Kompresi Teks Menggunakan Algoritma Huffman dan Md5 pada *Instant Messaging* Smartphone Android

M Taofik Chulkamdi, Sholeh Hadi Pramono, dan Erni Yudaningtyas

***Abstract*—Instant messaging is one of Android social media application that make user possible to send short messages directly in the same time (real time) using text to another user that online in the some network, more ever a lot of user using this application as an alternative communication via short message service (SMS). Remembering how important this application, so writer make an instant message application with small text transmission that will provide in the current needs with Huffman algorithm application. Writer also make security application model with message digest algorithm (MD5) that used to make the user comfortable so user not distributed by sniffing from another user.**

Keywords—Huffman, MD5, Instant Message, SMS

Abstrak— *Instant messaging* adalah salah satu aplikasi Android social media yang memungkinkan para pengguna dalam jaringan internet untuk mengirimkan pesan-pesan singkat secara langsung pada saat yang bersamaan (real time) dengan menggunakan teks kepada pengguna lainnya yang sedang terhubung kejaringan yang sama, bahkan banyak menggunakan aplikasi ini digunakan sebagai salah satu alternatif pengganti komunikasi *via* pesan singkat atau *Short Mesagge Sevice* (SMS). Mengingat begitu pentingnya aplikasi ini maka penulis membuat aplikasi *instant message* yang mempunyai ukuran teks transmisi yang berukuran kecil, yang mampu memenuhi kebutuhan saat ini dengan cara kompresi menggunakan algoritma Huffman. Disamping itu penulis juga membuat model security aplikasi dengan menggunakan algoritma *message digest* (MD5) yang digunakan untuk kenyamanan pengguna sehingga pengguna tidak terganggu dengan adanya *sniffing* yang dilakukan pihak pengguna lain.

Kata Kunci—Huffman, MD5, Instant Message. SMS.

I. PENDAHULUAN

SMARTPHONE Android tumbuh begitu pesat di Indonesia selain memberikan kemudahan dalam berkomunikasi, terdapat ribuan macam aplikasi yang dapat di *download* secara gratis dimana aplikasi tersebut dapat membuat *smartphone* Android menjadi lebih canggih dan multifungsi. Pada penelitian Abelson menjelaskan Android juga memberikan platform terbuka
bagi para pengembang *software* untuk menciptakan aplikasi mereka sendiri tanpa harus medapatkan *license* dari perusahaan tersebut untuk digunakan pada *smartphone*[1].

Aspek keamanan dalam proses pertukaran informasi *message* termasuk salah satu aspek yang paling diperhatikan pada saat ini untuk mengirimkan suatu pesan teks pada jaringan, kita menghadapi beberapa persoalan yaitu kerahasiaan (confidentiality), data integritas (menjamin pesan tidak diubah oleh orang lain), keaslian pesan (authentication), dan tidak ada penyangkalan (non-repudiation). Untuk itu diperlukan suatu teknik kriptografi untuk menangani persoalan ini dan diperlukan suatu teknik penyandian (enkripsi dan dekripsi) untuk menangani persoalan tersebut.

Pendse berpendapat bahwa dengan terus meningkatnya pertukaran informasi serta komunikasi, kebutuhan akan layanan data *internet* juga semakin meningkat sedangkan kapasitas *bandwidth* yang tersedia sangat terbatas sehingga tidak memungkinkan untuk mengirim pesan secara cepat dan tentunya harus menunggu respon balasan pesan dari yang kita kirim[2].

Menurut Chrocemore penggunaan data digital yang demikian besar berdampak pada kebutuhan media *storage* yang semakin besar pula. *Storage* yang besar tentu saja membutuhkan *cost* yang besar. Oleh karena itu, untuk mengatasi masalah tersebut dibutuhkan sebuah metode kompresi yang digunakan untuk memperkecil ukuran *file* yang akan ditransmisikan. Sehingga muncul ide menjadikan data berukuran lebih kecil dengan cara kompresi teks pada aplikasi *instant messaging*. Pemampatan teks bertujuan untuk menghasilkan teks yang ukurannya lebih kecil dari teks aslinya. Selain itu mampu menghemat media penyimpanan pada *smartphone* yang terbatas, kompresi teks ini sangat diperlukan agar *bandwidth* yang ada bisa digunakan secara *efisien*[3].

