ENKRIPSI TEKS SMS MENGGUNAKAN METODE *QUASIGROUP*

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

oleh :
RINALDI RUSLI
0410960047 - 96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2008

UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

ENKRIPSI TEKS SMS MENGGUNAKAN METODE *QUASIGROUP*

oleh :
RINALDI RUSLI
0410960003 - 96

Telah dipertahankan di depan Majelis Penguji
pada tanggal 07 Agustus 2008
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Pembimbing I	**Pembimbing II**
Dian Eka Rahmawati, S.Si, M.Kom	**Kasyful Amron, ST**
NIP. 132 300 324	**NIP. 132 304 118**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Agus Suryanto, M.Sc
NIP. 132 126 049

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**:**	**Rinaldi Rusli**
NIM	**:**	**0410960047-96**
Program Studi	**:**	**Ilmu Komputer**
Penulis Skripsi Berjudul	**:**	**Enkripsi Teks SMS Menggunakan Metode *Quasigroup***

Dengan ini menyatakan bahwa :

1. **Isi dari Skripsi yang saya buat adalah benar – benar karya sendiri dan tidak menjiplak karya orang lain, selain nama – nama yang termaktub di isi dan tertulis di daftar pustaka dalam Skripsi ini.**
2. **Apabila dikemudian hari ternyata Skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 07 Agustus 2008
Yang menyatakan,

(Rinaldi Rusli)
NIM. 0410960047

ENKRIPSI SMS MENGGUNAKAN METODE *QUASIGROUP*

ABSTRAK

Di era teknologi informasi seperti saat ini, kebutuhan akan komunikasi begitu besar, terutama komunikasi yang berbasis ponsel. Diprediksi pada tahun 2010, angka pengguna ponsel di Indonesia sudah mencapai 131 juta, dan rata-rata orang membeli ponsel, selain digunakan untuk melakukan panggilan atau telepon, mereka juga menggunakan ponsel sebagai alat untuk melakukan SMS(*short message service*). Sayangnya sebagai media komunikasi, SMS merupakan media yang tidak aman. Karena itulah dibutuhkan suatu solusi sehingga SMS bisa menjadi media yang lebih aman.

Salah satu solusi adalah enkripsi SMS. Metode enkripsi SMS yang telah dilakukan penelitian sebelumnya adalah metode *quasigroup*. Metode ini menggunakan prinsip *quasigroup* yaitu memiliki invers kiri dan kanan, yaitu untuk setiap $u, v \in Q$ ada $x, y \in Q$ yang tunggal sehingga $x * u = v$ dan $u * y = v$.

Berdasarkan analisis dan implementasi yang telah dilakukan, diperoleh bahwa penerapan dari metode ini ke dalam SMS tidak terlalu membebani memori ponsel karena membutuhkan ruang hanya 18% dari ukuran memory yang ada di ponsel. Selain itu didapatkan pula bahwa metode ini tahan terhadap serangan *bruteforce* yaitu *ciphertext only attack* dan *key only attack* dengan daya tahan sampai 100 tahun. Metode ini juga memenuhi syarat metode enkripsi yang baik yaitu kerahasiaan, autentikasi, integritas dan non-repudiasi.

SMS ENCRYPTION USING QUASIGROUP METHOD

ABSTRACT

In this information technology era, the need of communication is large, especially cellphone based communication. It predict that in Indonesia, the cellphone user increase until 131 million. And average user instead of using the cellphone for call, they use it to SMS (Short Message Service). Unfortunately, as the communication media SMS is not secure. That's why it need a solution that can make SMS more secure.

One of the solution is SMS encryption. A Method that already research in previous is quasigroup. This method use principal of quasigroup, that it have right and left invers, for every $u, v \in Q$ let $x, y \in Q$ is one so $x * u = v$ dan $u * y = v$

Based on analysis and implementation that already done, it found that the implementation of this method for SMS is not hard to cellphone memory, because it just need 18% of the size of cellphone memory. Moreover this method also resist for the bruteforce attack that is *ciphertext only attack* and key only attack with the resistant until 100 years. This method also suitable for the need of good method that is secret, authentication, integrity and non-repudiation.

UNIVERSITAS BRAWIJAYA

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat, taufik serta hidayah-Nya sehingga Skripsi yang berjudul "Enkripsi Teks SMS Menggunakan Metode *Quasigroup*" dapat diselesaikan.

Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, jurusan Matematika, fakultas MIPA, Universitas Brawijaya.

Banyak pihak yang berperan atas terselesaikannya penelitian dan penulisan Skripsi ini. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Dian Eka R, S.Si, M.Kom selaku dosen pembimbing 1 yang telah memberikan saran dan bimbingan atas penelitian "Enkripsi Teks SMS Menggunakan Metode Quasigroup".
2. Kasyful Amron, ST atas bimbingan dan masukan yang berguna dalam penulisan dan isi laporan.
3. Dr. Agus Suryanto, Msc., selaku ketua jurusan Matematika FMIPA UB Malang.
4. Wayan Firdaus Mahmudy, SSi, MT., selaku Ketua Program Studi Ilmu Komputer UB Malang.
5. Segenap bapak dan ibu dosen yang telah mendidik, dan mengamalkan ilmunya kepada penulis.
6. Segenap staf dan karyawan di Jurusan Matematika FMIPA UB.
7. Ayah, Ibu, dan adikku yang telah memberikan semangat dan mendoakan selama belajar di Program Studi Ilmu Komputer FMIPA UB dan dalam mengerjakan skripsi.
8. Pai yang rela meminjamkan ponselnya untuk uji coba program.
9. Syahrir, Idin, Reza, Dika, Muhlasin, Eli dan semua teman lain yang memberikan dorongan demi terselesaikannya skripsi ini.
10. Semua pihak lain yang telah membantu terselesaikannnya skripsi ini yang tidak bisa penulis sebutkan satu-persatu.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan, dan memiliki banyak kekurangan, sehingga dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang

membangun dari pembaca. Semoga penulisan skripsi ini bermanfaat bagi pembaca.

Malang, Agustus 2008

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

DAFTAR LAMPIRAN

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Sesuai dengan perkembangan teknologi informasi dan dari data yang diperoleh dari detikinet.com, pengguna ponsel di Indonesia sudah mencapai 94,7 juta. Diprediksikan pada tahun 2010, angka pengguna ponsel sudah mencapai 131 juta, dan rata-rata orang membeli ponsel, selain digunakan untuk melakukan panggilan atau telepon, mereka juga menggunakan ponsel sebagai alat untuk melakukan SMS(*short message service*), karena sms memiliki kelebihan lebih murah jika dibandingkan telepon. Selain itu saat ini juga mulai banyak bermunculan aplikasi – aplikasi berbasis sms, seperti sms banking, sms akademik, sms polling, dan lainnya. Dimana beberapa diantaranya membutuhkan inputan berupa data-data penting seperti password, nim, no akun, dan lainnya, untuk mengaktifkan aplikasi sms tersebut. Bisa dibayangkan apa yang terjadi jika data yang tersimpan atau terkirim dalam sms terbaca oleh orang lain, tentunya konsekuensi yang ditanggung akan sangat besar. Misalnya no akun bank atau *password* yang kita simpan dalam SMS yang kita miliki dibaca oleh orang yang tidak bertanggung jawab, maka besar kemungkinan terjadi penyalahgunaan data.

Hal ini didukung pula oleh Robin Good(2004) yang mengatakan bahwa "*SMS is not a secure environment,insecurity often occurs more easily on people rather than technology. The contents of SMS messages are known to the network operator's systems and personnel.Therefore, SMS is not an appropriate technology for secure communications"*.(SMS merupakan lingkungan yang tidak aman. ketidakamanan lebih mudah terjadi pada orang yang tidak mengerti teknologi. Isi dari SMS dapat diketahui oleh operator sistem dan personelnya. Sehingga, sms bukanlah teknologi yang layak untuk komunikasi yang aman).

Sehingga dari hal tersebut tentunya menjadi peluang untuk membahas masalah keamanan SMS, dalam hal ini solusi yang ditawarkan adalah dengan enkripsi. Untuk itulah akan dicoba membuat sebuah aplikasi yang bisa melakukan enkripsi dan deskripsi SMS. Sehingga bisa meningkatkan keamanan data yang ada di SMS.

Untuk membuat sebuah aplikasi enkripsi SMS, digunakan bahasa pemrograman Java dengan menggunakan paket J2ME (Java 2 Micro Edition) yang biasa digunakan untuk membangun sebuah aplikasi pada perangkat yang memiliki memori kecil seperti ponsel, pager dan PDA.(Chocolave Mic, 2003)

Metode yang digunakan untuk melakukan enkripsi ini adalah metode *quasigroup*. Digunakan metode ini, karena merupakan metode baru yang dikenalkan oleh Marko Hassinen dan belum banyak dibahas, sehingga pada karya tulis ini akan dibahas mengenai aplikasi metode ini dalam enkripsi SMS.

Quasigroup dapat didefinisikan sebagai himpunan berhingga *Q* bersama-sama dengan operasi biner * pada *Q* dimana invers kiri dan kanan selalu ada dan tunggal.(Fajar Yuliawan, 2003)Persamaan tersebut berarti bahwa operasi pada *quasigroup* dapat dibalik dan memiliki solusi tunggal.Dengan prinsip inilah digunakan algoritma *quasigroup*. Selain itu penggunaan algoritma ini juga mengacu pada pernyataan dari Marko Hassinen(2003) yang mengatakan bahwa *Quasigroup are well suited encryption of the mobile phone encryption.*(*Quasigroup* adalah enkripsi yang sesuai untuk enkripsi ponsel). Selain itu adanya keterbatasan memori pada ponsel sehingga dibutuhkan sebuah metode komputasi yang sederhana, sehingga tidak terlalu membebani memori pada ponsel. Dan metode ini memenuhi syarat tersebut.

1.2 Rumusan Masalah

a. Bagaimana penerapan metode *quasigroup* di sms?
b. Bagaimana kekuatan dari metode *quasigroup* terhadap serangan keamanan yang umum yaitu *bruteforce attack*?
c. Bagaimana karakteristik dari metode *quasigroup*?
d. Bagaimana kelayakan dari metode *quasigroup* sehingga bisa digunakan sebagai media enkripsi sms?

1.3 Batasan Masalah

a. Percobaan metode *quasigroup* ini dilakukan pada ponsel simulasi dan ponsel asli.
b. Pengiriman sms antar ponsel dilakukan secara simulasi.
c. Karakter masukan dibatasi untuk huruf dan angka.

d. Kelayakan diukur dalam hal hasil *ciphertext*, ukuran file, kecepatan proses dan penggunaan memori.

1.4 Tujuan Penelitian

a. Mengetahui penerapan metode *quasigroup* pada ponsel.
b. Mengetahui kekuatan metode *quasigroup* terhadap serangan *bruteforce*?
c. Mengetahui karakteristik dari metode *quasigroup.*
d. Mengetahui kelayakan metode *quasigroup* sehingga bisa dijadikan sebagai metode untuk mengamankan pesan sms.

1.5 Manfaat Penelitian

a. Memberikan masukan pada *vendor* ponsel untuk menggunakan algoritma *quasigroup* sebagai alternatif pengamanan sms pada ponsel yang dibuat.
b. Membuat sebuah aplikasi untuk melakukan enkripsi pada sms.

1.6 Metodologi Penelitian

Untuk mencapai tujuan yang dirumuskan sebelumnya, maka metodologi yang digunakan dalam penulisan tugas akhir ini adalah:

1. Studi Literatur
   Mempelajari teori-teori yang berhubungan dengan konsep enkripsi dengan metode *quasigroup*.
2. Pendefinisian dan analisis masalah
   Mendefinisikan dan menganalisis masalah untuk mencari solusi yang tepat.
3. Perancangan dan implementasi sistem
   Membuat rancangan model perangkat lunak dengan analisis terstruktur dan mengimplementasikan hasil rancangan tersebut yaitu membuat piranti lunak enkripsi dengan metode *quasigroup*.
4. Uji coba dan analisa hasil implementasi
   Menguji perangkat lunak untuk melakukan enkripsi teks SMS. Kemudian dilakukan análisis mengenai hasil enkripsi, kebutuhan memori, kecepatan proses, ukuran file, karakteristik metode, kesesuaian terhadap syarat enkripsi untuk SMS, dan análisis kekuatan terhadap serangan bruteforce.

