\gamma^a\right)^{-1} \bmod p,$$

dengan *m* adalah plainteks.

Bukti:

Diketahui kunci publik (p, α, β) dan kunci rahasia *a* pada algoritma ElGamal. Diberikan cipherteks (γ, δ), dari persamaan (5.1), (5.2) dan (5.3) diperoleh bahwa

$$\begin{aligned}
\delta.\left(\gamma^a\right)^{-1} &\equiv \left(\beta^k.m\right).\left(\gamma^a\right)^{-1} \pmod p \\
&\equiv \beta^k.m.\gamma^{-a} \pmod p \\
&\equiv \left(\alpha^a\right)^k.m.\left(\alpha^k\right)^{-a} \pmod p
\end{aligned}$$

$$\equiv \alpha^{a.k}.m.\alpha^{-a.k} \pmod{p}$$

$$\equiv m.\alpha^{0} \pmod{p}$$

$$\equiv m \pmod{p}.$$

Dengan demikian terbukti bahwa $m = \delta.\left(\gamma^{a}\right)^{-1} \bmod p$. □

Karena $\mathbb{Z}_p^*$ merupakan grup siklik yang mempunyai order $p-1$ dan $a \in \{0,1,..., p-2\}$, maka $\left(\gamma^{a}\right)^{-1} = \gamma^{-a} = \gamma^{p-1-a} \bmod p$.

Algoritma 5.5 : Algoritma Dekripsi

Input : Cipherteks (γ_i, δ_i), $i = 1, 2, ..., n$, kunci publik *p* dan kunci rahasia *a*.

Output : Pesan asli.

Langkah :

1. Untuk *i* dari 1 sampai *n* kerjakan :
   1.1. Hitung $\gamma_i^{\,p-1-a} \bmod p$.
   1.2. Hitung $m_i = \delta_i.\gamma_i^{\,p-1-a} \bmod p$.
2. Diperoleh plainteks $m_1, m_2, ..., m_n$.
3. Konversikan masing-masing bilangan $m_1, m_2, ..., m_n$ ke dalam karakter sesuai dengan kode ASCII-nya, kemudian hasilnya digabungkan kembali.

Contoh 5.2.3.1.

Berdasarkan Contoh 5.2.1.3 dan 5.2.2.1, Andi telah mengirimkan cipherteks kepada Budi. Cipherteks yang diperoleh Budi adalah sebagai berikut.

(716, 814)	(711, 344)	(1512, 1252)	(559, 2337)	(929, 1032)
(838, 1014)	(313, 1998)	(1259, 257)	(195, 1173)	(2183, 275)
(363, 960)	(875, 1089)	(2264, 1150)	(1898, 342)	(520, 1516)
(22, 359)	(1742, 830)	(1052, 1302)	(2153, 2087)	(235, 965)
(2099, 143)	(974, 1480)	(2528, 55)	(2145, 2305)	(2435, 2113)
(998, 1790)	(1497, 1207)	(329, 1233)	(1516, 960)	(2473, 2104)

Budi mempunyai kunci publik p = 2579 dan kunci rahasia a = 765. Selanjutnya, menggunakan Algoritma 5.5, Budi mendapatkan pesan asli yang dikirimkan Andi dengan melakukan perhitungan sebagai berikut.

Tabel 5.4. Proses dekripsi

i	γ_i	δ_i	$\gamma_i^{1813} \bmod 2579$	$m_i = \delta_i.\gamma_i^{1813} \bmod 2579$	Karakter m_i
1	716	814	1090	84	T
2	711	344	2047	101	e
3	1512	1252	2301	109	m
4	559	2337	202	117	u
5	929	1032	1552	105	i
6	838	1014	936	32	<spasi>
7	313	1998	133	97	a
8	1259	257	1887	107	k
9	195	1173	1128	117	u
10	2183	275	394	32	<spasi>
11	363	960	2203	100	d
12	875	1089	1826	105	i
13	2264	1150	767	32	<spasi>
14	1898	342	219	107	k
15	520	1516	1771	97	a
16	22	359	1825	109	m

i	γ_i	δ_i	$\gamma_i^{1813} \bmod 2579$	$m_i = \delta_i.\gamma_i^{1813} \bmod 2579$	Karakter m_i
17	1742	830	286	112	p
18	1052	1302	422	117	u
19	2153	2087	655	115	s
20	235	965	1502	32	<spasi>
21	2099	143	650	106	j
22	974	1480	1598	97	a
23	2528	55	424	109	m
24	2145	2305	847	32	<spasi>
25	2435	2113	1389	55	7
26	998	1790	925	32	<spasi>
27	1497	1207	1421	112	p
28	329	1233	2098	97	a
29	1516	960	438	103	g
30	2473	2104	1330	105	i

Berdasarkan perhitungan pada Tabel 5.4, Andi mengetahui pesan rahasia yang dikirimkan oleh Budi, yaitu ”Temui aku di kampus jam 7 pagi”.

BAB VI

IMPLEMENTASI DAN UJI COBA

Pada bab ini dibahas implementasi algoritma ElGamal serta konsep-konsep yang diberikan pada bab sebelumnya dalam sebuah program komputer yang ditulis menggunakan bahasa pemrograman Pascal, kemudian dilakukan uji coba pada program tersebut. Untuk selanjutnya, penulisan kode program, contoh program, nama fungsi, prosedur, nama file, nama direktori atau folder, dan output program ditulis menggunakan jenis huruf `Courier New` dengan ukuran 10pt, contoh: `writeln('kunci publik p=');`.

6.1. Sarana Implementasi

Untuk melakukan pembuatan program, penulis menggunakan spesifikasi perangkat keras dan perangkat lunak berikut.

Tabel 6.1. Spesifikasi perangkat keras

Jenis Perangkat	Spesifikasi
Nama Komputer	Acer Aspire 3004NWLMi
Processor	Mobile AMD® Sempron™ 3100+ 1,8 GHz
Memori	DDR 768 MB PC3200
Hardisk	40 GB
Monitor	TFT LCD 15.4 WXGA
Resolusi Monitor	1280 × 800

Tabel 6.2. Spesifikasi perangkat lunak

Jenis Perangkat	Spesifikasi
Sistem Operasi	Linux
Kernel Linux	Versi 2.6.18
Distribusi Linux	Debian GNU/Linux 4.0 (*Etch*)
Window Manager	GNOME 2.14.3
Kompiler Pascal	Free Pascal Compiler 2.0.4
Editor Teks	gedit 2.14.2
Terminal	GNOME Terminal 2.14.2

Debian GNU/Linux merupakan salah satu distribusi Linux yang cukup populer. Dikembangkan pertama kali pada tahun 1993. Pada saat tugas akhir ini ditulis, Debian GNU/Linux telah mencapai versi 4.0, versi ini dirilis pada tanggal 8 April 2007 dan diberi nama kode *Etch*. Debian GNU/Linux dapat diperoleh secara bebas dengan men-*download*-nya pada situs resminya di http://www.debian.org atau pada situs lainnya yang menyediakan (*mirror*) seperti di http://info.ugm.ac.id dan http://kambing.vlsm.org.

Free Pascal Compiler merupakan salah satu kompiler bahasa Pascal. Free Pascal Compiler atau cukup disebut dengan Free Pascal ini merupakan salah satu perangkat lunak yang bersifat *open source* dan dapat diperoleh secara gratis melalui website resminya di http://www.freepascal.org. Free Pascal kompatibel dengan kompiler Pascal lainnya seperti Turbo Pascal dan Borland Delphi, serta mendukung berbagai sistem operasi, seperti Linux, Microsoft Windows, MacOS dan FreeBSD.

6.2. Implementasi Algoritma ElGamal

Pada subbab ini dijelaskan tentang pembuatan program yang merupakan implementasi dari algoritma ElGamal menggunakan bahasa Pascal. *Source code* (kode program) dalam bahasa Pascal terdiri dari deklarasi nama program, unit, variabel-variabel dan tipe data, serta beberapa prosedur, fungsi-fungsi dan program utama. *Source code* yang lengkap diberikan pada bagian lampiran.

6.2.1. Deklarasi Nama Program, Unit, Variabel-Variabel dan Tipe Data

Pada implementasi ini, digunakan beberapa variabel dan tipe data. Karena pada algoritma ElGamal digunakan bilangan bulat yang cukup besar, maka beberapa variabel menggunakan tipe data *longint*. Serta menggunakan *crt* sebagai unit yang digunakan. Berikut ini diberikan deklarasi nama program, unit, dan variabel-variabel yang digunakan beserta tipe datanya.

```
program algoritma_elgamal;
uses crt;
var
   p,a,b,c,k,beta,alfa,x,x1,x2,y,y1,y2,q,ai,bs,u,v,r,d1,
   mp,t,n:longint;
   tombol:char;
   pesan:string;
   i,pilihan,j,w:integer;
   m,gamma,delta,d,gm,z:array[1..1000] of longint;
   f,ar,gr:real;
```

6.2.2. Beberapa Fungsi dan Prosedur

Berikut ini diberikan beberapa fungsi dan prosedur yang digunakan dalam program algoritma ElGamal. Ada beberapa prosedur yang tidak dijelaskan, karena

hanya merupakan bentuk lain dan pengembangan dari prosedur-prosedur sebelumnya yang telah dijelaskan.

