IMPLEMENTASI DAN PENGUJIAN KEAMANAN BYTE-ROTATION ENCRYPTION ALGORITHM (BREA) PADA APLIKASI INSTANT MESSAGING (IM)

SKRIPSI

Untuk memenuhi sebagian persyaratan untuk mencapai gelar Sarjana Komputer

Disusun Oleh :

MUHAMMAD NURWISESO WIBISONO

0910683067

DEPARTEMEN PENDIDIKAN NASIONAL

UNIVERSITAS BRAWIJAYA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

MALANG

2013

LEMBAR PERSETUJUAN

IMPLEMENTASI DAN PENGUJIAN KEAMANAN *BYTE-ROTATION ENCRYPTION ALGORITHM* (BREA) PADA APLIKASI *INSTANT MESSAGING* (IM)

SKRIPSI

Untuk memenuhi sebagian persyaratan untuk mencapai gelar Sarjana Komputer

Disusun oleh :

MUHAMMAD NURWISESO WIBISONO

NIM. 0910683067

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Aswin Suharsono, S.T., M.T	**Ir. Heru Nurwarsito, M.Kom**
NIK. 840919 06 1 1 0251	**NIP. 19650402 199002 1 001**

LEMBAR PENGESAHAN

IMPLEMENTASI DAN PENGUJIAN KEAMANAN *BYTE-ROTATION ENCRYPTION ALGORITHM* (BREA) PADA APLIKASI *INSTANT MESSAGING* (IM)

SKRIPSI

Untuk memenuhi persyaratan memperoleh gelar Sarjana Komputer

Disusun oleh :

MUHAMMAD NURWISESO WIBISONO

NIM. 0910683067

Skripsi ini telah diuji dan dinyatakan lulus pada

tanggal 17 Juli 2013

Penguji I	Penguji II
Kasyful Amron, S.T., M.Sc	**Barlian Henryranu Prasetio, S.T. ,M.T.**
NIP. 19750803 200312 1 003	**NIK. 821024 06 1 1 0254**

Penguji III

Ismiarta Aknuranda, S.T., M.Sc., Ph.D.

NIK. 740719 06 1 1 0079

Mengetahui

Ketua Program Studi Teknik Informatika

Drs. Marji, M.T.

NIP. 19670801 199203 1 001

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 daftar pustaka.

Apabila ternyata didalam 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 dan Pasal 70).

Malang, 23 Juli 2013

Mahasiswa,

Muhammad Nurwiseso Wibisono

NIM 0910683067

KATA PENGANTAR

Puji syukur penulis panjatkan kahadirat Tuhan Yang Maha Esa, karena Rahmat-Nya lah penulis dapat menyelesaikan skripsi ini. Adapun maksud penyusunan skripsi ini adalah untuk memenuhi salah satu persyaratan dalam menempuh ujian Sarjana Program Teknologi Informasi dan Ilmu Komputer. Judul skripsi yang disusun adalah: **"IMPLEMENTASI DAN PENGUJIAN KEAMANAN ALGORITMA ENKRIPSI *BYTE-ROTATION ENCRYPTION ALGORITHM* (BREA) PADA APLIKASI *INSTANT MESSAGING* (IM)"**

Melalui kesempatan ini, penulis ingin menyampaikan rasa hormat dan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan dan dukungan selama penulisan proposal skripsi, diantaranya:

1. Ibunda Suryantini dan Ayahanda Yanto Purnomo yang saya cintai, yang tidak henti hentinya memberikan dukungan penuh, semangat serta doa selama proses pengerjaan skripsi ini berlangsung.
2. Bapak Drs. Marji, M.T, selaku Ketua Program Studi Teknik Informatika Universitas Brawijaya Malang.
3. Bapak Aswin Suharsono, S.T, M.T selaku dosen pembimbing I yang telah memberikan pengarahan dalam pembuatan proposal skripsi ini.
4. Bapak Ir. Heru Nurwarsito, M.Kom selaku dosen pembimbing II yang telah memberikan saran – saran yang sangat berguna dalam penyelesaian proposal skripsi ini.
5. Ryan, Fengky, Tika, Rangga, Ika, Ardy, teman-teman forum diskusi skripsi, teman-teman angkatan 2009 dan pihak yang telah membantu yang tidak dapat saya sebutkan satu per satu.

Penulis menyadari bahwa proposal skripsi ini jauh dari sempurna, oleh karena itu untuk segala kritik dan saran yang membangun penulis ucapkan terima kasih. Penulis mengharapkan semoga proposal skripsi ini dapat berguna bagi yang membutuhkannya.

Malang, 23 Juli 2013

Penulis

ABSTRAK

Muhammad Nurwiseso Wibisono.2013. Implementasi Dan Pengujian Keamanan *Byte-Rotation Encryption Algorithm* (BREA) Pada Aplikasi *Instant Messaging* (IM). Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya, Malang. Dosen Pembimbing:Aswin Suharsono, S.T., M.T dan Ir. Heru Nurwarsito, M.Kom

Pesan Instan atau *Instant Messaging* telah menjadi hal yang umum digunakan di kalangan masyarakat dalam melakukan komunikasi. Namun media komunikasi ini tidak selalu aman dari serangan *Sniffer*. Kriptografi dapat menyelesaikan masalah tersebut. Dengan banyaknya pilihan untuk algoritma kriptografi, pembuat aplikasi dapat memilih berdasarkan kekuatan dan kecepatan dari algoritma kriptografi ketika di implementasikan. Penelitian ini akan mengimplementasikan *Byte-Rotation Encryption Algorithm* (BREA) dalam aplikasi pesan instan. Penelitian ini bertujuan untuk menguji seberapa cepat algoritma dapat di eksekusi dan seberapa besar pengaruh implementasi algoritma pada sistem pesan instan. Selain mengimplementasikan algoritma BREA, penelitian ini juga melakukan pengujian keamanan. Pengujian keamanan bertujuan untuk mengetahui seberapa aman algoritma tersebut ketika diimplementasikan kedalam pesan instan. Serangan yang digunakan untuk pengujian keamanan adalah serangan Brute Force. Sistem pesan instan akan di implementasikan kedalam aplikasi komputer menggunakan sistem *point-to-point* dalam sebuah jaringan lokal. Dari hasil pengujian aplikasi yang telah dilakukan, BREA memiliki waktu eksekusi yang tidak jauh berbeda dengan pesan instan yang tidak diberi enkripsi. Namun, BREA dapat dipecahkan dengan Brute Force dengan waktu kurang lebih 3 jam.

Kata Kunci : Pesan Instan, *Sniffer,* Kriptografi, *BREA, Brute Force*

ABSTRACT

Muhammad Nurwiseso Wibisono.2013. ***Implementation and Security Testing of Byte-Rotation Encryption Algorithm (BREA) on Instant Messaging (IM).*** *Information Technology and Computer Science Program, Brawijaya University,* Malang. *Advisor*:Aswin Suharsono, S.T., M.T dan Ir. Heru Nurwarsito, M.Kom

Instant Messaging has become a common thing among people in their use of communication. However, the communication medium is not always safe from Sniffer Attack. Cryptography can solve the problem. With so many options for cryptographic algorithms, application developers can choose based on the strength and speed of the cryptographic algorithms when implemented. This research will be implementing Byte-Rotation Encryption Algorithm (BREA) into an instant messaging application. This research aims to test how fast the algorithm can be executed and how much influence of implementation of the algorithm on an instant messaging system. In addition to implementing the algorithm BREA, this research also do security testing. Security testing aims to determine how safe the algorithms when implemented into an instant message. Attack method that used for security testing is Brute Force Attack. Instant messaging system will be implemented into computer applications using point-to-point system in a local network. From the results of application testing that has been done, BREA has execution time that is not much different from an instant message that is not encrypted. However, BREA can be solved by brute force with less than 3 hours.

Keywords : *Instant Messaging, Sniffer, Cryptography,BREA, Brute Force*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1. Latar Belakang

Chatting merupakan satu media yang dimanfaatkan oleh masyarakat untuk berkomunikasi. Sebuah layanan yang disediakan oleh berbagai macam *social networking* yang dapat menghubungkan pengguna dengan pengguna lainnya melalui media pesan tulis ini. Saat ini pengguna internet dapat melakukan hal-hal yang melanggar hak dan privasi dari pengguna lainnya. Salah satunya adalah penggunaan *sniffing* yaitu tindakan menangkap paket data yang bertebaran dalam sebuah jaringan[ZEF-11]. Dan salah satu media yang digunakan untuk melakukan aksi *sniffing* ini adalah media *chatting*. Tanpa sepengetahuan pengguna yang sedang ber-*chatting* ria pengguna lain dapat melakukan *sniffing* tersebut. Sehingga kenyamanan dan keamananpun mulai terancam.

Untuk menghindari pencurian data melalui *sniffing,* pembuat aplikasi dapat menggunakan berbagai cara. Salah satunya adalah kriptografi dimana data diacak menggunakan suatu kunci menjadi sesuatu yang sulit dibaca oleh seseorang yang tidak memiliki kunci tersebut[KSE-10]. Dengan kriptografi data yang dikirimkan lebih aman dibandingkan pengiriman teks biasa. *Byte-Rotation Encryption Algorithm* (BREA) merupakan salah satu algoritma yang baru. Dengan algoritma yang rumit, algoritma ini dapat dimanfaatkan untuk enkripsi pada aplikasi *Chatting*.

Berdasarkan hal tersebut, penulis merancang sebuah aplikasi *chatting* berbasis *Local Area Network* (LAN) dan menggunakan enkripsi *Byte-Rotation Encryption Algorithm* (BREA) yang akan di implementasikan dalam aplikasi *chatting* tersebut. Penulis ingin meneliti lebih lanjut mengenai kehandalan dari algoritma *Byte-Rotation Encryption Algorithm* (BREA) ketika di implementasikan kedalam sistem pesan instan (IM).

1.2. Rumusan masalah

Berdasarkan uraian latar belakang di atas, maka dapat dirumuskan permasalah pada tugas akhir ini yaitu sebagai berikut:

1. Bagaimana kriptografi *Byte-Rotation Encryption Algorithm* (BREA) di implementasikan kedalam Instant Messaging?
2. Berapa lama waktu yang dibutuhkan untuk melakukan proses dekripsi *Byte-Rotation Encryption Algorithm* (BREA) tanpa memiliki key yang digunakan untuk enkripsi?

1.3. Batasan Masalah

Agar permasalahan yang dirumuskan dapat lebih terfokus, maka penelitian tugas akhir ini dibatasi dalam hal:

1. Algoritma kriptografi yang akan diuji keamanannya adalah algoritma *Byte-Rotation Encryption Algorithm* (BREA) dan serangan yang digunakan untuk uji keamanan algoritma kriptografi tersebut adalah serangan *Brute Force*.
2. Pada penelitian ini tidak dilakukan proses *sniffing*. Diasumsikan bahwa data yang digunakan untuk penelitian ini telah dimiliki oleh *Cipheranalyst*.
3. Kehandalan yang dimaksud dalam penelitian ini adalah kehandalan dari segi kecepatan eksekusi BREA dan keamanan dari segi waktu yang dibutuhkan untuk memecahkan BREA tanpa memiliki key.

1.4. Tujuan

Mengimplementasikan kriptografi *Byte-Rotation Encryption Algorithm* kedalam aplikasi pesan instan dan menguji kehandalan kriptografi *Byte-Rotation Encryption Algorithm*.

1.5. Manfaat

Penelitian ini diharapkan dapat bermanfaat untuk berbagai pihak. Manfaat dari penelitian ini adalah sebagai berikut:

Bagi Penulis :

1. Memberikan keamanan pada pesan instan terhadap *sniffer* pada jaringan
2. Mengetahui kehandalan dari BREA ketika di implementasikan kedalam pesan instan
3. Dapat mengimplementasikan ilmu yang telah didapatkan dari perkuliahan di PTIIK Universitas Brawijaya

Bagi Pembaca :

1. Mengetahui seluk-beluk algoritma kriptografi *Byte-Rotation Encryption Algorithm* beserta tingkat keamanan saat diserang oleh serangan *Brute Force*.

1.6. Sistematika Penulisan

Sistematika penulisan laporan tugas akhir ini adalah sebagai berikut:

BAB I Pendahuluan

Memuat latar belakang permasalahan, identifikasi dan pembatasan masalah, rumusan masalah, tujuan dan manfaat, dan sistematika penulisan.

BAB II Dasar teori

Menjelaskan teori yang terkait dengan metode enkripsi *Byte-Rotation Encryption Algorithm*

BAB III Metode Penelitian

Membahas metode yang digunakan dalam penelitian yang terdiri dari studi literatur, perancangan, implementasi, pengujian dan analisis, serta pengambilan kesimpulan dan saran.

BAB IV Perancangan

Membahas analisis kebutuhan dan perancangan untuk pengujian keamanan aplikasi yang menggunakan metode enkripsi *Byte-Rotation Encryption Algorithm*

BAB V Implementasi

Membahas tentang implementasi dari *Byte-Rotation Encryption Algorithm*

BAB VI Pengujian dan Analisis

Memuat proses dan hasil pengujian terhadap metode yang telah direalisasikan.

BAB VII Penutup

Memuat kesimpulan serta saran yang diperoleh dari pembuatan dan pengujian metode untuk pengembangan lebih lanjut.