1.7 Sistematika Penulisan

Skripsi ini disusun berdasarkan sistematika penulisan sebagai berikut:

1. BAB I PENDAHULUAN
   Berisi latar belakang masalah, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan,
2. BAB II TINJAUAN PUSTAKA
   Menguraikan teori-teori yang berhubungan dengan SMS, Enkripsi, *Quasigroup*, Analisis Algoritma dan Serangan Bruteforce.
3. BAB III METODOLOGI PENELITIAN
   Pada bab ini akan dijelaskan mengenai metode-metode yang digunakan dalam melakukan enkripsi dan dekripsi, serta perancangan kegiatan yang akan dilakukan.
4. BAB IV HASIL DAN PEMBAHASAN
   Pada bab ini akan dilakukan implementasi sistem, pengujian dan analisa model sistem perangkat lunak yang dibangun.
5. BAB V KESIMPULAN DAN SARAN
   Berisi kesimpulan dari seluruh rangkaian penelitian serta saran kemungkinan pengembangannya.

BAB II
TINJAUAN PUSTAKA

2.1 SMS (Short Message Service)

Pengertian SMS menurut Ekawati Prihatini (2006), *Short Message Service* (SMS) adalah salah satu tipe *Instant Messaging* (IM) yang memungkinkan pengguna untuk bertukar pesan singkat kapan pun, walaupun pengguna sedang melakukan panggilan data atau suara. Untuk lebih mengerti SMS, akan diberikan pembahasan mengenai fitur yang ada di SMS sebagai berikut.

2.1.1 Fitur Dasar SMS

SMS pada umumnya memiliki 3 fitur dasar (Job De Haas, 2001), yaitu *Message Submission and Delivery, Status Report, Reply Path* dan *Addressing Mode.* Berikut merupakan penjelasan dari 3 fitur yang ada.

- *Message Submission and Delivery*

  Terdiri dari *message sending* dan *message delivery*. Pada *message sending*, pesan dikirm dari SIM (*Subscriber Identity Module)* ke SMSC (*SMS Center)*, dialamatkan ke SME(*Short Message Entity*) lain sebagai mobile user lain atau host internet. *Originator* SME(*Short Message Entity)* menentukan *validity period* dari pesan tersebut, pesan yang sudah tidak valid lagi akan dihapus oleh SMSC sepanjang pengiriman pesan. Fitur ini dikenal sebagai *Short Message-Mobile Originated* (SM-MO).

  Pada *message delivery*, pesan disampaikan oleh SMSC ke MS(*Message Server*). Dikenal sebagai *Short Message Mobile Terminated* (SM-MT). SM-MO dan SM-MT dapat dikirim / diterima saat *voice call* atau koneksi data sedang berlangsung.

- *Reply Path*

  Replay Path dapat diatur oleh SME asal (atau SMSC serving) untuk mengindikasikan bahwa SMSC serving dan mampu untuk meng-*handle* secara langsung reply dari SME penerima.

- *Addressing Mode*

  Addressing mode menggunakan MSISDN pada format [ITU-E.164]. *Email address* ditentukan oleh IETF pada format [RFC-2822] atau operator *specific numbering*.

2.1.2 Protocol Layer SMS

Seperti umumnya alat komunikasi jaringan, sms juga memiliki protokol yang terdiri atas beberapa layer. Terdapat 4 protokol layer yang membentuk sms (Denis Pankratov, 2004), yaitu : *Application Layer, Transfer Layer, Relay Layer* dan *Link Layer*. Berikut ini merupakan penjelasan dari keempat layer tersebut.

❖ *Application Layer*

Diimplementasi pada SME (*Short Message Entity)* dalam bentuk software aplikasi yang mengirim, menerima dan menginterpretasikan isi pesan (seperti : editor pesan dan games).

❖ *Transfer Layer*

Pesan dianggap sebagai serangkaian bilangan oktet yang mengandung informasi seperti panjang pesan, pengirim atau penerima pesan, tanggal penerimaan pesan.

❖ *Relay Layer*

Relay layer mengizinkan pengiriman pesan antar elemen network yang berbeda. Sebuah elemen network menyimpan pesan sementara jika elemen berikutnya dimana pesan akan di*forward* tidak tersedia.

❖ *Link Layer*

Link layer mengizinkan pengiriman pesan pada level physical. Untuk tujuan ini, pesan diprotek untuk mengatasi kesalahan low level channel.

2.2 Enkripsi

Enkripsi adalah sebuah proses untuk mengubah sebuah *plaintext* (data asli) menjadi susunan data yang tidak dimengerti atau tidak dapat dibaca oleh pihak lain (*ciphertext*). (Setiana, 2006) Enkripsi berasal dari bahasa Yunani *kryptos* yang artinya tersembunyi atau rahasia.

Metode enkripsi yang baik untuk SMS dengan keterbatasan yang ada tidak ditentukan oleh kerumitan dalam mengolah data atau

pesan yang akan disampaikan. Ada 4 syarat yang perlu dipenuhi,(Bruce Scheneir, 2003) yaitu:

1. Kerahasiaan
   plaintext hanya dapat dibaca oleh pihak yang memliki kewenangan.
2. Autentikasi
   Pengirim pesan harus dapat diidentifikasi dengan pasti, penyusup harus dipastikan tidak bisa berpura-pura menjadi orang lain.
3. Integritas
   Penerima pesan harus dapat memastikan bahwa pesan yang dia terima tidak dimodifikasi ketika sedang dalam proses transmisi data.
4. *Non-Repudiation*
   Pengirim pesan harus tidak bisa menyangkal pesan yang dia kirimkan.

2.3 Landasan Matematika Quasigroup

Ada beberapa landasan matematika yang harus diketahui sebelum membahas mengenai aplikasi *quasigroup* dalam kriptografi(Marko Hassinen,2003). Hal ini antara lain tentang *grupoid, quasigroup* sendiri dan *Latin square*.

2.3.1 *grupoid*

Sebuah *groupoid* adalah sebuah himpunan berhingga Q bersama-sama dengan sebuah operasi biner * pada Q yang memenuhi $a * b \in Q$ untuk semua $a,b \in Q$. Dengan kata lain, Q tertutup terhadap operasi * . Operasi biner * pada himpunan Q adalah suatu pemetaan yang mengawankan setiap pasangan himpunan terurut (a,b) anggota himpunan Q.

Sebuah *grupoid* yang memiliki n anggota dapat dinyatakan sebagai sebuah matriks $n \times n$ yang anggotanya merupakan anggota *grupoid* tersebut. Misalnya sebuah *grupoid* yang memiliki anggota {1,2,3,4} dapat dinyatakan dengan matriks suatu matriks M sebagai berikut

$$\begin{pmatrix} 1 & 3 & 2 & 4 \\ 2 & 1 & 3 & 4 \\ 3 & 4 & 1 & 2 \\ 1 & 2 & 3 & 4 \end{pmatrix}$$

Dari baris pertama matriks di atas, dapat diketahui bahwa *grupoid* di atas memenuhi 1 * 1 = 1, 1 * 2 = 3, 1 * 3 = 2, dan 1 * 4 = 4. selanjutnya dari baris keduanya, dapat diketahui bahwa 2 * 1 = 2, 2 * 2 = 1, 2 * 3 = 3, 2 * 4 = 5,dan seterusnya. Dari kolom pertama matriks tersebut, dapat diketahui juga bahwa 1 * 1 = 1, 2 * 1 = 2, 3 * 1 = 3 dan 4 * 1 = 1, dan seterusnya.

Secara umum, jika bilangan-bilangan 1,2,...,*n* diasosiasikan dengan setiap anggota *grupoid* *a*1, *a*2,..., *an*, maka matriks *M* yang merupakan representasi *grupoid* tersebut memiliki anggota

$$M[i,j] = ai * aj \ldots\ldots\ldots (2.1)$$

2.3.2 *Quasigroup*

Sebuah *quasigroup* *Q* adalah *grupoid* yang memiliki invers kiri dan kanan, yaitu untuk setiap $u, v \in Q$ ada $x, y \in Q$ yang tunggal sehingga $x * u = v$ dan $u * y = v$. Hal ini berarti juga bahwa operasi pada *quasigroup* dapat dibalik dan memiliki solusi tunggal. Dengan demikian,kedua operasi invers terhadap operasi pada *quasigroup* dapat didefinisikan, yaitu invers kiri (*left inverse*) dan invers kanan (*right inverse*).

Notasi invers kiri adalah \ dan notasi untuk invers kanan adalah /. Karena ketunggalan ini, daapt juga dikatakan bahwa operator \ bersama sama dengan himpunan berhingga *Q* mendefinisikan sebuah *quasigroup* (*Q*, \) dan untuk aljabar (*Q*, \, *). Demikian halnya dengan operator /. Dalam hal ini, berlaku persamaan

$$x * (x \setminus y) = y = x \setminus (x * y). \quad \ldots\ldots\ldots (2.2)$$

2.3.3 *Latin Square*

Representasi matriks dari sebuah *quasigroup* juga harus memiliki sebuah sifat tambahan, yaitu setiap kolom dan baris pada matriks harus merupakan permutasi dari anggota *quasigroup*.

Dengan kata lain, setiap kolom dan baris pada matris harus mengandung semua anggota-anggota *quasigroup*. Dalam kombinatorika, matriks semacam ini disebut juga sebagai *Latin Square*.

Contoh sebuah *latin square* yang paling sederhana adalah sebagai berikut

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{pmatrix}$$

Dari matriks tersebut terlihat bahwa setiap kolom dan baris merupakan permutasi dari 1,2,3,4. Matriks pertama mengandung tiga buah angka 4 dan sebuah angka 2 pada kolom keempat dan matriks kedua tidak mengandung angka 3 dan 4 sama sekali. Perhatikan matriks pada contoh tersebut. Pada baris ketiga terlihat bahwa 2 * 2 = 3, 2 * 3 = 4, 2 * 4 = 1 dan 2 * 1 = 2. Untuk memenuhi syarat *quasigroup* dengan demikian, 2 \ 3 = 2, 2 \ 4 = 3, 2 \ 1 = 4 dan 2 \ 2 = 1.

2.4 Analisis Algoritma

Menurut Cormen dkk (2003) algoritma adalah prosedur komputasi yang terdefinisi yang mengambil nilai atau sekumpulan nilai sebagai input, dan menghasilkan nilai atau sekumpulan nilai sebagai output. Sedangkan analisis algoritma adalah analisis yang digunakan untuk memprediksi *resource* yang diperlukan oleh sebuah algoritma.

Salah satu cara untuk memprediksi *resource* adalah berdasarkan *running time* yang disimbolkan dengan lambang t(n). *Running time* (t(n)) adalah jumlah operasi / langkah yang dieksekusi(*cost*) dalam unit waktu (*times*). Atau secara matematis bisa ditulis dalam persamaan :

$$t(n) = cost \times times \quad \ldots. (2.3)$$

Dalam melakukan penghitungan *running time* sebuah algoritma Cormen memberikan langkah-langkah sebagai berikut:

1. Ditentukan sebuah parameter yang mengindikasikan sebuah

input.

2. Diidentifikasi operasi dasar (langkah yang dieksekusi) yang dilakukan oleh sebuah algoritma.
3. Memeriksa apakah jumlah waktu untuk melakukan tiap operasi dasar hanya bergantung pada ukuran input, ataukah susunan input juga mempengaruhi. Jika mempengaruhi maka harus dipisahkan antara analisis *worst case, base case* dan *average case*.
4. Dengan rumus standard penjumlahan, disusun penjumlahan yang mengekspresikan berapa kali operasi dasar dilakukan pada algoritma.
5. Dijabarkan langkah ke empat, dengan menentukan tiap satu operasi dasar memerlukan sekali *running time* sehingga bisa ditentukan persamaan *running time* dari algoritma tersebut.

Dalam analisis *running time* berlaku *dominant term* dimana untuk jumlah input yang besar, suku dominan lebih mengindikasikan perilaku algoritma. Sedangkan untuk jumlah input yang kecil , umumnya berjalan sangat cepat, sehingga tidak terlalu mengindikasikan perilaku algoritma.