1. **Fungsi Pengecekan Bilangan Prima**

   Deklarasi: `function cekprima(a:longint):integer;`

   Deskripsi: Fungsi ini digunakan untuk mengecek suatu bilangan yang dimasukkan adalah bilangan prima atau komposit. Jika diberikan masukkan sebuah bilangan bulat, maka keluarannya adalah nilai 1 dan 0. Jika hasilnya adalah bilangan prima, maka outputnya 1, dan jika komposit, maka outputnya 0.

2. **Fungsi Pengecekan Bilangan Prima Aman**

   Deklarasi: `function cekprimaaman(p:longint):integer;`

   Deskripsi: Fungsi ini digunakan untuk mengecek suatu bilangan yang dimasukkan adalah bilangan prima aman atau tidak. Misalkan diberikan masukkan sebuah bilangan bulat. Jika hasilnya adalah bilangan prima aman, maka outputnya 1, dan jika bukan bilangan prima aman, maka outputnya 0.

3. **Fungsi Pencarian Bilangan Prima Aman Acak**

   Deklarasi: `function primaaman:longint;`

   Deskripsi: Fungsi ini digunakan untuk mencari bilangan prima aman secara acak. Bilangan prima aman yang dihasilkan berada di antara bilangan 259 dan 4259.

4. **Fungsi Fast Exponentiation**

   Deklarasi: `function fastexp(r:longint;s:longint;t:longint):longint;`

   Deskripsi: Fungsi ini digunakan untuk menghitung $r^s \bmod t$ menggunakan metode *fast exponentiation*, perintahnya adalah `fastexp(r,s,t)`.

5. Fungsi Tes Miller-Rabbin

Deklarasi: `function mrtest(p:longint; t:longint):integer;`

Deskripsi: Fungsi ini digunakan untuk mengecek suatu bilangan prima aman atau tidak menggunakan tes Miller-Rabbin. Jika hasilnya adalah bilangan prima, maka outputnya 1, dan jika bukan bilangan prima, maka outputnya 0.

6. Fungsi Menghitung Pembagi Persekutuan Terbesar

Deklarasi: `function gcd(a:longint; b:longint):longint;`

Deskripsi: Fungsi ini digunakan untuk menghitung nilai pembagi persekutuan terbesar dari dua bilangan bulat. Misalkan diberikan bilangan bulat *a* dan *b*, maka gunakan perintah `gcd(a,b)`.

7. Fungsi Pengecekan Elemen Primitif

Deklarasi: `function cekprimitif(alfa:longint; p:longint):integer;`

Deskripsi: Fungsi ini digunakan untuk mengecek apakah suatu bilangan merupakan elemen primitif. Fungsi ini meminta masukan berupa dua bilangan. Misalkan diberikan alfa dan bilangan prima aman *p*. Jika alfa benar merupakan elemen primitif $\mathbb{Z}_p^*$, maka outputnya 1, dan jika tidak, maka outputnya 0.

8. Fungsi Pencarian Elemen Primitif Acak

Deklarasi: `function primitif(p:longint):longint;`

Deskripsi: Fungsi ini digunakan untuk mencari elemen primitif aman secara acak. Fungsi ini membutuhkan masukan sebuah bilangan prima aman *p* untuk menentukan $\mathbb{Z}_p^*$.

9. Prosedur Pembangunan Kunci Otomatis

Deklarasi: `procedure keygen;` dan `procedure keygen_kunci_otomatis;`

Deskripsi: Prosedur `keygen;` digunakan untuk membuat kunci publik dan kunci rahasia secara otomatis. Sedangkan prosedur `keygen_kunci_otomatis;` digunakan untuk menentukan kunci yang digunakan.

10. Prosedur Menampilkan Logo

Deklarasi: `procedure logo;`

Deskripsi: Prosedur ini digunakan untuk menampilkan logo di bagian atas program.

11. Prosedur Memasukkan Kunci Pilihan

Deklarasi: `procedure masukkan_kunci_pilihan;`

Deskripsi: Prosedur ini digunakan untuk menerima masukan dan mengecek apakah masukan dapat digunakan sebagai kunci.

12. Prosedur Memasukkan Kunci Publik

Deklarasi: `procedure masukkan_kunci_publik;`

Deskripsi: Prosedur ini digunakan untuk memasukkan kunci publik.

13. Prosedur Memotong Pesan

Deklarasi: `procedure potong1karakter;`

Deskripsi: Prosedur ini digunakan untuk memotong pesan menjadi blok-blok dengan setiap bloknya adalah satu karakter pada pesan. Selanjutnya, setiap karakter dikonversi ke kode ASCII.

14. Prosedur Enkripsi

Deklarasi: `procedure enkripsi;`

Deskripsi: Prosedur ini digunakan untuk melakukan proses enkripsi pada pesan, dan diperoleh cipherteks.

15. Prosedur Dekripsi

Deklarasi: `procedure dekripsi;`

Deskripsi: Prosedur ini digunakan untuk melakukan proses dekripsi pada cipherteks untuk mendapatkan pesan aslinya.

16. Prosedur Representasi Bilangan Bulat b-adic

Deklarasi: `procedure badic;`

Deskripsi: Prosedur ini digunakan untuk menampilkan ekspansi *b*-adic dari suatu bilangan bulat dan bilangan basis.

17. Prosedur Algoritma Euclide

Deklarasi: `procedure euclide;`

Deskripsi: Prosedur ini digunakan untuk menghitung nilai gcd dari dua bilangan bulat menggunakan algoritma Euclide, serta ditampilkan proses perhitungannya.

18. Prosedur Algoritma Euclide yang Diperluas

Deklarasi: `procedure euclidediperluas;`

Deskripsi: Prosedur ini digunakan untuk menghitung nilai gcd dari dua bilangan bulat a dan b, serta nilai x dan y sedemikian hingga $a.x + b.y = \gcd(a,b)$ menggunakan algoritma Euclide yang diperluas, serta ditampilkan proses perhitungannya.

19. Prosedur Mencari Invers Pergandaan

Deklarasi: `procedure invers;`

Deskripsi: Prosedur ini digunakan untuk mencari invers pergandaan dari suatu bilangan yang merupakan elemen dari himpunan $\mathbb{Z}_m$. Prosedur ini meminta dua masukan, yaitu bilangan bulat positif *m* dan sebuah elemen $\mathbb{Z}_m$.

20. Prosedur Tes Fermat

Deklarasi: `procedure tesfermat;`

Deskripsi: Prosedur ini merupakan implementasi dari tes Fermat.

21. Prosedur Program Tambahan

Deklarasi: `procedure programtambahan;`

Deskripsi: Prosedur ini digunakan untuk menampilkan menu program-program tambahan yang mendukung algoritma ElGamal.

22. Prosedur Menu Utama

Deklarasi: `procedure menuutama;`

Deskripsi: Prosedur ini digunakan untuk menampilkan menu utama.

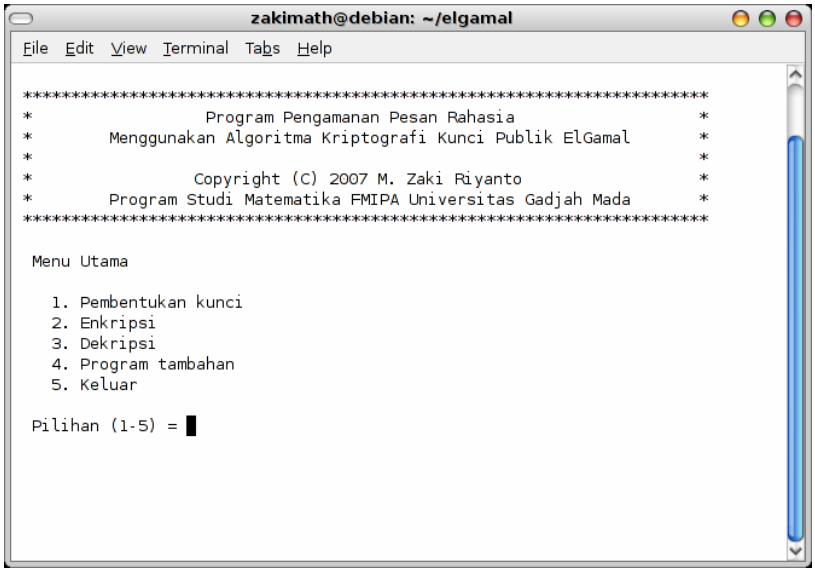
6.2.3. Program Utama

Pada bagian program utama ini hanya digunakan beberapa baris program saja, karena akan merujuk ke `procedure menuutama;`. Pada awal program, terlebih dahulu didefinisikan nilai $p = \alpha = \beta = a = 0$ agar dapat digunakan untuk mengecek apakah kunci telah dibuat atau belum. Berikut ini adalah kode programnya.

```
{Program Utama}
begin
   p:=0; alfa:=0; beta:=0; a:=0;
   menuutama;
end.
```

Program ini terdiri dari empat menu utama, yaitu pembentukan kunci, enkripsi, dekripsi dan program tambahan sebagai pendukung. Desain dari struktur menu-menu pada program ini dapat dijelaskan sebagai berikut.

