KOMPRESI SHORT MESSAGE SERVICE MENGGUNAKAN ALGORITMA ASYMMETRIC HALF BYTE

SKRIPSI

Untuk memenuhi sebagian persyaratan untuk mencapai gelar Sarjana Komputer

Disusun Oleh:

HERMANSYAH ISFANHANANI

0610963021

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
UNIVERSITAS BRAWIJAYA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
MALANG
2013

LEMBAR PERSETUJUAN

KOMPRESI SHORT MESSAGE SERVICE
MENGGUNAKAN ALGORITMA ASYMMETRIC HALF BYTE

SKRIPSI

Untuk memenuhi sebagian persyaratan untuk

mencapai gelar Sarjana Komputer

Disusun Oleh:

HERMANSYAH ISFANHANANI

0610963021

Telah diperiksa dan disetujui oleh

Dosen Pembimbing

Pembimbing I	Pembimbing II
Imam Cholissodin, S.Si., M.Kom.	**Novanto Yudistira, S.Kom., M.Sc.**
NIP. 850719 16 1 1 0422	**NIP. 831110 16 1 1 0425**

LEMBAR PENGESAHAN

KOMPRESI SHORT MESSAGE SERVICE

MENGGUNAKAN ALGORTIMA ASYMMETRIC HALF BYTE

SKRIPSI

Diajukan untuk memenuhi persyaratan memperolah gelar Sarjana Komputer

Disusun oleh :

HERMANSYAH ISFANHANANI

NIM. 0610963021

Skripsi ini telah diuji dan dinyatakan lulus tanggal 23 Agustus 2013

Penguji I	Penguji II
Edy Santoso, S.Si., M.Kom	**Ahmad Afif Supianto, S.Si., M.Kom**
NIP. 197404142003121004	**NIP. 82062316110425**

Penguji III

Issa Arwani, S.Kom., M.Sc
NIP. 83092206110074

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, MT
NIP. 196708011992031001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan dafta rpustaka.

Apabila ternyata di dalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 danPasal 70.

Malang, 20 Agustus 2013
Mahasiswa,

Hermansyah Isfanhanani
NIM. 0610963021

ABSTRAK

Kompresi SMS merupakan sebuah proses yang dilakukan dengan tujuan agar dapat meningkatkan jumlah karakter yang dapat dikirimkan dalam satu kali pengiriman SMS. Saat ini untuk satu kali pengiriman SMS maksimal dapat dituliskan 160 karakter. Jika jumlah karakter yang ditulis melebihi batas tersebut, maka biaya yang diperlukan juga akan bertambah karena pengiriman SMS harus dilakukan lebih dari satu kali. Penghematan biaya pengiriman SMS ini dapat dilakukan dengan cara melakukan kompresi pada SMS sebelum dikirim.

Salah satu metode kompresi yang sering digunakan terutama pada teks file teks adalah metode Half Byte. Metode ini memanfaatkan kesamaan 4 bit pertama yang sering sama pada teks. Bit-bit yang sama tersebut dapat dihilangkan sehingga akan mengurangi jumlah bit dari karakter aslinya. Asymmetric Half Byte merupakan metode kompresi yang dilakukan dengan memodifikasi algoritma *Half Byte* dimana jumlah bit kiri sebagai bit kunci diambil 3 bit, sehingga nantinya diharapkan akan semakin banyak deretan bit yang memiliki persamaan pada 3 bit awalnya yang secara otomatis menyebabkan deretan bit yang dapat dikompresi menjadi lebih banyak.

Berdasarkan Pengujian yang telah dilakukan menggunakan 20 data uji, diperoleh nilai rata – rata rasio kompresi SMS yang menggunakan metode Half Byte adalah sebesar 3,9%. Sedangkan nilai rata – rata rasio kompresi dengan menggunakan Asymmetric Half Byte sebesar 16,47%. Jika ditinjau dari pengiriman SMS dengan rasio tersebut, maka dapat diperoleh rata-rata jumlah karakter yang dapat dituliskan dalam satu kali pengiriman SMS sebanyak 186 karakter.

Kata Kunci : Kompresi, SMS, *Asymmetric Half Byte*

ABSTRACT

SMS compression is a process for increasing the number of characters that can be sent in one SMS delivery. Nowadays for one SMS, the maximum character that can be written is 160 characters. If the number of characters written exceeds this limit, then the costs will also increase because of sending SMS will be done more than once. SMS delivery cost savings can be done by doing compression before sending the SMS.

One of the commonly used methods of compression especially for text files is Half Byte method. This method utilizes the same first 4 bits that often equal on the text. These same bits can be removed so that it will reduce the number of bits from the original character. Asymmetric Half Byte is a compression method that is done by modifying the Half Byte algorithm where the number of left bits as key bits is taken 3 bits, so it is expected there will be more rows of bits that have the same first 3 bits that automatically causes more bits that can be compressed.

Based on the testing that has been performed using 20 test data, the average compression ratio of SMS obtained using Half Byte method is 3.9%. While the average compression ratio using Asymmetric Half Byte method is 16.47%. If sending an SMS based on this ratio, the average number of characters that can be written in a single SMS sending is 186 characters.

Keyword ***:*** *Compression, SMS, Asymmetric Half Byte*

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpah Bytan hidayah-Nya, Skripsi yang berjudul **"Kompresi Short Message Service Menggunakan Algoritma Asymmetric Half Byte"** ini dapat berjalan dengan baik. Skripsi ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, Program Teknologi Informatika dan Ilmu Komputer (PTIIK), Universitas Brawijaya Malang.

Shalawat serta salam tetap tercurahkan kepada Baginda Rasulullah Muhammad *Shalallahu 'alaihi wasallam*, makhluk paling mulia yang senantiasa memberikan cahaya petunjuk, seorang uswatun hasanah yang telah membawa agama Allah yaitu agama Islam menjadi agama yang *Rahmatan Lil 'Alamin*.

Dalam penyelesaian skripsi ini, penulis telah mendapat begitu banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Imam Cholissodin, S.Si., M.Kom., selaku dosen pembimbing pertama yang telah meluangkan waktu untuk memberikan pengarahan dan masukan bagi penulis.
2. Novanto Yudistira,S.Kom., M.Sc., selaku dosen pembimbing kedua yang telah memberikan bimbingan dalam penulisan laporan.
3. Drs. Marji., M.T., selaku Ketua Program Studi Informatika / Ilmu Komputer yang telah member motivasi terkait penyusunan skripsi.
4. Issa Arwani, S.Kom., M.Sc., selaku Sekretaris Program Studi Informatika / Ilmu Komputer yang telah memberikan kesempatan dan membantu mempercepat proses administratif penyusunan skripsi.
5. Wiwin Lukitohadi, S.Psi, SH, CHRM., selaku kepala unit konseling PTIIK yang telah memotivasi dan memberikan masukan terkait pelaksanaan penyusunan skripsi.

6. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Studi Teknik Informatika / Ilmu Komputer Program Teknologi Informasi dan Ilmu Komputer (PTIIK) Universitas Brawijaya.
7. Segenap staf dan karyawan di Program Teknologi Informatika dan Ilmu Komputer (PTIIK) Universitas Brawijaya yang telah banyak membantu penulis dalam pelaksanaan penyusunan skripsi ini.
8. Kedua orang tua, terima kasih atas semua doa, kasih sayang dan perhatian yang tulus serta dukungan yang telah diberikan.
9. Istri tercinta Yusrian yang tanpa kenal lelah selalu memberi dukungan, motivasi dan bantuan dalam menyelesaikan skripsi ini.
10. Rekan-rekan "Muke Gile 2006" di Program Studi Ilmu Komputer, Program Teknologi Informatika dan Ilmu Komputer (PTIIK) Universitas Brawijaya yang telah banyak memberikan bantuannya demi kelancaran pelaksanaan penyusunan skripsi ini.
11. Dan semua pihak yang telah terlibat baik secara langsung maupun tidak langsung yang tidak dapat penulis sebutkan satu persatu terima kasih atas semua bantuan yang telah diberikan.

Semoga skripsi ini bermanfaat bagi pembaca sekalian.Akhirnya, penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan dan mengandung banyak kekurangan, sehingga dengan segala kerendahan hati penulis mengharapkan kritik dan saran dari pembaca.

Malang, 20 Agustus 2013

Penulis

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

Sekarang ini perkembangan pesat terjadi dalam segala aspek. Salah satu aspek yang berkembang pesat dan menjadi pemicu berkembangnya aspek – aspek yang lain adalah komunikasi. Dalam kehidupan sehari – hari, disadari atau tidak, komunikasi adalah bagian dari kehidupan manusia itu sendiri. Untuk mendukung proses komunikasi, teknologi pada setiap jaman memberikan fitur – fitur yang beraneka ragam. Dari komunikasi verbal, tertulis, hingga visual. Dari berbagai jenis komunikasi, komunikasi tertulis merupakan jenis yang saat ini sangat bayak dipakai. Hal ini dibuktikan dengan adanya *e-mail*,*web Site, chatting*, dan lain sebagainya.

Seiring berkembanganya teknologi *mobile* nirkabel, penyedia layanan *mobile* pun meningkat pesat. Komunikasi nirkabel berbasis *text* pun tercipta. Salah satu teknologi nirkabel yang menggunakan *text* sebagai basis komunikasi adalah *short message service* (SMS). SMS yang diartikan dalam Bahasa Indonesia sebagai layanan pesan singkat atau surat masa singkat adalah sebuah layanan yang dilaksanakan menggunakan sebuah *Hand Phone* (HP) untuk mengirim atau menerima pesan – pesan pendek. SMS menjadi suatu komoditi yang menjanjikan untuk diperdagangkan. Sifatnya yang efisien, harganya yang murah, dan kecepatannya yang cukup memuaskan membuat SMS menempati ruang tersendiri dalam bidang komunikasi.

Sebuah pesan SMS maksimal terdiri dari 140 *bytes*. Dengan kata lain, sebuah pesan bisa memuat 140 karakter 8-bit atau 160 karakter 7-bit. Dengan demikian jika pengguna menuliskan SMS lebih dari 160 karakter maka biaya pengiriman yang akan dikeluarkan sama dengan pengiriman 2 SMS. Berdasarkan kasus tersebut penelitian ini bertujuan untuk meningkatkan efisiensi pengiriman data *text* yang diterapkan pada SMS, sehingga jumlah karakter yang dimuat dalam tiap SMS dapat bertambah.

Teknik dasar yang akan digunakan adalah memampatkan pesan SMS tersebut. Pemampatan atau kompresi adalah sebuah proses perubahan sebuah

input stream atau data awal menjadi sebuah data *stream* yang berbeda dengan ukuran yang lebih kecil. Salah satu metode kompresi yang cukup banyak dimanfaatkan adalah metode *Half Byte*.

Penelitian tentang imlementasi metode *Half Byte* untuk kompresi SMS sebelumnya pernah dilakukan oleh Yusrian Noviarti (2010). Pada penelitiannya dilakukan modifikasi pada metode *Half Byte* dengan melakukan modifikasi pada penataan data awalnya, sehingga menghasilkan tabel yang dinamis. Dari hasil penelitian tersebut didapatkan bahwa tingkat kompresi semakin baik jika jumlah karakter yang diinputkan semakin banyak tetapi memiliki variasi karakter yang sedikit. Tetapi semakin banyak variasi karakter yang diinputkan, maka hasil kompresi akan menjadi lebih besar karena *header* yang dihasilkan untuk mengenali variasi karakter tersebut menjadi lebih panjang.

Konsep dasar metode *Half Byte* adalah melakukan kompresi pada deretan karakter yang mempunyai 4 bit sebelah kiri yang sama secara berurutan. Saat karakter yang empat bit pertamanya sama diterima secara beruntun, algoritma ini mengkompres data tersebut dengan bit penanda kemudian karakter pertama dari deretan empat bit yang sama diikuti dengan pasangan empat bit terakhir deretan berikutnya dan ditutup dengan bit penutup (Sujaini, 2000).

Dalam penelitian ini akan dilakukan modifikasi pada algoritma *Half Byte* dimana algoritma ini baik jika di implementasikan pada karakter yang memiliki 4 bit pertama sama secara berurutan 7 karakter atau lebih. Namun jika karakter yang memiliki kesamaan 4 bit pertama secara berurutan kurang dari 7 maka algoritma *Half Byte* tidaklah efektif karena tidak akan terjadi pengurangan bit.

Modifikasi yang dilakukan adalah pengurangan jumlah bit kunci dari 4 menjadi 3 bit, dengan harapan akan semakin banyak karakter yang memiliki persamaan pada 3 bit awal dan secara otomatis menyebabkan jumlah karakter yang dapat dikompres menjadi lebih banyak. Pengambilan 3 bit sebagai kunci menyebabkan bit yang tersisa sebanyak 5 bit, berjumlah lebih banyak dibandingkan dengan metode *Half Byte* sebelumnya yang hanya memiliki sisa 4 bit tiap karakternya. Selanjutnya metode ini dinamakan *Asymmetric Half Byte*.