2.5 Serangan Brute Force Terhadap *Quasigroup*

Menurut Jerzy Wojdylo (2001) dikatakah bahwa serangan brute force dalam kriptografi adalah teknik serangan untuk menemukan sebuah *plaintext* dengan mencoba semua kemungkinan peluang kunci. Sedangkan serangan brute force pada metode *Quasigroup* adalah berdasarkan pada bujur sangkar *vigenere* dengan ukuran 26 baris kali 26 kolom yang berisi *latin square* dari huruf dan angka, dimana baris menunjukkan *key* sedangkan kolom adalah *plaintext*. Kombinasi antara *key* dan *plaintext* akan menghasilkan sebuah *ciphertext*.

Sesuai dengan ukuran dari ukuran bujur sangkar *vigenere* yang berukuran 26 x 26. Jika dilakukan serangan brute force, maka untuk tiap karakter *ciphertext* yang dihasilkan, maka masing-masing karakter akan memiliki peluang sebanyak 26 karakter. Misalkan karakter *ciphertext* A, kemunculan dalam bujursangkar adalah 26. Jadi untuk menentukan *plaintext* yang sesuai maka peluangnya adalah 26.

Jadi jika dirumuskan secara matematis, untuk peluang serangan bruteforce pada *vigenere* berlaku hubungan permutasi perulangan antara panjang *ciphertext* dengan jumlah karakter pada bujur sangkar vigenere , yaitu

$$n^r \quad \text{…. (2.4)}$$

Dengan n merupakan jumlah karakter pada bujur sangkar sedangkan r merupakan jumlah karakter yang diserang. Ada 2 macam serangan *bruteforce* yang umum digunakan yaitu *ciphertext only attack* dan *key only attack*. Pada serangan *ciphertext only attack*, teknik yang digunakan umumnya berdasarkan frekuensi kemunculan huruf. Dari huruf yang ada kemudian dicoba dengan melakukan penggantian ke huruf-huruf yang lain, sampai diketemukan *plaintext* yang bisa dibaca dan dimengerti. Selanjutnya adalah *key only attack*, dimana seorang penyerang menggunakan kombinasi antara *ciphertext* dan *key* sehingga didapatkan *plaintext* yang bisa dipahami dan dimengerti.

2.2.2.1. Analisa *Avalanche Effect*

Salah satu cara untuk menguji ketahanan kriptografi adalah dengan mengukur nilai *avalanche effect* kriptografi tersebut. (Sagita, 2005). *Avalanche effect* adalah perubahan satu bit dapat menghasilkan perubahan lebih dari satu bit setelah satu putaran. (Jaya, 2003) Nilai *avalanche effect* dirumuskan dengan:

$$Avalance\_effect(AE) = \frac{\sum bit\_berubah}{\sum bit\_total} * 100\% \quad (2.5)$$

Nilai A*valanche effect* optimal untuk suatu metode kriptografi adalah 50 %. (Sagita, 2005)

2.6 JAVA Wireless Messagging API

Menurut Enrique Ortiz(2002), dikatakan bahwa JAVA WMA (Wireless Messaging API) adalah "*is an optional package thus supports Java 2 Platform, Mobile Edition (J2ME) applications targeted at cell phones that can send and receive wireless messages"*(Java WMA adalah paket pilihan yang didukung oleh platform Java 2 Mobile Edition(J2ME) untuk aplikasi yang

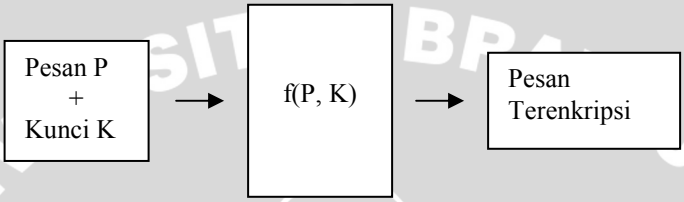
ditujukan pada ponsel,yang digunakan untuk mengirim dan menerima pesan *wireless*). Tabel 2.1 merupakan interface yang ada dari WMA.

Tabel 2.1 Metode dan Antar muka dari WMA
(Sumber : Enriq Ortiz, 2002)

Interface	Description	Methods
`Message`	Base `Message` interface, from which subinterfaces (such as `TextMessage` and `BinaryMessage`) are derived	`getAddress()`, `getTimestamp()`, `setAddress()`
`BinaryMessage`	Subinterface of `Message` that provides methods to set and get the binary payload	`getPayloadData()`, `setPayloadData()`
`TextMessage`	Subinterface of `Message` that provides methods to set and get the text payload	`getPayloadText()`, `setPayloadText()`
`MessageConnection`	Subinterface of the GCF Connection, which provides a factory of `Messages`, and methods to send and receive `Messages`	`newMessage()`, `receive()`, `send()`, `setMessageListener()`, `numberOfSegments()`
`MessageListener`	Defines the listener interface to implement asynchronous notification of `Message` objects	`notifyIncomingMessage()`

BAB III
METODOLOGI DAN PERANCANGAN

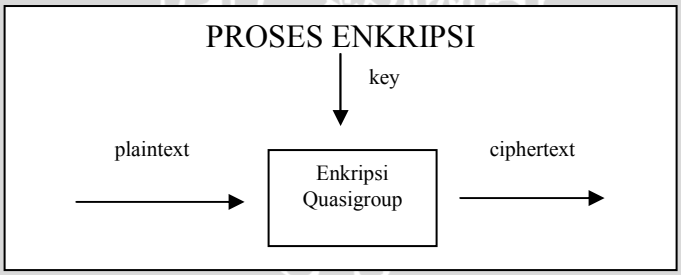
Enkripsi merupakan sebuah proses untuk mengubah sebuah *plaintext* menjadi susunan data yang tidak dimengerti atau tidak dapat dibaca oleh pihak lain. Untuk membuat sebuah aplikasi enkripsi, diagram yang digunakan adalah sebagai berikut.



Pada bab ini akan dijelaskan mengenai 2 proses program yaitu proses enkripsi dan proses dekripsi dengan menggunakan *Quasigroup*.

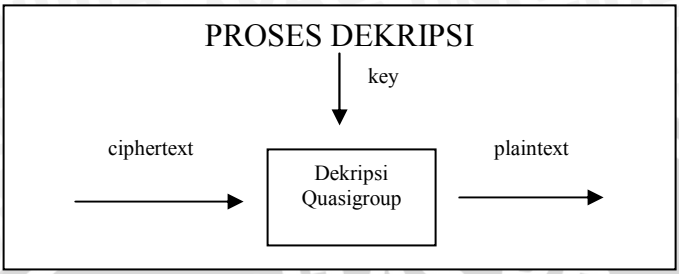
3.1 Proses Enkripsi dan Deskripsi

Terdapat 2 blok proses utama dalam program ini yaitu proses enkripsi dan dekripsi. Untuk proses enkripsi, blok proses digambarkan seperti gambar 3.1



Gambar 3.1 Proses Enkripsi

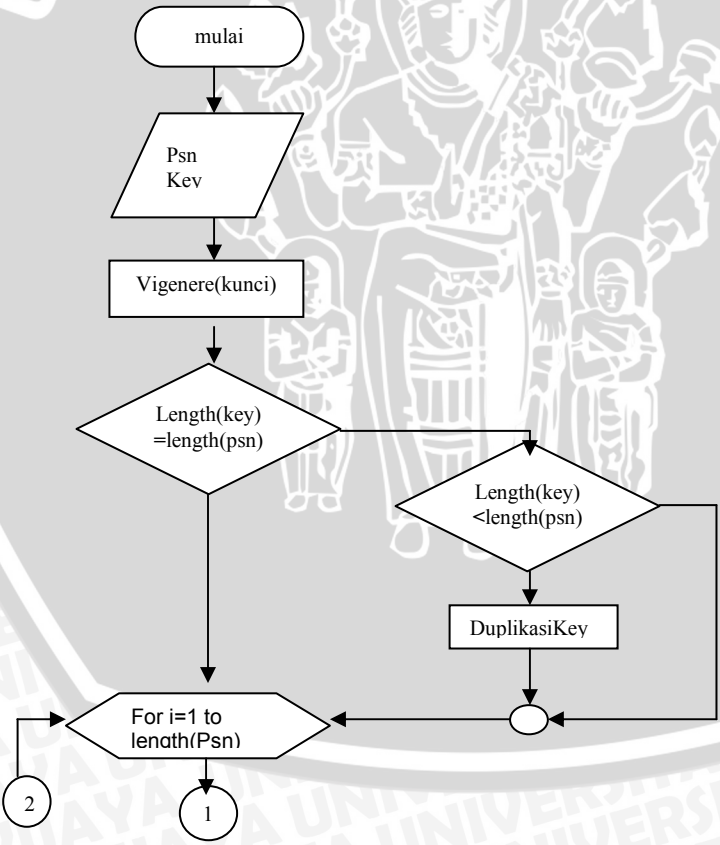
Sedangkan untuk proses dekripsi, yang merupakan proses untuk mengembalikan data terenkripsi (*ciphertext*) ke data asal digambarkan pada gambar 3.2.

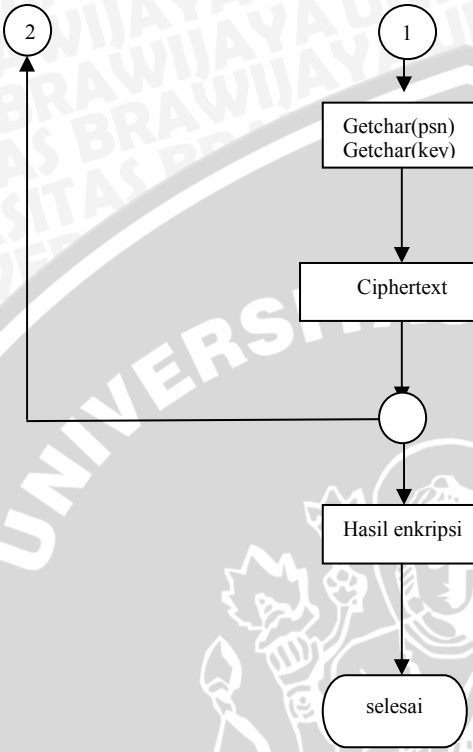


Gambar 3.2 Proses Dekripsi

3.2 Enkripsi *Quasigroup*

Metode yang digunakan untuk melakukan enkripsi ini adalah menggunakan *Quasigroup*. *Flowchart* enkripsi teks yang ada di sms, dapat dijelaskan pada gambar 3.3:





Gambar 3.3 *FlowChart* Enkripsi teks

Adapun penjabaran dari *flowchart* dari gambar 3.3 adalah sebagai berikut:

a. Dibuat sebuah bujur sangkar vigenere n x n yang memenuhi sifat *quasigroup* yaitu memiliki bentuk *latin square* (anggota himpunan matrix merupakan permutasi dari anggotanya). Dapat dilihat pada gambar 3.4.
b. Input pesan (*Plaintext*) merupakan data awal yang belum terenkripsi.*Plaintext* dibaca per-karakter dari kiri ke kanan
   Contoh : input *plaintext* = HAJI KE MEKAH
c. Input kunci (*Key*) merupakan bagian penting dalam enkripsi,berupa teks yang hanya diketahui oleh pengguna. Kunci(*Key*) yang digunakan pada *Quasigroup* memiliki syarat harus sama panjang sesuai dengan panjang *plaintext* inputan. Jika panjang kunci lebih pendek dari panjang *plaintext* maka

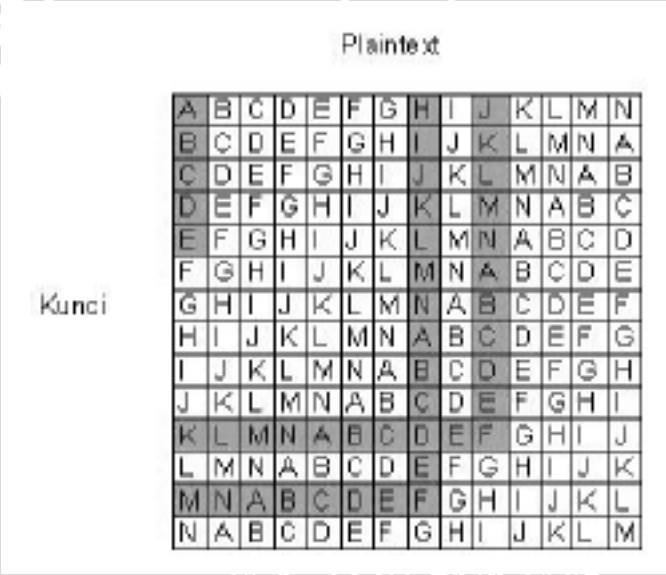
kunci diulang penggunaannya. Untuk spasi tidak diproses karena tidak terdapat pada matrix.