1. Pembentukan kunci
    1.1. Buat kunci otomatis
    1.2. Masukkan kunci pilihan Anda
    1.3. Kembali ke menu utama
2. Enkripsi
3. Dekripsi
4. Program tambahan
    4.1. Representasi bilangan bulat (b-adic)
    4.2. Algoritma Euclide untuk menghitung gcd(a,b)
    4.3. Algoritma Euclide yang diperluas
    4.4. Hitung $a^k \bmod p$ (metode *fast exponentiation*)
    4.5. Mencari invers pergandaan a mod m
    4.6. Tes keprimaan Fermat
    4.7. Tes keprimaan Miller-Rabbin
    4.8. Tes keprimaan biasa
    4.9. Tes bilangan prima aman
    4.10. Tampilkan semua bilangan prima di antara a dan b
    4.11. Tampilkan semua bilangan prima aman di antara a dan b
    4.12. Tes elemen primitif $\mathbb{Z}_p^*$, untuk p prima aman
    4.13. Tampilkan semua elemen primitif $\mathbb{Z}_p^*$, untuk p prima aman
    4.14. Cek apakah a dan b relatif prima
    4.15. Lihat elemen-elemen $\mathbb{Z}_n$ yang relatif prima dengan n
    4.16. Kembali ke menu utama
5. Keluar



Gambar 6.1. Tampilan program menu utama

6.3. Uji Coba Program

Pada subbab ini dijelaskan uji coba program yang telah dibuat. Pengujian meliputi tiga proses, yaitu proses pembentukan kunci, enkripsi dan dekripsi.

6.3.1. Bahan Pengujian

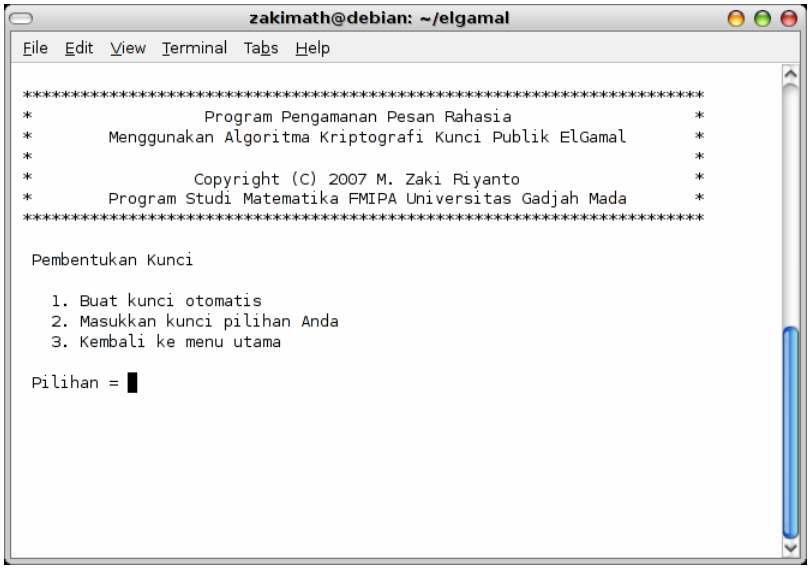
Untuk melakukan uji coba proses enkripsi dan dekripsi menggunakan program yang telah dibuat, digunakan sebuah teks pesan berupa kalimat dengan panjang atau banyaknya karakter maksimal adalah 1000. Teks pesan berupa huruf-huruf, angka dan tanda baca seperti titik, koma, spasi dan sebagainya yang tersedia pada *keyboard*. Pada pembentukan kunci, digunakan bilangan prima yang tidak terlalu besar, mengingat ukuran dari tipe data *longint* mempunyai nilai maksimum 2.147.483.647.

6.3.2. Pengujian Program

Pada uji coba ini dilakukan proses pembentukan kunci, enkripsi dan dekripsi menggunakan program yang telah dibuat.

6.3.2.1. Pengujian Proses Pembentukan Kunci

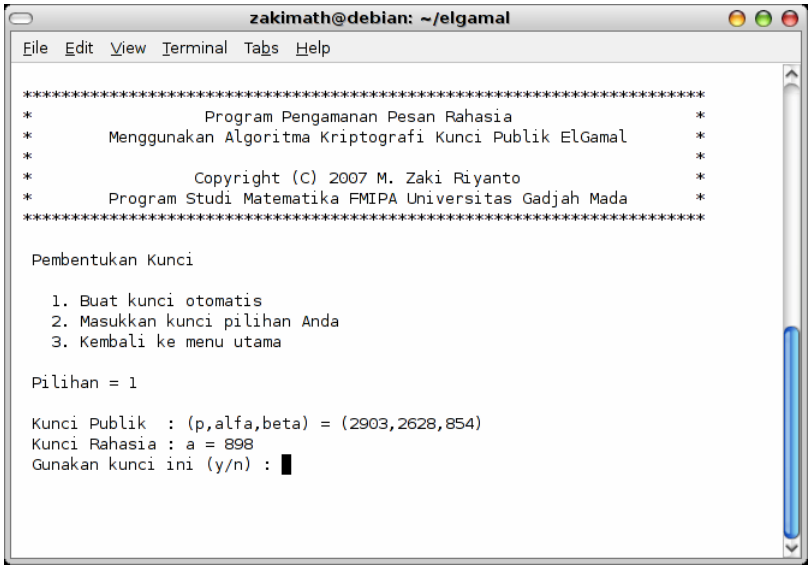
Untuk melakukan pembentukan kunci, dapat dilakukan dengan dua cara. Yaitu program akan membangun kunci acak secara otomatis, atau dapat dengan memasukkan pilihan kunci tertentu dan program akan mengecek apakah kunci yang dimasukkan memenuhi persyaratan sebagai kunci. Pada kotak dialog menu utama, dipilih menu **`1. Pembentukan kunci`**.



Gambar 6.2. Tampilan program pada menu pembentukan kunci

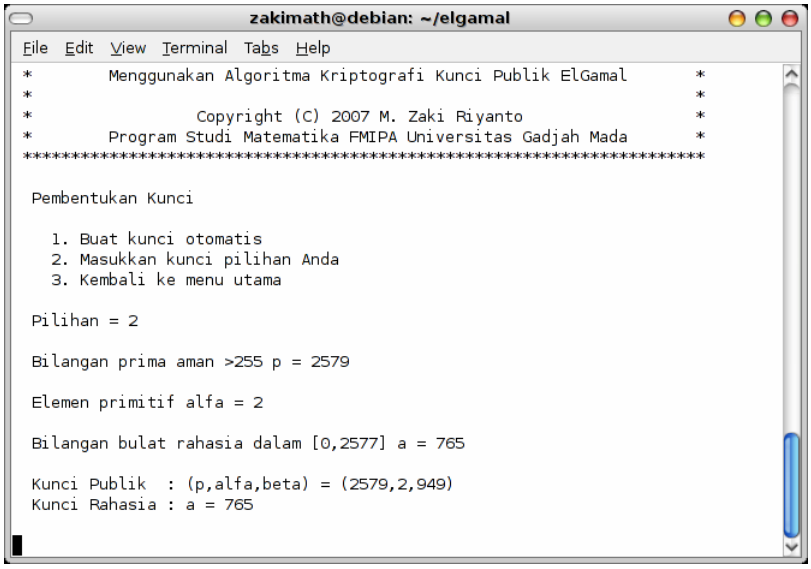
Kemudian ditentukan metode pembentukan kunci, masukan **`1`** untuk membuat kunci secara otomatis atau **`2`** untuk memasukkan pilihan kunci tertentu. Jika dipilih pembuatan kunci otomatis, maka program akan menampilkan kunci yang diperoleh

secara acak. Apabila kunci yang ditampilkan digunakan untuk proses selanjutnya, maka dipilih '**y**', dan apabila tidak, dipilih '**n**', kemudian program akan membuat kunci yang baru lagi secara acak. Pilihan **3** untuk kembali ke menu utama. Apabila masukan kunci pilihan tidak sesuai dengan persyaratan, maka program akan mengeluarkan peringatan.



Gambar 6.3. Tampilan program untuk membentuk kunci otomatis

Misalkan digunakan kunci seperti pada Contoh 5.2.1.3, yaitu $(p,\alpha,\beta)=(2579,2,949)$ sebagai kunci publik dan $a=765$ sebagai kunci rahasia. Pada menu pembentukan kunci, dipilih masukan **2**, kemudian kunci tersebut dimasukkan ke dalam program. Setelah pilihan kunci dimasukkan, maka program melakukan cek terlebih dahulu apakah kunci yang dimasukkan benar-benar sah untuk digunakan sebagai kunci. Jika ternyata tidak sah, maka program akan memberikan peringatan. Jika masukan kunci sah, maka program akan langsung memproses kuncinya. Lebih jelasnya dapat dilihat pada gambar di bawah ini.