Hal ini berakibat pada meningkatnya jumlah bit yang terangkai dalam *file* terkompresi untuk sebuah kalimat yang sama, yang kemudian menyebabkan

bertambahnya jumlah minimal karakter yang dikompres agar memperoleh hasil yang lebih kecil dari semula. Jika pada metode *Half Byte* harus ada minimal 7 karakter yang mempunyai persamaan, maka pada *Asymmetric Half Byte* dibutuhkan 11 karakter.

Permasalahan lain yang ditimbulkan dari gabungan 3 bit kunci dan kelipatan 5 bit sisa yaitu sering kali jumlah deret bit *file* terkompresi yang dihasilkan tidak habis dibagi 8, sehingga bit akhir yang tersisa sangat berpotensi menimbulkan kerancuan pada proses dekompresi. Inilah sebabnya dibutuhkan sebuah bit info yang berguna untuk menyimpan informasi tentang deret bit akhir sebuah *file* terkompresi.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka dirumuskan beberapa masalah berikut ini:

1. Bagaimana implementasi kompresi sms menggunakan metode *Half Byte*.
2. Bagaimana implementasi kompresi sms menggunakan metode *Asymmetric Half Byte.*
3. Bagaimana hasil perbandingan rasio kompresi metode *Asymmetric Half Byte* dengan metode *Half Byte* dalam kompresi sms.

1.3 Batasan Masalah

Batasan masalah dari skripsi ini adalah sebagai berikut :

1. SMS yang dikompres adalah SMS dengan karakter ASCII *printable Character.*
2. Karakter yang dikompres berjumlah 95 karakter (huruf latin) yang terdiri dari 26 *upper case* (A-Z), 26 *lower case* (a-z), 10 digit desimal (0-9), dan 33 karakter tanda baca yang umum digunakan.

1.4 Tujuan

Adapaun tujuan dari skripsi ini adalah :

1. Mengimplementasikan metode *Half Byte* pada kompresi SMS.
2. Mengimplementasikan metode *Asymmetric Half Byte* pada kompresi sms.

3. Menganalisa rasio kompresi metode *Half Byte* dan *Asymmetric Half Byte* dalam melakukan kompresi SMS.

1.5 Manfaat

Manfaat yang didapat dari skripsi ini adalah diperoleh sebuah aplikasi yang dapat mengkompresi teks pada SMS sehingga dapat meningkatkan jumlah karakter yang dapat dikirim dalam setiap pengiriman SMS sehingga dapat menghemat biaya.

1.6 Metodologi

Digunakan beberapa metode dalam pembuatan skripsiini, antara lain :

1. Studi Literatur

   Mempelajari konsep-konsep dasar maupun terapan dari berbagai jenis media yang mendukung pembuatan skripsi.

2. Pendefinisian dan Analisis Masalah

   Dari hasil studi literatur dapat dibentuk sebuah deskripsi umum sistem dan dapat dilakukan analisa kebutuhan sistem. Setelah itu dilakukan perancangan awal aplikasi, sehingga dihasilkan desain *interface* dan proses yang siap diimplementasikan.

3. Perancangan dan Implementasi Perangkat Lunak

   Perancangan tampilan aplikasi perangkat lunak yang kemudian akan diimplementasikan menggunakan J2ME.

4. Uji Coba dan Analisa Hasil Implementasi

   Pada tahap ini akan dilakukan uji coba dan evaluasi untuk kelayakan pemakaian sistem.

1.7 Sistematika Penulisan

Penulisan skripsiini dibagi menjadi beberapa bagian. Terdiri dari 6 bab, yaitu :

1. **BAB I. PENDAHULUAN**

   Berisi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika penulisan.

2. **BAB II. TINJAUAN PUSTAKA**

Pada bab ini dicantumkan beberapa dasar teori yang berkaitan dengan penyusunan Tugas Akhir, diantaranya *Short Message Service* (SMS), algoritma *Half Byte*, kompresi, ASCII, dan J2ME.

3. **BAB III. METODOLOGI DAN PERANCANGAN**

   Berisi penjelasan tentang metode dan proses perancangan yang akan dilakukan untuk melakukan kompresi. Beberapa poin yang dibahas diantaranya perancangan proses kompresi dan perancangan proses dekompresi.

4. **BAB IV. IMPLEMENTASI**

   Bab ini menerangkan hasil implementasi metodologi dan perancangan yang telah disebutkan pada bab III.

5. **BAB V. PENGUJIAN DAN ANALISIS**

   Berisi penjelasan strategi pengujian dan teknik pengujian yang dilakukan. Pada bagian akhir dilakukan penjelasan hasil pengujian secara keseluruhan.

6. **BAB VI. PENUTUP**

   Bab ini berisi kesimpulan-kesimpulan dari hasil penelitian dan saran-saran untuk perbaikan pada penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1 Short Message Service (SMS)

Short Message Service atau SMS pertama kali ditemukan oleh SGM pioners di Eropa. Standarisasi di bawah lembaga *Eutopan Telecommunications Standards Institute*. SMS dibuat untuk menyediakan infrastruktur transportasi pesan singkat yang mempunyai kapasitas maksimal 140 *bytes*. Skema dasar pengalamatan SMS adalah nomor *mobile phone* yang disebut MSISDN (Sigit, 2007). SMS berbentuk bilangan biner, terbagi atas 2 bagian yaitu *message header* untuk transportasi data dan *messsage body* sebagai *payload*.

SMS saat ini menjadi sebuah fitur mendasar dari setiap telepon selular. Fitur SMS akan berjalan ketika terdapat operator telepon seluler yang menyediakan layanan ini. Dengan SMS memungkinkan dua orang yang memiliki telepon selular dapat saling mengirimkan pesan teks satu sama lain. Teknologi yang mendukung SMS antara lain adalah GSM, TDMA dan CDMA. Dengan didukung oleh ketiga teknologi ini, SMS telah menjadi layanan data bergerak yang bersifat *universal*. Protokol yang bekerja pada SMS ini lebih dikenal dengan nama *Protocol Data Unit* atau PDU . SMS mampu mengirimkan dan menerima secara simultan data berupa teks antar jaringan operator selular.

Perkembangan SMS yang cukup pesat menjadikan layanan ini banyak diminati oleh pengguna telepon seluler. Menurut Ayuningtyas (2007) berbagai keunggulan dari SMS yaitu:

1. *Deliver Oriented Service*, pesan akan selalu diusahakan untuk dikirimkan ke tujuan. Jika suatu tujuan sedang tidak aktif, pesan akan disimpan di SMSC (*short message service centre)* sebagai *server* dan akan dikirim sesegera mungkin setelah nomor tujuan aktif. Pesan juga akan tetap terkirim walaupun pengguna sedang melakukan telepon karena transmisi SMS menggunakan kanal *signaling* bukan kanal suara.
2. Dapat dikirim ke berbagai tujuan/penerima pada saat bersamaan.
3. Dapat dikirim ke berbagai jenis penerima, seperti email, IP, maupun berbagai aplikasi lain.

4. Memiliki beberapa macam kegunaan bila dilakukan integrasi dengan berbagai aplikasi, seperti *chatting*, *voting*, kuis, reservasi, informasi tertentu, dan sebagainya.

Sebuah SMS terdiri atas 160 karakter untuk skema 7 bit, dan ada yang memakai skema 8 bit berjumlah 140 karakter, atau 70 karakter memakai skema 16 bit (Bambang, 2007). Maksud skema disini adalah pemakaian jenis tabel ASCII, jika suatu operator selular memakai skema 7 bit berarti tabel yang digunakan adalah tabel ASCII 7 bit. Untuk operator-operator yang berkembang di Indonesia saat ini kebanyakan memakai skema 7 bit. Penggabungan SMS dan kompresi SMS (terdiri lebih dari 160 karakter) telah dikembangkan. Tetapi fitur-fitur ini belum diimplementasikan oleh semua jaringan operator selular di dunia.

2.1.1 Mekanisme Kerja SMS

Menurut Adi Purnomo (2007), ketika suatu SMS dikirimkan ke suatu nomor tertentu, SMS tersebut tidak langsung dikirimkan ke nomor yang dimaksud, tetapi akan masuk ke SMSC operator telepon yang digunakan pengirim terlebih dahulu. SMS berfungsi sebagai penghubung antara HP pengirim dan HP penerima. Setelah SMS masuk ke SMSC, selanjutnya akan diteruskan ke *handphone* penerima, begitu juga sebaliknya.

Agar dapat mengirim dan menerima SMS harus melakukan koneksi ke SMSC. Adapun beberapa cara yang dapat dilakukan untuk melakukan koneksi ke SMS adalah sebagai berikut :

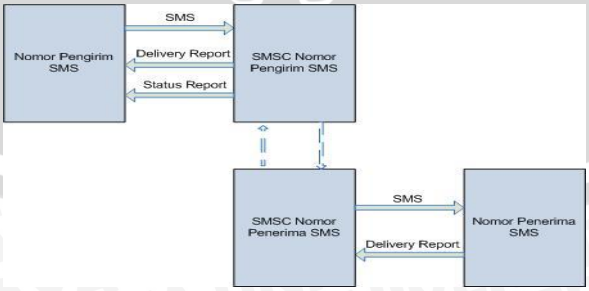
1. Menggunakan terminal, dapat berupa GSM modem atau *handphone*. Meskipun cara ini adalah cara yang paling mudah namun memiliki banyak kekurangan karena jumlah pesan yang dapat dikirimkan per menit nya sangat terbatas, yaitu sekitar 6 – 10 pesan.
2. Melakukan koneksi langsung dengan SMSC. Untuk melakukan hal ini dibutuhkan sebuah protokol penghubung.
3. Menggunakan *software* bantu yang saat ini banyak ditawarkan oleh vendor telekomunikasi, baik yang bersifat *freeware* maupun komersil.

Mekanisme kerja pengiriman SMS dibagi menjadi 2, yaitu :

1. Pengiriman SMS dalam satu operator, disebut intra-operator.
   SMS yang dikirimkan oleh nomor pengirim akan masuk ke SMSC operator nomor pengirim terlebih dulu, setelah itu SMS tersebut akan dikirimkan ke nomor yang dituju secara langsung. Nomor penerima akan mengirimkan sebuah *delivery report* yang disertai status *report* dari proses tersebut ke SMSC jika SMS telah diterima, yang kemudian akan diteruskan oleh SMSC ke nomor pengirim. Proses tersebut di atas ditunjukkan pada Gambar 2.1.

Gambar 2.1 Mekanisme Pengiriman SMS Intra-Operator

2. Pengiriman SMS pada 2 operator yang berbeda.
   Mekanisme ini melibatkan dua buah SMSC, yaitu SMSC operator pengirim dan SMSC operator penerima. Selain masuk ke SMSC operator pengirim, SMS akan dikirimkan ke SMSC operator penerima SMS sebelum dikirimkan ke nomor tujuan. *Delivery report* yang dihasilkan juga akan melalui jalur yang sama untuk sampai ke nomor pengirim. Komunikasi secara tidak langsung antara 2 operator yang berbeda ini dapat berlangsung setelah dilakukan kesepakatan kerjasama antar kedua operator tersebut.jika kesepakatan tidak terjadi akan menyebabkan SMS yang dikirim ke nomor tujuan yang berbeda operator tidak akan sampai ke nomor yang dituju. Mekanisme ini ditunjukkan pada Gambar 2.2



Gambar 2.2 Mekanisme Pengiriman SMS Inter-Operator

Sumber : Ayuningtyas, N. 2007. " Implementasi Kode Huffman Dalam Aplikasi Kompresi Teks Pada layanan SMS". ITB , Bandung.

2.2 ASCII

Kode Standar Amerika untuk Pertukaran Informasi atau ASCII (*American Standard Code for Information Interchange*) merupakan suatu standar internasional berupa rangkaian bit kode untuk mewakili teks, agar bisa dimengerti oleh banyak model komputer (Ravi, 2008).

Ada dua jenis kode yang sering digunakan, yaitu ASCII- 8 bit dan ASCII- 7 bit. Kode yang terdapat pada ASCII-8 bit jauh lebih lengkap dari ASCII-7 bit. Menurut Tino ASCII-7 bit mempunyai kombinasi kode 2^7 = 127 , dengan ketentuan sebagai berikut:

1. 26 kode untuk huruf kapital *(upper case)* dari A –Z.
2. 26 kode untuk huruf kecil *(lower case)* dari a –z.
3. 10 digit desimal dari 0 –9.
4. 34 karakter kontrol untuk informasi status operasi komputer.
5. 32 karakter khusus *(special characters).*

Kode ASCII dengan nilai kode 0 sampai dengan 31 dan 127 termasuk dalam status karakter-karakter kontrol yang tidak dapat dicetak (*non-printable characters*). Contohnya, tabel ASCII karakter 28 mewakili fungsi "*file separator*", atau karakter 8 yang mewakili *backspace*.