contoh : input kunci = MEKAH

maka dalam penggunaanya MEKAH diulang sampai panjangnya sesuai dengan panjang *plaintext*. Jadi kunci akan menjadi MEKA HM EKAHM

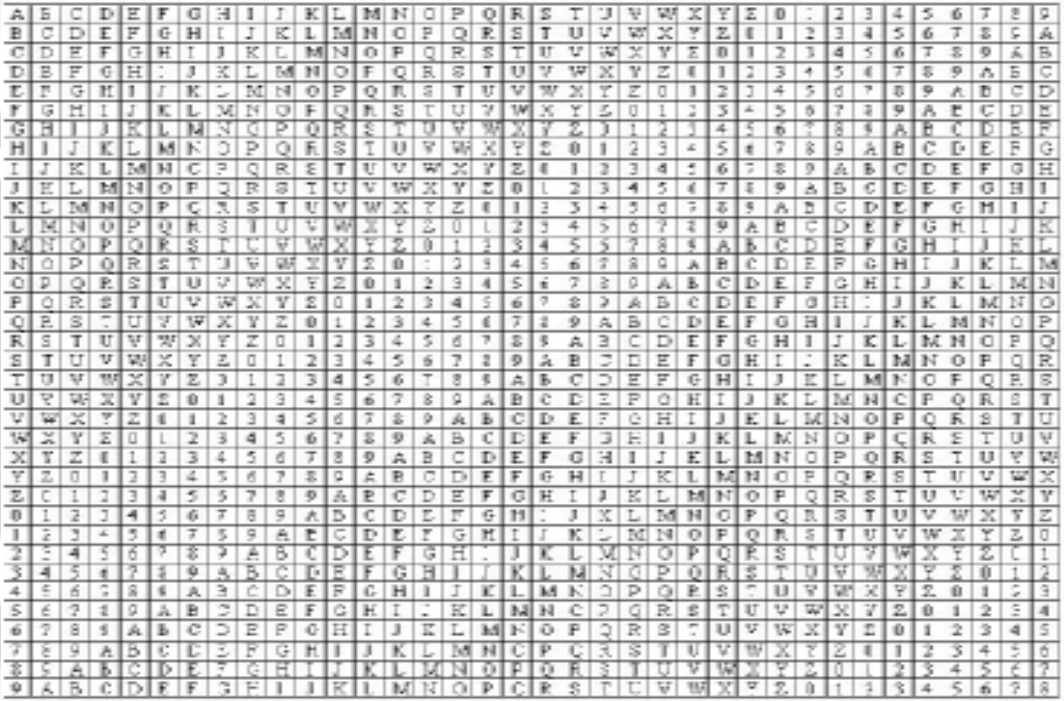
d. Membaca perpotongan antara karakter kunci (sumbu horisontal) dengan tiap karakter *plaintext* (sumbu vertikal) pada bujursangkar *vigenere*.

Contoh : misalkan karakter kunci = M dan karakter *plaintext* = H perpotongan menghasilkan karakter F, seperti yang ditunjukkan pada gambar 3.5



Gambar 3.5 perpotongan *plaintext* dan kunci

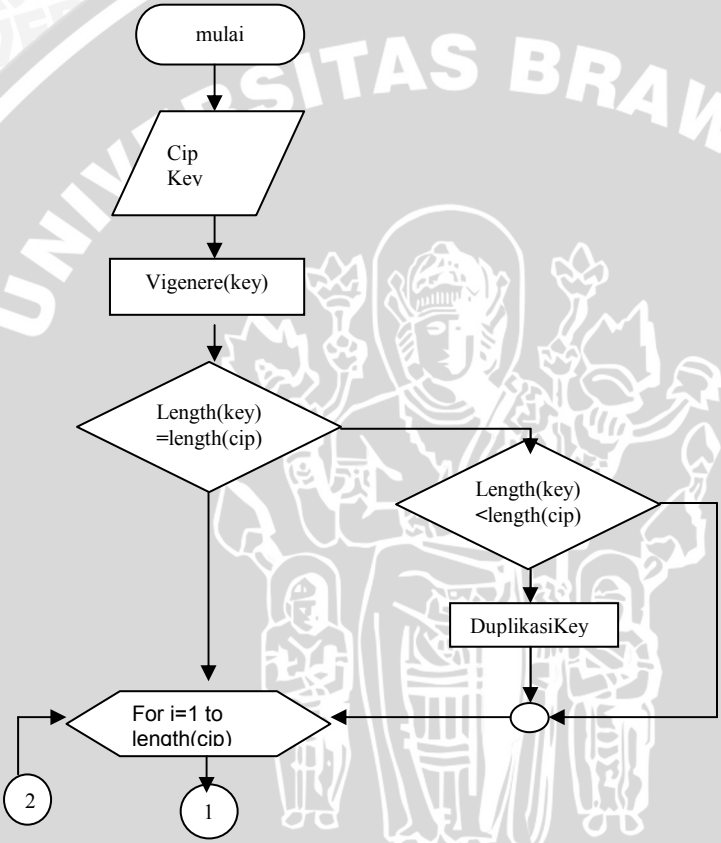
Pembacaan dilakukan tiap karakter sampai dengan akhir akan dihasilkan *ciphertext* = FEFI DC CAKHF.

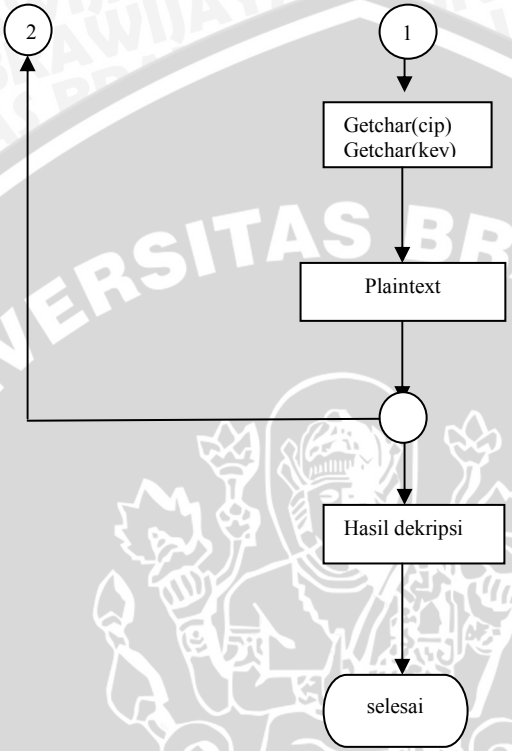


Gambar 3.4 Bujursangkar Vigenere dimodifikasi

3.3 Dekripsi *Quasigroup*

Pada proses dekripsi, dilakukan proses pembalikan dari kunci dengan *ciphertext*. Adapun f*lowchart* dari proses pembalikan atau dekripsi ini dapat dilihat pada gambar 3.6



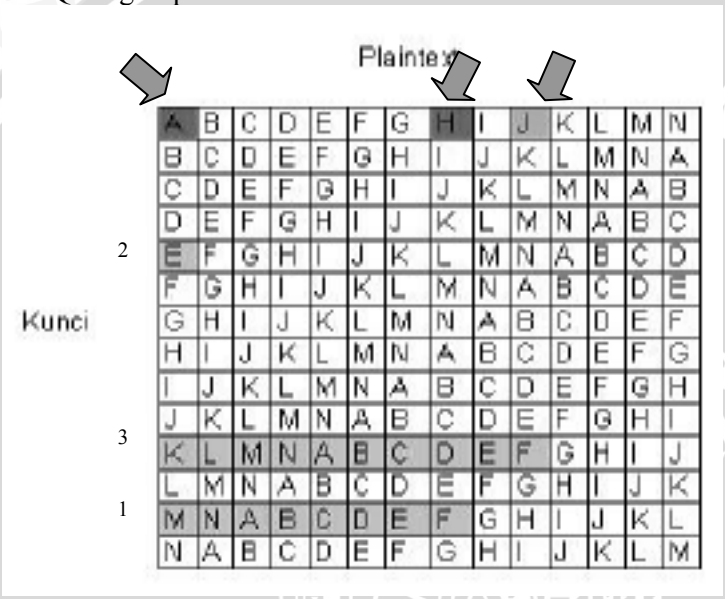


Gambar 3.6 *Flowchart* Dekripsi

Prinsip untuk melakukan dekripsi adalah posisi kunci berada di kolom ke 1 dan *plaintext* berada di baris ke 1. Dari sinilah cara untuk menemukan nilai *Plaintext* berdasarkan kunci dengan *cipher*. Langkah untuk melakukan dekripsi pesan adalah:

a. Untuk mencari huruf *plaintext*, maka cari huruf kunci ke-i di kolom 1, misal kunci berada di baris ke-x.
b. Pada baris ke-x tersebut, cari huruf *cipher* dengan bergerak ke sebelah kanan sampai menemukan cipher. Misal ditemukan pada posisi ke-y.

c. Dari posisi ke-y, dicari nilai baris ke-1, maka nilai *plaintext* = nilai baris ke-1 dari posisi y.
contoh :
ciphertext = F E F I DC CAKH F
kunci = MEKA HM EKAHM
gambar 3.7 merupakan contoh dekripsi algoritma Quasigroup.



Gambar 3.7 Proses Dekripsi Pesan

Pembacaan dilakukan sampai dengan akhir karakter, dan dari hasil dekripsi ini akan didapatkan nilai *plaintext* awal yaitu : HAJI KE MEKAH.

3.4 Perancangan Antar Muka

Perancangan interface dilakukan menggunakan *default device* yang disediakan oleh J2ME Wireless Toolkit. Gambar 3.7 merupakan perancangan *interface* untuk program enkripsi teks sms yang akan dibuat. Di dalam form sms, terdapat 2 menu yaitu menu enkripsi dan menu dekripsi.

Gambar 3.8 Antar muka ketika sms diketik

Kemudian jika menu enkripsi atau dekripsi dipilih, maka akan masuk ke form selanjutnya yaitu form kunci. Demi menjaga kerahasiaan kunci, input kunci yang diketikkan simbolnya diubah ke bentuk bintang/asterisk (*).

Gambar 3.9 Form kunci

3.5 Perancangan Analisis

3.5.1 Analisis Terhadap Penerapan Metode Enkripsi

Dari pengujian ini diharapkan bisa diketahui bagaimana penerapan metode enkripsi *quasigroup* sehingga bisa diaplikasikan pada pengiriman SMS. Pengujian yang dirancang adalah dengan melihat cara kerja, hasil input dan output apakah sesuai dengan teori yang telah ditentukan.

Untuk kejelasan dari pengujian ini dibuat sebuah tabel sebagai berikut:

Tabel 3.1 pengujian terhadap input dan output program yang dihasilkan

No	Input Plaintext	Key	Output Chipertext	Apakah Sesuai Teori ? (y/t)

Dengan tabel tersebut nantinya bisa dilihat apakah teori yang diberikan, jika diaplikasikan ke dalam program apakah memberikan hasil yang sama.

Selain hasil input dan output, akan dianalisis pula beberapa parameter lainnya yaitu penggunaan memori, *running time* dan ukuran file hasil kompilasi.

3.5.2 Analisis Kelayakan Metode *Quasigroup* Sebagai Metode Enkripsi SMS.

Untuk mengukur kelayakan dari metode enkripsi sehingga layak dijadikan metode enkripsi sebuah sms, telah disebutkan oleh Bruce Scheiner(2003). Sehingga dalam melakukan analisis, disesuaikan dengan teori yang disampaikan oleh Bruce Scheiner tersebut.

Adapun syarat yang disampaikan untuk kelayakan dalam enkripsi sms adalah kerahasiaan, autentikasi, integritas dan non repudiasi. Jika keempat syarat tersebut bisa terpenuhi, maka metode yang digunakan dalam penelitian ini layak digunakan sebagai metode untuk enkripsi. Tetapi jika tidak sesuai, maka diperoleh kesimpulan bahwa metode yang digunakan tidak layak. Sehingga bisa menjadi masukan bagi peneliti selanjutnya untuk mencari metode yang layak.

3.5.3 Analisis Terhadap Serangan *Bruteforce*

Untuk serangan *bruteforce* ini, dilakukan analisis bagaimana peluang sebuah *ciphertext* dikembalikan lagi nilainya ke *plaintext*. Akan dianalisis terhadap dua serangan, yaitu serangan terhadap *ciphertext* secara langsung (*ciphertext only attack*) dan serangan terhadap kunci (*key attack*).