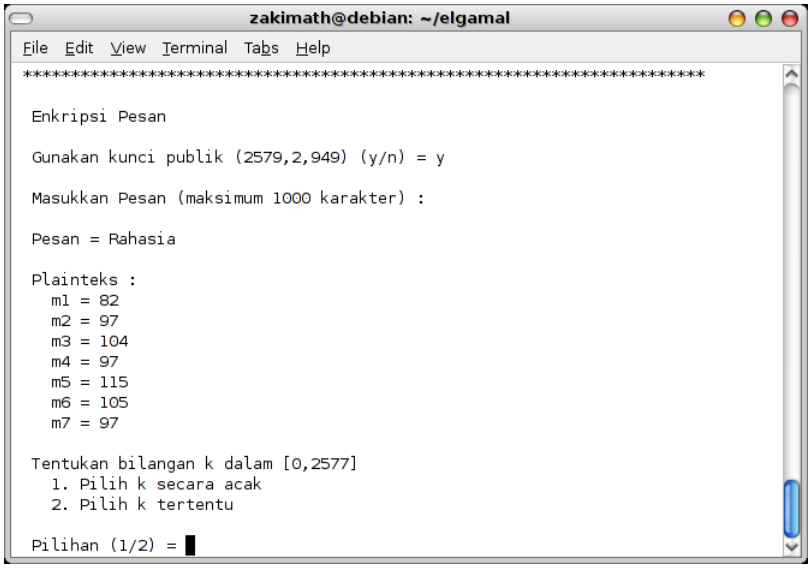


Gambar 6.4. Tampilan program untuk membentuk kunci pilihan

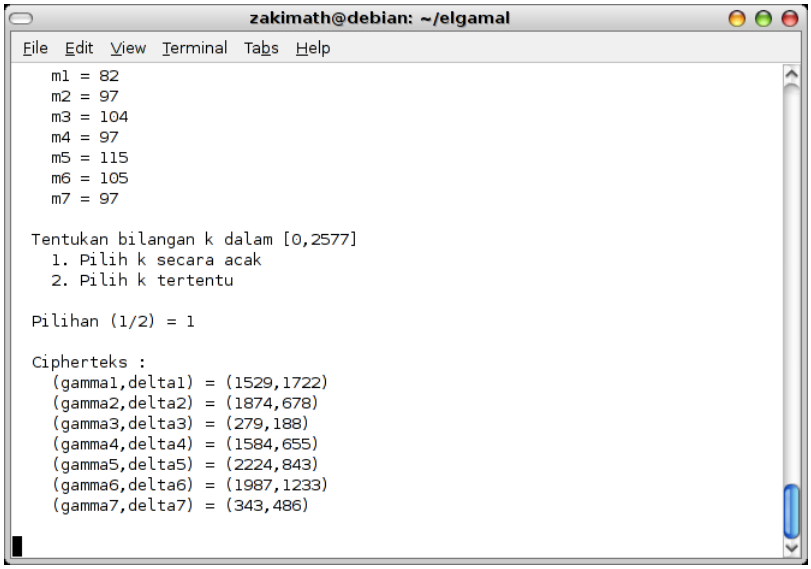
6.3.2.2. Pengujian Proses Enkripsi

Setelah melakukan pembentukan kunci, proses selanjutnya adalah enkripsi. Untuk melakukannya, pada menu utama dipilih **2**, muncul kotak dialog yang menanyakan apakah menggunakan kunci yang telah dibuat atau menggunakan kunci yang lain. Setelah itu dimasukkan teks pesan yang akan dienkripsi dengan jumlah maksimal 1000 karakter. Selanjutnya, program akan menampilkan plainteks dalam bentuk bilangan, kemudian program menanyakan apakah bilangan acak *k* ditentukan sesuai pilihan atau program mencarinya secara otomatis. Kemudian program akan melakukan proses enkripsi dan menampilkan cipherteksnya.

Misalkan proses enkripsi dilakukan menggunakan kunci publik $(2579, 2, 949)$ dan pesannya adalah ”Rahasia”, serta menentukan agar program menentukan bilangan *k* secara acak.



Gambar 6.5. Tampilan program proses enkripsi



Gambar 6.6. Tampilan program menampilkan hasil cipherteks

Pada Gambar 6.6, proses enkripsi pada pesan ”Rahasia” menghasilkan cipherteks sebagai berikut.

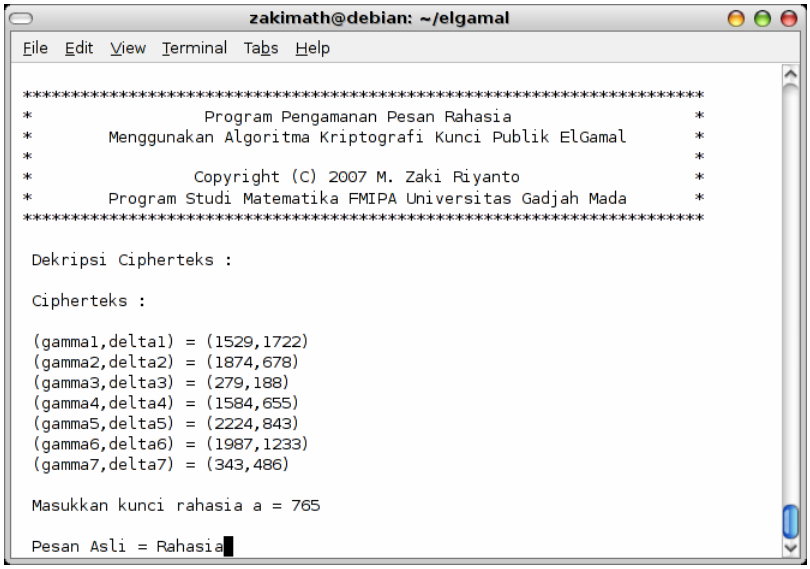
(1529, 1722) (1874, 678) (279, 188) (1584, 655) (2224, 843)

(1987, 1233) (343, 486).

6.3.2.3. Pengujian Proses Dekripsi

Untuk melakukan proses dekripsi, dipilih menu utama pilihan `3`. Program akan menanyakan apakah digunakan kunci publik yang telah dibuat atau kunci publik yang lain. Kemudian program meminta masukan kunci rahasia *a*. Apabila masukkan benar-benar merupakan kunci rahasianya, maka program akan menampilkan pesan aslinya, apabila dimasukkan kunci rahasia yang salah, maka program tetap menghitungnya tetapi akan dihasilkan pesan yang salah.

Misalkan akan didekripsi cipherteks pada Gambar 6.6 di atas berdasarkan kunci publik $(2579, 2, 949)$ dan kunci rahasia $a = 765$.

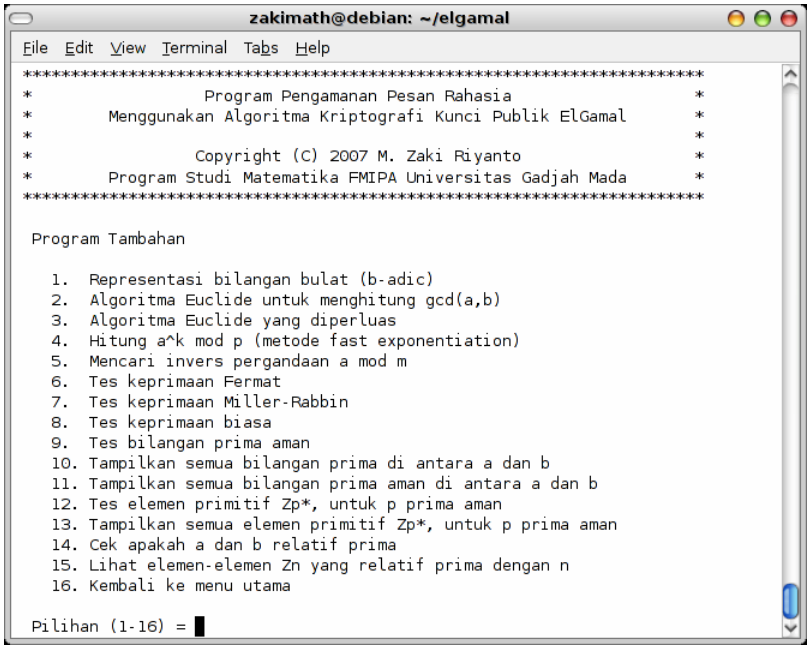


Gambar 6.7. Tampilan program menampilkan proses dekripsi

Pada Gambar 6.7, dapat dilihat bahwa proses dekripsi menghasilkan plainteks berupa pesan ”Rahasia” yang merupakan pesan aslinya.

Untuk mempermudah dalam mencari bilangan-bilangan yang akan digunakan sebagai kunci, telah diberikan sebuah menu tersendiri. Menu ini dapat dijalankan

pada menu utama pilihan **`4. Program tambahan`**. Pada menu ini terdapat beberapa program yang cukup bermanfaat, seperti metode *fast exponentiation*, tes keprimaan, tes elemen primitif, dan sebagainya.



Gambar 6.8. Tampilan program pada menu program tambahan

BAB VII

PENUTUP

7.1. Kesimpulan

Kesimpulan yang dapat diambil penulis setelah menyelesaikan pembuatan skripsi ini adalah :

1. Algoritma kriptografi asimetris, seperti algoritma ElGamal, sangat baik untuk mengatasi masalah distribusi kunci.
2. Diberikan bilangan prima *p*, maka grup pergandaan bilangan bulat modulo *p*, $\mathbb{Z}_p^*$ adalah siklik dan mempunyai $\varphi(p-1)$ elemen primitif.
3. Agar dapat mempermudah dalam menentukan elemen primitif grup $\mathbb{Z}_p^*$, maka penentuan bilangan prima p sebagai kunci publik sebaiknya harus diketahui faktorisasi prima dari $p-1$, seperti bilangan prima aman, yaitu $p = 2.q+1$, dengan q adalah bilangan prima.
4. Algoritma tes elemen primitif adalah sebagai berikut.

   Input : Bilangan prima aman $p \geq 5$ dan $\alpha \in \mathbb{Z}_p^*$.

   Output : Pernyataan "α adalah elemen primitif" atau "α bukan elemen primitif".

   Langkah :

   1. Hitung $q = \frac{p-1}{2}$.
   2. Hitung $\alpha^2 \bmod p$ dan $\alpha^q \bmod p$.
   3. Jika $\alpha^2 \bmod p = 1$, maka *output*("α bukan elemen primitif").

4. Jika $\alpha^q \bmod p = 1$, maka *output*("α bukan elemen primitif").

5. *Output*("α adalah elemen primitif").