Kode ke 32 adalah karakter *spasi*, menandakan pemisah antara kata, yang dihasilkan oleh *space-bar* dari *keyboard*. Kode 33 sampai 126 dikenal sebagai karakter-karakter yang dapat di cetak (*printable characters*), terdiri dari beberapa huruf, angka, tanda baca, dan beberapa simbol. Tabel 2.1 adalah beberapa contoh karakter ASCII 7 bit yang merupakan *printable characters* (Ravi, 2008).

Tabel 2.1 Kode ASCII-7 Bit

Binary	Oct	Dec	Hex	**Value**
011 0000	60	48	30	0
011 0001	61	49	31	1

011 0010	62	50	32	2
011 0011	63	51	33	3
011 0100	64	52	34	4
011 1010	72	58	3A	:
011 1011	73	59	3B	;
011 1100	74	60	3C	<
100 0001	101	65	41	A
100 0010	102	66	42	B
100 0011	103	67	43	C
100 0100	104	68	44	D

2.3 Kompresi

Kompresi atau *compression* adalah proses pemampatan ukuran sebuah data. Sebuah data terkadang memiliki sebuah informasi yang sama dan berulang-ulang. Sebuah informasi yang sama dan berulang-ulang membuat ukuran sebuah data menjadi besar. Pada transmisi data (*data transmision*), sebuah data berukuran besar akan membutuhkan waktu *transfer* lebih lama dibandingkan data berukuran kecil. Pada *file*, sebuah *file* yang berukuran besar menyita banyak ruangan pada media penyimpanan. Oleh karena itu, untuk menghindari masalah tersebut, maka dilakukan kompresi data. Dengan mengunakan algoritma-algoritma dan teknik-teknik tertentu, informasi yang sama dan berulang-ulang tersebut dikodekan sedemikian rupa sehingga data tersebut menjadi berukuran lebih kecil. *File* atau data yang sudah dikompres agar bisa digunakan kembali harus dikembalikan lagi seperti semula. Proses pengembalian sebuah *file* yang terkompres menjadi seperti *file* aslinya disebut dekompresi atau *Decompression*.

Jenis kompresi data berdasarkan *mode* penerimaan data oleh manusia:

1. *Dialoque Mode* yaitu proses penerimaan data dimana pengirim dan penerima seakan berdialog (*real time*), seperti pada contoh *video conference*. Kompresi data pada *mode* ini harus berada dalam batas penglihatan dan pendengaran manusia. Waktu tunda (*delay*) tidak boleh lebih dari 150 ms, dimana 50 ms untuk proses kompresi dan dekompresi dan 100 ms untuk mentransmisikan data dalam jaringan.
2. *Retrieval Mode* yaitu proses penerimaan data tidak dilakukan secara *real time*. *Retrieval Mode* dapat dilakukan secara *fast forward* maupun *fast rewind* di *client*, dan dapat dilakukan *random access* terhadap data dan dapat bersifat interaktif *(*Anynomous, 2005).

Jenis kompresi data berdasarkan output :

1. *Lossy Compression*
   - Teknik kompresi dimana terdapat data yang hilang selama proses kompresi. Artinya data hasil dekompresi tidak sama dengan data sebelum kompresi namun sudah cukup untuk digunakan. Contoh: Mp3, *streaming media*, JPEG, MPEG, dan WMA.
   - Kelebihannya yaitu ukuran file lebih kecil dibanding *loseless* namun masih tetap memenuhi syarat untuk digunakan.
   - Prinsip dasar dari *Lossy Compression* adalah membuang bagian-bagian data yang tidak digunakan, tidak dirasakan, tidak begitu dilihat oleh manusia sehingga manusia masih beranggapan bahwa data tersebut masih bisa digunakan walaupun sudah dikompresi.
2. *Loseless Compression*
   - Teknik kompresi dimana data hasil kompresi dapat didekompres lagi dan hasilnya tepat sama seperti data sebelum proses kompresi. Contoh aplikasi: ZIP, RAR, GZIP, 7-Zip
   - Data hasil dekompresi tetap sama dengan data sebelum dikompres, artinya tidak terdapat data yang hilang atau data kembali seperti sebelum proses kompresi.

Untuk Huffman *Coding* yang digunakan dalam penelitian ini termasuk dalam metode *Loseless Compression (*Anynomous, 2005).

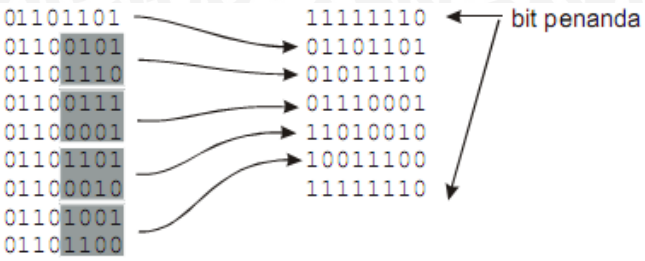
2.3.1 Algoritma *Half Byte*

Algoritma *Half Byte* memanfaatkan 4 bit pertama yang sering sama secara berurutan terutama pada *file* teks. Sebagai contoh, misalnya sebuah *file* berisi tulisan "mengambil", dalam heksadecimal dan biner karakter-karakter tersebut diterjemahkan menjadi sebagaimana terlihat pada Tabel 2.2 sebagai berikut.

Tabel 2.2 Tabel Contoh 1

Karakter	Decimal	Biner
m	109	01101101
e	101	01100101
n	110	01101110
g	103	01100111
a	97	01100001
m	109	01101101
b	98	01100010
i	105	01101001
l	108	01101100

Dari Tabel 2.2 di atas bisa diketahui bahwa karakter-karakter tersebut memiliki 4 bit pertama yang sama. Gejala seperti inilah yang dimanfaatkan oleh algoritma *Half Byte*. Saat karakter yang empat bit pertamanya sama diterima secara berurutan sebanyak 7 kali atau lebih, algoritma ini mengkompres data tersebut dengan memberikan bit penanda di depan karakter yang akan dikompres kemudian karakter pertama dari deretan karakter yang memiliki 4 bit pertama sama yang selanjutnya diikuti dengan pasangan empat bit terakhir deretan berikutnya dan ditutup dengan bit penutup. Proses kompresi ini ditunjukkan pada Gambar 2.3 di bawah ini.



Gambar 2.3 Proses Kompresi Algoritma *Half Byte*.

Sumber : Sujaini, Herry dan Yessy Mulyani. 2000. "Pemampatan File". ITB. Bandung

Deretan data di sebelah kiri merupakan deretan data pada file asli, sedangkan deretan di sebelah kanan merupakan deretan data yang telah dikompresi menggunakan algoritma *Half Byte*.

2.3.1.1 Proses *Encoding*

Encoding adalah cara menyusun string biner dari teks yang ada (Wardoyo, 2005).

Langkah-langkah yang dilakukan untuk meng-*encoding* suatu string biner menggunakan algoritma *Half Byte* (Sujaini, 2000) adalah :

1. Mengecek apakah ada deretan karakter yang mempunyai persamaan 4 bit pertama 7 kali berturut-turut atau lebih, jika memenuhi maka lakukan proses encoding
2. Memasukkan sebuah bit penanda di depan *file* yang akan dimampatkan. Bit penanda yang dipakai berupa 8 deretan bit (1 *byte*) dan merupakan karakter yang tidak atau jarang digunakan dalam teks yang terkait, dalam penelitian ini adalah karakter yang tidak digunakan pada SMS. Bit penanda dapat dipilih sembarang asalkan digunakan dengan konsisten mengingat fungsi dari bit penanda ini adalah untuk menandai bahwa karakter selanjutnya adalah karakter yang dimampatkan. Sehingga tidak akan membingungkan pada saat proses dekompresi.
3. Menambahkan karakter pertama dari deretan karakter 4 bit pertama sama dari *file* asli.

4. Menambahkan 4 bit kanan dari karakter ke-2 dan ke-3 pada *file* pemampatan. Dan mengulangi langkah ini sampai dengan akhir deretan 4 bit pertama yang sama.
5. Menutup *file* pemampatan dengan bit penanda.

Seperti telah disebutkan pada Bab I, algoritma *Half Byte* ini akan efektif jika jumlah karakter yang memiliki 4 bit pertama yang sama lebih dari atau sama dengan 7. Ada beberapa hal lain yang harus diperhatikan sebelum melakukan encoding menggunakan algoritma *Half Byte*, di antaranya adalah :

1. Jika pada *file* asli ditemukan karakter yang sama dengan bit penanda maka karakter tersebut harus ditulis sebanyak dua kali pada *file* pemampatan, untuk menghindari kesalahan mengenali apakah karakter tersebut merupakan bit penanda atau benar-benar karakter dari *file* asli pada saat dekompresi.
2. Jika pada saat penggabungan 4 bit kanan dari 2 karakter menghasilkan bit baru yang sama dengan bit penanda sehingga diduga sebagai bit penutp, maka deretan *file* tersebut tidak perlu dimampatkan.
3. Jika deretan karakter yang memenuhi syarat untuk dilakukan kompresi berjumlah genap, maka karakter yang terakhir tidak dimampatkan.

2.3.1.2 Proses Decoding

Decoding adalah proses penyusunan kembali sebuah data dari *string* biner menjadi sebuah karakter. Untuk melakukan proses *decoding* menggunakan algoritma *Half Byte* (Sujaini, 2000), langkah-langkah yang harus dilakukan adalah sebagai berikut :

1. Mengecek karakter pada hasil pemampatan dari awal hingga akhir, jika ditemukan bit penanda maka dilakukan proses pengembalian.
2. Mengecek karakter setalah bit penanda dan menambahkan karakter tersebut pada *file* pengembalian.
3. Mengecek karakter berikutnya, jika bukan bit penanda, mengambil 4 bit kanannya kemudian menggabungkannya dengan 4 bit kiri karakter di bawahnya.
4. Menambahkan hasil penggabungan tersebut pada *file* pengembalian.

5. Melakukan langkah – langkah tersebut di atas secara berulang – ulang hingga ditemukan bit penanda sebagai penutup.

2.3.2 Algortima Dynamic Half Byte

Metode kompresi yang digunakan menerapkan prinsip dasar dari metode *Half Byte* yang memanfaatkan 4 bit pertama pada masing-masing karakter yang sering sama terutama pada *file text*. Perbedaannya, pada penelitian ini digunakan tabel dinamis yang dibentuk berdasarkan teks SMS yang dituliskan oleh *user*.

Sistem akan mencari jenis – jenis karakter yang digunakan pada teks SMS, menyimpannya dalam sebuah *array* dan mengkodekan masing – masing karakter dengan tidak mengacu pada tabel ASCII. Pengkodean bermanfaat pada saat proses kompresi. Selain kode – kode tersebut, sistem juga akan menyimpan kode dari masing – masing karakter yang sesuai dengan tabel ASCII secara berurutan, yang selanjutnya digunakan sebagai *header*. Hal ini bertujuan untuk menghindari kesalahan dan kerancuan pada saat proses dekompresi.

2.3.2.1 Proses Kompresi Dynamic Half Byte

Proses ini dilakukan sebelum SMS dikirimkan, setelah pengguna selesai mengetikkan SMS. Langkah – langkah yang dilakukan pada proses kompresi adalah sebagai berikut :

1. *Input* data berupa *string* teksSMS
2. Melakukan proses pencarian variasi karakter dari teks SMS
3. Melakukan proses pembuatan header berdasarkan variasi karakter yang didapatkan
4. Mengkodekan tiap karakter pada SMS ke dalam bentuk biner sesuai dengan index karakter tersebut pada hasil dari langkah ke-2
5. Melakukan kompresi pada kode karakter yang dihasilkan

2.3.2.2 Proses Dekompresi Dynamic Half Byte

Proses dekompresi dilakukan pada saat SMS yang dikirimkan telah sampai ke *handphone* penerima. Agar SMS terkompresi yang dikirimkan dapat terbaca dengan benar, maka harus dilakukan proses dekompresi terlebih dahulu. Langkah – langkah pada proses dekompresi adalah sebagai berikut :

1. *Input* byte SMS yang terkompresi
2. Mengkonversi *byte* menjadi *string*
3. Membaca *header*, menginisialisasi karakter yang dipakai dalam SMS, kemudian memisahkan *header* dari teks SMS.
4. Melakukan dekompresi teks SMS
5. Kemudian melakukan pencocokan kode hasil dekompresi dengan *header* yang telah dipisahkan sebelumnya.
6. *Output string* hasilDekompres.