Dari jumlah peluang yang didapatkan, akan dilakukan konversi jumlah peluang ke dalam satuan waktu. Dengan mengasumsikan bahwa seorang penyerang menggunakan sebuah super komputer yang bisa memproses 1000000 (satu juta proses per detik). Setelah dilakukan konversi terhadap waktu, akan bisa dilihat berapa kekuatan metode *quasigroup* untuk diserang dengan menggunakan serangan *bruteforce*.

3.5.4 Analisis Karakteristik *Key* dan Pengaruhnya Terhadap *output*

Pengujian dalam enkripsi dilakukan untuk mengetahui karakteristik *quasigroup* terhadap serangan/modifikasi dengan mencari persentase *avalanche effect*. Pengujian dilakukan hanya untuk mengetahui sensitifitas pesan terhadap modifikasi. Pemodifikasian dilakukan dengan melakukan pengurangan karakter kunci secara bertahap sampai dengan tersisa 1 karakter kunci. Penyajian pengujian *quasigroup* atau persentase *avalanche effect* dapat dilihat pada tabel 3.2 :

Tabel 3.2 Rancangan Tabel *avalanche effect* metode *quasigroup*

pengujian ke	*Avalanche effect* Ciphertext (%)	*Avalanche effect* Plaintext (%)

BAB IV
HASIL DAN PEMBAHASAN

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan dalam sub bab ini adalah lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1 Lingkungan Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan aplikasi enkripsi dengan metode *quasigroup* ini adalah:

1. Prosesor Intel Celeron 440 1.8 GHz
2. Memori 1 GB
3. Harddisk dengan kapasitas 60 GB
4. *Keyboard*
5. *Mouse*
6. Ponsel dengan support *Java Virtual Machine* MIDP 2.0

4.1.2 Lingkungan Implementasi Perangkat Lunak

Perangkat Lunak yang digunakan dalam pengembangan aplikasi enkripsi dengan metode *quasigroup* ini yaitu:

1. Sistem operasi yang digunakan adalah *Microsoft Windows XP Profesional*, *Linux Mint* 5.0, dan *Java Virtual Machine*.
2. Sistem dibuat dengan menggunakan perangkat lunak NetBeans 5.5.1 dengan bahasa pemrograman Java.

4.2 Hasil Dari Aplikasi Enkripsi *Quasigroup*

Sesuai dengan metodologi dan perancangan yang telah dibahas pada bab sebelumnya. Pada bagian ini akan dijelaskan mengenai hasil dan cara kerja dari aplikasi enkripsi yang telah dibuat.

4.2.1 Struktur Data

Struktur data yang diciptakan dalam aplikasi enkripsi dengan menggunakan metode *quasigroup* bisa dilihat pada gambar 4.1

```
public String msg;
public String key;
public static String ALPHABETS[] =
{"A","B","C","D","E","F","G","H","I","J
","K","L","M","N","O","P","Q","R","S","
T","U","V","W","X","Y","Z","0","1","2",
"3","4","5","6","7","8","9"};
```

Gambar 4.1 struktur data enkripsi *quasigroup*

Struktur data enkripsi *quasigroup* dijelaskan pada tabel 4.1 sebagai berikut:

Tabel 4.1 Penjelasan dari stuktur data

msg	Digunakan mengambil nilai inputan pesan
key	Digunakan mengambil nilai inputan key
ALPHABETS[]	Array dimana diletakkan karakter alfabet sehingga memenuhi *vigenere*

4.2.2 Implementasi Metode *Quasigroup*

Dalam pembuatan aplikasi enkripsi *quasigroup* ini, digunakan bujur sangkar *vigenere* sebagai komponen utama dalam pengimplementasian program. Hasil dari kombinasi antara *key* dengan *plaintext* dalam bujur sangkar *vigenere* akan menghasilkan sebuah nilai *ciphertext*. Oleh karena itu, implementasi dalam program, yang dibutuhkan adalah sebuah representasi dari *key* yang ada, tidak perlu membuat sebuah bujur sangkar terlebih dahulu. Gambar 4.2 merupakan implementasi dalam *source code* untuk menghasilkan *key* yang sesuai dengan bujur sangkar *vigenere*.

```
private String[][] getRows(String key)
{
String rows[][] = new String[key.length()][ALPHABETS.length];
   for(int i=0;i<rows.length;i++){
      for(int j=0;j<rows[i].length;j++){
        String[] row = getNextAlphabets(""+key.charAt(i));
           for(int k=0;k<row.length;k++){
                rows[i][k] = row[k];
           }
       }
   }
        return rows;
}
```

Gambar 4.2 Fungsi untuk menghasilkan *key* yang sesuai bujur sangkar

4.2.2 Penerapan Aplikasi Enkripsi *Quasigroup*

Untuk membuktikan apakah *ciphertext* yang dihasilkan sesuai dengan teori, selain melakukan percobaan dengan menggunakan teks pada gambar 4.1. Dicobakan pula pada teks lain yang bisa dilihat pada tabel 4.2.

4.2 Tabel Enkripsi (digunakan kunci yaitu 'rahasia')

Input Plaintext	Output Ciphertext	Sesuai / Tidak
A	R	sesuai
Abcde	RBJDW	sesuai
Abcde fghij	RBJDW NGYIQ	sesuai
Zyxwv 12987	GY4WD 92Q8E	sesuai

Dari tabel 4.1 dapat dilihat bahwa hasil output *ciphertext* sesuai dengan teori pada bujursangkar *vigenere* modifikasi. Setelah melakukan pengujian terhadap hasil *cipher*, dilakukan pula pengembalian nilai *ciphertext* ke *plaintext*(dekripsi), apakah hasil dekripsi sesuai dengan input *plaintext*. Hasil pengujian dapat dilihat pada tabel 4.3.

4.3 Tabel Dekripsi (digunakan kunci yaitu 'rahasia')

Input Ciphertext	Output Plaintext	Sesuai / Tidak
R	A	sesuai
Rbjdw	ABCDE	sesuai
Rbjdw ngyiq	ABCDE FGHIJ	sesuai
Gy4wd 92q8e	ZYXWV 12987	sesuai

Setelah dilakukan percobaan untuk melakukan dekripsi, didapatkan bahwa aplikasi yang dihasilkan berhasil melakukan proses dekripsi sesuai dengan teori yang ada. Dapat disimpulkan bahwa hasil proses enkripsi dan dekripsi bisa berjalan dengan baik karena menghasilkan output sesuai dengan teori.

Aplikasi yang dikembangkan, hasilnya adalah berupa file berekstensi *.jar(*executable jar*) dan *.jad(*java aplication descriptor*), yang umumnya bisa dibaca oleh Java Virtual Machine (JVM) pada ponsel dengan support Java MIDP 2.0. Ukuran dari aplikasi yang dihasilkan untuk file dengan ekstensi *.jar atau file eksekusi hanya sebesar 9 kb, dan file *.jad atau media deskripsi java hanya sebesar 1 kb. Dari kedua file yang dihasilkan tersebut, dapat dibilang cukup kecil untuk ukuran ponsel dan tidak terlalu membebani memori. Hal ini dibuktikan dengan data memori yang menunjukkan grafik penggunaan memori pada saat aplikasi pertama kali dijalankan. Bisa dilihat pada gambar 4.6

Gambar 4.3 Aplikasi untuk memonitor penggunaan memori aplikasi yang dihasilkan oleh J2ME

Dari aplikasi memori monitor ini, bisa dilihat bahwa aplikasi yang telah dibuat, hanya membutuhkan memori awal sebesar 97080/531148 atau sekitar 18% dari keseluruhan memori yang ada. Tampaklah bahwa aplikasi yang dihasilkan memang tidak terlalu membebani memori, jika diterapkan pada ponsel yang sebenarnya.

Sedangkan untuk kecepatan dari proses enkripsi dari metode *quasigroup* ini, tidak membutuhkan waktu yang lama. Dari hasil pengujian yang dilakukan baik secara simulasi maupun dengan ponsel yang sebenarnya, waktu yang dibutuhkan untuk melakukan enkripsi maupun dekripsi kurang dari 1 detik. Untuk analisis komputasi lebih lanjut dibahas pada sub-bab 4.3 dan 4.4.

4.3 Karakteristik Metode *Quasigroup*

Dalam menggunakan metode *quasigroup* ini, berpegang pada prinsip *quasigroup* yang bisa dilihat pada persamaan 2.2. Pengimplementasiannya bisa dilihat bujursangkar *Vigenere* pada gambar 3.4. Perhatikan bahwa pada gambar tersebut, setiap baris dan kolom pada bujursangkar tersebut merupakan permutasi dari alfabet A, B, C, ..., Z yang digabungkan dengan angka 0, 1,...,9. Hal ini merupakan karakteristik dari *Latin Square*. Dan seperti telah dibahas sebelumnya, sebuah *Latin Square* merupakan representasi dari suatu *quasigroup.*

Elemen dari *quasigroup* adalah alfabet dan operasinya ditentukan dengan cara yang sama seperti proses enkripsi tiap huruf pada bujursangkar *Vigenere*, yaitu menarik garis vertikal dari huruf plainteks dan garis huruf mendatar dari huruf kunci, perpotongannya adalah hasil operasi. Hasil pengujian terhadap *ciphertext* yang dihasilkan pada proses enkripsi adalah sebagai berikut.

Sebagai contoh,jika dilihat bujursangkar *vigenere* pada gambar 4.7, dapat dilihat bahwa $C * E = G$, $G * C = I$, $E * F = J$ dan seterusnya.Dengan lambang * merupakan representasi dari metode enkripsi yang digunakan. Maka secara matematis, proses enkripsi dinyatakan sebagai berikut

$$Ci = EK(Pi) = Ki * Pi$$

Dimana *Pi*, *Ci* dan *Ki* berturut-turut menyatakan karakter ke-*i* pada plainteks, cipherteks dan kunci. Dan seperti penjelasan sebelumnya, jika panjang kunci lebih kecil daripada panjang plainteks, maka kunci diperpanjang (agar sama panjang dengan plainteks) dengan cara mengulang-ulang kunci tersebut.

Proses dekripsi juga dapat dinyatakan secara matematis (dalam notasi *quasigroup*)

$$Pi = DK(Ci) = Ki \setminus Ci$$

Dengan lambang \ merupakan lambang inverse atau kebalikan dari enkripsi yang dilakukan. Karena operasi invers kiri pada *quasigroup* bersifat tunggal, maka hasil operasi *Ki* \ *Ci* juga pasti hanya satu kemungkinan. Tidak mungkin ada dua atau lebih kemungkinan huruf plainteks. Inilah yang dimaksud dengan ketunggalan invers pada *quasigroup* bermanfaat pada proses enkripsi dan dekripsi.

Untuk melihat apakah ada hubungan antara *plaintext, key* dan *ciphertext* dalam bujursangkar *vigenere* yang telah dimodifikasi ini, dilakukan pengujian terhadap tiap karakter yang dienkripsi dan didekripsi pada tabel 4.2 dan 4.3. Hasil dari pengujian tampak pada tabel 4.4 dan 4.5. Diasumsikan pada tabel *vigenere* modifikasi, posisi A adalah posisi ke-0.