5. Algoritma pembentukan kunci adalah sebagai berikut.

   Input : Bilangan prima aman $p > 255$ dan elemen primitif $\alpha \in \mathbb{Z}_p^*$.

   Output : Kunci publik (p, α, β) dan kunci rahasia *a*.

   Langkah :

   1. Pilih $a \in \{0,1,..., p-2\}$.
   2. Hitung $\beta = \alpha^a \bmod p$.
   3. Publikasikan nilai *p*, α dan β, serta rahasiakan nilai *a*.

6. Algoritma enkripsi adalah sebagai berikut.

   Input : Pesan yang akan dienkripsi dan kunci publik (p, α, β).

   Output : Cipherteks (γ_i, δ_i), $i = 1, 2, ..., n$.

   Langkah :

   1. Pesan dipotong-potong ke dalam bentuk blok-blok pesan dengan setiap blok adalah satu karakter pesan.
   2. Konversikan masing-masing karakter ke dalam kode ASCII, maka diperoleh plainteks sebanyak *n* bilangan, yaitu $m_1, m_2, ..., m_n$.
   3. Untuk *i* dari 1 sampai *n* kerjakan :
      3.1. Pilih sebarang bilangan acak rahasia $k_i \in \{0,1,..., p-2\}$.
      3.2. Hitung $\gamma_i = \alpha^{k_i} \bmod p$.
      3.3. Hitung $\delta_i = \beta^{k_i} . m_i \bmod p$.
   4. Diperoleh cipherteks yaitu (γ_i, δ_i), $i = 1, 2, ..., n$.

7. Algoritma dekripsi adalah sebagai berikut.

   Input : Cipherteks (γ_i, δ_i), $i = 1, 2, ..., n$, kunci publik *p* dan kunci rahasia *a*.

   Output : Pesan asli.

   Langkah :

   1. Untuk *i* dari 1 sampai *n* kerjakan :

      1.1. Hitung $\gamma_i^{p-1-a} \bmod p$.

      1.2. Hitung $m_i = \delta_i . \gamma_i^{p-1-a} \bmod p$.

   2. Diperoleh plainteks $m_1, m_2, ..., m_n$.

   3. Konversikan masing-masing bilangan $m_1, m_2, ..., m_n$ ke dalam karakter sesuai dengan kode ASCII-nya, kemudian hasilnya digabungkan kembali.

8. Kelebihan dari algoritma ElGamal adalah proses enkripsi pada plainteks yang sama diperoleh cipherteks yang berbeda-beda, namun pada proses dekripsi diperoleh plainteks yang sama.

7.2. Saran

Setelah membahas dan mengimplementasikan algoritma ElGamal pada skripsi ini, penulis ingin menyampaikan beberapa saran berikut :

1. Apabila tidak ada masalah pada pendistribusian kunci, sangat dianjurkan untuk menggunakan algoritma kunci rahasia.

2. Untuk tetap menjaga keamanan cipherteks hasil enkripsi dengan algoritma ElGamal, kunci publik harus selalu dilindungi dari upaya manipulasi oleh pihak-pihak yang tidak bertanggung jawab.
3. Algoritma ElGamal ini didasarkan atas masalah logaritma diskret pada grup $\mathbb{Z}_p^*$. Perlu dilakukan penelitian dan pengembangan untuk mengaplikasikan masalah logaritma diskret pada suatu struktur aljabar lain yang lebih kompleks.
4. Perlu dilakukan penelitian untuk plainteks berupa *file* data, *image* (citra), video dan sebagainya.
5. Pembaca dapat melakukan penelitian tentang implementasi algoritma ElGamal untuk aplikasi tanda tangan digital dan pertukaran kunci. Seperti pada transaksi *online*, *internet banking*, lembaga intelejen, militer dan sebagainya, alat-alat telekomunikasi seperti telepon seluler (*handphone*) dan jaringan komunikasi nirkabel (*wireless*).
6. Mengimplementasikan algoritma ElGamal menggunakan bahasa pemrograman lain, seperti C/C++, Java, Visual Basic dan lain sebagainya.

DAFTAR PUSTAKA

Buchmann, Johannes A., 2000, *Introduction to Cryptography*, Springer-Verlag New York, Inc., USA.

Fraleigh, John B., 2000, *A First Course in Abstract Algebra*, Sixth Edition, Addison-Wesley Publishing Company, Inc., USA.

Menezes, Oorcshot, and Vanstone, 1996, *Handbook of Applied Cryptography*, CRC Press, Inc. USA.

Pranata, Antony, 2000, *Algoritma dan Pemrograman*, J&J Learning, Yogyakarta.

Rosen, Kenneth H., 1992, *Elementary Number Theory and Its Applications*, AT&T Bell Laboratories, USA.

Schneier, Bruce, 1996, *Applied Cryptography, Second Edition: Protocol, Algorithms and Source Code in C*, John Wiley and Sons, Inc.

Stinson, D.R., 1995, *Cryptography Theory and Practice*, CRC Press, Inc., Florida.

Wikipedia, 2006, *Cryptography*, http://en.wikipedia.org/wiki/Cryptography, 19 Mei 2006, 16:18.