1.7. Timeline penelitian

Berikut Timeline pengerjaan penelitian :

Tabel 1.1. Timeline Penelitian

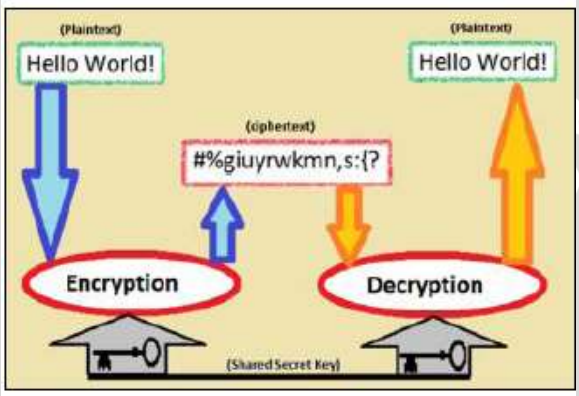
No	PROSES KEGIATAN :	BULAN DAN MINGGU KE:																			
		Bulan I				Bulan II				Bulan III				Bulan IV				Bulan V			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Studi literatur																				
2	Penyusunan dasar teori																				
3	Analisa dan perancangan perangkat lunak																				
4	Implementasi perangkat lunak																				
5	Pengujian perangkat lunak																				
6	Penulisan laporan penelitian																				

Sumber : Perancangan

BAB II
DASAR TEORI

2.1. Kriptografi

Kriptografi adalah ilmu mengenai teknik enkripsi yang dilakukan dengan cara mengacak data menggunakan suatu kunci enkripsi menjadi sesuatu yang sulit dibaca oleh seseorang yang tidak memiliki kunci dekripsi[KSE-10]. Pesan asli yang dirahasiakan dinamakan plainteks, sedangkan pesan acak hasil penyandian dinamakan cipherteks. Cipherteks dapat dikembalikan ke plainteks oleh orang yang berhak, biasanya orang tersebut mengetahui metode penyandian atau memiliki kunci (key) penyandian. Proses penyandian pesan disebut enkripsi dan proses pembalikan penyandiannya disebut dekripsi [WAP-12].



Gambar 2.1. Ilustrasi Kriptografi
Sumber : [WAP-12]

2.1.1. Pesan, Plainteks dan Cipherteks

Pesan adalah data atau informasi yang dapat dibaca dan dimengerti maknanya. Nama lain untuk pesan asli adalah plainteks (*plaintext*) atau teks-jelas (*cleartext*).

Pesan dapat berupa data atau informasi yang dikirim (melalui kurir, saluran komunikasi data, dsb) atau yang disimpan di dalam media perekaman (kertas, *storage* komputer, dsb).

2.1.2. Cipher Blok (Block Cipher)

Pada blok cipher, rangkaian bit-bit plainteks dibagi menjadi blok-blok bit dengan panjang sama, biasanya 64 bit (tapi adakalanya lebih). Algoritma enkripsi menghasilkan blok cipherteks yang – pada kebanyakan sistem kriptografi simetri – berukuran sama dengan blok plainteks. [MRT-04]

2.2. Matriks (Matematika)

Matriks adalah kumpulan elemen yang tersusun dalam sebuah persegi panjang[MAT-13]. Panjang dan isi dari matriks disesuaikan dengan kebutuhan. Adapun elemen disusun berdasarkan baris dan kolom. Contoh matriks 3x3 akan ditunjukkan pada gambar 2.2.

$$\begin{pmatrix} 2 & 1 & -1 \\ 0 & 4 & 3 \\ -5 & 0 & -2 \end{pmatrix}$$

Gambar 2.2. Ilustrasi Matriks 3x3
Sumber : [MAT-13]

Matriks memiliki pengindeksan. Indeks ditujukan untuk memudahkan pemanggilan nilai pada posisi tertentu. Seperti ilustrasi dibawah ini. σ_{11} Memiliki nilai yang berada pada baris 1 dan kolom 1. Jika mengacu pada Gambar 2, maka σ_{11} bernilai 2.

$$[\sigma] = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix}$$

Gambar 2.3. Ilustrasi Index Matriks
Sumber : [RDA-00]

Ada beberapa fungsi matematika matriks yang digunakan dalam penelitian ini. Adapun beberapa fungsi tersebut adalah sebagai berikut :

2.2.1. Transpose Matriks

Transpose matriks mengubah baris menjadi kolom dan kolom menjadi baris. Pada gambar 2.4 akan diilustrasikan tentang transpose matriks.

$$A = \begin{pmatrix} 2 & 4 & -1 \\ 0 & 3 & 5 \end{pmatrix}, A^t = \begin{pmatrix} 2 & 0 \\ 4 & 3 \\ -1 & 5 \end{pmatrix}$$

Gambar 2.4. Ilustrasi Transpose Matriks
Sumber : [MAT-13]

Matriks A merupakan matriks 2x3. Ketika di transpose maka menjadi matriks 3x2. Adapun nilai dari kolom berpindah menjadi nilai dari baris. Sesuai dengan fungsinya, transpose mengubah baris menjadi kolom dan kolom menjadi baris.

2.2.2. Penjumlahan Matriks

Penjumlahan matriks sama dengan penjumlahan pada umumnya. Berbeda dengan perkalian matriks, penjumlahan dilakukan dengan menjumlahkan sesuai dengan indexnya. Ilustrasi penjumlahan matriks akan ditunjukkan pada gambar 2.5.

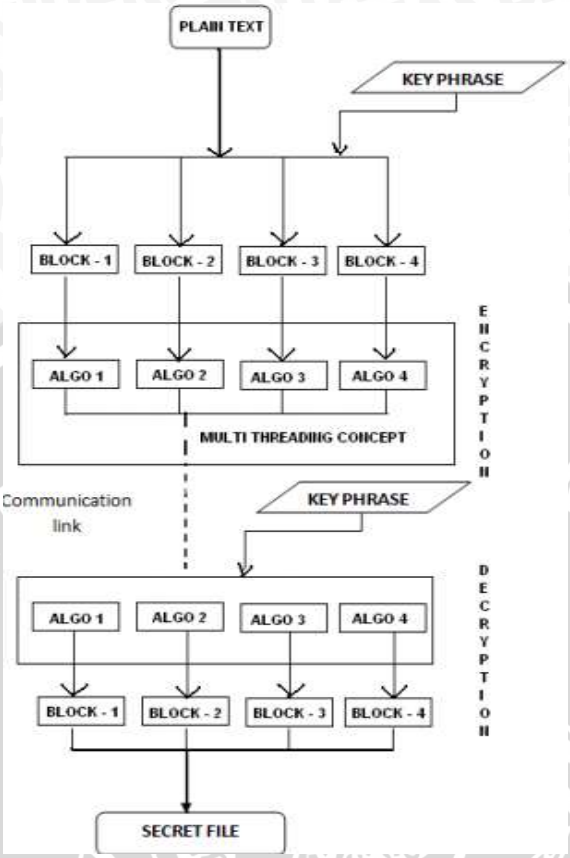
$$\begin{pmatrix} 1 & 2 \\ 0 & -2 \\ 3 & 7 \end{pmatrix} + \begin{pmatrix} 6 & -\frac{1}{4} \\ 1 & 2 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 7 & \frac{7}{4} \\ 1 & 0 \\ 3 & 7 \end{pmatrix}$$

Gambar 2.5. Ilustrasi Penjumlahan Matriks
Sumber : [MAT-13]

Pada ilustrasi, dimisalkan A_{11} (matriks kiri) bernilai 1, dan B_{11} (matriks kanan) bernilai 6, maka matriks C_{11} (hasil penjumlahan) bernilai 7. Begitu seterusnya dengan baris lainnya.

2.3. Byte-Rotation Encryption Algorithm (BREA)

BREA adalah sebuah algoritma yang disusun sedemikian rupa untuk mengamankan data. Algoritma berjalan dengan memanfaatkan Blok Cipher dan fungsi aljabar matriks yang tidak begitu rumit. Alur kerja dari *Byte-Rotation Encryption Algorithm* akan dijelaskan pada gambar 2.6.



Gambar 2.6. Alur BREA
Sumber : [BSU-12]

Dari ilustrasi di atas, dapat disimpulkan bahwa BREA memecah plaintext menjadi beberapa blok yang akan diproses menggunakan algoritma per blok tersebut menggunakan sebuah key. Adapun langkah dari algoritma ini adalah sebagai berikut:

2.3.1. Tahap awal : Input

Berikut akan dijelaskan secara singkat bagaimana teknik BREA ini bekerja. Dimisalkan input kalimat adalah 'RAYMOND SUITINGS'. Kalimat tersebut akan dipecah menjadi blok yang masing-masing blok terdiri dari 16 karakter. Blok yang digunakan adalah blok yang berisi matriks 4 x 4. Masing-masing karakter menempati satu slot matriks pada blok yang disediakan. Ilustrasi pengisian slot pada satu blok akan dijelaskan pada gambar 2.7.

M =

R	A	Y	M
O	N	D	S
U	I	T	I
N	G	S	0

Gambar 2.7. Posisi Awal Karakter
Sumber : [BSU-12]

Ketika jumlah huruf yang dibagi per-blok kurang dari jumlah slot pada matriks per-blok, maka akan diisi 0 (contoh pada matriks M_{44}). Setelah kalimat dimasukkan kedalam matriks maka di konversi menjadi angka. Konversi tersebut didasarkan pada ketentuan huruf A hingga Z direpresentasikan sebagai angka 1 hingga 26 secara berurut. Kemudian angka 1-9 direpresentasikan sebagai angka 27-35. Dan angka 0 bernilai tetap (0). Berikut ilustrasi perubahan blok huruf menjadi angka :

M_p =

18	1	25	13
15	14	4	19
21	9	20	9
14	7	19	0

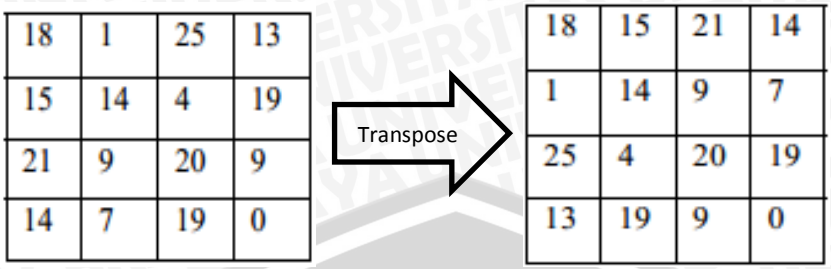
Gambar 2.8. Konversi Blok Huruf Menjadi Blok Angka
Sumber : [BSU-12]

2.3.2. Enkripsi

Untuk pengenkripsian data, BREA memanfaatkan transpose matriks, penjumlahan matriks, serta rotasi/pergeseran matriks baik vertikal maupun horizontal. Berikut fungsi yang berjalan pada BREA

2.3.2.1. Transpose Matriks

Setelah blok matriks di konversi menjadi angka, blok tersebut di transpose. Transpose matriks mengubah baris menjadi kolom atau sebaliknya.



Gambar 2.9. Ilustrasi Transpose
Sumber : [BSU-12]

Pada ilustrasi diatas telah menggambarkan bagaimana transpose dalam matriks berjalan. Terjadi pertukaran antara matriks dari baris 2 dan kolom 1 menjadi baris 1 dan kolom 2 dan begitu seterusnya. Ilustrasi hasil matriks transpose akan dijelaskan pada gambar 2.10.

Mp^T =

18	15	21	14
1	14	9	7
25	4	20	19
13	19	9	0

Gambar 2.10. Hasil Transpose Dari Blok Angka
Sumber : [BSU-12]

2.3.2.2. Key Generator

Key yang digunakan pada algoritma ini merupakan key yang dirandomisasi dari 1 hingga 26 dan dibatasi 16 karakter sesuai dengan jumlah slot pada matriks 4 x 4. Kemudian angka tersebut dimasukkan kedalam matriks untuk tahap selanjutnya. Ilustrasi randomisasi key akan ditunjukkan pada gambar 2.11.

K =

25	15	3	9
20	7	13	8
5	18	22	17
21	12	26	24

Gambar 2.11. Key Dari Hasil Randomisasi
Sumber : [BSU-12]

Setelah randomisasi, kemudian masing-masing slot pada matriks dilakukan proses matematis. Yaitu dengan masing-masing slot di modulus 2. Ilustrasi hasil dari modulus 2 akan ditunjukkan pada gambar 2.12.

1	1	1	1
0	1	1	0
1	0	0	1
1	0	0	0

Gambar 2.12. Hasil Key Modulus 2
Sumber : [BSU-12]

2.3.2.3. Penjumlahan Blok dan Key

Setelah mendapatkan key yang di buat oleh *key generator*, maka dilakukan penjumlahan matriks dengan key. Setiap blok yang berisi matriks akan dijumlahkan dengan key. Dimensi yang digunakan pada matriks dan key merupakan matriks 4 x 4. Ilustrasi penjumlahan blok dan key akan dijelaskan pada gambar 2.13.

C_{pk} =

18	15	21	14
1	14	9	7
25	4	20	19
13	19	9	0

+

1	1	1	1
0	1	1	0
1	0	0	1
1	0	0	0

Gambar 2.13. Penjumlahan Blok Dengan Key
Sumber : [BSU-12]

Setelah blok dijumlahkan dengan key, hasil tersebut akan digunakan untuk tahap selanjutnya. Ilustrasi hasil dari penjumlahan ditunjukkan pada gambar 2.14.