Berdasarkan pada deskripsi diatas bahwa keamanan serta ukuran data teks dalam komunikasi sangat dibutuhkan sehingga peneliti membuat aplikasi *instant message* yang mampu memenuhi dari aspek ukuran data teks dengan cara kompresi serta keamanan pesan menggunakan kriptografi, ada bermacam-macam algoritma kompresi yang umum digunakan, Namun penulis menggunakan algoritma Huffman untuk

M. Taofik Chulkamdi adalah Mahasiswa Program Studi Magister Teknik Elektro Universitas Brawijaya, Malang, Indonesia (email : chulkamdi@gmail.com).

Sholeh Hadi Pramono adalah Dosen Program Studi Magister Teknik Elektro Universitas Brawijaya, Malang, Indonesia (email : sholehpramono@ub.ac.id).

Erni Yudaningtyas adalah Dosen Program Studi Magister Teknik Elektro Universitas Brawijaya, Malang, Indonesia. (email : erni_yudaningtyas@yahoo.co.id).

kompresi teks. Sedangkan untuk keamanan pesan menggunakan algoritma Message Digest 5 (MD5).

II. DASAR TEORI

A. *Android*

Android adalah *Operating System* (OS) berbasis linux yang diperuntukan khusus untuk mobile device seperti *smartphone* atau PC *table*, persis seperti symbian yang dipergunakan oleh nokia dan blackberry OS, jelasnya seperti microsoft windows yang sangat dikenal baik oleh para pengguna komputer dan laptop. Pada perkembangannya sistem operasi Android telah mengalami beberapa perubahan dan perbaikan dan yang menarik versi keluaran Android diberi nama seperti nama-nama makanan[4].

B. *Encoding dan Decoding*

Encoding merupakan teknik untuk mendapatkan kode-kode tertentu (encoder). Dari kode-kode tersebut dapat diaplikasikan untuk pemampatan data dan keamanan data, dari data-data yang telah dikodekan tersebut, format-format isi dari data tersebut berbentuk kode-kode yang tidak bisa dibaca oleh *user*. Agar kode-kode tersebut bisa dibaca oleh *user*, maka kita perlu mengkodekan ulang data tersebut. Hal ini biasa dikenal dengan proses decoding, yaitu proses pengembalian kode-kode yang telah dibuat menjadi simbol-simbol yang dikenal oleh *user*. Proses decoder ini membaca *header* dari kode-kode yang berisi informasi simbol dan jumlah simbol yang digunakan, setelah pembacaan *header* proses encodern nama decoding (decoder).

C. *Kriptografi*

Kriptografi (cryptography) berasal dari bahasa Yunani yaitu dari kata crypto dan graphia yang berarti penulisan rahasia. Kriptografi adalah suatu ilmu yang mempelajari penulisan secara rahasia. Kriptografi merupakan bagian dari suatu cabang ilmu matematika yang disebut *cryptology*. Kriptografi bertujuan menjaga kerahasiaan informasi yang terkandung dalam data sehingga informasi tersebut tidak dapat diketahui oleh pihak yang tidak sah.

Kriptografi dapat memenuhi kebutuhan umum suatu transaksi:

1. Kerahasiaan (confidentiality)
2. Keutuhan (integrity)
3. Jaminan atas identitas dan keabsahan (authenticity)
4. Tidak bisa disangkal (non-repudiation)

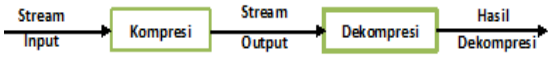
Enkripsi adalah proses mengubah suatu pesan asli (*plaintext*) menjadi suatu pesan dalam bahasa sandi (*ciphertext*).

$$\mathbf{C = E(M)}$$

Keterangan :
M = Pesan asli (Plaintext)
E = Proses enkripsi dengan *Key Private*
C = Chipertext (Plaintext yang terenkripsi)

D. *Compression*

Kompresi adalah proses mengkodekan informasi menggunakan bit atau *information bearing unit* lain yang lebih rendah dari pada refrensi data yang tidak terkodekan dengan suatu sistem enkoding tertentu. Gambar 1 mengilustrasikan proses kompresi dan dekompresi data.