_____, 2006, *ElGamal Encryption*, http://en.wikipedia.org/wiki/ElGamal_encryption, 19 Mei 2006, 16:46.

_____, 2006, *Public Key Cryptography*, http://en.wikipedia.org/wiki/Public-key_cryptography, 19 Mei 2006, 16:28.

Lampiran 1. Kode Program

```
{
 Source Code Program Algoritma Kriptografi ElGamal
 Oleh:
 Nama  : Muhamad Zaki Riyanto (2002)
 NIM   : 02/156792/PA/08944
 Prodi : Matematika
 Jurusan Matematika, FMIPA, Universitas Gadjah Mada Yogyakarta 2007

Copyright (C) 2007 Muh. Zaki Riyanto
Dipersilahkan menggunakan, merubah (modifikasi), mengembangkan dan
menyebarkan secara bebas untuk tujuan bukan komersial (nonprofit),
dengan syarat tidak menghapus atau merubah atribut pembuat kode
program dan pernyataan copyright yang disertakan.
Source code dapat di-download di http://zaki.math.web.id. Pertanyaan
atau    komentar    dapat    dilayangkan    melalui    email    di
zaki@mail.ugm.ac.id.
}

{Deklarasi nama program, unit, variabel dan tipe data}
program algoritma_elgamal;
uses crt;
var
p,a,b,c,k,beta,alfa,x,x1,x2,y,y1,y2,q,ai,bs,u,v,r,d1,mp,t,n:longint;
tombol:char;
pesan:string;
i,pilihan,j,w:integer;
m,gamma,delta,d,gm,z:array[1..1000] of longint;
f,ar,gr:real;

{Fungsi Pengecekan Bilangan Prima}
function cekprima(a:longint):integer;
var
   b,c:longint;
begin
     b:=1;
     repeat b:=b+1; c:=a mod b;
     until c=0;
     if a=b then cekprima:=1
     else cekprima:=0;
end;

{Fungsi Pengecekan Bilangan Prima Aman}
function cekprimaaman(p:longint):integer;
var
   y:longint;
begin
     if cekprima(p)=1 then
     begin
       y:=p-1; y:=y div 2;
       if cekprima(y)=1 then cekprimaaman:=1;
       if cekprima(y)=0 then cekprimaaman:=0;
     end
     else cekprimaaman:=0;
end;
```

```
{Fungsi Mencari Bilangan Prima Aman}
function primaaman:longint;
var
   rand:longint;
begin
     randomize;
     repeat
          repeat
                rand:=random(2000)+128+1;
                if rand mod 2 = 0 then rand:=rand+1;
          until cekprima(rand)=1;
     p:=(2*rand)+1;
     until cekprima(p)=1;
     primaaman:=p;
end;

{Metode Fast Exponentiation}
function fastexp(r:longint; s:longint; t:longint):longint;
var
      x,mtemp,atemp:longint;
begin
      atemp:=r; mtemp:=s; x:=1;
      while mtemp <> 0 do
        begin
           while mtemp mod 2 = 0 do
              begin
                mtemp:=mtemp div 2;
                atemp:=(atemp*atemp) mod t;
              end;
            mtemp:=mtemp-1; x:=(x*atemp) mod t;
         end;
    if x<0 then x:=(x+t) mod t;
    fastexp:=x;
end;

{Fungsi Tes Keprimaan Miller-Rabbin}
function mrtest(p:longint; t:longint):integer;
var
   x,y,s,m,a,r,j,i:longint;
begin
      mrtest:=1;
      randomize;
      x:=p; m:=0;
      repeat x:=x div 2; m:=m+1;
      until (x mod 2) <> 0;
      s:=m; r:=x;
      for i:=1 to t do
      begin
            a:=random(p-3)+2; y:=fastexp(a,r,p) mod p;
            j:=0;
            if (y<>1) and (y<>(p-1)) then
            begin
               j:=1;
               while (j<=(s-1)) and (y<>(p-1)) do
               begin
                  y:=(y*y) mod p;
```

```
                if (y=1) then mrtest:=0;
                j:=j+1;
             end;
             if y <> (p-1) then mrtest:=0;
          end;
     end;
end;

{Fungsi Untuk Menghitung gcd(a,b)}
function gcd(a:longint; b:longint):longint;
var
   r:longint;
begin
      if a<0 then a:=-a;
      if b<0 then b:=-b;
      while b <> 0 do
          begin
               r:=a mod b; a:=b; b:=r;
          end;
      gcd:=a;
end;

{Fungsi Untuk Mengecek Elemen Primitif}
function cekprimitif(alfa:longint; p:longint):integer;
var
   b,q:longint;
begin
     q:=(p-1) div 2; b:=fastexp(alfa,2,p);
     if b=1 then cekprimitif:=0
     else
         b:=fastexp(alfa,q,p);
         if b=1 then cekprimitif:=0
         else cekprimitif:=1;
end;

{Fungsi Untuk Mencari Elemen Primitif acak}
function primitif(p:longint):longint;
var
   alfa:longint;
begin
     repeat
           randomize;
           alfa:=random(p-2)+1;
     until cekprimitif(alfa,p)=1;
     primitif:=alfa;
end;

{Prosedur Pembangunan Kunci (Secara Acak)}
procedure keygen;
begin
     p:=primaaman; alfa:=primitif(p); a:=random(p-3)+1;
     beta:=fastexp(alfa,a,p);
     writeln('  Kunci Publik  : (p,alfa,beta) =
                ',p,',',alfa,',',beta,')');
     writeln('  Kunci Rahasia : a = ',a);
end;
```

```
{Prosedur Menampilkan Logo Atas}
procedure logo;
begin
writeln;
writeln('*******************************************************');
writeln('*          Program Pengamanan Pesan Rahasia           *');
writeln('*Menggunakan Algoritma Kriptografi Kunci Publik
          ElGamal*');
writeln('*                                                     *');
writeln('*Copyright (C) 2007 M. Zaki Riyanto,
          http://zaki.math.web.id  *');
writeln('*Program Studi Matematika FMIPA Universitas Gadjah Mada*');
writeln(' *******************************************************');
writeln;
end;

procedure keygen_kunci_otomatis;
begin
     repeat
           keygen;
           write('  Gunakan kunci ini (y/n) : ');readln(tombol);
           writeln;
     until tombol='y';
end;

procedure masukkan_kunci_pilihan;
begin
     repeat
           repeat
           write('  Bilangan prima aman >255 p = ');readln(p);
           if cekprimaaman(p)=0
              then writeln('  ',p,' bukan bilangan prima aman');
           if p<255 then writeln('  Masukkan p>255');writeln;
           until cekprimaaman(p)=1;
     until p>255;
     repeat
           write('  Elemen primitif alfa = ');readln(alfa);
           if cekprimitif(alfa,p)=0
              then writeln('  ',alfa,' bukan elemen primitif');
           writeln;
     until cekprimitif(alfa,p)=1;
     write('  Bilangan bulat rahasia dalam [0,',p-2,'] a = ');
     readln(a);
     if a > (p-2) then
        begin
           writeln('  ',a,' di luar interval');writeln;
           write('  Bilangan bulat rahasia dalam [0,',p-2,'] a = ');
           readln(a);
        end
     else if a < 0 then
        begin
           writeln('  ',a,' di luar interval');writeln;
           write('  Bilangan bulat rahasia dalam [0,',p-2,'] a = ');
           readln(a);
        end;
     beta:=fastexp(alfa,a,p);
```

```
     writeln;
     writeln('  Kunci Publik  : (p,alfa,beta) =
            (',p,',',alfa,',',beta,')');
     writeln('  Kunci Rahasia : a = ',a);
     writeln;
     readln;
end;

{prosedur Pembentukan Kunci}
procedure pembentukankunci;
begin
     clrscr;
     logo;
     writeln('  Pembentukan Kunci');
     writeln;
     writeln('    1. Buat kunci otomatis');
     writeln('    2. Masukkan kunci pilihan Anda');
     writeln('    3. Kembali ke menu utama');
     writeln;
     write('  Pilihan = ');readln(pilihan);
     writeln;
     case pilihan of
          1: begin
                keygen_kunci_otomatis;
             end;
          2: begin
                masukkan_kunci_pilihan;
             end;
          3: begin end;
     end;
end;

{Prosedur memasukkan kunci publik}
procedure masukkan_kunci_publik;
begin
       writeln('  Masukkan kunci publik (p,alfa,beta) :');
       write('    p = ');readln(p);
       write('    alfa = ');readln(alfa);
       write('    beta = ');readln(beta);
end;

{Prosedur memasukkan kunci publik dekripsi}
procedure masukkan_kunci_publik_dekripsi;
begin
     write('  Masukkan kunci publik p = ');readln(p);
end;

{Memotong Pesan Menjadi Blok-Blok per karakter }
procedure potong1karakter;
begin
     for i:=1 to n do
          begin
               m[i]:=ord(pesan[i]);
          end;
end;
```

```
{Prosedur Enkripsi}
procedure enkripsi;
begin
     clrscr;
     logo;
     writeln('  Enkripsi Pesan');writeln;
     if (alfa+beta)=0 then masukkan_kunci_publik
     else
         begin
            write('  Gunakan kunci publik (',p,',',alfa,',',beta,')
                  (y/n) = ');
            readln(tombol);
            if tombol='n' then masukkan_kunci_publik;
         end;
     {Memasukkan Plainteks}
     writeln;
     writeln('  Masukkan Pesan (maksimum 1000 karakter) :');
     writeln;
     write('  Pesan = ');readln(pesan);
     n:=ord(pesan[0]);
     writeln;
     potong1karakter;
       writeln('  Plainteks :');
       for i:=1 to n do
       begin
            writeln('    m',i,' = ',m[i]);
       end;
     writeln;
     writeln('  Tentukan bilangan k dalam [0,',p-2,']');
     writeln('    1. Pilih k secara acak');
     writeln('    2. Pilih k tertentu');
     writeln;
     write('  Pilihan (1/2) = ');readln(pilihan);writeln;
      case pilihan of
      1: begin
            writeln('  Cipherteks :');
            for i:=1 to n do
            begin
                  k:=random(p-2);
                  gamma[i]:=fastexp(alfa,k,p);
                  delta[i]:=(m[i]*fastexp(beta,k,p)) mod p;
                  write('    (gamma',i,',','delta',i,') = ');
                  write('(',gamma[i],',',delta[i],')');
                  writeln;
            end;
         end;
      2: begin
            writeln('  Cipherteks :');
            for i:=1 to n do
            begin
                  write('    k',i,' = ');read(k);
                  gamma[i]:=fastexp(alfa,k,p);
                  delta[i]:=(m[i]*fastexp(beta,k,p)) mod p;
                  write('      (gamma',i,',','delta',i,') = ');
                  write('(',gamma[i],',',delta[i],')');
                  writeln;
```

```
                readln;
            end;
         end;
       end;
     writeln; readln;
end;

procedure menu_enkripsi;
begin
     clrscr;
     logo;
     writeln('  Enkripsi');
     writeln;
     writeln('    1. Enkripsi pesan ');
     writeln('    2. Kembali ke menu utama');
     writeln;
     write('  Pilihan (1/2) = ');readln(pilihan);
     writeln;
     case pilihan of
          1: begin
                  enkripsi;
             end;
          2: begin end;
     end;
end;

{Prosedur Dekripsi}
procedure dekripsi;
begin
     clrscr;
     logo;
     writeln('  Dekripsi Cipherteks Pesan');writeln;
     if (p+a)=0 then
     begin
          write('  Masukkan kunci publik p = ');readln(p);
     end
     else
         begin
              write('  Gunakan kunci publik p = ',p,' (y/n) = ');
              readln(tombol);
              writeln;
              if tombol='n' then  masukkan_kunci_publik_dekripsi;
         end;
     writeln;
     writeln('  Masukkan cipherteks (gamma, delta), masukkan 0 untuk
              berhenti ');
     writeln;
     j:=0;
     repeat
      j:=j+1;
      write('  gamma',j,' = ');readln(gm[j]);
      write('  delta',j,' = ');readln(d[j]);
      writeln;
     until
          gm[j]=0;
     writeln;
```

```
     write('  Masukkan kunci rahasia a = ');readln(a);
     writeln;
     write('  Pesan Asli = ');
     for i:=1 to (j-1) do
     begin
          m[i]:=(fastexp(gm[i],p-1-a,p)*d[i]) mod p;
               write(chr(m[i]));
     end;
     readln;
end;

procedure dekripsi_hasil;
begin
     clrscr;
     logo;
     writeln('  Dekripsi Cipherteks :');
     writeln;
     writeln('  Cipherteks :');
     writeln;
     for i:=1 to n do
       begin
            write('  (gamma',i,',','delta',i,') = ');
            writeln('(',gamma[i],',',delta[i],')');
       end;
     writeln;
     write('  Masukkan kunci rahasia a = ');readln(a);
     writeln;
     write('  Pesan Asli = ');
     for i:=1 to n do
     begin
          m[i]:=(fastexp(gamma[i],p-1-a,p)*delta[i]) mod p;
          write(chr(m[i]));
     end;
     readln;
end;

procedure menu_dekripsi;
begin
     clrscr; logo;
     writeln('  Dekripsi Cipherteks');
     writeln;
     writeln('    1. Dekripsi cipherteks ');
     writeln('    2. Dekripsi cipherteks hasil sebelumnya');
     writeln('    3. Kembali ke menu utama');
     writeln;
     write('  Pilihan (1-3) = ');readln(pilihan); writeln;
     case pilihan of
          1: begin
                  dekripsi;
             end;
          2: begin
                  dekripsi_hasil;
             end;
          3: begin end;
     end;
end;
```

```
{Prosedur Representasi Bilangan Bulat (b-adic)}
Procedure badic;
begin
     writeln('  Representasi Bilangan Bulat (b-adic)');writeln;
     write('  Masukkan bilangan bulat nonnegatif a = ');readln(a);
     write('  Masukkan bilangan bulat basis >1 b = ');readln(b);
     writeln; write('  ',a,' = (');
     ar:=a; gr:=b; bs:=b;
     f:=ln(ar)/ln(gr); n:=trunc(f);
     i:=0; x:=a; q:=x div b; ai:=x-(q*b); c:=ai;
     while q>0 do
          begin
               i:=i+1; x:=q;
               q:=x div b; ai:=x-(q*b);
               z[i]:=ai;
          end;
     for i:=n downto 1 do
            begin
                 write(z[i],' ');
            end;
      write(c,') basis ',bs);
end;

{Algoritma Euclide}
Procedure euclide;
begin
     writeln('  Algoritma Euclide untuk menghitung gcd(a,b)');
     writeln;
     repeat
         writeln('  Masukkan bilangan bulat a dan b dengan a > b');
         write('    a = ');readln(a);
         write('    b = ');readln(b);
         writeln;
         if (a<b) then writeln('  Pemilihan bilangan salah!');
     until (a > b);
     writeln;
     x1:=a; x2:=b;
     if x1<0 then x1:=-x2;
     if x2<0 then x2:=-x2;
     r:=x1 mod x2;
     if r=0 then x1:=x2
     else
     begin
       while r <> 0 do
          begin
               r:=x1 mod x2;
               q:=x1 div x2;
               writeln('  ',x1,' = (',q,').(',x2,') + ',r);
                   x1:=x2;