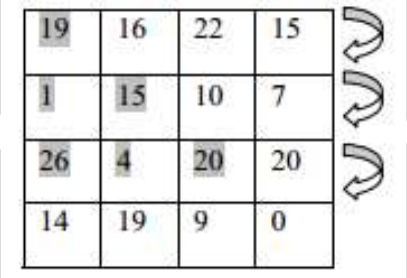
C_{pk} =

19	16	22	15
1	15	10	7
26	4	20	20
14	19	9	0

Gambar 2.14. Hasil Penjumlahan Blok dan Key
Sumber : [BSU-12]

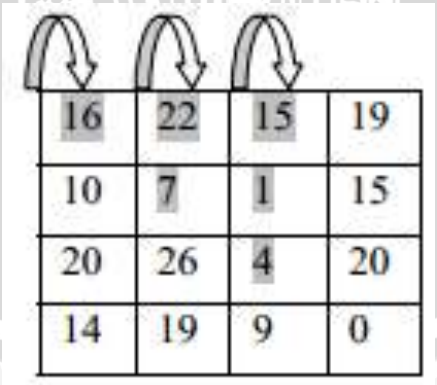
2.3.2.4. Pergeseran Matriks

Pada algoritma ini, digunakan pergeseran matriks. Pergeseran yang dilakukan adalah pergeseran secara horizontal, kemudian dilanjutkan dengan pergeseran secara vertikal. Setelah mendapatkan hasil penjumlahan blok dan key, matriks tersebut digeser secara horizontal, kemudian digeser secara vertikal. Gambar 2.15. menunjukkan ilustrasi pergeseran matriks secara horizontal.



Gambar 2.15. Pergeseran Secara Horizontal
Sumber : [BSU-12]

Masing-masing baris bergeser sesuai dengan baris dimana slot tersebut berada. Pergeseran yang dilakukan adalah pergeseran ke kiri. Sedangkan banyak geseran yang dilakukan bergantung pada di baris mana ia berada. Pada algoritma ini diterangkan bahwa baris pertama mengalami 1 pergeseran, baris kedua mengalami 2 pergeseran, dan baris ketiga mengalami 3 pergeseran. Sedangkan baris ke empat tidak mengalami pergeseran. Gambar 2.16. menunjukkan hasil pergeseran matriks secara horizontal dan ilustrasi dari pergeseran secara vertikal.



Gambar 2.16. Pergeseran Secara Vertikal
Sumber : [BSU-12]

Setelah digeser secara horizontal, maka dilakukan pergeseran secara vertikal. Pergeseran vertikal ini adalah pergeseran ke atas. Adapun skema yang

dilakukan sama dengan pergeseran horizontal. Kolom pertama mengalami 1 pergeseran, kolom kedua mengalami 2 pergeseran, kolom ketiga akan mengalami 3 pergeseran dan kolom keempat tidak mengalami pergeseran. Gambar 2.17. menunjukkan hasil dari pergeseran secara vertikal.

10	26	9	19
20	19	15	15
14	22	1	20
16	7	4	0

Gambar 2.17. Hasil Pergeseran Secara Vertikal
Sumber : [BSU-12]

2.3.3. Pengembalian dari Angka menjadi Huruf

Setelah tahap enkripsi selesai, matriks angka akan dikonversi menjadi huruf sesuai dengan ketentuan awal yang telah dijelaskan mengenai konversi huruf ke angka. Gambar 2.18. menunjukkan hasil konversi dari angka ke huruf

J	Z	I	S
T	S	O	O
N	V	A	T
P	G	D	0

Gambar 2.18. Hasil Konversi Angka ke Huruf
Sumber : [BSU-12]

Setelah dikembalikan menjadi huruf, blok data hasil dari pergeseran vertikal akan dijadikan satu dan dikembalikan kedalam bentuk teks (*ciphertext*).

2.4. Jenis Serangan Cipher

Menurut Eric Conrad [ECO-13], serangan pada cipher dapat berupa serangan *Known Plaintext* dan *Ciphertext Only*. *Known Plaintext* adalah serangan dimana seseorang dapat mengakses *plaintext* yang dibuat dan *ciphertext* dari hasil *plaintext* tersebut untuk dipelajari. Sedangkan serangan *Ciphertext Only* dilakukan hanya dengan menggunakan *ciphertext* saja.

2.5. *Hash* dan *Message Digest 5*

Hash merupakan sebuah fungsi yang digunakan untuk menerima masukan berupa teks yang panjangnya tidak dibatasi dan mengkonversikan teks awal menjadi teks yang sudah diproses dengan panjang teks yang tetap. Pada umumnya *hash* bersifat satu arah yaitu tidak dapat dikembalikan menjadi teks semula. Salah satu contoh dari fungsi hash satu arah adalah *Message Digest 5* [MRF-04].

Message Digest 5 (MD5) merupakan hash satu arah yang umum digunakan saat ini. MD5 merupakan perbaikan dari MD4 yang telah berhasil dipecahkan oleh *cryptanalyst.* Adapun hasil dari MD5 adalah hexadesimal yang berjumlah 32 karakter.

2.6. Data Kriptografi AES, DES Dan 3DES

Ada berbagai macam algoritma kriptografi yang digunakan dalam mengamankan data. Beberapa algoritma standar yang umum digunakan adalah AES, DES dan 3DES (Pengembangan dari DES). Pada tabel 2.1, akan ditunjukkan hasil penelitian yang dilakukan oleh Hamdan O. Alanazi beserta rekanannya mengenai perbandingan AES, DES dan 3DES terhadap 9 faktor terkait.

Tabel 2.1. Perbandingan AES, 3DES dan DES Terhadap 9 Faktor

Factors	AES	3DES	DES
Key Length	128,192,or 256 bits	(k1,k2, and k3)168 bits (k1 and k2 is same) 112 bits	56 bits
Cipher Type	Symmetric block cipher	Symmetric block cipher	Symmetric block cipher
Block Size	128,192, or 256 bits	64 bits	64 bits
Developed	2000	1987	1977

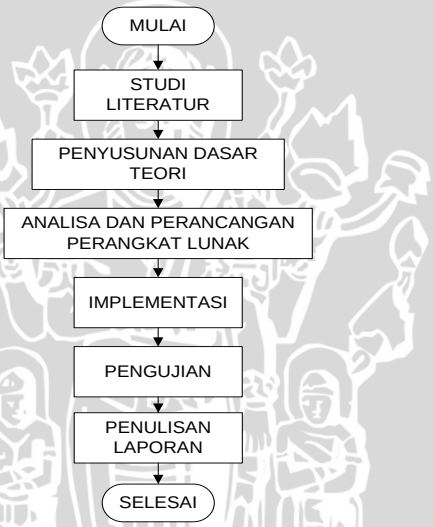
Cryptanalysis resistance	Strong againts differential, truncated differential linear, interpolation and square attack	Vulnerable to differential, Brute Force attacker could be analyze plain text using different cryptanalysis	Vulnerable to differential and linear cryptanalysis; weak subtitution tables
Security	Considered secure	One only weak which is Exit in DES	Proven inadequate
Possible Keys	2^{128}, 2^{192}, or 2^{256}	2^{112}, 2^{168}	2^{56}
Possible ASCII Printable Character Keys	95^{15}, 95^{24}, or 95^{32}	95^{14} or 95^{21}	95^{7}
Time required to check all possible keys at 50 billion keys per second**	For a 128-bit key: $5x10^{21}$ years	For a 112-bit key : 800 days	For a 56-bit key : 400 days

Sumber :[AHO-10]

Dari data diatas dapat disimpulkan bahwa berdasarkan waktu yang dibutuhkan untuk memecahkan masing-masing algoritma, maka tingkat keamanan paling tinggi merupakan kriptografi AES ($5x10^{21}$ tahun), kemudian 3DES (800 hari), dan DES (400 hari).

BAB III
METODOLOGI PENELITIAN

Pada bab ini dijelaskan langkah-langkah yang akan dilakukan dalam pengerjaan tugas akhir, yaitu studi literature, penyusunan dasar teori, analisa dan perancangan, implementasi, analisis dan pengujian dari aplikasi perangkat lunak yang akan dibuat, hingga penulisan laporan. Kesimpulan dan saran disertakan sebagai catatan atas aplikasi dan kemungkinan arah pengembangan aplikasi selanjutnya. Gambar 3.1. mengilustrasikan diagram alir runtutan pengerjaan penelitian ini:



Gambar 3.1. Flowchart Runtutan Pengerjaan Penelitian
Sumber : Perancangan

3.1. Studi Literatur

Studi literatur mempelajari mengenai penjelasan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori pendukung tersebut diperoleh dari buku, jurnal, e-book, dan dokumentasi project.

3.2. Penyusunan Dasar Teori

Penyusunan dasar teori dilakukan setelah mendapatkan referensi yang tepat untuk mendukung penulisan penelitian ini. Teori-teori pendukung tersebut meliputi:

1. Kriptografi
2. Matriks (Matematika)
3. Metode Byte-Rotation Encryption Algorithm

3.3. Analisa Dan Perancangan

Analisa kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan oleh sistem yang akan dibangun dan diuji. Analisa kebutuhan dilakukan dengan mengidentifikasi kebutuhan sistem dan siapa saja yang terlibat didalamnya. Berikut analisis kebutuhan dalam penelitian yang akan dilakukan

3.3.1. Kebutuhan Jaringan

Jaringan yang dibutuhkan dalam penggunaan program ini adalah sebagai berikut :

1. Jaringan dapat menggunakan kabel maupun nirkabel.
2. Jaringan yang digunakan adalah jaringan *Local Area Network* (LAN).
3. Dapat melakukan koneksi antar komputer.

3.3.2. Kebutuhan Perangkat Lunak Pesan Instan

Fungsi-fungsi yang dimiliki oleh perangkat lunak pesan instan adalah sebagai berikut:

1. Program harus mampu menggunakan algoritma *Byte-Rotation Encryption Algorithm* yang memiliki berbagai macam fungsi seperti penjumlahan, rotasi, dan penyimpanan data kedalam array.
2. Program pesan instan harus mampu berinteraksi dengan program lain pada jaringan.

3.3.3. Kebutuhan Perangkat Lunak Pengujian Keamanan

Fungsi-fungsi yang dimiliki oleh perangkat lunak pengujian keamanan adalah sebagai berikut:

1. Program harus mampu menggunakan algoritma *Byte-Rotation Encryption Algorithm* yang memiliki berbagai macam fungsi seperti penjumlahan, rotasi, dan penyimpanan data kedalam array.
2. Program pengujian keamanan mampu menyimpan kamus yang ada kedalam array dan mampu mencoba memecahkan *Byte-Rotation Encryption Algorithm*.

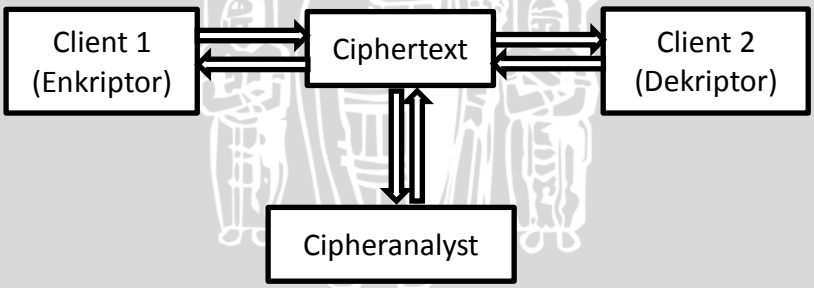
3.3.4. Kebutuhan Data

Data yang digunakan oleh perangkat lunak ini adalah sebagai berikut:

1. Data berupa kamus kata yang digunakan untuk pengujian keamanan.
2. Data berupa teks pesan instan yang digunakan untuk berkomunikasi.

3.3.5. Skenario Penelitian

Penelitian ini ditujukan untuk implementasi dan menguji keamanan BREA pada aplikasi berbasis chatting. Untuk pengujian dibutuhkan 3 jenis elemen. Client 1, Client 2, dan *Cipheranalyst*. Ilustrasi tentang skenario penelitian akan dijelaskan pada gambar 3.2.

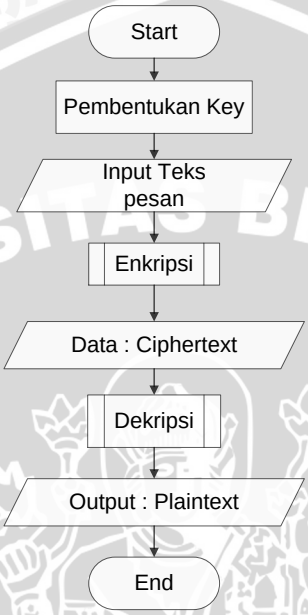


Gambar 3.2. Skenario Penelitian
Sumber : Perancangan

Teknologi yang diimplementasikan dalam aplikasi pesan instan ini adalah menggunakan *Socket Programming*. *Client 1* melakukan proses *Listen*. Kemudian inisiasi koneksi via *Socket* oleh *client* dan koneksi chat berjalan. Namun untuk menguji keamanan dibutuhkan *Cipheranalyst* yang berperan sebagai 'pencuri informasi' dari *Client 1* Maupun *Client 2*.

3.3.6. Diagram Alir Proses

Diagram alir menggunakan notasi-notasi untuk menggambarkan arus data yang membantu dalam proses memahami jalannya aplikasi. Gambar 3.3. mengilustrasikan diagram alir mengenai proses kriptografi BREA berjalan.



Gambar 3.3. Diagram Alir Proses
Sumber : Perancangan

Program ini dimulai dengan melakukan pembentukan key terlebih dahulu. Pembentukan key dilakukan ketika program pada client 1 dan client 2 akan melakukan inisiasi koneksi. Kemudian dilakukan input teks oleh client 1. Teks yang di inputkan akan di enkripsi sesuai dengan BREA. Setelah di enkripsi data akan dikirim menuju client 2 dan akan di dekripsi sesuai dengan BREA yang menjadi hasil keluaran dari proses dekripsi yaitu plainteks.