Gambar. 1. Proses pengolahan data

Menurut Shanon, *entropy coding* digunakan untuk meminimalkan kesalahan dan menggunakan channel untuk informasi tranmisi dengan kapasitasnya. Semua media baik berupa teks, grafik, audio, atau video mempunyai *redudansi*. *Redudansi* (berulang) terjadi ketika representasi dari sebuah media yang berkapasitas x bytes dan y bytes (y<x), maka x merupakan suatu *redudansi* relatif terhadap y [6].

Redudansi dapat diketahui dengan terlebih dahulu mengetahui secara bertahap nilai-nilai *Compression Factor* dan *Compression Ratio*, persamaan (1) dan (2).

$$CF = \frac{\text{Ukuran Runtun Bit Input}}{\text{Ukuran Runtun Bit Output}} \quad (1)$$

$$CR = \frac{\text{Ukuran Runtun Bit Output}}{\text{Ukuran Runtun Bit Output}} \quad (2)$$

Keterangan :
CF = *Compression Factor*
CR = *Compression Ratio*

Compression Ratio adalah ukuran persentase teks yang telah berhasil dimampaatkan. Secara matematis rasio pemampatan teks ditulis dalam persamaan (3) dan (4):

Enthropy adalah jumlah informasi rata-rata dalam setiap huruf, *enteropy* ini digunakan mengetahui nilai *Redudancy* [7].

$$E = \sum_{i=1}^{n} P_1 . \log_2(P_i) = \sum_{i=1}^{n} P_i . \log_2(\frac{1}{P_1}) \quad (3)$$

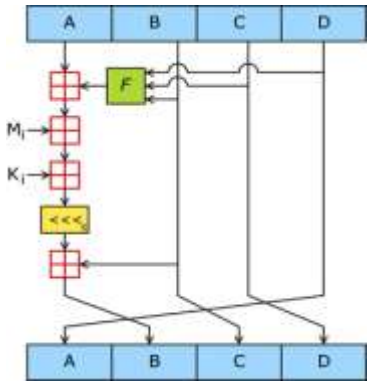
$$R = \log_2(n) + \sum_{i=1}^{n} P_1 ' \log_2(P_i) \quad (4)$$

Keterangan :
E = *Enthropy*
R = *Redudancy*
P_i = Peluang Kemuculan

E. *MD5*

MD5 adalah salah satu dari serangkaian algortima *message digest* yang didesain oleh Profesor Ronald Rivest dari MIT (1994). Saat kerja analitik menunjukkan bahwa pendahulu MD5-MD4 mulai tidak aman, MD5 kemudian didesain pada tahun 1991 sebagai pengganti dari MD4 (kelemahan MD4 ditemukan oleh Hans Dobbertin (1991). Pada tahun 1993, den Boer dan Bosselaers memberikan awal hasil dari penemuan *pseudo-collision* dari fungsi kompresi MD5. Dua vektor inisialisasi berbeda I dan J dengan beda 4 bit diantara keduanya.

MD5 mengolah blok 512 bit, dibagi kedalam 16 subblok berukuran 32 bit. Keluaran algoritma diset menjadi 4 blok yang masing-masing berukuran 32 bit yang setelah digabungkan akan membentuk nilai hash 128 bit untuk model perhitunganya pada Gambar 2.



Gambar. 2. Blok MD5

1. Penambahan bit-bit pengganjal (padding bits) : Pesan ditambahkan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 modulo 512. Jika panjang pesan jumlahnya 448 bit, makan pesan tersebut ditambah dengan 512 bit menjadi 960 bit, jadi panjang bit-bit pengganjal adalah antara 1 sampai 512. Bit-bit pengganjal terdiri dari sebuah bit 1 dan diikuti dengan bit 0 yang menjadi sisanya.
2. Penambahan nilai panjang pesan semula : Pesan yang telah diberi bit-bit pengganjal selanjutnya ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula. Jika panjang pesan >264 maka diambil adalah panjangnya dalam modulo 264.
3. Inisialisasi penyangga (buffer) MD : MD5 membutuhkan 4 buah penyangga atau buffer yang masing-masing panjangnya 4 x 32 = 128 bit. 4 buffer tersebut menampung hasil antara dan hasil akhir. Setiap penyangga diinisialisasi dengan nilai Hexadesimal, contoh : 89ABCDEF.
   word A: 01 23 45 67
   word B: 89 ab cd ef
   word C: fe dc ba 98
   word D: 76 54 32 10
4. Pengolahan pesan dalam blok berukuran 512 bit : Pesan dibagi menjadi L buah blok yang masing-masing memiliki panjang 512 bit (Y0 s.d YL-1). Setiap blok diproses dengan penyangga MD menjadi keluaran 128 bit dan hasil ini disebut proses HMD5.