               x2:=r;
            end;
     end;
     writeln;
     writeln('  Nilai gcd(',a,',',b,') = ',x1);
end;
```

```
{Prosedur Algoritma Euclide yang diperluas}
Procedure euclidediperluas;
begin
     writeln('  Algoritma Euclide yang diperluas');
     writeln;
     repeat
         writeln('  Masukkan bilangan bulat a dan b dengan a >= b');
         write('    a = ');readln(a);
         write('    b = ');readln(b);
         writeln;
         if (a<b) then writeln('  Pemilihan bilangan salah!');
     until (a > b);
     u:=a; v:=b;
     if b=0 then
        begin
            d1:=a; x:=1; y:=0;
            writeln('  Nilai gcd(',u,',',v,') = ',d1);
            writeln('  Nilai x = ',x);
            writeln('  Nilai y = ',y);
         end
      else
         begin
            x2:=1; x1:=0;
            y2:=0; y1:=1;
            while b>0 do
            begin
               q:=a div b; r:=a-(q*b); x:=x2-(q*x1); y:=y2-(q*y1);
               a:=b; b:=r; x2:=x1; x1:=x; y2:=y1; y1:=y;
            end;
            d1:=a; x:=x2; y:=y2;
            writeln('  Nilai gcd(',u,',',v,') = ',d1);
            writeln('  Nilai x = ',x);
            writeln('  Nilai y = ',y);
         end;
end;

{Prosedur Mencari Invers Pergandaan}
procedure invers;
begin
     writeln('  Mencari Invers Pergandaan a mod m');
     writeln;
     write('  Masukkan bilangan bulat a = ');readln(a);
     write('  Masukkan bilangan bulat positif m = ');readln(mp);
     writeln;
     u:=a; v:=mp; a:=a mod v;
     if mp=0 then
        begin
            d1:=a; x:=1; y:=0;
            x:=x+v; x:=x mod v;
            if d1>1 then
               writeln('  ',u,' tidak punya invers mod ',v)
            else writeln('  Invers ',u,' modulo ',v,' adalah ',x);
         end
    else
        begin
            x2:=1; x1:=0; y2:=0; y1:=1;
```

```
            while mp>0 do
            begin
               q:=a div mp; r:=a-(q*mp);
               x:=x2-(q*x1); y:=y2-(q*y1);
               a:=mp; mp:=r; x2:=x1;
               x1:=x; y2:=y1; y1:=y;
            end;
            d1:=a; x:=x2; y:=y2;
            x:=x+v; x:=x mod v;
            if d1>1 then
                  writeln('  ',u,' tidak punya invers mod ',v)
            else writeln('  Invers ',u,' modulo ',v,' adalah ',x);
         end;
end;

{Prosedur Tes Fermat}
Procedure tesfermat;
begin
     writeln('  Tes Keprimaan Fermat');
     writeln;
     write('  Masukkan bilangan bulat positif ganjil >2 p = ');
     readln(p);
     write('  Masukkan bilangan parameter positif t = ');readln(t);
     for i:=1 to t do
     begin
         writeln;
         write(' Ambil bilangan bulat a di dalam [2, ',p-1,'] a =');
         readln(a);
         y:=fastexp(a,p-1,p);
         writeln;
         if y <> 1 then writeln('  ',p,' adalah bilangan komposit')
         else writeln('  ',p,' adalah bilangan prima');
     end;
end;

{Prosedur Menu Tes Miller-Rabbin}
procedure menutesmillerrabbin;
begin
     writeln('  Tes Keprimaan Miller-Rabbin');
     repeat
           writeln;
           write('  Masukkan bilangan bulat positif ganjil >2 p =');
           readln(p);
           if (p mod 2)=0 then writeln('  ',p,' genap');
     until (p mod 2)=1;
     write('  Masukkan bilangan parameter positif t = ');readln(t);
     x:=p; k:=0; writeln;
     repeat
            x:=x div 2; k:=k+1;
     until x mod 2) <> 0;
     writeln('  ',p-1,' = ',x,' x 2^',k); writeln;
     if mrtest(p,t)=1 then writeln('  ',p,' adalah bilangan prima');
     if mrtest(p,t)=0
        then writeln('  ',p,' adalah bilangan komposit');
end;
```

```
{Prosedur Menu Metode Fast Exponentiation}
procedure menufastexp;
begin
     writeln('  Menghitung a^k mod p menggunakan metode fast
exponentiation');
     writeln;
     write('  Masukkan bilangan bulat positif a = ');readln(a);
     write('  Masukkan bilangan bulat positif k = ');readln(k);
     write('  Masukkan bilangan bulat positif p = ');readln(p);
     writeln;
     write('  ',a,'^',k,' mod ',p,' = ',fastexp(a,k,p));
     writeln;
end;

{Prosedur Menu Tes Prima Biasa}
procedure menutesprima;
begin
     writeln('  Tes Bilangan Prima Biasa');
     writeln;
     write('  Masukkan bilangan bulat >1 p = ');readln(p);
     writeln;
     if cekprima(p)=1
      then writeln('  ',p,' adalah bilangan prima');
     if cekprima(p)=0
      then writeln('  ',p,' adalah bilangan komposit');
     writeln;
end;

{Prosedur Menu Tes Prima Aman}
procedure menutesprimaaman;
begin
     writeln('  Tes Bilangan Prima Aman');
     writeln;
     writeln('  Masukkan bilangan bulat p > 1 ');
     writeln;
     write('  p = ');readln(p); writeln;
     if cekprima(p)=0
      then writeln('  ',p,' adalah bilangan komposit');
     if cekprima(p)=1 then
     begin
          if cekprimaaman(p)=1
            then writeln('  ',p,' adalah bilangan prima aman');
          if cekprimaaman(p)=0
            then writeln('  ',p,' adalah bilangan prima, tetapi
      bukan bilangan prima aman');
     end;
     writeln;
end;

{Prosedur Menu Tes Elemen Primitif}
procedure menutesprimitif;
begin
     writeln('  Tes elemen primitif Zp*, untuk p prima aman');
     writeln;
     repeat
           repeat
```

```
                write('  Masukkan bilangan prima aman p = ');
                readln(p);
                writeln;
                if cekprima(p)=0 then
                begin
                     writeln('  ',p,' bukan bilangan prima');
                     writeln;
                end;
           until cekprima(p)=1;
           if cekprimaaman(p)=0 then
           begin
              writeln; writeln('  ',p,' bukan bilangan prima aman');
           end;
     until cekprimaaman(p)=1;
     writeln('  Masukkan sebarang a elemen Z',p,'* : (a=0 untuk
           berhenti)');
     repeat
           write('  a = ');readln(a);
           writeln;
           if cekprimitif(a,p)=1
              then writeln('  ',a,' adalah elemen primitif');
           if cekprimitif(a,p)=0
              then writeln('  ',a,' bukan elemen primitif');
           writeln;
     until a=0;
end;

{Prosedur Menu Lihat Semua Elemen Primitif}
procedure lihatsemuaprimitif;
begin
     writeln('  Tampilkan semua elemen primitif Zp*, untuk p prima
           aman :');
     writeln;
     repeat
           repeat
                write('  Masukkan bilangan prima aman p = ');
                readln(p);
                writeln;
                if cekprima(p)=0 then
                begin
                     writeln('  ',p,' bukan bilangan prima');
                     writeln;
                end;
           until cekprima(p)=1;
           if cekprimaaman(p)=0 then
           begin
              writeln; writeln('  ',p,' bukan bilangan prima aman');
           end;
     until cekprimaaman(p)=1;
     writeln;
     writeln('  Elemen-elemen primitif Z',p,'* adalah :');
     writeln;
     w:=0;
     for x:=2 to (p-1) do
     begin
          if cekprimitif(x,p)=1 then
```

```
          begin
               w:=w+1;
               write('  ',x,'  ');
          end;
     end;
     writeln; writeln; writeln('  Banyaknya bilangan = ',w);
end;

{Prosedur Menu Lihat Semua Bilangan Prima}
procedure menulihatprima;
begin
     writeln('  Tampilkan semua bilangan prima di antara a dan b');
     writeln;
     write('  Masukkan bilangan >1, a = ');readln(a);
     write('  Masukkan bilangan >',a,', b = ');readln(b);
     writeln; w:=0;
     writeln('  Bilangan-bilangan prima di antara ',a,' dan ',b,'
              adalah :');
     for i:=a to b do
     begin
          if cekprima(i)=1 then
        begin
            w:=w+1; write('  ',i);
        end;
     end;
     writeln; writeln; writeln('  Banyaknya bilangan = ',w);
end;

{Prosedur Menu Lihat Semua Bilangan Prima Aman}
procedure menulihatprimaaman;
begin
  writeln(' Tampilkan semua bilangan prima aman di antara a dan b');
  writeln;
  repeat
           write('  Masukkan bilangan >4, a = ');readln(a);
           if a<5 then writeln('  Pemilihan bilangan salah');
           writeln;
     until a>4;
     repeat
           write('  Masukkan bilangan >',a,' b = ');readln(b);
           if b<a then writeln('  Pemilihan bilangan salah');
           writeln;
     until b>a;
     writeln;
     writeln('  Bilangan-bilangan prima aman di antara ',a,' dan
             ',b,' adalah :');
     w:=0;
     for i:=a to b do
     begin
          if cekprimaaman(i)=1 then
        begin
            w:=w+1; write('  ',i);
        end;
     end;
     writeln; writeln; writeln('  Banyaknya bilangan = ',w);
end;
```

```
{cek relatif prima a dan b}
procedure menu_cek_relatif_prima;
begin
     writeln('  Cek apakah a dan b relatif prima'); writeln;
     write('  Masukkan bilangan a = ');readln(a);
     write('  Masukkan bilangan b = ');readln(b);
     writeln;
     if gcd(a,b)=1 then writeln('  ',a,' dan ',b,' relatif prima')
     else writeln('  ',a,' dan ',b,' tidak relatif prima');
     writeln;
end;

{Lihat Elemen Zn Yang Relatif Prima dengan n}
procedure menu_lihat_elemen_relatif_prima;
begin
     writeln('  Lihat elemen-elemen Zn yang relatif prima dengan
               n');
     writeln;
     write('  Masukkan bilangan >2 n = ');readln(n);
     writeln;
     writeln('  Elemen-elemen Z',n,' yang relatif prima dengan ',n,'
               adalah :');
     w:=0;
     for i:=1 to (n-1) do
     begin
          if gcd(i,n)=1 then
             begin
                  write('  ',i); w:=w+1;
             end;
      end;
      writeln; writeln; writeln('  Banyaknya bilangan = ',w);
end;

{Prosedur Program Tambahan}
procedure programtambahan;
begin
     clrscr;
     logo;
     writeln('  Program Tambahan'); writeln;
     writeln('    1.  Representasi bilangan bulat (b-adic)');
     writeln('    2.  Algoritma Euclide untuk menghitung gcd(a,b)');
     writeln('    3.  Algoritma Euclide yang diperluas');
     writeln('    4.  Hitung a^k mod p (metode fast
                      exponentiation)');
     writeln('    5.  Mencari invers pergandaan a mod m');
     writeln('    6.  Tes keprimaan Fermat');
     writeln('    7.  Tes keprimaan Miller-Rabbin');
     writeln('    8.  Tes keprimaan biasa');
     writeln('    9.  Tes bilangan prima aman');
     writeln('    10. Tampilkan semua bilangan prima di antara a dan
                      b');
     writeln('    11. Tampilkan semua bilangan prima aman di antara
                      a dan b');
     writeln('    12. Tes elemen primitif Zp*, untuk p prima aman');
     writeln('    13. Tampilkan semua elemen primitif Zp*, untuk p
                      prima aman');
```

```
     writeln('    14. Cek apakah a dan b relatif prima');
     writeln('    15. Lihat elemen-elemen Zn yang relatif prima
                      dengan n');
     writeln('    16. Kembali ke menu utama');
     writeln;
     write('  Pilihan (1-16) = ');readln(pilihan); writeln;
     case pilihan of
          1: begin
                  clrscr; logo; badic; readln;
             end;
          2: begin
                  clrscr; logo; euclide; readln;
             end;
          3: begin
                  clrscr; logo; euclidediperluas; readln;
             end;
          4: begin
                  clrscr; logo; menufastexp; readln;
             end;
          5: begin
                  clrscr; logo; invers; readln;
             end;
          6: begin
                  clrscr; logo; tesfermat; readln;
             end;
          7: begin
                  clrscr; logo; menutesmillerrabbin; readkey;
             end;
          8: begin
                  clrscr; logo; menutesprima; readkey;
             end;
          9: begin
                  clrscr; logo; menutesprimaaman; readkey;
             end;
          10: begin
                  clrscr; logo; menulihatprima; readkey;
             end;
          11: begin
                  clrscr; logo; menulihatprimaaman; readkey;
             end;
          12: begin
                   clrscr; logo; menutesprimitif;
              end;
          13: begin
                   clrscr; logo; lihatsemuaprimitif; readkey;
              end;
          14: begin
                   clrscr; logo; menu_cek_relatif_prima; readkey;
              end;
          15: begin
                   clrscr; logo; menu_lihat_elemen_relatif_prima;
                   readkey;
              end;
          16: begin clrscr; end;
     end;
end;
```

```
{Prosedur Menampilkan Menu Depan}
procedure menuutama;
begin
     clrscr;
     logo;
     writeln('  Menu Utama');
     writeln;
     writeln('    1. Pembentukan kunci');
     writeln('    2. Enkripsi');
     writeln('    3. Dekripsi');
     writeln('    4. Program tambahan');
     writeln('    5. Keluar');
     writeln;
     write('  Pilihan (1-5) = ');readln(pilihan);
     writeln;
     case pilihan of
       1: begin
            pembentukankunci;
            menuutama;
          end;
       2: begin
            menu_enkripsi;
            menuutama;
          end;
       3: begin
            menu_dekripsi;
            menuutama;
          end;
       4: begin
            programtambahan;
            menuutama;
          end;
       5: begin clrscr; end;
     end;
end;

{Program Utama}
begin
     p:=0; alfa:=0; beta:=0; a:=0;
     menuutama;
end.

{
### ======================== end of file ======================== ###
    created : M. Zaki Riyanto, last modified: 28.08.07 23.06.20
    any comments at zaki@mail.ugm.ac.id
}
```