3.4. Implementasi

Implementasi aplikasi dilakukan dengan mengacu pada diagram alir proses. Implementasi perangkat lunak menggunakan bahasa pemrograman Python. Implementasi meliputi :

- Pembuatan/ penggunaan *user interface* sebagai tool untuk implementasi aplikasi
- Implementasi BREA pada pesan instan.

- implementasi Brute Force untuk BREA dengan tujuan pengujian keamanan.

3.5. Pengujian

Dalam penelitian ini, pengujian akan dilakukan 2 tahap. Yaitu :

1. Pengujian Perangkat Lunak
2. Pengujian Keamanan BREA

3.5.1. Pengujian Perangkat Lunak

Pengujian perangkat lunak pada penelitian ini dilakukan untuk menujukkan bahwa perangkat lunak telah mampu bekerja sesuai dengan spesifikasi kebutuhan serta menguji kehandalan BREA ketika di implementasikan kedalam aplikasi pesan instan. Adapun pengujian yang akan dilakukan adalah sebagai berikut:

1. Perbandingan pengaruh waktu antara pesan instan yang tidak menggunakan enkripsi dengan pesan instan yang menggunakan BREA. Contoh tabel pengujian akan ditunjukkan pada tabel 3.1.

Tabel 3.1. Contoh Tabel Pengujian Pesan Instan

No	Kata	Jumlah Karakter	Key	Waktu Eksekusi	
				Dengan BREA	Tanpa BREA
1					
...					
10					
Rata-Rata					

Sumber : Perancangan

2. Pengujian Algoritma BREA. Pengujian ini ditujukan untuk mengetahui kecepatan Enkripsi dan Dekripsi yang dapat dilakukan oleh BREA. Contoh tabel pengujian akan ditunjukkan pada tabel 3.2.

Tabel 3.2. Contoh Tabel Pengujian Algoritma

No	Kata	Jumlah Karakter	Key	Waktu Eksekusi	
				Enkripsi	Dekripsi
1					
...					
10					
Rata-Rata					

Sumber : Perancangan

3.5.2. Pengujian Keamanan BREA

Pengujian Keamanan BREA dilakukan untuk menguji keamanan dari algoritma ini ketika di implementasikan kedalam aplikasi pesan instan (IM). Pengujian ini ditujukan untuk mengetahui berapa lama waktu yang dibutuhkan untuk melakukan dekripsi BREA tanpa memiliki key. Adapun yang akan dilakukan dalam pengujian keamanan adalah melakukan serangan *Brute Force* terhadap key. Kemudian melakukan analisis cipher. Berikut detil dari penelitian yang dilakukan:

3.5.2.1. Serangan Brute Force

Pada penelitian ini, Brute Force digunakan untuk melakukan pencarian key secara *exhaustive* yang berarti pencarian semua kemungkinan key yang digunakan. Serangan Brute Force didalam penelitian ini dilakukan secara berurut (*sequential*). Untuk mendapatkan key yang sesungguhnya, nilai key akan ditambah 1 setiap kali key yang digunakan tidak sesuai dengan aturan yang ditentukan (Dalam kasus ini, aturan yang dimaksud adalah pencocokan antara sumber dan teks setelah didekripsi dengan key Brute Force). Key BREA menggunakan 16 bit, sehingga nilai desimal maksimum ketika semua bit key bernilai 1 adalah 65535. Brute Force ini akan digunakan untuk analisis cipher.

3.5.2.2. Analisis Cipher

Dalam penelitian ini akan digunakan 2 tipe serangan. Yaitu :

a. Known Plain Text and Ciphertext Attack

Serangan tipe ini memanfaatkan plainteks dan cipherteks dalam pendekripsian secara Brute Force. Serangan ini membandingkan cipherteks yang sudah di dekripsi secara Brute Force dengan plainteks yang sebenarnya diketikkan oleh pengirim. Dalam penelitian ini akan diuji berapa lama waktu yang dibutuhkan untuk melakukan Brute Force dengan metode ini. Contoh tabel pengujian akan ditunjukkan pada tabel 3.3.

Tabel 3.3. Contoh Tabel Pengujian Pesan Instan

No	Kata	Jumlah Karakter	Ciphertext	Jumlah Karakter	Waktu Eksekusi
1					
...					
10					
Rata-Rata					

Sumber : Perancangan

b. Known Ciphertext Attack

Serangan tipe ini hanya menggunakan *ciphertext* sebagai satu-satunya informasi yang diterima dari pengirim tanpa mengetahui key namun mengetahui pola algoritma yang digunakan. Tanpa key dan plainteks, serangan ini memanfaatkan kamus sebagai pembanding kata yang di dekripsi secara Brute Force. Serangan ini melakukan validasi apakah kata tersebut merupakan kata yang *valid* atau dekripsi yang salah. Pada penelitian ini akan diuji apakah *ciphertext* yang dikirimkan baik dari client 1 maupun client 2 dapat di dekripsi sesuai dengan penjelasan diatas. Contoh tabel pengujian akan ditunjukkan pada tabel 3.4.

Tabel 3.4. Contoh Tabel Pengujian Pesan Instan

No	Ciphertext	Jumlah Karakter	Waktu Eksekusi	Plainteks ditemukan?
1				
...				
10				
Rata-Rata				

Sumber : Perancangan

3.6. Penulisan Laporan

Laporan Penelitian ditulis selama semua proses pengerjaan tugas akhir. Laporan berisi dokumentasi perancangan aplikasi serta hasil penelitian yang berupa informasi mengenai kehandalan dan keamanan BREA.

BAB IV
PERANCANGAN

Bab ini membahas mengenai perancangan implementasi BREA terhadap aplikasi pesan instan. Perancangan yang dilakukan meliputi analisa kebutuhan, perancangan perangkat lunak, dan contoh penghitungan manual sistem yang akan dibangun. Tahap analisis kebutuhan terdiri dari dua langkah yaitu analisis data yang diperlukan, identifikasi aktor. Proses perancangan perangkat lunak mempunyai tiga tahap, yaitu perancangan diagram blok, perancangan proses sistem, dan perancangan algoritma.

4.1. Analisis Kebutuhan

Analisis kebutuhan bertujuan untuk mendapatkan semua kebutuhan yang diperlukan oleh sistem yang akan dibangun. Analisis kebutuhan dilakukan dengan mengidentifikasi kebutuhan sistem dan siapa saja yang terlibat di dalamnya. Berikut analisis kebutuhan untuk implementasi dan pengujian BREA.

4.1.1. Analisis Data

Analisis data bertujuan untuk mendapatkan struktur penyimpanan data yang dibutuhkan sistem perangkat lunak untuk pengujian keamanan BREA. Berikut analisis data yang dibutuhkan untuk penelitian ini :

1. Data berupa teks pesan instan yang diambil dari ChatBox milik forum gogamers(http://forum.gogamers.us/misc.php?do=ccarc)
2. Data kamus kata dasar bahasa indonesia serta kamus kata yang sering digunakan dalam jejaring sosial
3. Data hasil enkripsi (*ciphertext*) yang digunakan untuk pengujian keamanan BREA

4.1.2. Identifikasi Aktor

Tabel 4.1. Identifikasi Aktor

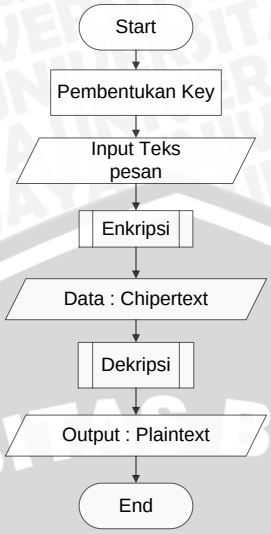
Aktor	Deskripsi
Client 1	Client 1 merupakan aktor pengguna yang menjadi inisiator pesan instan. Client 1 menentukan port dimana koneksi pesan instan akan dilakukan
Client 2	Client 2 merupakan aktor pengguna yang melakukan inisiasi menuju Client 1. Client 2 menggunakan port dan alamat IP milik Client 1 untuk terhubung dengan pesan instan
Cipheranalyst	*Cipheranalyst* merupakan aktor pengguna yang melakukan pengujian keamanan terhadap BREA dengan cara melakukan analisis *cipher*

Sumber : Perancangan

4.2. Perancangan Proses

Perancangan proses merupakan perancangan tahap atau urutan sistem untuk melakukan proses kriptografi BREA pada aplikasi pesan instan. Data masukan yang digunakan pada aplikasi ini adalah input berupa pesan teks. Input pesan ini dapat dilakukan oleh client 1 dan client 2. Pesan teks yang dikirimkan akan di enkripsi dan di dekripsi sesuai dengan alur yang sudah ada

Proses pertama yang dilakukan oleh kedua client adalah melakukan sinkronisasi key yang akan digunakan. Setelah sinkronisasi, client 1 dapat melakukan input teks pada aplikasi dan mengirimkan tersebut kepada client 2. Teks yang dikirimkan akan di enkripsi menggunakan BREA. Kemudian client 2 akan melakukan dekripsi menggunakan BREA. setelah di dekripsi maka akan muncul hasil keluaran berupa teks yang dikirimkan oleh client 1. Diagram alir proses yang dilakukan akan ditunjukkan pada gambar 4.1.



Gambar 4.1. Diagram Alir Proses
Sumber : Perancangan

Sesuai dengan diagram alir proses dan tabel aktor, maka akan ada 4 proses yang digunakan untuk implementasi algoritma ini. Yaitu:

4.2.1. Generator Key

Pada program ini, key dirancang dapat dimasukkan oleh user. Key dapat berupa teks apa saja selama kedua client menggunakan teks yang sama. Kemudian key tersebut akan di konversi berdasarkan library MD5. Dengan menggunakan library MD5, maka key yang sama ketika dikonversi MD5 akan memiliki hasil hexadesimal yang sama. Kemudian hasil dari konversi MD5 tersebut akan di modulus 2. Algoritma yang digunakan akan ditunjukkan pada tabel 4.2.

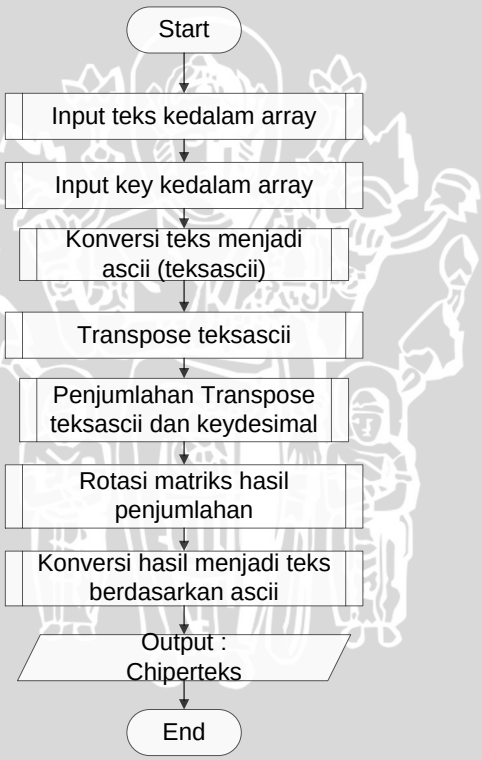
Tabel 4.2. Algoritma Generator Key

Nama Algoritma : Generator Key
Deskripsi : ➔ Masukan : Key dari User ➔ Proses : • Proses Key menjadi MD5 dengan mengambil 16 karakter saja • Konversi MD5 menjadi desimal • Modulus 2 hasil konversi • Menyimpan Key yang digunakan Output : Key

Sumber : Perancangan

4.2.2. Enkripsi

Sesuai dengan dasar teori, BREA menggunakan beberapa langkah dalam melakukan proses enkripsi dan dekripsi data. dimulai dari memasukkan teks dan key kedalam array. Kemudian melakukan konversi array teks menjadi array desimal berdasarkan tabel ASCII. Kemudian melakukan transpose pada array desimal. Setelah di transpose maka array tersebut akan dijumlahkan dengan key sesuai dengan indeks array yang dimiliki. Kemudian dilakukan rotasi secara horizontal dan vertikal, kemudian dikembalikan ke bentuk teks. Alur dari proses enkripsi BREA akan dijelaskan pada gambar 4.2.



Gambar 4.2. Proses Enkripsi BREA
Sumber : Perancangan

4.2.2.1. Input Teks / Key Kedalam Array

Proses input teks/key kedalam array merupakan proses untuk memasukkan teks/key kedalam array 4x4. Jika melebihi slot array tersebut maka

akan dibuat array lain yang dimensinya 4x4. Algoritma yang digunakan akan ditunjukkan pada tabel 4.3.

Tabel 4.3. Algoritma Input Teks / Key Kedalam Array

Nama Algoritma : Input Teks/ Key Kedalam Array
Deskripsi : ➔ Masukan : Teks pesan instan / Key Proses : • Melakukan input masukan kedalam list • Untuk setiap 16 karakter yang dimiliki oleh list, maka o indeks *block* bertambah 1 • Untuk setiap *block* yang ada : o Jika huruf kurang dari 16 maka ▪ Membuat 4 indeks array untuk baris (0,1,2,3) ▪ Untuk setiap indeks baris • Membuat list kolom • Mengisi kolom dengan huruf sesuai urutan ▪ Memasukkan list kolom pada indeks baris o Memasukkan indeks baris pada *block* • Mengembalikan nilai *block* Output : Pengembalian nilai *block*

Sumber : Perancangan

4.2.2.2. Konversi Karakter Menjadi Desimal berdasarkan ASCII

Proses ini dilakukan untuk merubah karakter menjadi desimal berdasarkan tabel ASCII. Algoritma yang digunakan akan ditunjukkan pada tabel 4.4.