2.4 J2ME

Awalnya, *Java* dibangun pada tahun 1991 oleh James Gosling. Pada mulanya diberi nama oak, namun kemudian diubah menjadi Java karena telah ada sebuah bahasa yang diberi nama Oak. *Platform* Java dibedakan menjadi beberapa edisi, yaitu :

1. *Java* 2 *Platform, Standard Edition* (J2SE), digunakan untuk aplikasi desktop.
2. *Java* 2 *Platform, Enterprise Edition* (J2EE), digunakan untuk aplikasi *enterprise* yang terfokus pada pengembangan sisi *webserver*.
3. *Java* 2 *Platform, Micro Edition* (J2ME), digunakan pada perangkat *mobile*.
4. *Javacard*

J2ME adalah satu set spesifikasi dan teknologi yang fokus kepada perangkat konsumen.Perangkat ini memiliki jumlah memori yang terbatas, menghabiskan sedikit daya dari baterei,layar yang kecil dan bandwith jaringan yang rendah.

Dengan perkembangbiakan perangkat mobile konsumer dari telepon, PDA, kotak permainan keperalatan-peralatan rumah, Java menyediakan suatu lingkungan yang portable untuk mengembangkan dan menjalankan aplikasi pada

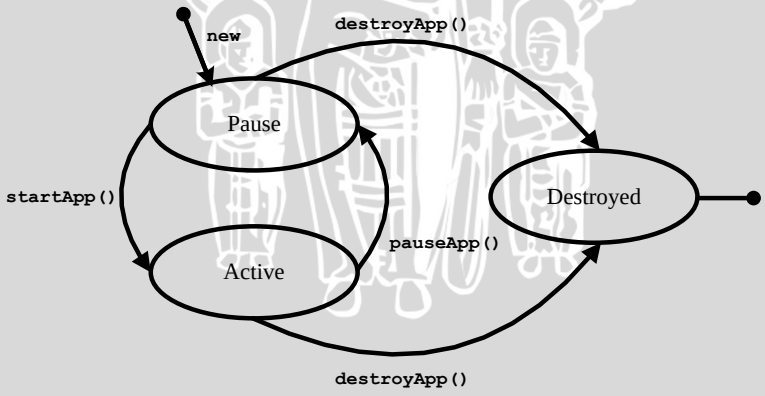
perangkat ini.Program J2ME, seperti semua program JAVA adalah diterjemahkan oleh VM. Program-program tersebut dikompile ke dalam bytecode dan diterjemahkan denga Java Virtual Machine(JVM).

Inti dari J2ME terletak pada configuration dan profile-profile. Suatu configuration menggambarkan lingkungan runtime dasar dari suatu sistem J2ME. Ia menggambarkan core library, virtual machine, fitur keamanan dan jaringan.

2.4.1 Pengenalan Midlet

MIDlet merupakan bagian dari *package* javax.microedition.midlet. Perangkat *Application Management Software* (AMS) berinteraksi langsung dengan MIDlet dengan *method* MIDlet *create, start, pause, dan destroy*. Suatu MIDlet tidak mempunyai *method public static void main(String[] args). Method* tersebut tidak dikenali oleh AMS sebagai titik awal sebuah program.

Siklus MIDlet dapat dilihat pada Gambar 2.4.



Gambar 2.4 Siklus hidup MIDlet

2.3 Rasio Kompresi

Rasio kompresi merupakan perbandingan ukuran *file* asli dan ukuran *file* setelah kompresi. Persentase kompresi dapat dinyatakan dengan persamaan (Sayood, 2000):

$$R = (1 - (\frac{K}{A})) * 100\%$$

Keterangan :

A = Ukuran *file* asli (sebelum kompresi)

K = Ukuran *file* setelah kompresi

R = Persentase kompresi

Sedangkan untuk menghitung jumlah karakter rata-rata yang dapat dihasilkan setelah dilakukan kompresi dapat dinyatakan dengan persamaan sebagai berikut:

$$JumlahKarakterRata - Rata = K + (K * R)$$

Keterangan :

K = Jumlah karakter SMS normal

R = Rata-rata rasio kompresi *Asymmetric Half Byte*

BAB III
METODOLOGI DAN PERANCANGAN

Dalam bab ini akan dijelaskan mengenai metode perancangan yang digunakan dan langkah – langkah yang akan dilakukan untuk melakukan kompresi *short message service* menggunakan metode *Asymmetric Half Byte.*

3.1 **Studi *literature***

Dalam penelitian ini dibutuhkan studi *literature* untuk merealisasikan tujuan dan penyelesaian masalah, Teori – teori berkaitan dengan Algoritma *Half Byte* digunakan sebagai dasar penelitian yang didapat dari buku, jurnal, dan *browsing* dari internet. Dari data yg diperoleh akan diolah sehingga dapat digunakan untuk analisis. Setelah dilakukan analisis maka dapat diimplementasikan ke dalam aplikasi perangkat lunak.

3.2 **Data Penelitian**

Untuk mengetahui bagaimana kinerja *system* kompresi digunakan data pengujian yang berasal dari beberapa input berupa *text* sms.

3.3 **Deskripsi Umum Sistem**

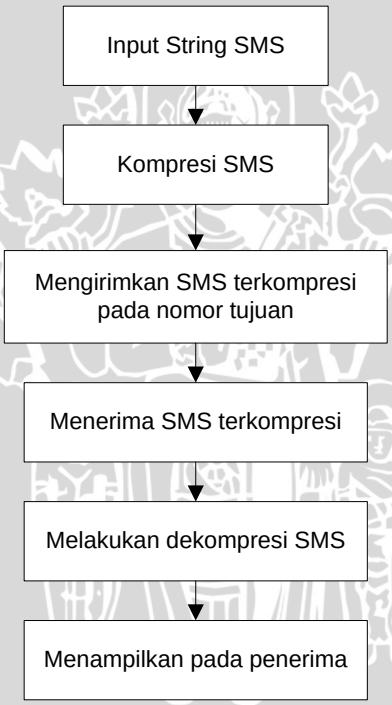
Sistem yang akan dibangun pada penelitian ini merupakan sebuah aplikasi SMS yang akan dijalankan pada *handphone*. Dengan menggunakan aplikasi ini *user* dapat melakukan kompresi SMS, mengirim dan menerima SMS yang kemudian didekompresi untuk mendapatkan teks SMS sesuai dengan teks yang asli seperti sebelum dilakukan kompresi.

Proses kompresi yang dilakukan bertujuan untuk efisiensi jumlah karakter yang dikirim dalam 1 SMS yang saat ini hanya dapat memuat 160 karakter, sehingga dapat menambah jumlah karakter dalam setiap SMS dengan ukuran tetap dan biaya yang sama. Metode yang digunakan adalah modifikasi dari metode *Half Byte* yang mermpunyai prinsip dasar memanfaatkan persamaan dari 4 bit pertama setiap karakter yang sering digunakan dalam sebuah *file text*. Perbedaannya, penelitian ini akan memanfaatkan persamaan dari 3 bit pertama masing-masing karakter berdasarkan *table* ASCII agar didapatkan lebih banyak

karakter yang dapat dikompresi, sehingga ukuran *file* hasilnya pun akan semakin kecil. Hasil modifikasi algoritma *Half Byte* ini kemudian dinamakan algoritma *Asymmetric Half Byte*.

Tahap selanjutnya adalah melakukan analisa dan perancangan perangkat lunak. Perangkat lunak yang dimaksudkan akan mengkompresi SMS yang dikirimkan, dan melakukan dekompresi sebelum pesan ditampilkan pada penerima. Perangkat lunak ini harus *terinstall* pada masing-masing *handphone* baik pengirim maupun penerima SMS.

Tahap-tahap yang dilakukan oleh *system* secara umum diilustrasikan pada Gambar 3.1.



Gambar 3.1. Tahapan umum system

3.4 **Perancangan Proses**

Pertama kali akan dijelaskan mengapa jumlah bit kunci yang digunakan sebanyak 3 bit. Table perhitungan preproses dari beberapa kemungkinan bit kunci yang digunakan adalah sebagai berikut, dengan rumus yang digunakan untuk menentukan jumlah bit akhir setelah kompresi yaitu:

Jumlah Bit Akhir = 8 bit info + 8 bit penanda + jumlah bit kunci + (jumlah bit sisa * jumlah karakter) + 8 bit penanda

Table 3.1. Rumus perhitungan jumlah bit akhir setelah kompresi dalam preproses Kompresi Menggunakan *Asymmetric Half Byte*

Table 3.2. Preproses Kompresi Menggunakan *Half Byte* (4-4)

Jumlah Karakter (Byte)	Jumlah Bit Awal (bit)	Jumlah Bit Akhir (bit)	Jumlah Karakter Akhir (Byte)	Rasio (%)
1	8	24	3	-200%
2	16	28	4	-100%
3	24	32	4	-33%
4	32	36	5	-25%
5	40	40	5	0%
6	48	44	6	0%
7	56	48	6	14%
8	64	52	7	13%
9	72	56	7	22%
10	80	60	8	20%
11	88	64	8	27%
12	96	68	9	25%
13	104	72	9	31%
14	112	76	10	29%
15	120	80	10	33%
16	128	84	11	31%
17	136	88	11	35%
18	144	92	12	33%
19	152	96	12	37%
20	160	100	13	35%

Table 3.3. Preproses Kompresi Menggunakan *Asymmetric Half Byte* (3-5)

Jumlah Karakter (Byte)	Jumlah Bit Awal (bit)	Jumlah Bit Akhir (bit)	Jumlah Karakter Akhir (Byte)	Rasio (%)
1	8	32	4	-300%
2	16	37	5	-150%
3	24	42	6	-100%
4	32	47	6	-50%

5	40	52	7	-40%
6	48	57	8	-33%
7	56	62	8	-14%
8	64	67	9	-13%
9	72	72	9	0%
10	80	77	10	0%
11	88	82	11	0%
12	96	87	11	8%
13	104	92	12	8%
14	112	97	13	7%
15	120	102	13	13%
16	128	107	14	13%
17	136	112	14	18%
18	144	117	15	17%
19	152	122	16	16%
20	160	127	16	20%

Table 3.4. Preproses Kompresi Menggunakan Modifikasi *Half Byte* (2-6)

Jumlah Karakter (Byte)	Jumlah Bit Awal (bit)	Jumlah Bit Akhir (bit)	Jumlah Karakter Akhir (Byte)	Rasio (%)
1	8	32	4	-300%
2	16	38	5	-150%
3	24	44	6	-100%
4	32	50	7	-75%
5	40	56	7	-40%
6	48	62	8	-33%
7	56	68	9	-29%
8	64	74	10	-25%
9	72	80	10	-11%
10	80	86	11	-10%
11	88	92	12	-9%
12	96	98	13	-8%
13	104	104	13	0%
14	112	110	14	0%
15	120	116	15	0%
16	128	122	16	0%
17	136	128	16	6%
18	144	134	17	6%
19	152	140	18	5%
20	160	146	19	5%

Table 3.5. Preproses Kompresi Menggunakan Modifikasi *Half Byte* (5-3)

Jumlah Karakter (Byte)	Jumlah Bit Awal (bit)	Jumlah Bit Akhir (bit)	Jumlah Karakter Akhir (Byte)	Rasio (%)
1	8	32	4	-300%
2	16	35	5	-150%
3	24	38	5	-67%
4	32	41	6	-50%
5	40	44	6	-20%
6	48	47	6	0%
7	56	50	7	0%
8	64	53	7	13%
9	72	56	7	22%
10	80	59	8	20%
11	88	62	8	27%
12	96	65	9	25%
13	104	68	9	31%
14	112	71	9	36%
15	120	74	10	33%
16	128	77	10	38%
17	136	80	10	41%
18	144	83	11	39%
19	152	86	11	42%
20	160	89	12	40%

Table 3.6. Perbandingan Rasio Preproses Kompresi Metode *Half Byte* dan *Asymmetric Half Byte*

Jumlah Karakter (Byte)	Jumlah Bit Awal (bit)	*Half Byte*	AHB (3-5)	AHB (2-6)	AHB (5-3)
1	8	-200%	-300%	-300%	-300%
2	16	-100%	-150%	-150%	-150%
3	24	-33%	-100%	-100%	-67%
4	32	-25%	-50%	-75%	-50%
5	40	0%	-40%	-40%	-20%
6	48	0%	-33%	-33%	0%
7	56	14%	-14%	-29%	0%
8	64	13%	-13%	-25%	13%
9	72	22%	0%	-11%	22%
10	80	20%	0%	-10%	20%
11	88	27%	0%	-9%	27%
12	96	25%	8%	-8%	25%
13	104	31%	8%	0%	31%
14	112	29%	7%	0%	36%

15	120	33%	13%	0%	33%
16	128	31%	13%	0%	38%
17	136	35%	18%	6%	41%
18	144	33%	17%	6%	39%
19	152	37%	16%	5%	42%
20	160	35%	20%	5%	40%

Dari table 3.6. dapat dilihat bahwa dengan metode *Asymmetric Half Byte* 3-5 diperoleh persentase rasio lebih baik jika dibandingkan dengan *Asymmetric Half Byte* 2-6, tetapi lebih kecil jika dibandingkan dengan *Asymmetric Half Byte* 5-3. Namun pada kenyataannya akan lebih sulit untuk mendapatkan persamaan 5 bit awal dari sebuah karakter pada 8 karakter berturut-turut seperti yang dibutuhkan *Asymmetric Half Byte* 5-3. Oleh sebab itu *Asymmetric Half Byte* 3-5 mempunyai peluang lebih baik untuk diimplementasikan.