III. METODE PENELITIAN

Metodologi penelitian merupakan suatu proses yang digunakan untuk memecahkan suatu masalah secara logis, dimana memerlukan data-data untuk mendukung terlaksananya suatu penelitian..

Teknik yang akan digunakan untuk menganalisis hasil dari penelitian ini adalah melakukan perbandingan menggunakan media tabel dengan cara memasukkan semua data-data yang didapat dari hasil eksperimen kedalam sebuah tabel, serta melakukan perbandingan terhadap hasil penelitian dengan kompresi Huffman dan tanpa kompresi serta melakukan uji enkripsi menggunakan algoritma MD5. Suatu hasil dapat dikatakan *valid* jika teks kompresi ditransmisikan lebih cepat dibandingkan tanpa kompresi dan setelah dilakukan kompresi akan menghasilkan teks yang sama persis dengan teks aslinya sedangkan untuk algoritma MD5 dikatatakan valid jika proses enkripsi dan dekripsi mampu mengautentikasi keaslian *password* yang telah diinputkan.

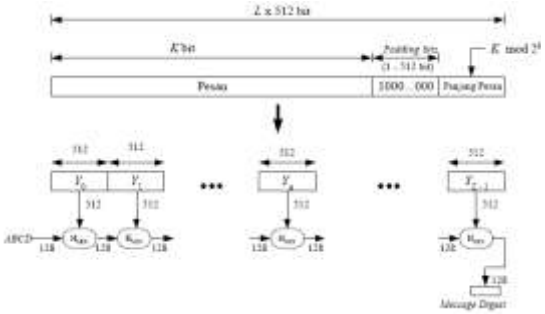
Gambar. 3. Proses pengolahan data

Desain topologi yang digunakan adalah desain topologi *star*. Pada desain topologi Gambar 3 Mikrotik difungsikan sebagai gateway pada jaringan sekaligus sebagai manajement *Traffic bandwith* . Semua akses lalulintas data baik yang masuk maupun keluar jaringan akan melewati Mikrotik RB750 terlebih dahulu untuk diperiksa baru kemudian dilanjutkan ke *server*.

Pada Gambar 3 peneliti menggunakan ip tipe C dengan karakter pada IP tipe C bahwa tiga bit pertama IP *address* kelas C selalu diset 111 sedangkan *network* ID terdiri dari 24 bit dan host ID 8 bit sisanya sehingga dapat terbentuk sekitar 2 juta *network* dengan masing-masing *network* memiliki 256 host bit pertama antara 192 - 223, Jumlah 6.384 *Range* IP 192.0.0.x.x - 223.255.255.x.x jumlah IP sebanyak 254 IP.

IV. PENGUJIAN

A. *Pengujian MD5*



Gambar 4 Alur Sistem MD5

Pengujian verifikasi ini bertujuan untuk mengetahui apakah *prototype* MD5 yang terbentuk telah berjalan sesuai dengan rancangan awal. Pengujian verifikasi yang dilakukan adalah verifikasi rancangan antarmuka pengguna dan verifikasi antarmuka. Cara kerja kriptografi algoritma MD5 adalah menerima input berupa pesan dengan ukuran sembarang dan menghasilkan *message diggest* yang memiliki panjang 128 bit. Berikut ilustrasi dari pembuatan *message diggest* pada kriptografi algoritma MD terdapat pada

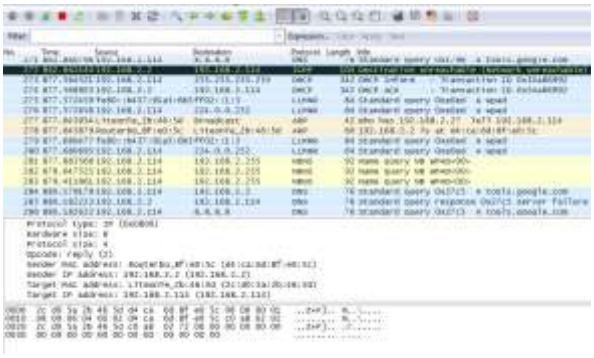
Gambar 4.