Tabel 4.4. Algoritma Konversi Karakter Menjadi Desimal

Nama Algoritma : Konversi karakter Menjadi desimal berdasarkan ASCII
Deskripsi : ➔ Masukan : *block* Proses : • Untuk setiap *block* yang ada : o Untuk setiap indeks baris (4) ▪ Untuk setiap indeks kolom (4) • Konversi dari huruf menjadi angka berdasarkan ascii • Pengembalian nilai *block* Output : Pengembalian nilai *block*

Sumber : Perancangan

4.2.2.3. Transpose Blok

Proses ini dilakukan untuk melakukan transpose pada setiap array yang ada pada blok. Algoritma yang digunakan akan ditunjukkan pada tabel 4.5.

Tabel 4.5. Algoritma Transpose Blok

Nama Algoritma : Transpose *block*

Deskripsi :
➔ Masukan : *block*
Proses :
• Membuat array temp
• Untuk setiap *block* yang ada:
  ○ X bernilai 0
  ○ Untuk x kurang dari panjang *block* dikurangi satu
  ▪ Y bernilai x ditambah 1
  ▪ Untuk y kurang dari panjang *block*
  • Pertukaran *block*(x,y) menjadi *block*(y,x)
  • Y ditambah 1
  ▪ X ditambah 1
• Pengembalian nilai *block*
Output : Pengembalian nilai *block*

Sumber : Perancangan

4.2.2.4. Penjumlahan Blok dengan Key

Proses ini dilakukan untuk menjumlahkan nilai array yang ada pada blok dengan key yang sudah di buat oleh kedua client. Algoritma yang digunakan akan ditunjukkan pada tabel 4.6.

Tabel 4.6. Algoritma Penjumlahan Blok dengan key

Nama Algoritma : Penjumlahan *block* dengan key

Deskripsi :
➔ Masukan : *block*
Proses :
• Membuat array sejenis dengan *block*
• Untuk setiap *block* yang ada :
  ○ Untuk setiap indeks baris (4)
  ▪ Untuk setiap indeks kolom (4)
  • Melakukan penjumlahan kolom dengan key sesuai indeks
  • Jika kolom bernilai lebih dari 127 maka kolom dikurangi 127
• Pengembalian nilai *block*

Output : Pengembalian nilai *block*

Sumber : Perancangan

4.2.2.5. Rotasi Blok secara Horizontal

Proses ini dilakukan untuk melakukan rotasi secara horizontal. Rotasi yang dilakukan pada proses enkripsi adalah rotasi secara horizontal ke kiri. Jumlah pergeseran disesuaikan dengan jumlah baris (kecuali baris 4). Algoritma yang digunakan akan ditunjukkan pada tabel 4.7.

Tabel 4.7. Algoritma Rotasi Blok secara horizontal

Nama Algoritma : Rotasi *block* secara Horizontal

Deskripsi :
➔ Masukan : *block*
Proses :
• Untuk setiap *block* yang ada :
  ○ Untuk setiap indeks baris (4)
  ▪ Melakukan pergeseran yaitu:
  ▪ Baris 1 bergeser ke kiri sebanyak 1 kali
  ▪ Baris 2 bergeser ke kiri sebanyak 2 kali
  ▪ Baris 3 bergeser ke kiri sebanyak 3 kali
• Pengembalian nilai *block*
Output : Pengembalian nilai *block*

Sumber : Perancangan

4.2.2.6. Rotasi Blok secara Vertikal

Proses ini dilakukan untuk melakukan rotasi secara vertikal. Rotasi yang dilakukan pada proses enkripsi adalah rotasi secara vertikal ke atas. Jumlah pergeseran disesuaikan dengan jumlah kolom (kecuali kolom 4). Algoritma yang digunakan akan ditunjukkan pada tabel 4.8.

Tabel 4.8. Algoritma Rotasi Blok secara vertikal

Nama Algoritma : Rotasi *block* secara Vertikal

Deskripsi :
➔ Masukan : *block*
Proses :
• Membuat array sejenis dengan *block*
• Untuk setiap *block* yang ada :
  ○ Untuk setiap indeks baris (4)
  ▪ Untuk setiap indeks kolom (4)

• Melakukan pergeseran yaitu: • Kolom 1 bergeser ke atas sebanyak 1 kali • Kolom 2 bergeser ke atas sebanyak 2 kali • Kolom 3 bergeser ke atas sebanyak 3 kali • Pengembalian nilai *block* Output : Pengembalian nilai *block*

Sumber : Perancangan

4.2.2.7. Konversi desimal Menjadi karakter berdasarkan ASCII

Proses ini dilakukan untuk mengembalikan nilai desimal menjadi karakter berdasarkan tabel ASCII agar data dapat dilihat ke bentuk teks. Algoritma yang digunakan akan ditunjukkan pada tabel 4.9.

Tabel 4.9. Algoritma konversi desimal menjadi karakter

Nama Algoritma : Konversi desimal Menjadi karakter berdasarkan ASCII
Deskripsi : ➔ Masukan : *block* Proses : • Untuk setiap *block* yang ada : o Untuk setiap indeks baris (4) ▪ Untuk setiap indeks kolom (4) • Konversi dari huruf menjadi angka berdasarkan ascii • Pengembalian nilai *block* Output : Pengembalian nilai *block*

Sumber : Perancangan

4.2.2.8. Mengembalikan Blok menjadi Teks

Proses ini dilakukan untuk menggabungkan huruf yang ada pada blok menjadi teks agar data dapat dilihat ke bentuk teks. Algoritma yang digunakan akan ditunjukkan pada tabel 4.10.

Tabel 4.10. Algoritma mengembalikan Blok menjadi teks

Nama Algoritma : Mengembalikan *block* menjadi teks
Deskripsi : ➔ Masukan : *block* Proses : • Membuat array teks • Untuk setiap *block* yang ada:

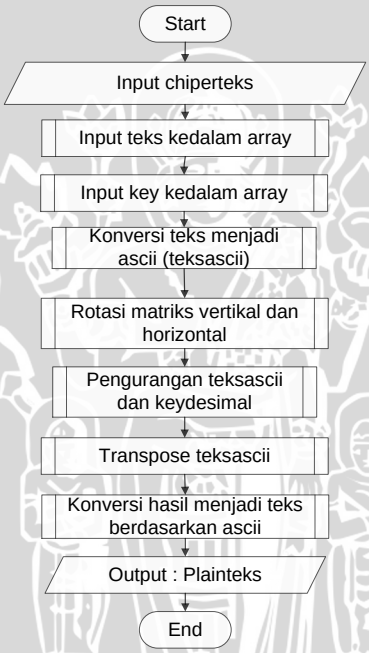
- Menggabungkan huruf dan menyimpan kedalam array teks
- Pengembalian string hasil penggabungan array teks

Output : Ciphertext hasil enkripsi BREA

Sumber : Perancangan

4.2.3. Dekripsi

Dekripsi merupakan proses kebalikan dari enkripsi. Pada BREA, algoritma yang akan berubah dari algoritma yang digunakan pada enkripsi adalah pembalikan rotasi secara vertikal dan horizontal, dan pengurangan nilai desimal teks dengan key. Ilustrasi dekripsi akan ditunjukkan pada gambar 4.3. :



Gambar 4.3. Proses Dekripsi BREA
Sumber : Perancangan

4.2.3.1. Input Teks / Key kedalam Array

Proses input teks/key kedalam array merupakan proses untuk memasukkan teks/key kedalam array 4x4. Jika melebihi slot array tersebut maka akan dibuat array lain yang dimensinya 4x4. Algoritma yang digunakan akan ditunjukkan pada tabel 4.11.

Tabel 4.11. Algoritma input teks / key kedalam array

Nama Algoritma : Input Teks/ Key Kedalam Array
Deskripsi : ➔ Masukan : Teks pesan instan / Key Proses : • Melakukan input masukan kedalam list • Untuk setiap 16 karakter yang dimiliki oleh list, maka o indeks *block* bertambah 1 • Untuk setiap *block* yang ada : o Jika huruf kurang dari 16 maka ▪ Membuat 4 indeks array untuk baris (0,1,2,3) ▪ Untuk setiap indeks baris • Membuat list kolom • Mengisi kolom dengan huruf sesuai urutan ▪ Memasukkan list kolom pada indeks baris o Memasukkan indeks baris pada *block* • Mengembalikan nilai *block* Output : Pengembalian nilai *block*

Sumber : Perancangan

4.2.3.2. Konversi Teks Menjadi Desimal berdasarkan ASCII

Proses ini dilakukan untuk merubah karakter menjadi desimal berdasarkan tabel ASCII. Algoritma yang digunakan akan ditunjukkan pada tabel 4.12.

Tabel 4.12. Algoritma Konversi Karakter Menjadi Desimal

Nama Algoritma : Konversi Teks Menjadi desimal berdasarkan ASCII
Deskripsi : ➔ Masukan : *block* Proses : • Untuk setiap *block* yang ada : o Untuk setiap indeks baris (4) ▪ Untuk setiap indeks kolom (4) • Konversi dari huruf menjadi angka berdasarkan ascii • Pengembalian nilai *block* Output : Pengembalian nilai *block*

Sumber : Perancangan

4.2.3.3. Rotasi Blok Secara Vertikal

Proses ini merupakan kebalikan dari proses enkripsi dimana dilakukan rotasi secara vertikal ke bawah. Jumlah pergeseran disesuaikan dengan jumlah kolom (kecuali kolom 4). Algoritma yang digunakan akan ditunjukkan pada tabel 4.13.

Tabel 4.13. Algoritma Rotasi Blok secara vertikal

Nama Algoritma : Rotasi *block* secara Vertikal
Deskripsi : ➔ Masukan : *block* Proses : • Membuat array sejenis dengan *block* • Untuk setiap *block* yang ada : o Untuk setiap indeks baris (4) ▪ Untuk setiap indeks kolom (4) • Melakukan pergeseran yaitu: • Kolom 1 bergeser ke bawah sebanyak 1 kali • Kolom 2 bergeser ke bawah sebanyak 2 kali • Kolom 3 bergeser ke bawah sebanyak 3 kali • Pengembalian nilai *block* Output : Pengembalian nilai *block*

Sumber : Perancangan

4.2.3.4. Rotasi Blok secara Horizontal

Proses ini merupakan kebalikan dari proses enkripsi dimana dilakukan rotasi secara horizontal ke kanan. Jumlah pergeseran disesuaikan dengan jumlah baris (kecuali baris 4). Algoritma yang digunakan akan ditunjukkan pada tabel 4.14.

Tabel 4.14. Algoritma Rotasi Blok secara horizontal

Nama Algoritma : Rotasi *block* secara Horizontal
Deskripsi : ➔ Masukan : *block* Proses : • Untuk setiap *block* yang ada : o Untuk setiap indeks baris (4) ▪ Melakukan pergeseran yaitu: ▪ Baris 1 bergeser ke kiri sebanyak 1 kali ▪ Baris 2 bergeser ke kiri sebanyak 2 kali ▪ Baris 3 bergeser ke kiri sebanyak 3 kali

• Pengembalian nilai *block* Output : Pengembalian nilai *block*

Sumber : Perancangan

4.2.3.5. Penngurangan Blok dengan Key

Proses merupakan kebalikan dari proses enkripsi dimana dilakukan untuk mengurangkan nilai array yang ada pada blok dengan key yang sudah di buat oleh kedua client. Algoritma yang digunakan akan ditunjukkan pada tabel 4.15.

Tabel 4.15. Algoritma Pengurangan Blok dengan key

Nama Algoritma : Pengurangan *block* dengan key
Deskripsi : ➔ Masukan : *block* Proses : • Membuat array sejenis dengan *block* • Untuk setiap *block* yang ada : o Untuk setiap indeks baris (4) ▪ Untuk setiap indeks kolom (4) • Melakukan pengurangan kolom dengan key sesuai indeks • Jika kolom bernilai kurang dari 127 maka kolom ditambah 127 • Pengembalian nilai *block* Output : Pengembalian nilai *block*

Sumber : Perancangan

4.2.3.6. Transpose Blok

Proses ini dilakukan untuk melakukan transpose pada setiap array yang ada pada block. Algoritma yang digunakan akan ditunjukkan pada tabel 4.16.

Tabel 4.16. Algoritma Transpose block

Nama Algoritma : Transpose *block*
Deskripsi : ➔ Masukan : *block* Proses : • Untuk setiap *block* yang ada: o X bernilai 0 o Untuk x kurang dari panjang *block* dikurangi satu ▪ Y bernilai x ditambah 1 ▪ Untuk y kurang dari panjang *block*

• Pertukaran *block*(x,y) menjadi *block*(y,x) • Y ditambah 1 ▪ X ditambah 1 • Pengembalian nilai *block* Output : Pengembalian nilai *block*

Sumber : Perancangan

4.2.3.7. Konversi Desimal Menjadi Karakter berdasarkan ASCII

Proses ini dilakukan untuk mengembalikan nilai desimal menjadi karakter berdasarkan tabel ASCII agar data dapat dilihat ke bentuk teks. Algoritma yang digunakan akan ditunjukkan pada tabel 4.17.

Tabel 4.17. Algoritma Konversi Desimal Menjadi Karakter

Nama Algoritma : Konversi desimal Menjadi karakter berdasarkan ASCII
Deskripsi : ➔ Masukan : *block* Proses : • Untuk setiap *block* yang ada : o Untuk setiap indeks baris (4) ▪ Untuk setiap indeks kolom (4) • Konversi dari huruf menjadi angka berdasarkan ascii • Pengembalian nilai *block* Output : Pengembalian nilai *block*

Sumber : Perancangan

4.2.3.8. Mengembalikan Blok menjadi Teks

Proses ini dilakukan untuk menggabungkan huruf yang ada pada blok menjadi teks agar data dapat dilihat ke bentuk teks. Algoritma yang digunakan akan ditunjukkan pada tabel 4.18.