Secara umum, tahapan proses dalam *system* kompresi SMS menggunakan Algoritma *Asymmetric Half Byte* ini terdiri dari 2 operasi, yaitu kompresi dan dekompresi. Setelah itu akan dihitung nilai perbandingan rasio SMS terkompresi dengan SMS tanpa kompresi.

3.4.1 Perancangan Proses Kompresi *Asimetric Half Byte*

Proses-proses berikut dilakukan sebelum SMS dikirimkan kepada penerima. Langkah-langkah yang dilakukan untuk meng-*encoding* suatu string biner menggunakan algoritma *Asymmetric Half Byte* adalah :

6. Mencari kode ASCII masing-masing karakter dalam pesan teks dan mengubahnya menjadi *binary string*. Khusus untuk karakter spasi (“ ”) dan titik (“.”) kode ASCII-nya akan ditukar dengan karakter *separator* (“|”) dan (“~”) berturut-turut. Hal ini dilakukan karena baik spasi (“ ”) maupun titik (“.”) lebih sering dipakai dalam *text* SMS dibandingkan (“|”) dan (“~”).
7. Mengecek apakah ada deretan karakter yang mempunyai persamaan 3 bit pertama 11 kali berturut-turut atau lebih, jika memenuhi maka dilanjutkan dengan proses *encoding*
8. Menambahkan sebuah bit penanda di depan *file* yang akan dikompresi. Bit penanda yang dipakai berupa 8 deretan bit (1 *byte*) dan merupakan karakter yang tidak atau jarang digunakan dalam sebuah teks, dalam penelitian ini

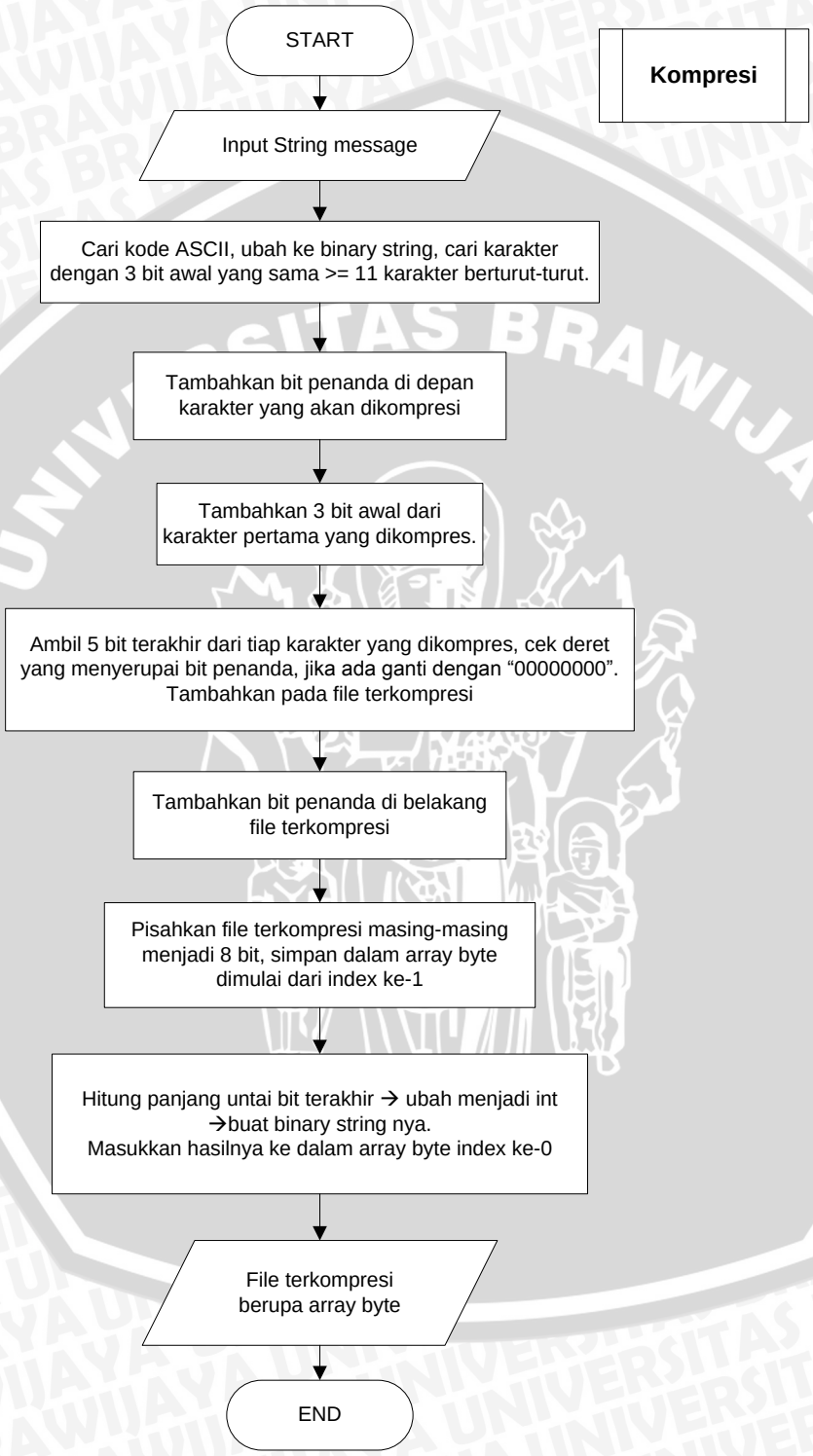
adalah karakter yang tidak digunakan pada SMS. Bit penanda dapat dipilih sembarang asalkan digunakan dengan konsisten mengingat fungsi dari bit penanda ini adalah untuk menandai bahwa karakter selanjutnya adalah karakter yang dikompresi. Sehingga tidak akan membingungkan pada saat proses dekompresi. Pada penelitian ini menggunakan "11111111"

9. Menambahkan 3 bit awal karakter pertama dari deretan karakter yang mempunyai persamaan 3 bit awal ke dalam *file* terkompresi.
10. Menambahkan 5 bit kanan dari karakter ke-1, ke-2 dan seterusnya sampai dengan untai bit terakhir yang mempunyai persamaan 3 bit awal pada *file* terkompresi. Dan mengulangi langkah ini sampai dengan karakter terakhir yang mempunyai deretan 3 bit pertama yang sama.
11. Melakukan pengecekan pada untai bit yang dihasilkan dari penggabungan 5 bit pada langkah ke 5. Jika ditemukan deret bit yang menyerupai bit penanda ("11111111") maka dilakukan penggantian menjadi "00000000" pada deret tersebut.
12. Menutup *file* terkompresi dengan bit penanda.
13. Membagi deretan bit *binary string* yang dihasilkan pada *file* terkompresi masing-masing menjadi 8 bit, dan mengkodekan potongan *binary string* tersebut menjadi kode ASCII yang selanjutnya diubah menjadi *byte* dan menyimpannya dalam sebuah *array byte*.
14. Dari pemisahan *binary string* tersebut dapat diketahui panjang untai *binary string* terakhir yang dibutuhkan untuk menentukan bit info, yang kemudian akan dimasukkan ke dalam index ke-0 *array byte* SMS terkompresi yang dikirimkan. Bit info diperoleh menggunakan *role* yang telah ditentukan dan ditunjukkan pada *Table* 3.7 . Bit info tersebut diperlukan pada proses dekompresi yaitu untuk mengetahui *binary string* dari karakter terakhir dari *file* terkompresi.

```
BIT INFO = DeretBitAkhir.startsWith("1") ? Integer.toBinaryString(0) :
Integer.toBinaryString( DeretBitAkhir.length() ) ;
```

Table 3.7 *Role* untuk menentukan bit info

Flowchart dari langkah-langkah kompresi *Asymmteric Half Byte* adalah sebagai berikut.

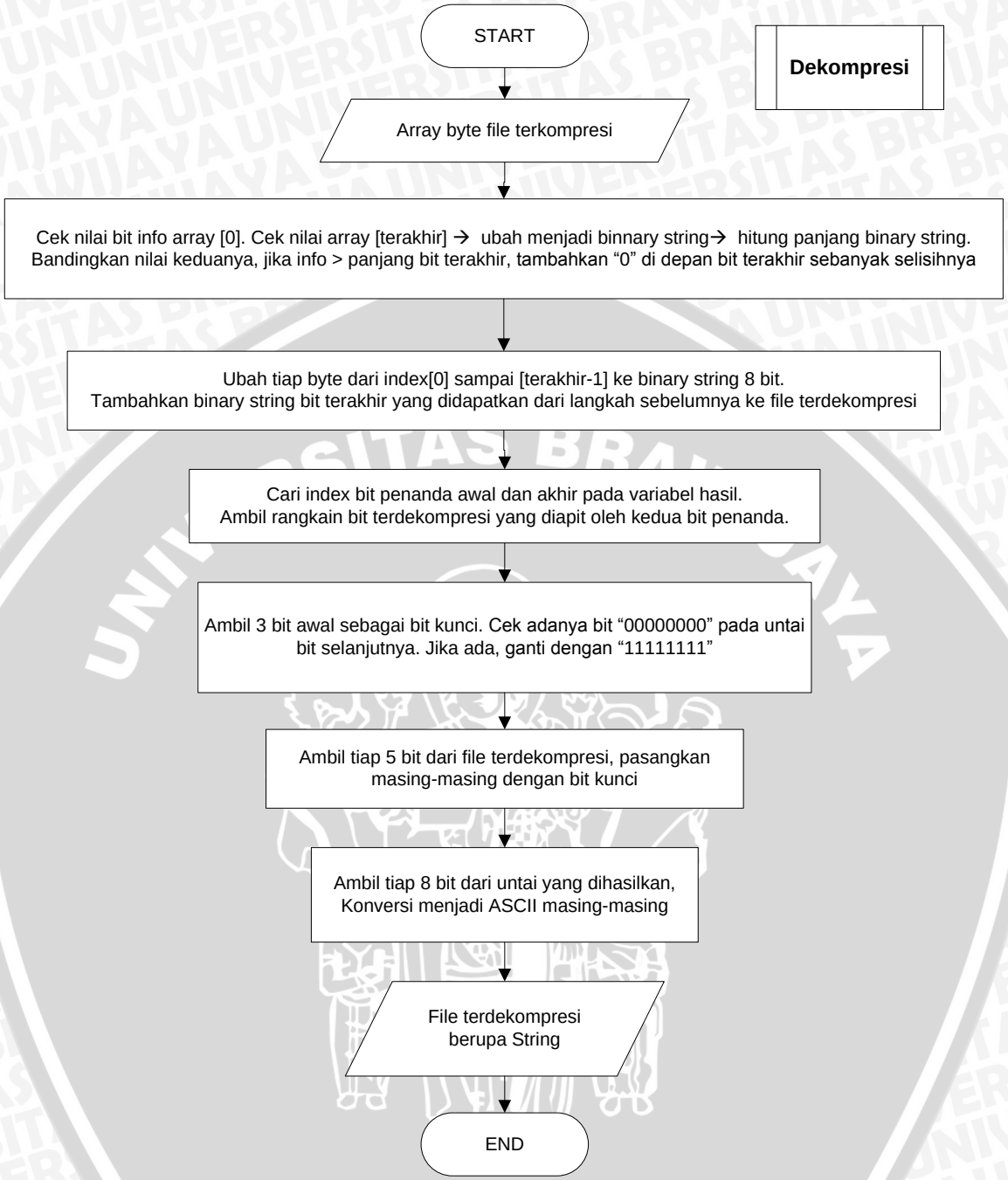


Gambar 3.2. *Flowchart* keseluruhan proses kompresi *Asimetric Half Byte*

3.4.2 Perancangan Proses Dekompresi *Asimetric Half Byte*

Setelah pesan teks melalui proses kompresi seperti yang dijelaskan sebelumnya, selanjutnya dilakukan proses dekompresi agar pesan yang telah di encode dapat dibaca oleh penerima. Langkah-langkahnya yaitu :

1. Mengambil nilai *index* ke-0 dari *array byte* kemudian dikonversi menjadi *integer*, simpan dalam sebuah *variable* info yang bertipe *int*.
2. Melakukan pengecekan apakah nilai tersebut >= 16, jika iya maka info = 16 – info.
3. Melakukan konversi tiap *byte* dalam *array* menjadi *binary string*, dimulai dari *index* ke-1 sampai dengan (*index* terakhir - 1), menyeragamkan panjang *binary string* masing-masing menjadi 8 bit dengan menambahkan "0" di depannya dan hasilnya disimpan dalam sebuah *variable* bertipe *String*.
4. Mengkonversi nilai pada *index* terakhir ke dalam *binary string* tanpa menambahkan "0" di depannya seperti pada langkah 3, kemudian membandingkan panjang *binary string* yang dihasilkan dengan *variable* info. Jika *variable* info lebih besar dari panjang *binary string* tersebut, tambahkan "0" di depan *binary string* yang dihasilkan sebanyak selisih jumlahnya. Menambahkan hasilnya ke dalam *variable* hasil pada langkah ke-3.
5. Mencari *index* bit penanda awal dan bit penanda akhir pada variabel hasil. Setelah itu akan didapatkan rangkain bit terkompresi yang diapit oleh kedua bit penanda.
6. Memisahkan 3 bit awal yang merupakan bit kunci dari rangkain bit terkompresi dan mengecek adanya rangkaian bit "00000000" pada sisa untai bit yang ada. Jika ada, ubah rangkaian bit tersebut menjadi "11111111".
7. Mengambil tiap 5 bit dari untaian bit yang dihasilkan pada langkah ke-6 dan menyambungnya dengan bit kunci yang telah dipisahkan sebelumnya. Melakukan perulangan ini sampai akhir untaian.