Berdasarkan penerapan algoritma didapatkan hasil pada Tabel 1.

TABEL 1
HASIL PENGUJIAN MD5

no	*Password*	*Md5*
1	12345	827ccb0eea8a706c4c34a16891f84e7b
2	abcde	ab56b4d92b40713acc5af89985d4b786
3	123de	d3278bfde7d7391b33527e5ad03c51a9
4	abc12	b2157e7b2ae716a747597717f1efb7a0
5	ab123	467953075fb15d5fcb13eb010e7cbe2b

Pada pengujian berikutnya dilakukan pengujian terhadap *imunitas password* MD5 menggunakan aplikasi whireshark dengan teknik *sniffing* terlihat pada Gambar 5:



Gambar 5 Whireshark

Gambar 6 Hasil *sniffing*

Dari hasil uji coba sniffing pada Gambar 6 *password* yang seharusnya dibaca oleh aplikasi whireshark tidak bisa terbaca karena adanya enkripsi MD5.

B. Pengujian Kompresi Huffman

Untuk simulasi perhitungan algoritma Huffman yang diimplementasikan pada halaman *message* dengan karakter input "ABACCDA" didapatkan tabel ASCII ditunjukan pada Tabel 2.

TABEL 2
SIMBOL ASCII

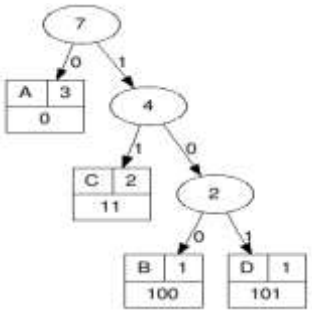
Simbol	Kode ASCII
A	01000001
B	01000010
C	01000011
D	01000100

Dengan mengikuti ketentuan pengkodean di atas, *string* 'ABACCDA' dipresentasikan menjadi rangkaian bit : 01000001010000100100000101000011010000011010001000100000 1

Kemudian membuat lintasan dari akar kedaun, akan dihasilkan kode untuk setiap simbol. Dari Gambar 6 diperoleh kode untuk masing-masing simbol asal sebagai berikut :

A = 0, B = 100, C = 11, D = 101

Sehingga simbol yang paling sering muncul memiliki kode dengan jumlah bit minimum dan tidak ada simbol yang kodenya merupakan kode awalan untuk simbol yang lain. Dengan cara ini, kode yang diawali 0 dapat dipastikan adalah A, sedangkan kode yang diawali 1 mungkin B, C atau D, yang harusnya diperiksa lagi dari bit-bit selanjutnya.



Gambar 6 Pohon Huffman

Ketika pengujian diperoleh waktu yang diperlukan pada saat kompresi dan dekompresi sedikit berubah-ubah, maka pengujian untuk setiap data teks dilakukan sebanyak sepuluh kali untuk data yang sama, kemudian diambil nilai rata-ratanya. Dalam pengujian pada Tabel 3 menggunakan model teks transmisi dengan kapasitas ukuran data 16 bit – 23.112 bit.

TABEL 3
KOMPRESI DATA TEXT

Nama File	Ukuran File Asli	Ukuran File Kompres	Persentase
Uji 1	4664 bit	3063 bit	34,3%
Uji 2	23.112 bit	15.061 bit	34,8%
Uji 3	6488 bit	4554 bit	29,8%
Uji 4	48 bit	18 bit	62,5%
Uji 5	104 bit	38 bit	63,4%
Uji 6	152 bit	65 bit	57,2%
Uji 7	32 bit	12 bit	62,5%
Uji 8	16 bit	4 bit	75%
Uji 9	96 bit	24 bit	75%

Berdasarkan pengujian ukuran data teks asli terhadap data teks kompresi ditunjukan pada Gambar 7 dan 8.