Tabel 4.4 Hubungan *Plaintext, Key, dan Ciphertext* Pada Enkripsi (lihat tabel *vigenere* modifikasi)

Plaintext (posisi)	*Key* (posisi)	*Ciphertext* (posisi)
A (0)	R (17)	R (17)
B (1)	A (0)	B (1)
C (2)	H (7)	J (9)
D (3)	A (0)	D (3)
E (4)	S (18)	W (22)
Z (25)	**R (17)**	**G (6) -> ? (42)**
Y(24)	A (0)	Y (24)
X(23)	H (7)	4 (30)
V(21)	**S (18)**	**D (3) -> ? (39)**
1 (27)	I (8)	9 (35)
9 (35)	**R (17)**	**Q (16) -> ? (52)**

Tabel 4.5 Hubungan *Ciphertext*, *Key,* dan *Plaintext* Pada Dekripsi (lihat tabel *vigener* modifikasi)

Ciphertext	*Key*	*Plaintext*
R (17)	R (17)	A (0)
B (1)	A (0)	B (1)
J (9)	H (7)	C (2)
D (3)	A (0)	D (3)

W (22)	S (18)	E (4)
G (6)	**R (17)**	**Z (25) -> ?(-13)**
Y (24)	A (0)	Y(24)
4 (30)	H (7)	X(23)
D (3)	**S (18)**	**V(21) -> ?(-15)**
9 (35)	I (8)	1 (27)
Q (16)	**R (17)**	**9 (35) -> ? (-1)**

Dari tabel 4.4 yaitu tabel hubungan *plaintext*, *key* dan *ciphertext* pada metode enkripsi, dihasilkan bahwa posisi *plaintext* jika ditambahkan dengan posisi *key* akan menghasilkan posisi dari *ciphertext*. Tetapi keanehan muncul pada karakter Z(25) yang dikombinasikan dengan S(17), seharusnya hasil penjumlahan posisinya adalah posisi (42). Tetapi posisi yang muncul adalah posisi 6 yang dihuni oleh karakter G. Karakter *plaintext* lainnya seperti V dan 9 juga menghasilkan posisi *ciphertext* yang tidak sesuai dengan penjumlahan posisi. Dari sini muncul pertanyaan bagaimanakah posisi 42 menjadi posisi 6, posisi 39(kombinasi *plaintext* V, *key* S) menjadi posisi 3 dan posisi 52 menjadi posisi 16.

Dan setelah dipelajari lagi, disinilah letak dari prinsip *quasigroup*. Posisi yang ada di bujursangkar *vigenere* jika lebih dari jumlah karakter yang ada di bujursangkar, maka nilainya dikembalikan lagi, ke posisi awal. Hal ini menjawab pertanyaan bagaimanakah posisi yang seharusnya 42, menjadi posisi 6 yang dihuni karakter G. Dari sini dapat diartikan bahwa operasi * pada enkripsi *quasigroup* yang digunakan memiliki arti sebagai berikut.

$$Ci = EK(Pi) = Ki * Pi = Ki + Pi \bmod 36$$

Jadi dapat disimpulkan bahwa, proses enkripsi dari metode ini adalah hasil penjumlahan antara posisi *key* dengan posisi *plaintext* yang dibagi dengan mod 36. Dimana angka 36 sesuai dengan ukuran dari bujursangkar yaitu 36 x 36.

Sedangkan untuk dekripsi, tidak berbeda dengan proses enkripsi. Dengan mengacu pada tabel 4.5 dapat dilihat bahwa, formula untuk menentukan nilai *plaintext* adalah posisi *ciphertext* dikurangi posisi *key*. Atau bisa dituliskan sebagai.

$$Pi = DK(Ci) = Ki \setminus Ci = Ci - Ki \bmod 36$$

Sehingga dari sini dapat disimpulkan bahwa proses dekripsi, menggunakan pembalikan dengan cara mengurangkan antara posisi *ciphertext* dengan posisi *key* yang di bagi dengan mod 36.

4.4 Analisis Performa Metode Enkripsi *Quasigroup*

Untuk mengukur performa metode enkripsi *quasigroup* ini, adalah menggunakan waktu untuk melakukan proses enkripsi yang dibandingkan dengan jumlah karakter input *plaintext*. Hasil dari pengujian yang dilakukan tampak pada tabel 4.6.

Tabel 4.6 Waktu proses enkripsi

Inputan Karakter Plaintext	Waktu Proses(milidetik)
1	31
5	46
10	47
20	47
30	47
40	47
50	47
60	62
70	62
80	62
90	62
100	62
110	63
120	63
130	63
140	63
150	63
160	63

Dari tabel 4.6 terlihat bahwa waktu yang dibutuhkan untuk melakukan proses enkripsi dipengaruhi oleh jumlah karakter input tetapi tidak signifikan untuk tiap karakter. Kenaikan yang dialami menunjukkan bahwa waktu proses naik sedikit demi sedikit. Untuk

mengetahui kenapa waktu yang dibutuhkan dalam melakukan proses enkripsi naik dengan performa seperti tabel 4.6. Maka perlu dilakukan analisis *running time* dari hasil algoritma *quasigroup* ini. Dengan mengikuti langkah yang disarankan oleh Cormen dkk (2003) langkah pertama adalah melihat cara kerja dari algoritma, sehingga bisa menentukan sebuah parameter yang mengindikasikan sebuah input.

Cara kerja dan parameter yang dibutuhkan dari metode enkripsi *quasigroup* dapat dituliskan sebagai berikut.

- Metode ini digunakan untuk melakukan enkripsi teks sms yang telah dikombinasikan dengan key. Langkah pertama adalah membuat baris key yang ditunjukkan pada fungsi *getrows(key)*, setelah itu dicari posisi dari karakter *plaintext*, gabungan dari *key* dan *plaintext* ini akan menghasilkan posisi dari *ciphertext* yang ditunjukkan dengan rows[counter++][alphapos]. Metode enkripsi *quasigroup* bisa dilihat pada gambar 4.4
- Input : sekumpulan karakter pesan sebagai *plaintext* dan sekumpulan karakter *key*.
- Output : sekumpulan karakter hasil enkripsi sebagai *chipertext*.

```
    Enkripsi ( String msg, String key)
    {
1   String rows[][] = getRows(key)
2   for(int i=0;i<msg.length();i++){
3    if(!" ".equals(""+msg.charAt(i))){
4      String alpha = ""+msg.charAt(i);
5      int alphaPos = this.getAlphabetPosition(alpha,
    SMSkirim.ALFABET);
6      result = result+rows[counter++][alphaPos];
7    counter = counter == key.length?0:counter;
     }
8    else result+=" ";
    }
```

Gambar 4.4 metode enkripsi *quasigroup*

Setelah diketahui parameter yang dibutuhkan, dilanjutkan pada langkah kedua yaitu mengidentifikasi operasi dasar yang dilakukan oleh sebuah algoritma. Dari gambar 4.8 dapat dilihat bahwa terdapat 8 operasi yang dilakukan oleh metode *quasigroup*.

Langkah ketiga adalah memeriksa apakah jumlah waktu untuk melakukan tiap operasi dasar hanya bergantung pada ukuran input. Dari gambar 4.8 dapat dilihat bahwa metode ini hanya bergantung pada input yaitu *plaintext* dan *key*. Karena metode ini hanya bergantung pada input, maka tidak diperlukan analisis *worst case, best case* dan *average case* secara terpisah.

Langkah keempat adalah disusun penjumlahan yang mengekspresikan berapa kali operasi dasar dilakukan pada algoritma. Pada langkah keempat ini, dilakukan analisis sebagai berikut.

- Ukuran input : m,n
- Unit waktu eksekusi : cost c (konstanta)

Tabel 4.7 setup jumlah waktu tiap operasi dasar enkripsi *quasigroup*

Enkripsi(String msg, String Key)	**Cost**	**Waktu**
1 **Rows[][] = getRows(key)**	C1	m
2 **for(int i=0;i<msg.length(); i++)**	C2	n
3 {**if(!"".equals(""+msg.charAt(i)))**	C3	n-1
4 **String alpha = ""+ msg.charAt(i);**	C4	n-1
5 **int alphaPos = this.getAlphabetPosition(alpha, SMSkirim.ALFABET);**	C5	n-1
6 **result = result+rows[counter++][alphaPos];**	C6	n-1
7 **counter = counter == key.length?0:counter;**	C7	n-1
8**}else result+=" "; }**	C8	n-1

Dari tabel 4.7 dapat dilihat bahwa setelah dilakukan operasi for, nilai waktu operasi lainnya berubah menjadi n-1, hal ini dikarenakan operasi for hanya dilakukan mulai dari 0 sampai kurang dari panjang pesan asli, sehingga bisa waktu yang digunakan pun berkurang 1 angka. Karena for menaungi beberapa operasi di bawahnya, secara otomatis waktu operasi lainnya juga mengikuti waktu dari operasi for.

Langkah terakhir untuk menentukan *running time* dari metode enkripsi *quasigroup* ini adalah menggunakan formula standard dan aturan penjumlahan dalam mengidentifikasi *order of growth*. Langkah kelima ini dituliskan sebagai berikut. *running time* dari metode ini adalah

$$T(n) = c1m + c2n + c3(n-1) + c4(n-1) + c5(n-1) + c6(n-1) + c7(n-1)+c8(n-1)$$
$$= c1m+(c2+c3+c4+c5+c6+c7+c8)n - (c3+c4+c5+ c6+ c7+c8)$$

running time tersebut bisa diekspresikan dengan persamaan : **am + bn - c** (a, b, c konstanta)

dari persamaan yang telah ditentukan dengan menggunakan formula standar, ditemukan bahwa persamaan yang dihasilkan *dominant term* adalah persamaan linear. Sehingga dari sini dapat disimpulkan bahwa *running time* dari enkripsi *quasigroup* adalah $\theta(n)$.

Hal ini tentunya sesuai dengan hasil percobaan bahwa *running time* yang dihasilkan memiliki kenaikan yang tidak signifikan tetapi naik sedikit demi sedikit berdasarkan jumlah inputan. Tetapi meski hasil analisis *running time* menunjukkan persamaan linear, waktu proses enkripsi yang dibutuhkan tetap relatif sangat singkat untuk inputan maksimum 160 karakter hanya dibutuhkan 63 milidetik. Sehingga hal ini mendukung kelayakan digunakan sebagai metode untuk enkripsi di SMS.

4.5 Analisis Performa Metode Dekripsi *Quasigroup*

Setelah dilakukan analisis performa dari metode enkripsi, dilakukan pula analisis performa untuk metode dekripsi. Untuk mengukur performa metode dekripsi *quasigroup* ini, sama dengan metode enkripsi adalah menggunakan waktu untuk melakukan proses enkripsi yang dibandingkan dengan jumlah karakter input *plaintext*. Hasil dari pengujian yang dilakukan tampak pada tabel 4.8.

Tabel 4.8 Waktu proses dekripsi

Inputan Karakter Plaintext	Waktu Proses(milidetik)
1	31
5	46

10	47
20	47
30	47
40	47
50	47
60	62
70	62
80	62
90	62
100	62
110	63
120	63
130	63
140	63
150	63
160	63

Dari tabel 4.8 terlihat bahwa waktu yang dibutuhkan untuk melakukan proses dekripsi hampir sama dengan waktu yang dibutuhkan untuk proses enkripsi, dan dipengaruhi oleh jumlah karakter input meski tidak terlalu signifikan. Waktu yang dilakukan untuk melakukan proses dekripsi memiliki rentang antara 31 – 63 milidetik atau jika dibulatkan ke bawah maka waktu proses tidak sampai 0.1 detik. Secara sekilas tampak bahwa metode dekripsi ini mirip dengan metode enkripsi. Tetapi untuk membuktikan apakah keduanya memiliki kesamaan dalam hal pemrosesan, maka akan dilakukan juga analisis performa untuk metode dekripsi *quasigroup*.

Langkah untuk melakukan analisis performa, tetap menggunakan langkah-langkah yang disarankan oleh Cormen dkk (2003). Langkah pertama adalah melihat cara kerja dari algoritma, sehingga bisa menentukan sebuah parameter yang mengindikasikan sebuah input. Cara kerja dan parameter dari metode dekripsi *quasigroup* dituliskan sebagai berikut:

- Metode ini digunakan untuk melakukan dekripsi , dengan langkah posisi *chipertext* ditarik garis lurus dengan *key* sehingga bisa mengembalikan nilai dari *plaintext* ke nilai awal sehingga bisa dibaca oleh pengguna. Hal ini tampak pada result += ALPHABETS[pos].

- Input : sekumpulan teks *chipertext* dan sekumpulan *key*
- Output : sekumpulan teks *plaintext*

```
    Decrypt(String msg,String key)
    {
1     String[][] rows = this.getRows(key);
2     for(int i=0;i<msg.length();i++){
3           if(!" ".equals(""+msg.charAt(i))){
4       int pos = this.getAlphabetPosition(""+msg.charAt(i),
      rows[counter++]);
5     result+= ALPHABETS[pos];
6     counter = counter == key.length?0:counter;
7   } else  result+=" ";
```

Gambar 4.5 metode dekripsi *quasigroup*

Setelah diketahui parameter yang dibutuhkan, dilanjutkan pada langkah kedua yaitu mengidentifikasi operasi dasar yang dilakukan oleh sebuah algoritma. Dari gambar 4.9 dapat dilihat bahwa operasi yang dilakukan pada metode ini lebih sedikit dari metode enkripsi, hanya terdapat 7 operasi yang dilakukan oleh metode dekripsi *quasigroup*.