Gambar 3.3. *Flowchart* keseluruhan proses Dekompresi *Asymmetric Half Byte*

3.5 Perhitungan Manual

3.5.1 Proses Kompresi

Diketahui sebuah *text* SMS berisi kalimat **"Sahabat sejati."**. Maka langkah-langkah komputasi manual kompresi menggunakan *Asymmetric Half Byte* kalimat tersebut adalah sebagai berikut:

1. Mencari kode ASCII masing-masing karakter
2. Mengkonversi kode ASCII menjadi *binary string*. Khusus untuk karakter spasi (" ") dan titik (".") kode ASCII-nya akan ditukar dengan karakter separator ("|") dan ("~") berturut-turut. Hal ini dilakukan karena baik spasi (" ") maupun titik (".") lebih sering dipakai dalam *text* SMS dibandingkan ("|") dan ("~").
3. Mencari deret *binary string* yang mempunyai 3 bit awal yang sama sampai dengan 11 karakter atau lebih. Pada kalimat ini semua karakter mempunyai 3 bit awal yang sama kecuali huruf pertama. Lebih jelas akan ditunjukkan pada table 3.1.
4. Memasukkan *binary string* karakter pertama ("S") ke dalam *variable* penampung. Hal ini dapat langsung dilakukan karena huruf "S" tidak mempunyai 3 bit awal yang sama dengan karakter-karakter berikutnya, sehingga tidak perlu mengalami proses kompresi terlebih dahulu.
5. Menambahkan bit penanda pada *variable* penampung *file* terkompresi, diikuti dengan menambahkan 3 bit awal dari karakter pertama yang memiliki kesamaan 3 bit awal dengan karakter-karakter berikutnya ("a"), 3 bit ini merupakan bit kunci.
6. Menambahkan 5 bit terakhir masing-masing karakter mulai dari karakter pertama yang memiliki kesamaan 3 bit awal dengan karakter-karakter berikutnya ("a") hingga karakter terakhir yang memiliki kesamaan 3 bit awal yaitu "." (titik) pada *file* terkompresi.
7. Menambahkan bit penanda sebagai penutup karakter-karakter terkompresi. Hasil dari penggabungan tersebut ditunjukkan dalam table 3.2.
8. Membagi deretan bit *binary string* yang dihasilkan pada *file* terkompresi masing-masing menjadi 8 bit, dan mengkodekan potongan *binary string* tersebut menjadi kode ASCII yang selanjutnya diubah menjadi *byte* dan

menyimpannya dalam sebuah *array* seperti ditunjukkan pada table 3.3. Dari table 3.3 dapat diperoleh bit info menggunakan role yang telah ditentukan dan menambahkan bit info ke dalam *array index* ke-0. Diilustrasikan dengan table 3.2

Table 3.8 Pengkodean karakter menjadi ASCII dan *binary string*

No	Karakter	ASCII	Binary String
1	S	83	01010011
2	a	97	01100001
3	h	104	01101000
4	a	97	01100001
5	b	98	01100010
6	a	97	01100001
7	t	116	01110100
8	spasi	124	01111100
9	s	115	01110011
10	e	101	01100101
11	j	106	01101010
12	a	97	01100001
13	t	116	01110100
14	i	105	01101001
15	titik	126	01111110

Table 3.9 Hasil penggabungan *binary string file* terkompresi

0101001111111111011000010100000001000100000110100
1110010011001010101000001101000100111110111111111

Table 3.10. Pengkodean *file* terkompresi menjadi ASCII

Index	Binary String	ASCII
1	01010011	83
2	11111111	-1
3	01100001	97
4	01000000	64
5	01000100	68
6	00011010	26
7	01110010	114
8	01100101	101
9	01010000	80
10	01101000	104
11	10011111	-97
12	01111111	-1
13	1	1

Table 3.11. Hasil pengkodean *file* terkompresi menjadi ASCII lengkap dengan bit info

Index	Binary String	ASCII
0	00000000	0
1	01010011	83
2	11111111	-1
3	01100001	97
4	01000000	64
5	01000100	68
6	00011010	26
7	01110010	114
8	01100101	101
9	01010000	80
10	01101000	104

11	10011111	-97
12	01111111	-1
13	1	1

3.5.2 **Proses Dekompresi**

Tahapan dekompresi dari perhitungan di atas adalah sebagai berikut :

1. Mengambil nilai index ke-0 dari array byte hasil proses kompresi kemudian dikonversi menjadi integer, simpan dalam sebuah variable info yang bertipe int. Array[0] atau bit info pada perhitungan ini bernilai 0.
2. Melakukan pengecekan apakah nilai tersebut >= 16, jika iya maka info = 16 – info.
3. Melakukan konversi tiap byte dalam array menjadi binary string, dimulai dari array[1] sampai dengan array[12], menyeragamkan panjang binary string masing-masing menjadi 8 bit dengan menambahkan “0” di depan binary string dan hasilnya disimpan dalam sebuah variable bertipe String.
4. Mengkonversi array[13] yang bernilai 1 ke dalam binary → “1”, kemudian membandingkan panjang binary string yang dihasilkan dengan var info. Variable info bernilai 0 diperoleh pada langkah 1, sedangkan binary string yang dihasilkan array terakhir panjangnya 1. Karena bit info tidak lebih besar dari panjang binary string array terakhir, maka tidak perlu menambahkan “0” di depan binary string tersebut dan dapat langsung ditambahkan pada file terdekompresi.
5. Mencari index bit penanda awal dan bit penanda akhir pada variabel hasil. Setelah itu akan didapatkan rangkain bit terkompresi yang diapit oleh kedua bit penanda.
6. Memisahkan 3 bit awal yang merupakan bit kunci dari rangkain bit terkompresi dan mengecek adanya rangkaian bit “00000000” pada sisa untai bit yang ada. Jika ada, ubah rangkaian bit tersebut menjadi “11111111”.

7. Mengambil tiap 5 bit dari untaian bit yang dihasilkan pada langkah ke-6 dan menyambungnya dengan bit kunci yang telah dipisahkan sebelumnya. Melakukan perulangan ini sampai akhir untaian.

Table 3.12. Konversi ASCII menjadi binary string pada file terkompresi

Index	ASCII	Binary String
1	83	01010011
2	-1	11111111
3	97	01100001
4	64	01000000
5	68	01000100
6	26	00011010
7	114	01110010
8	101	01100101
9	80	01010000
10	104	01101000
11	-97	10011111
12	-1	01111111
13	1	1

Table 3.13. File terkompresi yang sudah dipisahkan dengan bit penanda

011000010100000001000100000110100111001001100101010101000001101000100111110

3.6 Perancangan Antarmuka Sistem

Perancangan Antarmuka merupakan perancangan halaman aplikasi yang nantinya akan berinteraksi secara langsung dengan *user*. Perancangan yang akan dibuat adalah sebagai berikut :

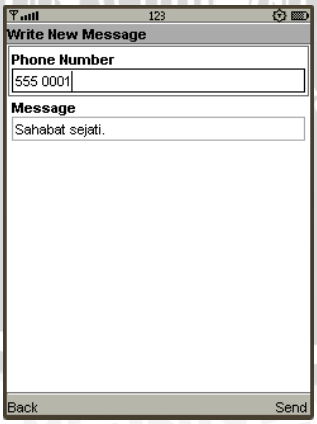
1. Menu utama.

   Menu utama adalah *Antarmuka* awal yang ditampilkan kepada *user*. Pada menu utama ini diberikan *list* menu untuk memilih *write new*, *inbox*, *outbox*, dan *exit*. Pengguna dapat memilih salah satu menu yang tersedia. Untuk mengakhiri penggunaan aplikasi, *user* dapat menekan *list* menu *exit*. *User interface* dari menu utama ditunjukkan pada Gambar 3.13.

Gambar 3.3 *Antarmuka* Menu Utama

2. *Write new*.

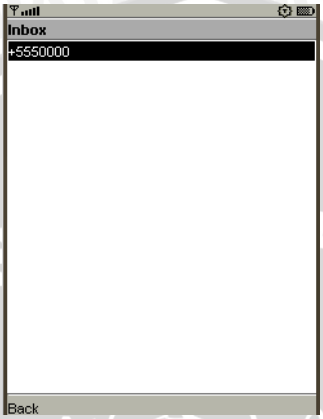
   Menu *write new* merupakan menu yang disediakan sebagai tempat mengetik pesan bagi *user*. Menu ini menggunakan. Pada menu ini terdapat 2 tombol, yaitu tombol untuk *back* dan *send* untuk mengirimkan pesan. Menu *write new* ditunjukkan pada Gambar 3.14.



Gambar 3.4 *Antarmuka* Menu *Write New*

3. *Inbox*

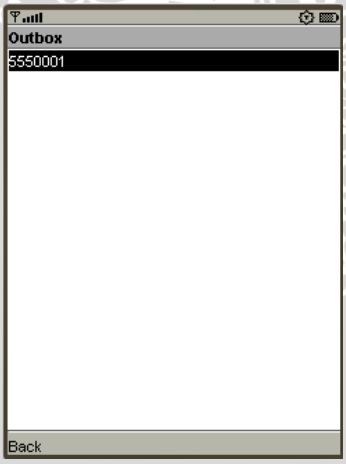
Antarmuka inbox berisi *list* dari pesan yang masuk. Terdapat tombol *back* untuk kembali ke menu awal. Gambar 3.15 menunjukkan *Antarmuka* dari menu *inbox*.



Gambar 3.5 *Antarmuka* Menu *Inbox*

4. *Outbox*

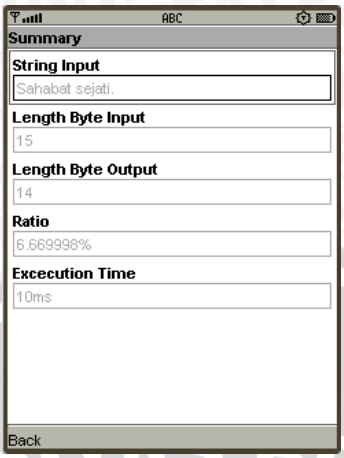
Menu ini berisi *list* pesan – pesan yang telah. *Antarmuka* menu *outbox* ini ditunjukkan pada Gambar 3.16.



Gambar 3.6 *Antarmuka* Menu *Outbox*

5. Info pesan.

Interface ini akan ditampilkan ketika *user* telah mengirimkan SMS. Pada *interface* ini ditampilkan info dari SMS yang diketikkan, berupa jumlah byte sebelum dan sesudah dilakukan kompresi, dan rasio kompresi dari SMS tersebut. *Interface* ini ditunjukkan pada Gambar 3.17.



Gambar 3.7 *Antarmuka* Info Pesan

3.7 Perancangan Pengujian

Pengujian yang akan dilakukan pada perangkat lunak ini adalah pengujian rasio SMS sebelum dilakukan kompresi dibandingkan dengan setelah proses kompresi berlangsung. Ada 2 jenis pengujian yang akan dilakukan, yaitu:

1. Pengujian pertama dilakukan menggunakan metode *Half Byte*. Pengujian ini dilakukan untuk membandingkan hasil rasio kompresi sebelum dan sesudah menggunakan *Asymmetric Half Byte*.
2. Pengujian kedua dilakukan menggunakan metode *Asymmetric Half Byte*. Pengujian ini dilakukan untuk mengetahui hasil rasio kompresi metode tersebut jika diimplementasikan

Tabel yang nantinya digunakan untuk pengujian I ditunjukkan pada Tabel 3.5. Sedangkan Tabel yang digunakan untuk pengujian II ditunjukkan pada Tabel 3.6.