Gambar 7 Ukuran file kompresi

Kompresi teks berhasil dibuat tingkat kompresi hingga mencapai 75% jika ukuran *file* yang digunakan sebesar 16 bit dan 96 bit dan masing-masing jenis mempunyai frekuensi kemunculan karakter yang hampir sama. Kompresi teks ini kurang berhasil jika isi teks terlalu sedikit karena karakter yang muncul tidak terjadi

perulangan, sedangkan untuk *ratio* kompresi yang paling kecil didapatkan sebesar 34,3%.

Gambar 8 Ukuran file kompresi

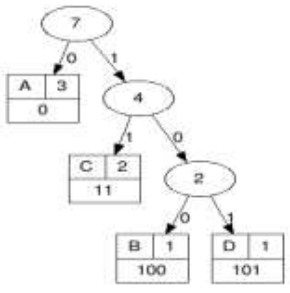
Berdasarkan pengujian ukuran symbol ASCII terhadap satu simbol pada karakter kompresi huffman terdapat pada Gambar 9.

Gambar 9 Perbandingan ASCII Terhadap Karakter Teks

Dari hasil uji coba yang sudah dilakukan, didapatkan setiap pengkodean ASCII 8 bit dengan kompresi Huffman didapatkan pengkodean setiap satu simbol karakter teks menjadi 1 – 4,6 bit.

C. Pengujian Dekompresi Algoritma Huffman

Dekompresi adalah proses pengembalian kode-kode biner yang telah dibuat menjadi karakter-karakter yang kita kenal. Proses dekompresi ini membaca *header* dari kode-kode yang berisi informasi karakter dan jumlah karakter yang digunakan, maka kode isi pesan diurai kedalam pohon Huffman, kemudian pohon Huffman ditelusuri sampai semua kode isi pesan berbentuk karakter yang sesuai dengan inputan. Untuk melakukan proses pengembalian kedata asli dengan algoritma Huffman, maka langkah-langkahnya terdapat pada Gambar 10.



Gambar 10 Dekompresi Huffman

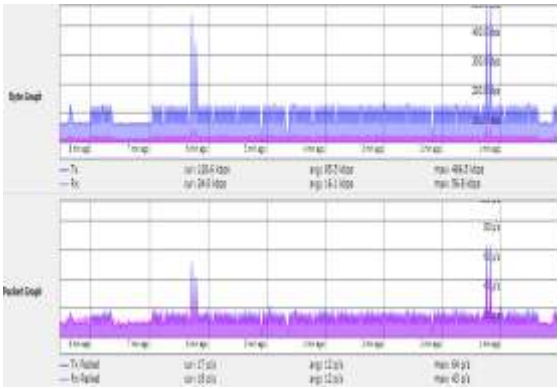
Dalam pengujian dekompresi atau proses pengembalian kode-kode yang telah terkompresi menjadi simbol-simbol yang dikenali, dari hasil pengujian data teks hasil dekompresi ditunjukan pada Tabel 4.

TABEL 4
HASIL UJI DEKOMPRESI

No Uji	Ukuran File Asli/bit	Ukuran File Kompres/bit	ASCII	Rata-rata pengkodean Setiap Simbol/bit	Entropi
1	4664	3063	8	4.254	1,052
2	23.112	15.061	8	4.213	1,053
3	6488	4554	8	4.615	1,051
4	48	18	8	2 bit	1,043
5	104	38	8	1.923	1,036
6	152	65	8	2.421	1,057
7	32	12	8	2	1,057
8	16	4	8	1	1
9	96	24	8	1	1

Hasil uji coba teknik dekompresi huffman memiliki nilai entropi yang paling baik didapatkan pada pengujian 8 dan 9 karena hasil entropi bernilai 1 sama dengan besaran pengkodean simbol yang bernilai 1.

Pada tahap ini pengujian dilakukan menggunakan Router Board 750 dengan model klien transmisi,untuk hasil graphnya terletak pada gambar 11.