Langkah ketiga adalah memeriksa apakah jumlah waktu untuk melakukan tiap operasi dasar hanya bergantung pada ukuran input. Dari gambar 4.9 dapat dilihat bahwa metode ini hanya bergantung pada input yaitu *ciphertext* dan *key*. Karena metode ini hanya bergantung pada input, maka tidak diperlukan analisis *worst case, best case* dan *average case* secara terpisah.

Langkah keempat adalah melakukan setup yang mengekspresikan jumlah waktu untuk tiap operasi dasar algoritma yang dieksekusi. Pada langkah keempat ini, dilakukan analisis sebagai berikut.

- ukuran input : m, n
- unit waktu eksekusi : cost c

Tabel 4.9 setup jumlah waktu operasi dasar dari metode dekripsi *quasigroup*

Decrypt(String msg,String key)	Cost	Waktu
1 String[][] rows = this.getRows (key);	C1	m
2 for(int i=0;i<msg.length(); i++){	C2	n
3 if(!" ".equals(""+msg.charAt(i))){	C3	n-1

4 int pos = this.getAlphabetPosition(""+ msg.charAt(i), rows[counter++]);	**C4**	**n-1**
5 result+= ALPHABETS[pos];	**C5**	**n-1**
6 counter = counter == key.length? 0:counter;	**C6**	**n-1**
7 } else result+=" ";	**C7**	**n-1**

Dari tabel 4.9 dapat dilihat bahwa setelah dilakukan operasi for, nilai waktu operasi lainnya berubah menjadi n-1, hal ini dikarenakan operasi for hanya dilakukan mulai dari 0 sampai kurang dari panjang pesan asli, sehingga bisa waktu yang digunakan pun berkurang 1 angka. Karena for menaungi beberapa operasi di bawahnya, secara otomatis waktu operasi lainnya juga mengikuti waktu dari operasi for.

Langkah terakhir untuk menentukan *running time* dari metode enkripsi *quasigroup* ini adalah menggunakan formula standard dan aturan penjumlahan dalam mengidentifikasi *order of growth*. Langkah kelima ini dituliskan sebagai berikut. *running time* dari metode ini adalah

$$T(n) = c1m + c2n + c3(n-1) + c4(n-1) + c5(n-1) + c6(n-1) + c7(n-1)$$
$$= c1m+(c2+c3+c4+c5+c6+c7)n - (c3+c4+c5+ c6+ c7)$$

running time tsb bisa diekspresikan dengan persamaan :

am + bn - c (a,b,c konstanta)

dari persamaan yang telah ditentukan dengan menggunakan formula standar, ditemukan bahwa persamaan yang dihasilkan adalah persamaan linear. Sehingga dari sini dapat disimpulkan bahwa *running time* dari dekripsi *quasigroup* adalah $\theta(n)$.

Hal ini mirip dengan proses dekripsi, bahwa hasil dari fakta yang ada dengan analisis adalah sama. Tetapi dari sini terbukti bahwa proses enkripsi dan dekripsi dari metode *quasigroup* memiliki kesamaan dalam hal *running time* $\theta(n)$. Sehingga hal ini menjawab kenapa keduanya membutuhkan waktu proses yang hampir sama. Dari pembuktian ini, tentunya lebih memperkuat kenapa metode ini layak digunakan sebagai media enkripsi pada SMS.

4.6 Karakteristik *Key* dan Pengaruhnya Terhadap *Output* (*Avalanche Effect*)

Seperti yang ditulis pada bab 3.5.4 tentang pengujian enkripsi *quasigroup*, pengujian dilakukan sebanyak 15 kali dengan kunci sama tetapi panjangnya secara bertahap dikurangi satu karakter dari belakang. Dalam hal ini digunakan kunci 'rahasianegaraku'. Pengujian dibatasi untuk mengetahui sensifitas dari metode *quasigroup* terhadap perubahan. Pengujian untuk mengetahui pengaruh terhadap modifikasi kunci dilakukan dengan melihat pengaruh panjang kunci terhadap *avalanche effect* yang dihasilkan oleh ciphertext dan plaintext.

Untuk pengujian *ciphertext,* hasil *ciphertext* awal digunakan sebagai pembanding utama, untuk melihat perbedaan antara *ciphertext* awal dengan *cipehertext* yang dihasilkan dari pengurangan panjang kunci

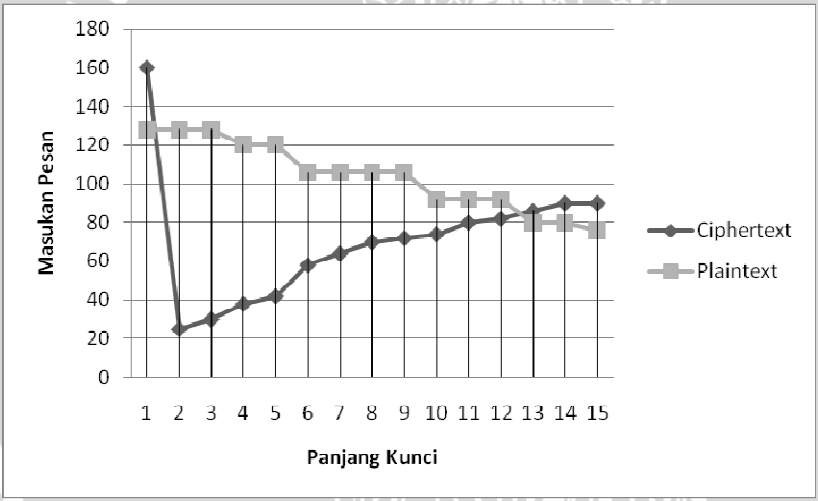
Sedangkan uji coba kedua adalah melihat peubahan antara *ciphertext* yang dihasilkan dari pengurangan panjang kunci terhadap *plaintext*. Tabel uji *avalanche effect* enkripsi *quasigroup* dapat dilihat pada tabel 4.10.

Tabel 4.10 Tabel uji *avalanche effect* berdasarkan panjang kunci

		Ciphertext		Plaintext	
Panjang Plaintext	Panjang Kunci	Beda	%	Beda	%
160	15	160	100	128	80
160	14	25	15.625	128	80
160	13	30	18.75	128	80
160	12	38	23.75	120	75
160	11	42	26.25	120	75
160	10	58	36.25	106	66.25
160	9	64	40	106	66.25
160	8	70	43.75	106	66.25
160	7	72	45	106	66.25
160	6	74	46.25	92	57.5
160	5	80	50	92	57.5
160	4	82	51.25	92	57.5

160	3	86	53.75	80	50
160	2	90	56.25	80	50
160	1	90	56.25	76	47.5

Dari hasil uji coba dapat dilihat bahwa panjang kunci sangat berpengaruh pada *ciphertext* yang dihasilkan. Dapat dilihat pada tabel, tampak bahwa panjang kunci 15 menghasilkan persentase beda sebesar 80%. Hal ini berarti bahwa dengan panjang kunci 15, maka karakter yang berubah adalah sebesar 80% dari karakter *plaintext*, sisanya sama dengan karakter *plaintext* sebenarnya. Sedangkan jika panjang kunci semakin kecil, misalnya hanya diisi 4 karakter. Maka persentase beda *ciphertext* yang dihasilkan semakin kecil pula hanya mencapai 57.5%. Untuk lebih jelasnya, dapat dilihat pada gambar 4.5



Gambar 4.5 grafik hubungan kunci dengan *avalanche effect*

Dari gambar 4.5 dapat dilihat bahwa untuk beda antar *ciphertext* didapatkan bahwa jika jarak panjang kunci semakin besar, maka semakin besar pula beda *ciphertext* yang dihasilkan. Dan sebaliknya jika jarak panjang kunci semakin besar, maka semakin kecil beda *plaintext* yang dihasilkan.

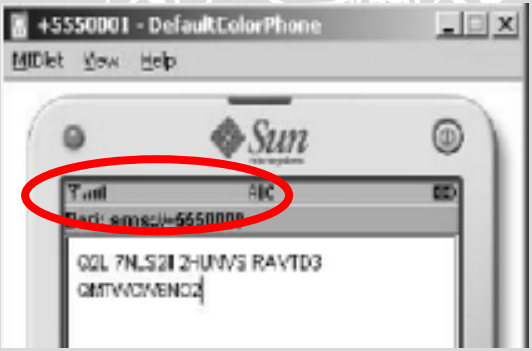
4.7 Fitur Yang Dihasilkan

Hasil dari pengembangan program pada J2ME telah memenuhi syarat sms pada umumnya sesuai dengan pernyataan Job De Haas (2001) yaitu adanya :

1. *message submission* atau pesan yang diketik ditujukan kemana. Dalam hal ini adalah nomor dari ponsel lain yang terletak pada simulator. Bisa dilihat pada gambar 4.6

Gambar 4.6 *message submission* dari aplikasi yang telah dibuat

2. *reply path* atau ketika seorang pengguna menerima sms, ditampilkan darimana sms tersebut berasal. Bisa dilihat pada gambar 4.7



Gambar 4.7 PONSEL dengan nomor +5550001 menerima sms dari nomor +5550000 (diberi lingkaran)

3. *addressing mode*, sudah disediakan oleh J2ME sehingga ketika no tujuan diketik, maka secara otomatis akan dikirim ke no tujuan tersebut.

4.8 Bekerja di Layer

Dalam pengembangan aplikasi enkripsi sms ini, pertama-tama dikerjakan pada layer applikasi. Kemudian setelah simulator dijalankan, program yang dikembangkan ini berada di keempat layer yang ada, yaitu layer aplikasi, layer transfer, layer relay dan layer link. Berikut merupakan penjelasan dari layer-layer yang dikerjakan.

4.8.1 Layer aplikasi

Pada layer aplikasi ini dituliskan source code dengan menggunakan bahasa pemrograman java. Setelah dilakukan kompilasi terhadap source code, file yang dihasilkan kemudian diaplikasikan baik melalui simulator maupun ponsel yang sebenarnya.

4.8.2 Layer transfer

Pada layer transfer dalam J2ME terdapat sebuah fungsi yang diambil dari class *javax.wireless.messaging* sehingga bisa diambil teks yang dikirimkan melalui metode *getpayloadtext()* dan *setpayloadtext(),* yang berfungsi untuk mengeset pesan yang akan dikirimkan. Setelah dikirimkan penerima sms mengaktifkan metode *getpayloadtext()* untuk mendapatkan pesan yang dikirimkan tadi. Dengan pengaktifan metode *getpayloadtext()* inilah layer transfer bekerja. Pesan yang dikirimkan adalah pesan yang telah dienkripsi, sesuai dengan tujuan yaitu membuat sebuah pesan tidak bisa dibaca oleh operator. Sehingga kerahasiaan isi sms hanya bisa diketahui oleh pengirim dan penerima.

4.8.3 Layer relay

Untuk layer relay ini, ditangani juga oleh class *javax.wireless.messaging*, dalam hal ini digunakan *interface message connection* dengan metode *send*() dan *receive*(). Metode *send()* digunakan untuk pengiriman pesan, sedangkan *receive*() untuk menerima pesan yang dikirimkan.

4.8.4 Layer link

Layer link telah disediakan oleh simulator J2ME, dimana J2ME telah membuat sebuah simulasi jaringan yang bisa menghubungkan antara 2 atau lebih ponsel sehingga bisa saling terhubung.

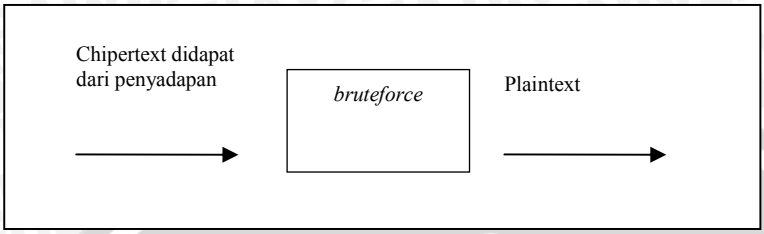
4.9 Analisis Kekuatan dari Metode *Quasigroup*

Pada bagian ini, akan coba dibahas mengenai kekuatan keamanan metode kriptografi *Quasigroup* yang digunakan dengan sebuah metode serangan yang umum digunakan oleh orang awam. Dikarenakan tidak adanya suatu teknik yang digunakan untuk menguji metode enkripsi adalah aman secara mutlak, maka akan dilakukan pendekatan untuk melihat apakah orang lain dapat berfikir atau menemukan suatu jalan untuk menghancurkan metode kriptografi ini. Dalam sub bab ini, akan dianalisis dua buah kemungkinan penyerangan terhadap metode *quasigroup* yaitu pertama penyerang mencoba untuk membuka *chipertext* dengan melakukan metode *brute force* atau dengan kata lain seorang penyerang mencoba-coba setiap peluang, sampai diketemukan padanan kata yang sesuai dengan *chipertext*.