Tabel 4.18. Algoritma Mengembalikan Blok Menjadi Teks

Nama Algoritma : Mengembalikan *block* menjadi teks
Deskripsi : ➔ Masukan : *block* Proses : • Membuat array teks • Untuk setiap *block* yang ada: o Menggabungkan huruf dan menyimpan kedalam array teks

• Pengembalian string hasil penggabungan array teks Output : -

Sumber : Perancangan

4.2.4. Cipheranalyst

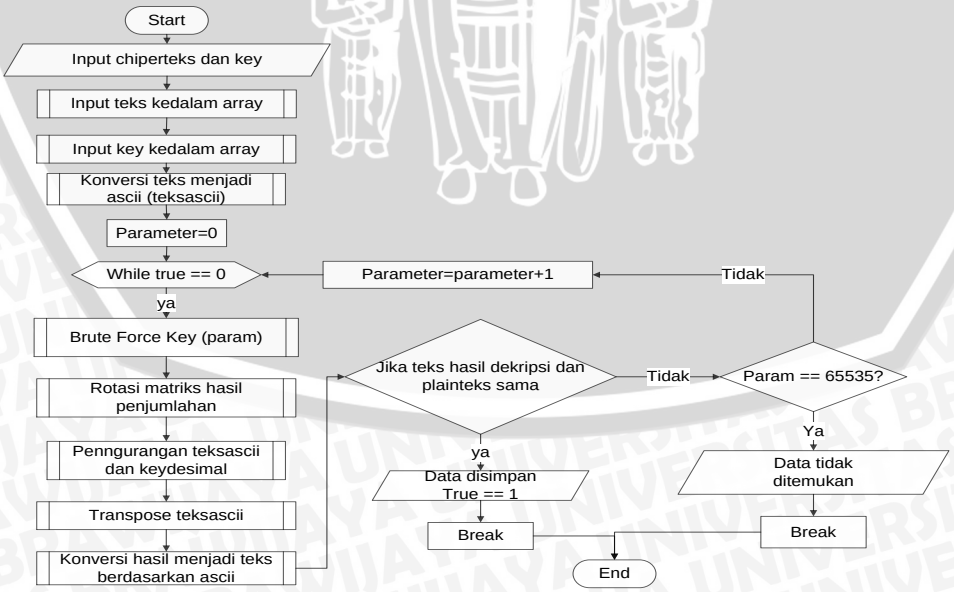
Cipheranalyst berperan menjadi pendengar diantara masing-masing Client. Tujuan utama dari *Cipheranalyst* adalah untuk mencoba memecahkan teks yang dikirim oleh Client satu dengan lainnya. Setelah *Cipheranalyst* mendapatkan informasi yang didapat dari salah satu pihak, maka pengujian keamanan dapat dilakukan.

Akan ada 2 hal yang dilakukan oleh *Cipheranalyst*. Adapun 2 hal tersebut adalah :

1. *Known Plaintext Attack*
2. *Known Ciphertext Attack*

4.2.4.1. Known Plaintext Attack

Proses ini sama dengan proses dekripsi BREA. hanya saja perbedaannya adalah penggunaan key. Key pada proses ini didapat dari hasil serangan Brute Force. Ilustrasi alur serangan akan ditunjukkan pada gambar 4.4.



Gambar 4.4. Proses Brute Force skema Known Plaintext
Sumber : Perancangan

Pada proses ini, Brute Force dilakukan dengan cara menggunakan parameter untuk setiap kali looping. Parameter akan bertambah 1 ketika teks tidak sesuai dengan kamus dan tidak melebihi nilai parameter maksimum yaitu 65535 . Pemanggilan fungsi Brute Force Key digunakan untuk mengubah nilai desimal parameter menjadi biner yang nantinya akan digunakan sebagai key untuk dekripsi. Algoritma yang digunakan akan ditunjukkan pada tabel 4.19.

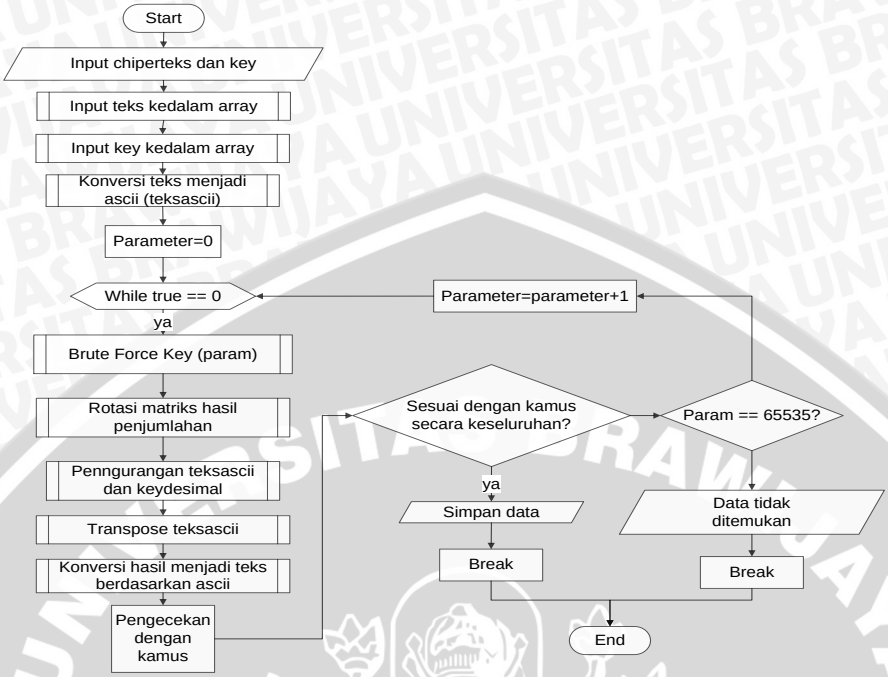
Tabel 4.19. Algoritma Brute Force Key

Nama Algoritma : Brute Force Key
Deskripsi : ➔ Masukan : nilai parameter Proses : • Menjadikan parameter menjadi biner • Memasukkan biner kedalam list • Menghilangkan 2 karakter paling depan Output : pengembalian nilai biner

Sumber : Perancangan

4.2.4.2. Known Ciphertext Attack

Proses ini lebih rumit dibandingkan dengan serangan Known Plaintext Attack. Untuk alur proses, hanya berbeda di bagian pengecekan dengan kamus. Berikut alur dari serangan Known Plaintext Only Attack akan dijelaskan pada gambar 4.5.



Gambar 4.5. Proses Brute Force skema Known Ciphertext Only
Sumber : Perancangan

Untuk pengecekan dengan kamus, penulis membuat sistem pencocokan dengan maksimum 3 kata pertama untuk mempercepat proses pencocokan data. Hal ini dilakukan karena variasi dan panjang teks yang di inputkan oleh client 1 maupun client 2 dapat mempengaruhi lama waktu untuk serangan Brute Force. Algoritma yang digunakan akan ditunjukkan pada tabel 4.20.

Tabel 4.20. Algoritma Pengecekan Kamus

Nama Algoritma : Pengecekan kata dengan kamus
Deskripsi : ➔ Masukan : *data* yang sudah di dekripsi secara Brute Force Proses : • Memecah kata yang sudah di dekripsi berdasarkan spasi • Mengkategorikan berdasarkan panjang hasil pecahan kata: ○ Jika 1 kata : ▪ Melakukan pencocokan pada kamus. ▪ Jika cocok, maka menyimpan data pada notepad dan keluar dari program ○ Jika 2 kata : ▪ Melakukan pencocokan pada kamus untuk 2 kata ▪ Jika keduanya cocok maka menyimpan data pada notepad dan keluar dari program ○ Jika 3 kata atau lebih:

▪ Melakukan pencocokan pada kamus untuk 3 kata pertama ▪ Jika ketiganya cocok maka menyimpan data pada notepad dan keluar dari program Output : data yang tersimpan di notepad

Sumber : Perancangan

BAB V
IMPLEMENTASI

Bab ini membahas mengenai tahapan implementasi sistem perangkat lunak untuk aplikasi pesan instan menggunakan kriptografi BREA yang telah didapatkan dari analisis kebutuhan dan proses perancangan perangkat lunak. Implementasi terdiri atas penjelasan tentang spesifikasi sistem, batasan-batasan dalam implementasi, dan implementasi algoritma.

5.1. Spesifikasi Lingkungan Sistem

Implementasi dan pengujian keamanan pada BREA dikembangkan dalam lingkungan implementasi yang terdiri dari perangkat keras dan perangkat lunak.

5.1.1. Spesifikasi Lingkungan Perangkat Keras

Spesifikasi lingkungan perangkat keras yang dipakai dalam proses pengembangan untuk implementasi dan pengujian keamanan pada BREA dijelaskan pada Tabel 5.1.

Tabel 5.1. Spesifikasi Lingkungan Perangkat Keras Komputer

(Client 1)	
Processor	Intel (R) Core (TM) i7-2630QM CPU @ 2.00 GHz
Memory (RAM)	4 GB
Harddisk	750 GB
Laptop Toshiba (Client 2)	
Processor	Intel (R) Core (TM)2 Duo CPU T5500 @1.66 GHz
Memory (RAM)	2 GB
Harddisk	120 GB

Sumber : Implementasi

5.1.2. Spesifikasi Lingkungan Perangkat Lunak

Spesifikasi lingkungan perangkat lunak yang dipakai dalam proses pengembangan untuk implementasi dan pengujian keamanan pada BREA dijelaskan pada Tabel 5.2.

Tabel 5.2. Spesifikasi Lingkungan Perangkat Lunak Komputer

Laptop Lenovo (Client 1)	
Operating System	Microsoft Windows 7 Home Premium 64-bit
Programming Language	Python
Laptop Toshiba (Client 2)	
Operating System	Microsoft Windows XP SP 3 32-bit
Programming Language	Python

Sumber : Implementasi

5.2. Batasan – Batasan Implementasi

Beberapa batasan dalam implementasi BREA kedalam aplikasi pesan instan adalah sebagai berikut :

3. Kriptografi yang di implementasikan dalam pesan instan adalah BREA yang di implementasikan kedalam bahasa pemrograman *Python*
4. Antarmuka pesan instan di ambil dan di edit dari aplikasi pesan instan milik chprice (Sumber : https://github.com/chprice/Python-Chat-Program)
5. Aplikasi pesan instan menggunakan sistem *Point-to-Point* menggunakan *Socket Programming*.
6. Kamus yang digunakan adalah kamus bahasa indonesia baku dari KBBI (Sumber : http://perpus.unimus.ac.id/wp-content/uploads/2012/05/Kamus-Besar-Bahasa-Indonesia.pdf) dan kamus bahasa jejaring sosial dari geovedi (Sumber : https://github.com/ardwort/freq-dist-id)

5.3. Implementasi Algoritma

Implementasi dan uji keamanan BREA menggunakan berbagai macam algoritma sesuai dengan perancangan. Berikut implementasi dari perancangan yang sudah dibuat :

5.3.1. Key

Sesuai dengan perancangan, key generator digunakan dengan enkripsi MD5 yang nantinya dimodulus 2. Potongan program yang digunakan ditunjukkan pada tabel 5.3.

Tabel 5.3. Implementasi Algoritma Key

```
1.  def keygenerator(text):
2.      keys=text
3.      hash=hashlib.md5()
4.      hash.update(keys)
5.      hkeys=hash.hexdigest()
6.      x=0
7.      listhkeys=[]
8.      while x<len(hkeys)-16:
9.          dechkeys=int(hkeys[x],16)
10.         listhkeys.append(dechkeys%2)
11.         x=x+1
12.     return listhkeys
```

Sumber : Implementasi

Fungsi hash (baris 3-5) digunakan untuk mengkonversi teks menjadi md5. Kemudian baris 8 digunakan untuk mengambil 16 karakter pertama saja. Dan yang dimasukkan kedalam array adalah hexadesimal yang dikonversi kedalam desimal pada baris 9. Baris 10 merupakan pemasukan desimal yang sudah dimodulus 2 kedalam array.

5.3.2. Enkripsi

Dalam perancangan tahap enkripsi, ada berbagai macam fungsi yang digunakan untuk melakukan enkripsi tersebut. Potongan program yang digunakan ditunjukkan pada tabel 5.4.

Tabel 5.4. Implementasi Algoritma Enkripsi

```
1.  def enkripsi(text,key):
2.      keyblock=toblock(key)
3.      block=toblock(text)
4.      dblock=converttoascii(toblock(text))
5.      transposed=transpose(dblock)
6.      sums=sumfunction(transposed,keyblock,text)
7.      rotatedhorizontal=rotatehorizontal(sums)
8.      rotatedvertical=rotatevertical(rotatedhorizontal)
9.      encryptedblock=converttoletter(rotatedvertical)
10.     encrypted=totext(encryptedblock)
11.     return encrypted
```

Sumber : Implementasi

5.3.2.1. Input Teks / Key Kedalam Array

Sesuai dengan perancangan algoritma, input teks / key kedalam array melakukan pengecekan terhadap panjang huruf yang dimiliki. Potongan program yang digunakan ditunjukkan pada tabel 5.5.

Tabel 5.5. Implementasi Pengecekan panjang karakter

```
1.      #pembuatan list untuk block per 16 karakter text
2.      while x<len(text):
3.          block.append(y)
4.          y=y+1
5.          x=x+16
```

Sumber : Implementasi

Pada potongan program ini ditunjukkan bahwa setiap 16 karakter yang ada maka list blok akan menambahkan nilai y kedalam listnya menggunakan fungsi *append*. Blok digunakan untuk membuat indeks untuk array 4x4 setiap 16 karakter.