Tabel 3.5 Tabel pengujian rasio kompresi dengan metode *Half Byte*

SMS ke-	Jumlah karakter Awal	Jumlah Karakter Akhir	Rasio (%)

Tabel 3.6 Tabel pengujian rasio kompresi dengan metode *Asymmetric Half Byte*

SMS ke-	Jumlah karakter Awal	Jumlah Karakter Akhir	Rasio (%)

BAB IV
IMPLEMENTASI

Bab ini menjelaskan tentang implementasi dari perancangan pada bab sebelumnya. Implementasi sistem meliputi implementasi perangkat keras dan implementasi perangkat lunak.

4.1. Implementasi Perangkat Keras

Perangkat keras yang digunakan dalam pengembangan system adalah sebagai berikut :

1. *Personal Computer*
2. Prosesor AMD Turion X2
3. Memori 1 GB
4. *Harddisk* kapasitas 160 GB
5. Monitor 14”

4.2. Implementasi Perangkat Lunak

Perangkat lunak yang akan digunakan adalah sebagai berikut:

1. Sistem Operasi Windows 7
2. JCreator Pro
3. JDK Version 6.10
4. NetBeans 6.5
5. Notepad ++
6. Microsoft Excel

4.3. Implementasi Proses

Berdasarkan analisa dan perancangan proses yang terdapat dalam bab 3, maka pada subbab ini akan dijelaskan implementasi proses – proses tersebut.

4.3.1. Implementasi Proses Kompresi

Pada proses ini dilakukan proses kompresi teks

```
String pesan ="";
      String hasil ="";
      String prev= "";
      byte[] dataByte= message.getBytes();
```

```
        int count=0;
        int start = 0;
        int hb = 0;
        for (int i=0; i<dataByte.length; i++){
                int t = (int)dataByte[i];
                if((int)dataByte[i]<0){
                        t += 256;
                }
                String temp = Integer.toBinaryString(t);
                if((int)dataByte[i]==32){  //kode utk karakter spasi diganti
kode 124 yg sebenarnya utk karakter separator "|"
                        temp = "01111100";
                } else if((int)dataByte[i] == 46){      //kode utk karakter
titik diganti kode 126 yg sebenarnya utk karakter separator "~"
                        temp = "01111111";
                } else if((int)dataByte[i] == -1){      //karakter yg sama
dengan bit penanda diganti dengan "00000000"
                        temp = "00000000";
                }

                if(temp.length() <8){
                        int m = temp.length();
                        for(int s =0; s< 8 - m; s++){
                                temp = "0" + temp;
                        }
                }
                pesan += temp;
        }

        for(int i=0; i<pesan.length(); i+=8){
                String temp = pesan.substring(i, i+8);
                if(getKey(temp).equals(prev) && i<pesan.length()-8){
                        count++;
                        hasil = hasil + temp;
                }
                else{
                        hasil = hasil + temp;
                        String temp3="";
                        if(getKey(temp).equals(prev) && i==pesan.length()-8){
                                count++;
                        }
                        hb = count + 1;
                        if(hb>=11){
                                //temp3 = variabel utk nyimpen karakter2 yg
dikompres
                                if(getKey(temp).equals(prev) &&
i==pesan.length()-8){
                                        temp3 =
hasil.substring((hasil.length())-(hb)*8,hasil.length());
                                        start = (hasil.length())-(hb)*8;
                                }
                                else{
                                        temp3 = hasil.substring((hasil.length()-
8)-(hb)*8,
                                                        hasil.length()-8);  //8
bit terakhir(karakter terakhir) tidak diikutkan.
```

```
                                start = (hasil.length()-8)-(hb)*8;
                        }
                        if(hb%2==0){
                                temp3 = temp3.substring(0,
temp3.length()-8);
                                hb--;
                        }
                        for(int rep =0; rep<hb*8; rep+=8){
                                terkompres +=
temp3.substring(rep+3, rep+8);
                        }
                        for(int rep = 0; rep <= terkompres.length();
rep+=5){      //checking gabungan bit kanan yang sama dengan bit tanda
                                if(terkompres.length()-rep > 8 &&
                                        terkompres.substring(rep,
rep+8).equals("11111111")){
                                        terkompres =
terkompres.substring(0, rep)+"00000000"+terkompres.substring(rep+8,
terkompres.length());
                                        rep+=5;
                                }
                        }
                        subKompresi = "11111111" + prev + terkompres
+"11111111";
                        hasil = hasil.substring(0, start) +
subKompresi +
                         hasil.substring(start+hb*8, hasil.length());
                    }
                    count=0;
                }
                terkompres ="";
                subKompresi ="";
                prev = getKey(temp);
        }
        return hasil;
    }
```

4.3.2 Implementasi Proses Dekompresi

File terkompresi yang dihasilkan melalui proses kompresi sebelumnya akan dikembalikan menjadi teks awal menggunakan proses dekompresi sebagai berikut :

```
public static String dekompresi(byte[] datas){
        String data ="";
        byte[] dataByte = datas;
        int jumlah = (int)dataByte[0];

        if(jumlah >= 16){
                jumlah = 16 - jumlah;
        }
        String xx ="";
        for (int ee=1; ee<dataByte.length-1; ee++){
                int t = (int)dataByte[ee];
                if((int)dataByte[ee]<0){
```

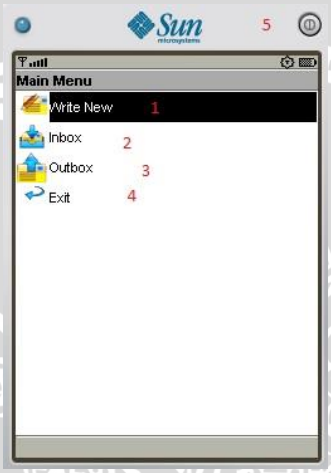
```
                t += 256;
            }
            String temp = Integer.toBinaryString(t);
            if(temp.length() <8){
                int m = temp.length();
                for(int s =0; s< 8 - m; s++){
                    temp = "0" + temp;
                }
            }
            xx += temp;
        }
        int byteAkhir = (int)dataByte[dataByte.length - 1];
        if(byteAkhir<0){
            byteAkhir += 256;
        }

        String temp2 = Integer.toBinaryString(byteAkhir);
        int temp2Int = (int)dataByte[dataByte.length - 1];
        data = xx;

        int index = -1;
        for (int i =0; i<data.length(); i+=8){
            index = data.indexOf("11111111", i);
            break;
        }

        // untuk mencari jumlah NOL yang tadinya diabaikan
        if(jumlah != 0 && temp2.length() < jumlah){
            for(int k =0; k <jumlah - temp2.length(); k++){
                temp2 = "0" + temp2;
            }
        }
        else if(index == -1){
            if(temp2.length() <8){
                int m = temp2.length();
                for(int s =0; s< 8 - m; s++){
                    temp2 = "0" + temp2;
                }
            }
        }
        data += temp2;

        String stkompres="";
        int indStart=0;
        int indEnd =0;
        String key="";
        String hasil ="";
        for(int x =0; x<=data.length(); x+=8){
            int b = data.indexOf("11111111", x);
            System.out.println("b  : "+b+"\nx : "+x+"\ndata.length
="+data.length());
            if(b==x && x<data.length()){
                indStart = b+8;
                int tempEnd = data.indexOf("11111111", indStart+1);
                while(tempEnd!= -1 && tempEnd<=data.length()-9 &&
                    data.substring(tempEnd,
tempEnd+9).equals("111111111")){
```

```
                    tempEnd++;
                    if(data.length()-9<=tempEnd){
                        tempEnd++;
                        break;
                    }
                }
                indEnd = tempEnd;
                key = data.substring(indStart, indStart+3);
                stkompres = data.substring(indStart+3, indEnd);
                x = indEnd;
                for(int y = 0; y<= stkompres.length(); y+=5){
                    if(y != 0 && stkompres.length()-y >= 8 &&
stkompres.substring(y, y+8).equals("00000000")){
                        String tt ="";
                        tt = stkompres.substring(0, y)+ "11111111" +
stkompres.substring(y+8, stkompres.length());
                        stkompres = tt;
                    }
                    if(y+5 <= stkompres.length()){
                        hasil += key + stkompres.substring(y, y+5);
                        System.out.println("hasil 1 : "+hasil );
                    }
                }
            }
            else{
                if(x+8 <= data.length()){
                    if(data.substring(x, x+8).equals("00000000")){
                        hasil+="11111111";
                    }
                    else{
                        hasil += data.substring(x, x+8);
                    }
                }
                else{
                    hasil += data.substring(x, data.length());
                }
            }
                }
            String fin ="";
            for(int y=0; y <= hasil.length()-8; y+=8){
                String temp = hasil.substring(y, y+8);
                int i = Integer.valueOf(temp, 2).intValue();
                if((char)i=='|'){
                    i = 32;
                } else if(i == 127){
                    i = 46;
                }
                fin += (char)i;
            }
            return fin;
        }
```

4.4 Implementasi Antar Muka

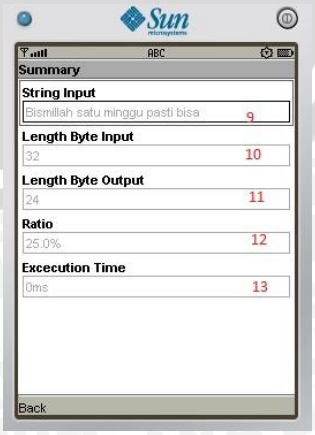
Berdasarkan rancangan antarmuka pada bab sebelumnya dibuatlah sebuah aplikasi yang menyerupai aplikasi SMS pada umumnya. Untuk tampilan pada uji coba digunakan emulator sun java wireless toolkit 2.5.2 for CLDC. Beberapa tampilan dari aplikasi yang dikembangkan dapat dilihat pada gambar berikut ini.

4.4.1 Tampilan Awal



Keterangan :

1. Menu Untuk menulis SMS
2. Menu untuk melihat SMS yang masuk
3. Menu untuk melihat SMS yang telah terkirim
4. Menu untuk keluar ke tampilan awal
5. Menu untuk keluar dari aplikasi



6. Text Box Untuk memasukkan nomer tujuan SMS
7. Text Box berisi inputan text yang akan dikirim.
8. Menu Untuk excute/mengirim text sms yang telah di input kan.
9. Text Field Tampilan SMS Awal
10. Text Field Menampilkan Byte SMS awal
11. Text Field Menampilkan Panjang Byte SMS setelah proses Kompresi
12. Text Field Menampilkan Rasio Kompresi
13. Text Field Menampilkan Kecepatan Komputasi
14. ilkan SMS hasil Kompresi, bentuk SMS awal dan akhir sama, tidak ada penambahan atau pengurangan jumlah karakter.

BAB V
PENGUJIAN DAN ANALISIS

5.1 Pengujian

Pada bab ini akan dilakukan pengujian untuk menganalisis tingkat efisiensi dari metode yang diterapkan dengan menganalisi tingkat rasio nya. Untuk pengujian rasio kompresi SMS dilakukan melalui 2 tahap pengujian. Pengujian pertama menggunakan metode kompresi Half Byte dan pengujian kedua menggunakan Asymmetric Half Byte.

Pengujian dilakukan dengan memberi inputan text pada program, yang terdiri dari berbagai variasi inputan, mulai dari yang memiliki karakter puluhan sampai ratusan. Data Uji yang digunakan dalam melakukan pengujian ditunjukkan pada Table 5.1 .