Kemungkinan kedua, penyerang menyerang kunci dengan melakukan metode *bruteforce* atau dia melakukan penyerangan dengan mencoba setiap peluang kunci yang digunakan sehingga menghasilkan *chipertext* yang dianggap sesuai *plaintext*.

4.9.1 Serangan Kepada Chipertext dengan Bruteforce (*Chipertext Only Attack*)

Analisis terhadap kemungkinan serangan pertama adalah serangan terhadap chipertext yang dihasilkan dari enkripsi *quasigroup*. Serangan ini sangat mungkin dilakukan oleh pihak-pihak yang tidak bisa menyentuh secara langsung ponsel korban seperti operator jaringan telekomunikasi. Jadi penyerang mendapatkan teks yang dikirim dengan melakukan penyadapan, sehingga penyerang bisa bisa memperoleh pesan yang dikirimkan. Skema dari serangan ini dapat digambarkan seperti tampak pada gambar 4.8.



Gambar 4.8 Serangan *bruteforce* terhadap chipertext

Untuk melakukan serangan bruteforce terhadap chipertext ini, seorang penyerang akan melakukan percobaan terhadap setiap karakter sehingga ditemukan padanan kata yang sesuai. Waktu yang diperlukan untuk memperoleh *plaintext* yang diharapkan selalu berbanding lurus dengan panjang kunci yang dimiliki oleh metode *quasigroup*. Dan makin cepat prosesor yang dipakai untuk melakukan serangan ini, makin cepat pula waktu yang dibutuhkan untuk memperoleh *plaintext* tersebut.

Dari hal tersebut diperkirakan seorang penyerang akan memiliki waktu percobaan seperti tabel berikut:

Table 4.10 tabel perkiraan peluang serangan *bruteforce*

Karakter Chipertext	Peluang	Total Peluang
1	36	36
2	36 x 36	1296
3	36 x 36 x 36	46656
…		
160	36^{160}	1.1×10^{249}

Tabel 4.10 menunjukkan bahwa jika seseorang melakukan serangan *bruteforce* jumlah ciphertext berbanding lurus dengan peluang yang diperlukan, tampak bahwa seorang penyerang penyerang memiliki 36^n peluang, dimana n adalah jumlah karakter *ciphertext*.

Untuk membuktikan peluang ini, dilakukan uji coba dengan menggunakan program *bruteforce* . Program ini digunakan untuk menghitung waktu untuk melakukan penyerangan terhadap kunci yang dicari. Dalam menggunakan program ini, diujikan pada komputer prosesor Intel Pentium 4 dengan kecepatan prosesor sesuai dengan data yang diperoleh dari

(http://id.wikipedia.org/wiki/Instruksi_per_detik) sebesar 9726 MIPS (*Million instruction per second*). Dengan bantuan program ini didapatkan waktu untuk melakukan penyerangan *bruteforce* seperti pada tabel 10.

Tabel 10. Dilakukan uji coba dengan *password calculator*

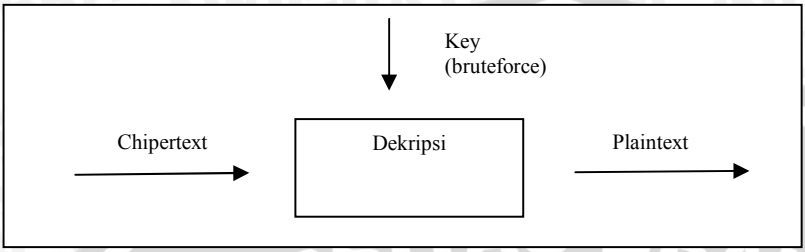
Jumlah karakter kunci	Kecepatan (MIPS)	Jumlah komputer	Waktu yang dibutuhkan
1	9726	1	1 menit
2	9726	1	1 menit
3	9726	1	1 menit
4	9726	1	1 menit
5	9726	1	1 menit
6	9726	1	2 menit
7	9726	1	14 menit
8	9726	1	9 jam
9	9726	1	13 hari
10	9726	1	15 bulan
11	9726	1	44 tahun

Dari hasil tersebut dapat dilihat bahwa untuk mendapatkan kunci yang sesuai sehingga diperoleh kunci yang sesuai dengan metode *quasigroup* tidak semudah yang diperkirakan. Semakin panjang kunci yang dimasukkan maka akan semakin lama pula waktu yang dibutuhkan untuk menemukan kunci.

4.9.2 Serangan Kepada Key dengan Bruteforce (*Key Attack*)

Analisis kedua adalah seorang penyerang melakukan penyerangan dengan menggunakan *bruteforce* terhadap key untuk menemukan plaintext yang sesuai. Serangan ini kemungkinan dilakukan oleh pihak-pihak yang bisa menyentuh secara langsung fisik ponsel dari korban seperti saudara, teman atau copet. Setelah mendapatkan ponsel, penyerang yang membaca *chipertext* bisa melakukan penyerangan *bruteforce* dengan mencoba-coba *key* dan melihat hasil *plaintext*.

Skema dari serangan ini digambarkan pada gambar 4.9



Gambar 4.9 serangan *bruteforce* terhadap key

Dengan menggunakan serangan ini, terdapat dua kemungkinan, yaitu kemungkinan terbaik dan terburuk. Untuk kemungkinan terbaik adalah seorang penyerang sudah mengetahui panjang kunci sedangkan kemungkinan terburuk seorang penyerang tidak mengetahui panjang kunci.

Untuk kemungkinan terbaik, jika seorang penyerang telah mengetahui panjang dari key yang ada, maka peluang penyerang memperoleh key yang sesuai adalah berlaku hubungan 36^n, dengan n adalah nilai panjang kunci.Tetapi jika tidak diketahui panjang key, maka percobaan maksimal untuk mendapatkan cipherteks adalah 32 karakter. Karena diberikan batasan key hanya sampai 32 karakter sehingga peluang seorang penyerang bruteforce adalah 36 pangkat 32 atau memiliki peluang sekitar 4.5×10^{34}

4.10 Kelayakan Metode Enkripsi *Quasigroup* Untuk Digunakan Pada Ponsel

Pada jaringan selular, lebar *bandwith* sangat terbatas sehingga metode enkripsi selain harus memenuhi standar keamanan, juga harus menjaga agar hasil file enkripsi tetap kecil. Oleh karena keterbatasan tersebut, solusi yang ditawarkan untuk enkripsi SMS adalah kerahasiaan algoritmanya.

Dari hasil enkripsi SMS dengan menggunakan metode *quasigroup*, metode enkripsi yang digunakan sudah memenuhi keempat syarat metode enkripsi yang baik untuk SMS seperti disebutkan oleh Bruce Scheneir(2003) yaitu:

1. Kerahasiaan

Plaintext yang dihasilkan oleh metode ini berupa tulisan acak yang tidak bisa dibaca oleh orang lain.

2. Autentikasi

   Karena menggunakan media ponsel, sesuai dengan fitur dasar yang dimiliki, autentikasi bisa dilihat pada fitur *reply path* dimana nomor pengirim bisa terlihat pada layar penerima.

3. Integritas

   Dengan adanya enkripsi *quasigroup* ini, seorang pengirim sms tidak perlu merasa was-was bahwa pesan yang dikirim akan dibaca oleh pihak yang tidak berhak seperti operator jaringan atau lainnya. Karena *plaintext* yang dihasilkan sudah diacak sesuai dengan bujursangkar *Vigenere*, dan untuk mengembalikan nilai *plaintext* dibutuhkan *key* yang sesuai.

4. Non-Repudiation

   Sesuai dengan fitur yang ada di sms, maka seorang pengirim tidak akan menyangkal sms yang telah dikirimnya.

Sehingga karena sudah memenuhi keempat syarat tersebut, dapat dikatakan bahwa metode kriptografi *quasigroup* sudah layak digunakan sebagai metode enkripsi yang baik untuk SMS.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah melakukan kegiatan pembuatan aplikasi enkripsi SMS dan analisis terhadap metode *quasigroup*, didapatkan kesimpulan sebagai berikut:

1. metode *quasigroup* bisa berjalan sesuai hasil yang diharapkan dengan menggunakan program J2ME dan sudah diujikan pada ponsel maupun melalui simulasi komputer.
2. Setelah dilakukan analisis mengenai kekuatan terhadap serangan *bruteforce ciphertext only attack* dan *key only attack*, diperoleh bahwa peluang untuk mengubah *ciphertext* yaitu 1.1×10^{249}, sedangkan untuk *key attack* peluang untuk mendapatkan kunci adalah $4.5 \times 10^{34.}$
3. Karakteristik kunci dari metode *quasigroup* sangat bergantung pada panjang kunci dan kemiripan dengan kunci sebenarnya. Semakin mirip kunci palsu dengan kunci sebenarnya, maka hasil dari dekripsi akan semakin bisa dimengerti.
4. Setelah dilakukan analisis mengenai running time, karakteristik dan syarat kelayakan terhadap sebuah metode enkripsi SMS. Diperoleh kesimpulan bahwa metode *quasigroup* dianggap layak sebagai metode untuk melakukan enkripsi terhadap SMS. Kelebihan dari metode *quasigroup* ini terletak pada kecepatan proses dan *chipertext* yang dihasilkan. Dari pengujian yang dilakukan pada inputan *plaintext* SMS maksimal(160 karakter), waktu yang diperlukan untuk melakukan enkripsi atau dekripsi tidak lebih dari 1 detik dan hasil *chipertext* juga tidak berlebih sesuai dengan *plaintext* yang diinputkan.

5.2 Saran

Untuk kegiatan lebih lanjut dari kegiatan yang telah dilakukan, saran yang mungkin bisa membantu kelanjutan karya ini:

1. Pengembangan metode *quasigroup* dengan penghilangan sifat komutatif, atau dengan kata lain bujur sangkar vigenere yang dihasilkan tidak menggunakan *latin square*. Jadi bujur sangkar *vigenere* berbentuk acak sehingga hal ini akan menambah kesulitan seorang penyerang untuk mengubah *chipertext* menjadi *plaintext* yang sesuai.
2. Karakter yang ada masih terbatas pada huruf dan angka, ke depan disarankan supaya mensupport huruf, angka, tanda baca dan simbol-simbol sehingga benar-benar lebih bisa diaplikasikan ke dalam dunia nyata.

DAFTAR PUSTAKA

Cormen, dkk. 2003. *Introduction To Algorithm 2*. McGraw Hill : New York

De Has, Job. 2001. *Mobile Security : SMS and WAP nada.* (http://www.itsec.gov.cn/webportal/download.ppt)

Good, Robin. 2002. *SMS Offer No Guarantee of Privacy*. (http://www3.gartner.com/resources/111700/111720/111720.pdf)

Hassinen, Marko.2003.*Secure SMS Messaging Using Quasigroup Encryption and Java SMS API.* Kuopio University : Kuopio.

Mic, Chocolave.2003. *Pengenalan J2ME*. (http://ilmukomputer.com)

Ortiz, Enrique. 2002. *The Wireless Messaging API*. (http://developers.sun.com/mobility/index.html)

Pankratrov, Denis. 2004. *SMS Spoofing*. Brown University : West Bromwich.

Prihatini, Ekawati.2006. *Aspek Keamanan SMS*. ITB : Bandung.

Scheiner, Bruce. 2003. *Applied Crytptography*. Prentice Hall: New York.

Setiana, 2006. *Steganografi dengan menggunakan algoritma Least Significant Bit*(*LSB)*. Universitas Brawijaya : Malang.

Prihatini, Ekawati.2006. *Aspek Keamanan SMS Pada Jalur Komunikasi Short Message Service*. ITB : Bandung.

Wojdylo, Jerzy. 2001. *Cryptography*. Waterloo University : Canada

Yuliawan, Fajar. 2003. *Studi Mengenai Aplikasi Teori Quasigroup dalam Kriptografi*. ITB : Bandung.