Setelah dilakukan pengisian blok, maka akan dilanjutkan proses memasukkan karakter kedalam array. Potongan program yang digunakan ditunjukkan pada tabel 5.6.

Tabel 5.6. Implementasi Input Teks kedalam Array

```
1.      v=0
2.      h=0
3.      while v<len(block):
4.          x=0
5.          #pengisi matriks 4x4 sejumlah 16 karakter
6.          while x<16:
7.              index=[0,1,2,3]
8.              y=0
9.              #matriks 4x4
10.             while y<4:
11.                 inindex=[]
12.                 z=0
13.                 while z<4:
14.                     try:
15.                         inindex.append(text[h])
16.                     except:
17.                         inindex.append('')
18.                     z=z+1
19.                     h=h+1
20.                     x=x+1
21.                 index[y]=inindex
22.                 y=y+1
23.             block[v]=index
24.             v=v+1
```

Sumber : Implementasi

Dalam sebuah array, terdapat baris dan kolom yang harus di isi. Sesuai dengan BREA maka dibutuhkan 4 baris dan 4 kolom untuk proses enkripsi. Pada potongan program, variabel *index* merupakan baris (baris 7) dan *inindex* merupakan kolom (baris 11). Setiap kolom akan di isi terlebih dahulu, kemudian

dimasukkan kedalam baris yaitu list *index*. Setelah semua baris terkumpul, maka dimasukkan kedalam variabel blok. Jika panjang karakter kurang dari jumlah slot array yang disediakan, maka secara otomatis akan dimasukkan karakter (‘’) kedalam array.

5.3.2.2. Konversi Karakter Menjadi Desimal berdasarkan ASCII

Setelah melakukan input teks / key kedalam array maka selanjutnya karakter dalam array tersebut di konversi menjadi desimal berdasarkan tabel ASCII. Potongan program yang digunakan ditunjukkan pada tabel 5.7.

Tabel 5.7. Implementasi Konversi Karakter menjadi Desimal

```
1.  def converttoascii(ablock):
2.      x=0
3.      while x<len(ablock):
4.          y=0
5.          while y<4:
6.              z=0
7.              while z<4:
8.                  try:
9.                      ablock[x][y][z]=ord(ablock[x][y][z])
10.                 except:
11.                     ablock[x][y][z]=32 #karakter spasi
12.                 z=z+1
13.             y=y+1
14.         x=x+1
15.     return ablock
```

Sumber : Implementasi

ablock merupakan hasil dari input teks kedalam array. Proses konversi dari karakter menjadi desimal menggunakan fungsi *ord* (baris 9). Jika terjadi error dikarenakan karakter (‘’) , maka secara otomatis akan berubah menjadi angka 32 yaitu desimal ASCII untuk spasi (baris 11) .

5.3.2.3. Transpose Blok

Setelah dikonversi menjadi desimal, maka array tersebut akan di transpose. Potongan program yang digunakan ditunjukkan pada tabel 5.8.

Tabel 5.8. Implementasi Transpose Blok

```
1.  def transpose(trblock):
2.      z=0
3.      temp=[]
4.      while z<len(trblock):
```

```
5.          x=0
6.          while x<len(trblock[z])-1:
7.              y=x+1
8.              while y<len(trblock[z]):
9.                  temp=trblock[z][x][y]
10.                 trblock[z][x][y]=trblock[z][y][x]
11.                 trblock[z][y][x]=temp
12.                 y=y+1
13.             x=x+1
14.         z=z+1
15.     return trblock
```

Sumber : Implementasi

Pada program ini ditunjukkan proses penukaran nilai pada slot array. List temp (baris 3) digunakan sebagai variabel sementara yang menyimpan nilai untuk pertukaran.

5.3.2.4. Penjumlahan Blok dengan Key

Setelah dilakukan transpose maka langkah selanjutnya adalah penjumlahan blok dengan key. Potongan program yang digunakan ditunjukkan pada tabel 5.9.

Tabel 5.9. Implementasi Penjumlahan Blok dengan Key

```
1.  def sumfunction(sblock,key,text):
2.      sum=toblock(text)
3.      x=0
4.      while x<len(sblock):
5.          y=0
6.          while y<4:
7.              z=0
8.              while z<4:
9.                  sum[x][y][z]=sblock[x][y][z]+key[0][y][z]
10.                 if sum[x][y][z]>127:
11.                     sum[x][y][z]=sum[x][y][z]-127
12.                 else:
13.                     None
14.                 z=z+1
15.             y=y+1
16.         x=x+1
17.     return sum
```

Sumber : Implementasi

Pada baris 9 ditunjukkan penjumlahan nilai antara blok dengan key. Seluruh blok akan dijumlahkan dengan key yang sama sehingga indeks key pertama adalah 0. Kemudian jika hasil penjumlahan melebihi 127, maka akan

dikurangi 127. Hal ini dilakukan karena nilai maksimum desimal ASCII adalah 127.

5.3.2.5. Rotasi Blok secara Horizontal

Pada BREA, rotasi pertama yang dilakukan saat enkripsi adalah rotasi secara horizontal. Potongan program yang digunakan ditunjukkan pada tabel 5.10.

Tabel 5.10. Implementasi Rotasi Blok Horizontal

```
1.  def rotatehorizontal(hblock):
2.      x=0
3.      temp=[]
4.      while x<len(hblock):
5.          y=0
6.          while y<3:
7.              v=y+1
8.              while v!=0:
9.                  z=0
10.                 while z<3:
11.                     temp=hblock[x][y][z]
12.                     hblock[x][y][z]=hblock[x][y][z+1]
13.                     hblock[x][y][z+1]=temp
14.                     z=z+1
15.                 v=v-1
16.             y=y+1
17.         x=x+1
18.     return hblock
```

Sumber : Implementasi

Pada program ini digunakan variabel v sebagai pengendali pergeseran. Variabel v didasarkan dari variabel y yang digunakan untuk menunjukkan indeks baris. Variabel y tidak mengalami pergeseran karena variabel y merupakan indeks baris. Variabel z digunakan untuk indeks yang menggeser array secara horizontal (baris 11-13)

5.3.2.6. Rotasi Blok secara Vertikal

Pada BREA, rotasi yang dilakukan selanjutnya adalah rotasi secara horizontal. Potongan program yang digunakan ditunjukkan pada tabel 5.11.

Tabel 5.11. Implementasi Rotasi Blok Secara Vertikal

```
1.  def rotatevertical(vblock):
2.      x=0
3.      temp=[]
4.      while x<len(vblock):
5.          z=0
```

```
6.          while z<3:
7.              v=z+1
8.              while v!=0:
9.                  y=0
10.                 while y<3:
11.                     temp=vblock[x][y][z]
12.                     vblock[x][y][z]=vblock[x][y+1][z]
13.                     vblock[x][y+1][z]=temp
14.                     y=y+1
15.                 v=v-1
16.             z=z+1
17.         x=x+1
18.     return vblock
```

Sumber : Implementasi

Pada program ini digunakan variabel v sebagai pengendali pergeseran. Variabel v didasarkan dari variabel z yang digunakan untuk menunjukkan indeks baris. Variabel z tidak mengalami pergeseran karena variabel z merupakan indeks kolom. Variabel y digunakan untuk indeks yang menggeser array secara horizontal (baris 11-13).

5.3.2.7. Konversi desimal Menjadi karakter berdasarkan ASCII

Setelah proses rotasi selesai, maka blok akan dikonversi kembali menjadi karakter berdasarkan tabel ASCII. Potongan program yang digunakan ditunjukkan pada tabel 5.12.

Tabel 5.12. Implementasi Konversi Desimal menjadi Karakter

```
1.  def converttoletter(lblock):
2.      x=0
3.      while x<len(lblock):
4.          y=0
5.          while y<4:
6.              z=0
7.              while z<4:
8.                  try:
9.                      lblock[x][y][z]=chr(lblock[x][y][z])
10.                 except:
11.                     lblock[x][y][z]=''
12.                 z=z+1
13.             y=y+1
14.         x=x+1
15.     return lblock
```

Sumber : Implementasi

Pada baris 9 merupakan pengembalian desimal menjadi karakter dengan menggunakan fungsi chr. Jika terjadi error, maka blok tersebut akan diisi dengan karakter (‘’).

5.3.2.8. Mengembalikan Blok menjadi Teks

Setelah dikembalikan menjadi karakter, maka karakter akan dikumpulkan menjadi kesatuan. Potongan program yang digunakan ditunjukkan pada tabel 5.13.

Tabel 5.13. Implementasi Pengembalian Blok menjadi Teks

```
13. def totext(tblock):
14.     kembali=[]
15.     x=0
16.     while x<len(tblock):
17.         y=0
18.         while y<4:
19.             temp=''.join(tblock[x][y])
20.             kembali.append(temp)
21.             y=y+1
22.         x=x+1
23.     return str(''.join(kembali))
```

Sumber : Implementasi

Variabel kembali digunakan untuk menyimpan hasil penggabungan karakter tiap blok. Pada baris 7 dilakukan penggabungan untuk blok sesuai dengan indeks. Penggabungan menggunakan ‘’.join(). kemudian pada baris 8 merupakan penyimpanan hasil temp kedalam variabel kembali menggunakan fungsi append.

5.3.3. Dekripsi

Dalam perancangan tahap dekripsi, ada berbagai macam fungsi yang digunakan untuk melakukan dekripsi tersebut. Implementasi untuk algoritma dekripsi akan ditunjukkan pada tabel 5.14.

Tabel 5.14. Implementasi Algoritma Dekripsi

```
12. def dekripsi(text,key):
13.     keyblock=toblock(key)
14.     block=toblock(text)
15.     dblock=converttoascii(toblock(text))
16.     reversedvertical=reversevertical(dblock)
17.     reversedhorizontal=reversehorizontal(dblock)
18.     sums=minfunction(reversedhorizontal,keyblock,text)
19.     transposed=transpose(sums)
20.     decryptedblock=converttoletter(transposed)
21.     decrypted=totext(decryptedblock)
```

```
22.     return decrypted
```

Sumber : Implementasi

5.3.3.1. Input Teks / Key kedalam Array

Sesuai dengan perancangan algoritma, input teks / key kedalam array melakukan pengecekan terhadap panjang huruf yang dimiliki. Potongan program yang digunakan ditunjukkan pada tabel 5.15.

Tabel 5.15. Implementasi Pengecekan panjang Kalimat

```
1.      #pembuatan list untuk block per 16 karakter text
2.      while x<len(text):
3.          block.append(y)
4.          y=y+1
5.          x=x+16
```

Sumber : Implementasi

Pada potongan program ini ditunjukkan bahwa setiap 16 karakter yang ada maka list blok akan menambahkan nilai y kedalam listnya menggunakan fungsi *append*. Blok digunakan untuk membuat indeks untuk array 4x4 setiap 16 karakter.

Setelah dilakukan pengisian blok, maka akan dilanjutkan proses memasukkan karakter kedalam array. Potongan program yang digunakan ditunjukkan pada tabel 5.16.

Tabel 5.16. Implementasi Input Teks/Key kedalam Array

```
1.      v=0
2.      h=0
3.      while v<len(block):
4.          x=0
5.          #pengisi matriks 4x4 sejumlah 16 karakter
6.          while x<16:
7.              index=[0,1,2,3]
8.              y=0
9.              #matriks 4x4
10.             while y<4:
11.                 inindex=[]
12.                 z=0
13.                 while z<4:
14.                     try:
15.                         inindex.append(text[h])
16.                     except:
17.                         inindex.append('')
18.                     z=z+1
19.                     h=h+1
20.                     x=x+1
21.                 index[y]=inindex
```

```
22.                 y=y+1
23.         block[v]=index
24.         v=v+1
```

Sumber : Implementasi

Dalam sebuah array, terdapat baris dan kolom yang harus di isi. Sesuai dengan BREA maka dibutuhkan 4 baris dan 4 kolom untuk proses enkripsi. Pada potongan program, variabel *index* merupakan baris (baris 7) dan *inindex* merupakan kolom (baris 11). Setiap kolom akan di isi terlebih dahulu, kemudian dimasukkan kedalam baris yaitu list *index*. Setelah semua baris terkumpul, maka dimasukkan kedalam variabel blok. Jika panjang karakter kurang dari jumlah slot array yang disediakan, maka secara otomatis akan dimasukkan karakter (‘’) kedalam array.

5.3.3.2. Konversi Teks Menjadi Desimal berdasarkan ASCII

Setelah melakukan input teks / key kedalam array maka selanjutnya karakter dalam array tersebut di konversi menjadi desimal berdasarkan tabel ASCII. Potongan program yang digunakan ditunjukkan pada tabel 5.17.

Tabel 5.17. Implementasi Konversi Teks Menjadi Desimal

```
1.  def converttoascii(ablock):
2.      x=0
3.      while x<len(ablock):
4.          y=0
5.          while y<4:
6.              z=0
7.              while z<4:
8.                  try:
9.                      ablock[x][y][z]=ord(ablock[x][y][z])
10.                 except:
11.                     ablock[x][y][z]=32 #karakter spasi
12.                 z=z+1
13.             y=y+1
14.         x=x+1
15.     return ablock
```

Sumber : Implementasi

ablock merupakan hasil dari input teks kedalam array. Proses konversi dari karakter menjadi desimal menggunakan fungsi *ord* (baris 9). Jika terjadi error dikarenakan karakter (‘’) , maka secara otomatis akan berubah menjadi angka 32 yaitu desimal ASCII untuk spasi (baris 11) .