Gambar 11 Graph Router

Tabel 5
UJI TRANSMISI

No	Ukuran File Asli	Ukuran File Kompres	Kecepatan Transmisi
Uji 1	4664 bit	3063 bit	18Kbps
Uji 2	23.112 bit	15.061 bit	60Kbps
Uji 3	6488 bit	4554 bit	25Kbps
Uji 4	48 bit	18 bit	14Kbps
Uji 5	104 bit	38 bit	15Kbps
Uji 6	152 bit	65 bit	17Kbps
Uji 7	32 bit	12 bit	14Kbps
Uji 8	16 bit	4 bit	13Kbps
Uji 9	96 bit	24 bit	17Kbps

Dari hasil uji coba transmisi pada Tabel 5 terlihat bahwa data teks berhasil di transmisikan dengan baik dan menghasilkan kecepatan transmisi antara 13-60 Kbps dengan ukuran data teks 4-15.061 bit.

Pada Gambar 11 menunjukan halaman *signup* digunakan untuk input pendaftaran jika *user* belum

mempunyai *account identitas* terdiri dari *username*, *password*, *password again*, e-*mail*.

Gambar 11 *Register*

Pada Gambar 12 halaman ini digunakan untuk masuk kedalam sistem stelah melakukan *registrasi* pada halaman *signup* halaman ini merupakan hasil setelah proses *registrasi* berhasil, dan juga merupakan halaman utama yang menandakan bahwa pengguna sudah memiliki hak akses untuk menjalankan aplikasi terdiri dari *username* dan *password*.

Gambar 12 *Signup*

Setelah langkah *register*, *login* dan pencarian teman Gambar 13 ditunjukan halaman yang digunakan untuk komunikasi transmisi data teks kepada *user* lain yang terhubung pada jaringan *server*.

Gambar 13 Halaman Pesan

V. KESIMPULAN DAN SARAN

Aplikasi *instant message* berbasis Android ini dapat dijalankan pada *smartphone* yang *compatible* dengan Android OS dengan spesifikasi minimal OS Ginger 2.0. Berdasarkan algoritma MD5 yang sudah diimplementasikan didapatkan kesimpulan Format *password* yang digunakan dapat berupa data *alfanumerik*, sehingga tidak mengganggu pengguna dalam membuat *password* pengguna dapat membuat *password* dengan *alfabet*, *numerik* atau gabungan keduanya, semakin panjang kunci semakin lama waktu yang dibutuhkan untuk pencarian kunci sehingga semakin bagus sistem kriptografi tersebut. Sedangkan untuk Algoritma Huffman dapat diterapkan dalam pembuatan Aplikasi Kompresi teks Berbasis Android Menggunakan Algoritma Huffman. Berdasarkan hasil uji coba, rasio hasil kompresi dan dekompresi teks yang dilakukan pada tiga jenis ponsel menghasilkan rasio kompresi yang sama.

Perlu ditambahkan fitur-fitur seperti *edit profil* bagi *register u*ser, tampilan dapat dibuat lebih rapi dan teratur dan penambahan animasi seperti *smilley* agar lebih menarik. Penambahan fitur *Create room*, dan *Delete room* bagi *register user*. Algoritma MD5 sesuai untuk aplikasi ini menjaga integritas *password*. Semakin besar ukuran kunci semakin lama untuk melakukan pencarian terhadap kunci, pengguna harus mengingat *password* yang telah dibuat karena yang disimpan telah mengalami perubahan menjadi pesan ringkas *message digest* yang berbeda dari *password* aslinya

REFERENSI

[1] , F.,C. Collins, and R. Sen. 2012. Unlocking Android – A Developer's Guide. First Edition. Manning Pub. Greenwich.

[2] Pendse, P., J, Greene. 2013. A Wellness Android Application with Social Networking Capabilities. (24) : 1-6.

[3] Chrochemore, M. and T. Lecroq. 2000. data compression algorithms. http://citeseer.nj.nec.com/19499.htm. Diakses 18 April 2014.

[4] Lee, Wei-Meng. 2011. Beginning Android™ Application development. Wiley Publishing. Indianapolis. Indiana.

[5] Utomo, E. http://mekoutomo.blogspot.com/2012/12/tabel-ascii.html. Diakses 12 Desember 2012.

[6] Richardson, I. E. G. 2003. H. 264 and MPEG-4 Video Compression Video Coding for Next-generation Multimedia. Wiley. England.

[7] Kodituwakku, S.R., U. S. Amarasinghe. 2010. Comparison of lossless data compression algorithms For data. IJCSE (1) 4 : 416-425. ISSN : 0976-5166.