Table 5.1 Tabel Uji

SMS Ke-	Jumlah Karakter	Isi SMS
1	17	12345678912345600
2	16	12345678:., ?/!@
3	19	:P:d:)):(X_X;):*^_^
4	33	LE CEPATTT PULANGGGG!!!!!!!!!!!!
5	37	Sian9 94an, l3h p3san j3rs3y Ny 9ugk?
6	49	Mas saya mau pesan jersey 100pcs harga berapa ya?
7	72	Gan Harga Sepatu Nike Nya Brapa?Kalo Sekalian Ongkir Ke Samarinda Brapa?
8	93	Hum pulang jam brapa? bunda nitip beliin susu sama sabun buat cuci2, jgn malam2 ya pulangnya,
9	117	Mas saya telfon kok gak diangkat? Mau nipu ya? Saya sudah transfer kemaren kok sampai sekarang belum sampai paketnya?
10	141	LE DIMANA?? KOK JAM SEGINI BELUM PULANG? RAWAN LHO MALAM2 GINI NAIK MOTOR SENDIRIAN, DI JALAN KEMBAR DEPAN ITU KALO JAM SEGINI SEPI DAN GELAP
11	149	dek nanti pulang jam brapa? Emas titip beliin makanan ya, dirumah ga ada masakan, lalapan ayam yay g

		puedesss.atau kalo gak ada nasi padang juga gpp.
12	150	Sebuah SMS terdiri atas 160 karakter untuk skema 7 bit, dan ada yang memakai skema 8 bit berjumlah 140 karakter, atau 70 karakter memakai skema 16 bit
13	158	*Short Message Service* atau SMS pertama kali ditemukan oleh SGM pioners di Eropa. Standarisasi di bawah lembaga *Eutopan Telecommunications Standards Institute*.
14	166	SMS saat ini menjadi sebuah fitur mendasar dari setiap telepon selular. Fitur SMS akan berjalan ketika terdapat operator telepon seluler yang menyediakan layanan ini. Dengan SMS memungkinkan dua orang yang memiliki telepon selular dapat saling mengirimkan pesan teks satu sama lain.
15	187	Loseless Compression Teknik kompresi dimana data hasil kompresi dapat didekompres lagi dan hasilnya tepat sama seperti data sebelum proses kompresi. Contoh aplikasi: ZIP, RAR, GZIP, 7-Zip
16	187	Mengambil tiap 5 bit dari untaian bit yang dihasilkan pada langkah ke-6 dan menyambungnya dengan bit kunci yang telah dipisahkan sebelumnya. Melakukan perulangan ini sampai akhir untaian.
17	204	MIDlet merupakan bagian dari *package* javax.microedition.midlet. Perangkat *Application Management Software* (AMS) berinteraksi langsung dengan MIDlet dengan *method* MIDlet *create, start, pause, dan destroy*.
18	218	*File* atau data yang sudah dikompres agar bisa digunakan kembali harus dikembalikan lagi seperti semula. Proses pengembalian sebuah *file* yang terkompres menjadi seperti *file* aslinya disebut dekompresi atau *Decompression*
19	245	Loosy Compression Teknik kompresi dimana terdapat data yang hilang selama proses kompresi. Artinya data hasil dekompresi tidak sama dengan data sebelum kompresi namun sudah cukup untuk digunakan. Contoh: Mp3, *streaming media*, JPEG, MPEG, dan WMA
20	254	Bit penanda dapat dipilih sembarang asalkan digunakan dengan konsisten mengingat fungsi dari bit penanda ini adalah untuk menandai bahwa karakter selanjutnya

		adalah karakter yang dimampatkan. Sehingga tidak akan membingungkan pada saat proses dekompresi.

5.2 Skenario Pengujian

Sesuai dengan rancangan pengujian pada bab 3, proses pengujian dilakukan 2 tahap, yang pertama proses pengujian dilakukan dengan menghitung rasio kompresi menggunakan metode Half Byte, yang kedua proses pengujian dilakukan dengan menghitung rasio kompresi menggunakan metode *Asymmetric Half Byte*. Hasil uji dari pengujian pertama ditampilkan pada table 5.2, hasil uji dari pengujian kedua ditampilkan pada table 5.3 .

RASIO = (1- (output / input)) *100

Dimana :

- Input : Jumlah karakter awal teks SMS
- Output : Jumlah karakter akhir setelah proses kompresi

Tabel 5.2 Data Kompresi SMS Menggunakan *Half Byte*

SMS Ke-	Jumlah Karakter (*Byte*)	Hasil Kompresi *Half Byte (Byte)*	Rasio (%)
1	17	11	35.29
2	16	14	12.50
3	19	19	0
4	33	28	15.15
5	37	37	0
6	49	49	0
7	72	71	1.39
8	93	92	1.08
9	117	116	0.85
10	141	141	0

11	149	148	0.67
12	151	148	1.33
13	159	156	1.27
14	166	164	1.20
15	187	186	0.53
16	187	185	1.07
17	204	203	0.49
18	218	210	3.67
19	245	245	0
20	254	250	1.57

Dari table 5.2 dapat diketahui bahwa rasio kompresi terbesar didapatkan pada SMS ke-1 dengan rasio kompresi 35,29 %. Sedangkan rasio kompresi paling kecil didapatkan pada lebih dari 1 data uji, yaitu sebesar 0%, dengan kata lain tidak terjadi proses kompresi menggunakan Half Byte pada file tersebut karena syarat untuk melakukan kompresi Half Byte yaitu mempunyai deret 7 karakter yang memiliki 4 bit awal yang sama tidak terpenuhi.

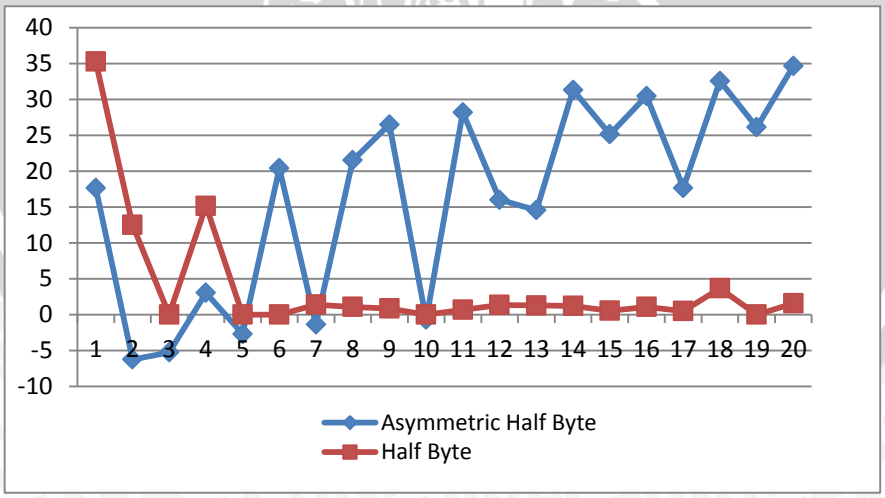
Tabel 5.3 Data Kompresi SMS Menggunakan *Asymmetric Half Byte*

SMS Ke-	**Jumlah Karakter (*Byte*)**	**Hasil Kompresi *Asymmetric Half Byte* (*Byte*)**	**Rasio (%)**
1	17	14	17.65
2	16	17	-6.25
3	19	20	-5.26
4	33	32	3.03
5	37	38	-2.70
6	49	39	20.41
7	72	73	-1.39
8	93	73	21.51
9	117	86	26.50

10	141	142	-0.71
11	149	107	28.19
12	151	126	16.00
13	159	135	14.56
14	166	114	31.33
15	187	140	25.13
16	187	130	30.48
17	204	168	17.65
18	218	147	32.57
19	245	181	26.12
20	254	166	34.65

Dari table 5.3 dapat diketahui bahwa rasio kompresi terbesar didapatkan pada SMS ke-20 dengan rasio kompresi 34,65%. Sedangkan rasio kompresi terkecil didapatkan pada SMS ke-2 dengan rasio kompresi -6,25%, dengan kata lain untuk hasil kompresi terjadi penambahan jumlah bit yg menyebabkan jumlah byte yg dihasilkan lebih besar karena terjadi kompresi atau tidak tetap ditambahkan 1 bit info pada file yang akan dikirim.

Grafik 5.1. Perbandingan Rasio Kompresi



5.3 Analisis Hasil

Hasil evaluasi perhitungan rasio kompresi menunjukkan nilai rasio yang bervariasi pada tiap data uji. Pada pengujian pertama menggunakan metode Half Byte, nilai rasio kompresi cenderung tidak banyak berubah antara satu data uji dengan data uji lainnya. Rasio rata-rata 3,90 %. Hal itu disebabkan karena karakter yang diinputkan seringkali tidak memiliki karakteristik untaian yang dapat dikompresi dengan menggunakan metode Half Byte.

Pengujian ke-2 dilakukan dengan menggunakan metode *Asymmetric Half Byte*. Rata – rata nilai rasio kompresi untuk metode ini sebesar 16,47%. Hal ini dapat terjadi karena karakter-karakter yang digunakan dalam data uji cenderung memiliki persamaan 3 bit awal sehingga dapat dikompresi menggunakan metode *Asymmetric Half Byte*.

Dari hasil pengujian ini pula dapat dilihat bahwa Asymetric Half Byte dapat menyebabkan pertambahan jumlah karakter file terkompresi pada data uji tertentu. Penambahan karakter ini berasal dari adanya bit info yang ditambahkan di awal file terkompresi. Seperti yang terlihat pada hasil pengujian pada data uji ke-2 dengan penambahan sebesar 1 byte. Sedangkan pengujian dengan metode 1 pada data uji yang sama menghasilkan rasio yang lebih baik yaitu sebesar 12.50% dengan pengurangan 2 byte karakter.

Persentase rasio terbaik menggunakan metode Half Byte diperoleh pada data uji ke-1 dengan rasio sebesar 35.29%, sedangkan *Asymmetric Half Byte* hanya menghasilkan rasio sebesar 17.65% pada data uji yang sama. Data uji ke-1 memiliki kedekatan kode ASCII antara 1 karakter dengan karakter lainnya, dan panjang pesan teks yang akan dikompresi bisa dikatakan tidak terlalu banyak. Dengan kondisi tersebut, metode Half Byte mempunyai keunggulan dibandingkan *Asymmetric Half Byte*, karena pada metode assymetric Half Byte bit kunci yang diringkas hanya 3 bit, beda dengan metode Half Byte yang diringkas 4 bit sehingga hasil kompresi metode assymetric Half Byte lebih panjang daripada metode Half Byte. Selain itu juga karena pada metode half Byte memiliki

persyaratan untuk proses kompresi yang lebih mudah yaitu 7 karakter, sedangkan *Asymmetric Half Byte* 11 karaker sama.

Dari hasil uji menggunakan Metode *Asymmetric Half Byte* pada data uji SMS ke-14,15,16,18 dapat dilihat terjadi efisiensi jumlah sms yang dikirim. Sebagai salah satu contoh pada SMS ke-14 karakter awal 166 byte yang berarti SMS tersebut seharusnya akan terkirim dalam 2 SMS (1 SMS maks 160 karakter). Tapi setelah penerapan metode diatas dihasilkan 114 byte yang artinya metode ini berhasil memberikan efisiensi dalam mengurangi jumlah karakter dalam 1 kali pengiriman SMS.

Jumlah karakter rata-rata = K + (K * R)

= 160 + (160 * 16.47%)

= 186,352 karakter

Dimana :

- K : Jumlah karakter SMS normal
- R : Rata-rata rasio kompresi *Asymmetric Half Byte*

BAB VI
KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari hasil implementasi dan analisis system serta saran untuk pengembangan system.

6.1 Kesimpulan

Dari hasil implementasi, pengujian dan analisis metode *Asymmetric Half Byte* pada data uji berupa input teks SMS dapat disimpulkan bahwa :

1. Dari pengujian yang dilakukan didapatkan nilai rata-rata rasio kompresi yang menggunakan Half Byte sebesar 3.9 %.
2. Berdasarkan pengujian dengan metode *Asymmetric Half Byte* diperoleh nilai rata-rata rasio kompresi sebesar 16.47%. dan ditinjau dari pengiriman SMS dengan rasio tersebut, rata-rata jumlah karakter yang dapat dikirimkan dalam satu kali SMS seanyak 186 karakter.
3. Metode *Asymmetric Half Byte* menghasilkan rasio kompresi terbaik jika diimplementasikan pada teks SMS yang lebih dari 11 karakter dan terdiri dari karakter *lower case*.

6.2 Saran

Beberapa saran yang dapat dipertimbangkan dalam pengembangan penelitian ini adalah :

1. Menggunakan metode kompresi *Asymmetric Half Byte* untuk mengkompresi data teks yang lain.
2. Menggabungkan metode *Asymmetric Half Byte* dengan metode lainnya untuk melakukan kompresi SMS

DAFTAR PUSTAKA

[AYU-07] Ayuningtyas, N. 2007. "Implementasi Kode Huffman dalam Aplikasi Kompresi Teks pada Layanan SMS". Jurusan Teknik Informatika Institut Teknologi Bandung. Bandung.

[KOE-09] Koesharwanto, Nanang. 2009. "Kompresi Citra Menggunakan Metode Kompresi Half Byte". Program Studi Ilmu Komputer Jurusan Matematika Universitas Brawijaya Malang. Malang.

[ORT-10]]Ortiz, Enrique. "The MIDP 2.0 Push Registry". http://developers.sun.com/mobility/midp/articles/pushreg/. Tanggal akses : 5 Agustus 2010.

[RAH-07] Raharjo, B., Imam. H dan A. Haryono. 2007. "Tuntunan Pemrograman Java Untuk Handphone". Informatika. Bandung.

[RAV-08] Ravi, K. 2008. "ASCII Table: 7-bit". Dept. of Physiology University of Wisconsin. Madison. http://www.physiology. wisc.edu/comp/docs/ascii/. Tanggal akses : 5 Agustus 2012.

[SAY-00] Sayood, K. 2000. "Introduction to Data Compression". Second Edition. Morgan Kaufmann Publishers.

[SUJ-00] Sujaini, Herry dan Yessi Mulyani. 2000. "Pemampatan File". Institut Teknologi Bandung. Bandung.

[YUS-10] Yusrian N. 2010. "Kompresi Short Message Service Menggunakan *Dynamic Half Byte*".Universitas Brawijaya Malang. Malang