Lampiran 2. Tabel Kode ASCII

Kode ASCII (0 – 127)

No.	Kode
0	NULL (null)
1	SOH (start of heading)
2	STX (start of text)
3	ETX (end of text)
4	EOT (end of transmission)
5	ENQ (enquiry
6	ACK (acknowledge)
7	BEL (bell)
8	BS (backspace)
9	TAB (horizontal tab)
10	LF (new line)
11	VT (vertical tab)
12	FF (new page)
13	CR (carriage return)
14	SO (shift out)
15	SI (shift in)
16	DLE (data link escape)
17	DC1 (device control 1)
18	DC1 (device control 2)
19	DC1 (device control 3)
20	DC1 (device control 4)
21	NAK (negative acknowledge)
22	SYN (synchronus idle)
23	ETB (end of trans. blok)
24	CAN (cancel)
25	EM (end of medium)
26	SUB (substitute)
27	ESC (escape)
28	FS (file separator)
29	GS (group separator)
30	RS (record separator)
31	US (unit separator)
32	Space
33	!
34	“
35	#
36	$
37	%
38	&
39	‘
40	(
41	)
42	*
43	+

No.	Kode
44	,
45	-
46	.
47	/
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?
64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W

No.	Kode
88	X
89	Y
90	Z
91	[
92	\
93	]
94	^
95	_
96	`
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{
124	|
125	}
126	~
127	DEL

Kode ASCII *Extended* (128 – 255)

128	Ç	144	É	161	í	177	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	163	ú	179	│	195	├	211	╙	227	π	243	≤
131	â	147	ô	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
134	å	150	û	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
136	ê	152	ÿ	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
137	ë	153	Ö	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	171	½	187	╗	203	╦	219	█	235	δ	251	√
139	ï	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	ⁿ
140	î	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
141	ì	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	
143	Å	160	á	176	░	192	└	208	╨	224	α	240	≡		

Lampiran 3.

DATA PRIBADI PENULIS

Nama Lengkap	:	Muhamad Zaki Riyanto
NIM	:	02/156792/PA/08944
Tempat, Tgl. Lahir	:	Yogyakarta, 13 Januari 1984
Alamat	:	Karangkajen Mg III/911 Brontokusuman, Mergangsan, Yogyakarta 55153
No. HP	:	085697673865, (0274)7008105
Yahoo! Messanger	:	zakimath
E-mail	:	zaki@mail.ugm.ac.id
Homepage	:	http://zaki.math.web.id

Riwayat Pendidikan:

2002 – 2008	:	S1 Program Studi Matematika FMIPA UGM
1999 – 2002	:	SMU Negeri 5 Yogyakarta
1996 – 1999	:	SMP Negeri 4 Yogyakarta
1993 – 1996	:	SD Muh. Bodon “Kotagede”, Yogyakarta
1990 – 1993	:	SD Muh. Banguntapan, Bantul, Yogyakarta