5.3.3.3. Rotasi Blok Secara Vertikal

Pada BREA, rotasi yang dilakukan selanjutnya adalah rotasi secara horizontal. Potongan program yang digunakan ditunjukkan pada tabel 5.18.

Tabel 5.18. Implementasi Rotasi Blok Secara Vertikal

```
1.      x=0
2.      temp=[]
3.      while x<len(vrblock):
4.          z=0
5.          while z<3:
6.              v=z+1
7.              while v!=0:
8.                  y=2
9.                  while y>=0:
10.                     temp=vrblock[x][y][z]
11.                     vrblock[x][y][z]=vrblock[x][y-1][z]
12.                     vrblock[x][y-1][z]=temp
13.                     y=y-1
14.                 v=v-1
15.             z=z+1
16.         x=x+1
17.     return vrblock
```

Sumber : Implementasi

Pada program ini digunakan variabel v sebagai pengendali pergeseran. Variabel v didasarkan dari variabel z yang digunakan untuk menunjukkan indeks baris. Variabel z tidak mengalami pergeseran karena variabel z merupakan indeks kolom. Variabel y digunakan untuk indeks yang menggeser array secara horizontal (baris 11-13).

5.3.3.4. Rotasi Blok secara Horizontal

Pada BREA, rotasi pertama yang dilakukan saat enkripsi adalah rotasi secara horizontal. Potongan program yang digunakan ditunjukkan pada tabel 5.19.

Tabel 5.19. Rotasi Blok Secara Horizontal

```
1.  x=0
2.      temp=[]
3.      while x<len(hrblock):
4.          y=0
5.          while y<3:
6.              v=y+1
7.              while v!=0:
8.                  z=2
9.                  while z>=0:
10.                     temp=hrblock[x][y][z]
11.                     hrblock[x][y][z]=hrblock[x][y][z-1]
```

```
12.                     hrblock[x][y][z-1]=temp
13.                     z=z-1
14.                 v=v-1
15.             y=y+1
16.         x=x+1
17.     return hrblock
```

Sumber : Implementasi

Pada program ini digunakan variabel v sebagai pengendali pergeseran. Variabel v didasarkan dari variabel y yang digunakan untuk menunjukkan indeks baris. Variabel y tidak mengalami pergeseran karena variabel y merupakan indeks baris. Variabel z digunakan untuk indeks yang menggeser array secara horizontal (baris 11-13).

5.3.3.5. Pengurangan Blok dengan Key

Setelah dilakukan transpose maka langkah selanjutnya adalah penjumlahan blok dengan key. Potongan program yang digunakan ditunjukkan pada tabel 5.20.

Tabel 5.20. Implementasi Pengurangan Blok dengan Key

```
1.  def minfunction(mblock,key,text):
2.      sum=toblock(text)
3.      x=0
4.      while x<len(mblock):
5.          y=0
6.          while y<4:
7.              z=0
8.              while z<4:
9.                  sum[x][y][z]=mblock[x][y][z]-key[0][y][z]
10.                 if sum[x][y][z]<0:
11.                     sum[x][y][z]=sum[x][y][z]+127
12.                 else:
13.                     None
14.                 z=z+1
15.             y=y+1
16.         x=x+1
17.     return sum
```

Sumber : Implementasi

Pada baris 9 ditunjukkan penjumlahan nilai antara blok dengan key. Seluruh blok akan dijumlahkan dengan key yang sama sehingga indeks key pertama adalah 0. Kemudian jika hasil pengurangan bernilai kurang dari 0, maka akan ditambah 127.

5.3.3.6. Transpose Blok

Setelah dikonversi menjadi desimal, maka array tersebut akan di transpose. Potongan program yang digunakan akan dijelaskan pada tabel 5.21.

Tabel 5.21. Implementasi Transpose Blok

```
1.  def transpose(trblock):
2.      z=0
3.      temp=[]
4.      while z<len(trblock):
5.          x=0
6.          while x<len(trblock[z])-1:
7.              y=x+1
8.              while y<len(trblock[z]):
9.                  temp=trblock[z][x][y]
10.                 trblock[z][x][y]=trblock[z][y][x]
11.                 trblock[z][y][x]=temp
12.                 y=y+1
13.             x=x+1
14.         z=z+1
15.     return trblock
```

Sumber : Implementasi

Pada program ini ditunjukkan proses penukaran nilai pada slot array. List temp (baris 3) digunakan sebagai variabel sementara yang menyimpan nilai untuk pertukaran.

5.3.3.7. Konversi Desimal Menjadi Karakter berdasarkan ASCII

Setelah proses rotasi selesai, maka blok akan dikonversi kembali menjadi karakter berdasarkan tabel ASCII. Potongan program yang digunakan akan dijelaskan pada tabel 5.22.

Tabel 5.22. Implementasi Konversi Desimal Menjadi Karakter

```
1.  def converttoletter(lblock):
2.      x=0
3.      while x<len(lblock):
4.          y=0
5.          while y<4:
6.              z=0
7.              while z<4:
8.                  try:
9.                      lblock[x][y][z]=chr(lblock[x][y][z])
10.                 except:
11.                     lblock[x][y][z]=''
12.                 z=z+1
13.             y=y+1
14.         x=x+1
15.     return lblock
```

Sumber : Implementasi

Pada baris 9 merupakan pengembalian desimal menjadi karakter dengan menggunakan fungsi chr. Jika terjadi error, maka blok tersebut akan diisi dengan karakter (‘’).

5.3.3.8. Mengembalikan Blok menjadi Teks

Setelah dikembalikan menjadi karakter, maka karakter akan dikumpulkan menjadi kesatuan. Potongan program yang digunakan ditunjukkan pada tabel 5.23.

Tabel 5.23. Implementasi Pengembalian Blok Menjadi Teks

```
1.  def totext(tblock):
2.      kembali=[]
3.      x=0
4.      while x<len(tblock):
5.          y=0
6.          while y<4:
7.              temp=''.join(tblock[x][y])
8.              kembali.append(temp)
9.              y=y+1
10.         x=x+1
11.     return str(''.join(kembali))
```

Sumber : Implementasi

Variabel kembali digunakan untuk menyimpan hasil penggabungan karakter tiap blok. Pada baris 7 dilakukan penggabungan untuk blok sesuai dengan indeks. Penggabungan menggunakan ‘’.join(). kemudian pada baris 8 merupakan penyimpanan hasil temp kedalam variabel kembali menggunakan fungsi append.

5.3.4. Cipheranalyst

Pada perancangan proses untuk pengujian keamanan adalah peran dari *Cipheranalyst*. Adapun implementasi dari perancangan proses tersebut adalah sebagai berikut :

5.3.4.1. Known Plaintext (KP) Attack

Implementasi dari perancangan untuk serangan tipe *Known Plaintext Attack* akan ditunjukkan pada tabel 5.24.

Tabel 5.24. Implementasi Pengecekan KP

```
1.  true=0
2.  param=0
3.  now=datetime.now()
4.  while true==0:
5.      keyblock=bruteforce(param)
6.      sums=minfunction(reversedhorizontal,keyblock)
7.      transposed=transpose(sums)
8.      decryptedblock=converttoletter(transposed)
9.      decrypted=totext(decryptedblock)
10.     if (decrypted==text[1])is True:
11.         true=1
12.         then=datetime.now()
13.         print ('FOUND!!')
14.         break
15.     else:
16.         None
17.     #print (param)
18.     param=param+1
```

Sumber : Implementasi

Sesuai dengan perancangan, program Brute Force dimulai setelah konversi teks menjadi ASCII. Pada potongan program ini menunjukkan pemanggilan fungsi Brute Force pada baris 5 yang memiliki parameter param. Param akan bertambah setiap kali hasil dekripsi tidak sesuai dengan ciphertext. Program akan berhenti ketika hasil dekripsi dengan plainteks cocok. Jika cocok, maka nilai true akan berubah menjadi 1 dan program akan keluar dari loop. Tabel 5.25. menunjukkan algoritma Brute Force yang digunakan.

Tabel 5.25. Implementasi Brute Force

```
1.  def bruteforce(number):
2.      biner=bin(number)
3.      biners=list(biner)
4.      biners.pop(0)
5.      biners.pop(0)
6.      while len(biners)<16:
7.          biners.insert(0,'0')
8.      biners=toblock(biners)
9.      biners=converttonumber(biners)
10.     return biners
```

Sumber : Implementasi

Brute force akan mengubah nilai parameter menjadi key sesuai dengan key pada BREA yang menggunakan 1 dan 0 atau bisa dikategorikan sebagai biner. Program akan membinerkan nilai parameter, kemudian memasukkan kedalam list

(baris 2-3). Jika karakter key kurang dari 16 maka akan dimasukkan angka 0 di indeks paling depan (indeks 0).

5.3.4.2. Known Ciphertext Only (KC) Attack

Pada perancangan proses untuk serangan tipe Known Ciphertext Only digunakan pengecekan setelah di dekripsi. Pengecekan bertujuan agar mempersingkat waktu pencarian. Implementasi dari pengecekan akan dijelaskan tabel 5.26.

Tabel 5.26. Implementasi Pengecekan berdasarkan Panjang Kata

```
1.  true=0
2.  param=0
3.  now=datetime.now()
4.  starter(text)
5.  while true==0:
6.      keyblock,binkey=bruteforce(param)
7.      sums=minfunction(reversedhorizontal,keyblock)
8.      transposed=transpose(sums)
9.      decryptedblock=converttoletter(transposed)
10.     decrypted=totext(decryptedblock)
11.     decrypted=decrypted.split()
12.     print (param,decrypted,len(decrypted))
13.     if len(decrypted)==1:
14.         paramsingle,index=checker(decrypted[0])
15.         print (paramsingle,index)
16.         if paramsingle==1:
17.             #saving time function
18.             then=datetime.now()
19.             times=[]
20.             delta=then-now
21.             timedelta(now,then,delta)
22.             fix(binkey,decrypted)
23.             true=1
24.             break
25.     elif len(decrypted)==2:
26.         param1,index1=checker(decrypted[0])
27.         param2,index2=checker(decrypted[1])
28.         params=[]
29.         params.append(param1)
30.         params.append(param2)
31.         index=[]
32.         index.append(index1)
33.         index.append(index2)
34.         if params[0]==1 and params[1]==1:
35.             #saving time function
36.             then=datetime.now()
37.             times=[]
38.             delta=then-now
39.             timedelta(now,then,delta)
40.             fix(binkey,decrypted)
```

```
41.                 true=1
42.                 break
43.             elif params[0]==1 or params[1]==1:
44.                 #saving time function
45.                 then=datetime.now()
46.                 times=[]
47.                 delta=then-now
48.                 timedelta(now,then,delta)
49.                 probability=len(params)/2*100
50.                 temp=[]
51.                 temp.append(params.index(1))
52.                 candidate(decrypted,probability,binkey,2,temp)
53.         elif len(decrypted)>2:
54.             param1,index1=checker(decrypted[0])
55.             param2,index2=checker(decrypted[1])
56.             param3,index3=checker(decrypted[2])
57.             params=[]
58.             params.append(param1)
59.             params.append(param2)
60.             params.append(param3)
61.             index=[]
62.             index.append(index1)
63.             index.append(index2)
64.             index.append(index3)
65.             if params[0]==1 and params[1]==1 and params[2]==1:
66.                 #saving time function
67.                 then=datetime.now()
68.                 times=[]
69.                 delta=then-now
70.                 timedelta(now,then,delta)
71.                 fix(binkey,decrypted)
72.                 true=1
73.                 break
74.             elif params[0]==1 or params[1]==1 or params[2]==1:
75.                 x=0
76.                 temp=[]
77.                 while x<len(params):
78.                     if params[x]==1:
79.                         temp.append(x)
80.                     x=x+1
81.                 probability=len(temp)/3*100
82.                 #saving time function
83.                 then=datetime.now()
84.                 times=[]
85.                 delta=then-now
86.                 timedelta(now,then,delta)
87.                 candidate(decrypted,probability,binkey,3,temp)
88.         param=param+1                   temp=[]
89.                 temp.append(params.index(1))
90.                 candidate(decrypted,probability,binkey,2,temp)
91.         elif len(decrypted)>2:
92.             param1,index1=checker(decrypted[0])
93.             param2,index2=checker(decrypted[1])
94.             param3,index3=checker(decrypted[2])
95.             params=[]
96.             params.append(param1)
```

```
97.             params.append(param2)
98.             params.append(param3)
99.             index=[]
100.            index.append(index1)
101.            index.append(index2)
102.            index.append(index3)
103.            if params[0]==1 and params[1]==1 and params[2]==1:
104.                #saving time function
105.                then=datetime.now()
106.                times=[]
107.                delta=then-now
108.                timedelta(now,then,delta)
109.                fix(binkey,decrypted)
110.                true=1
111.                break
112.            elif params[0]==1 or params[1]==1 or params[2]==1:
113.                x=0
114.                temp=[]
115.                while x<len(params):
116.                    if params[x]==1:
117.                        temp.append(x)
118.                    x=x+1
119.                probability=len(temp)/3*100
120.                #print ('probability : '+str(probability)+'%')
121.                 #saving time function
122.                then=datetime.now()
123.                times=[]
124.                delta=then-now
125.                timedelta(now,then,delta)
126.                candidate(decrypted,probability,binkey,3,temp)
127.            #print ('it is more than one')
128.        param=param+1
```

Sumber : Implementasi

Pengecekan dilakukan dengan 3 kondisi. Yaitu 1 kata, 2 kata, dan lebih dari 3 kata. Untuk kondisi 1 kata, jika 1 kata tersebut cocok dengan kata di kamus maka program akan keluar dari loop dan true bernilai 1. Untuk 2 kata, maka 2 kata tersebut harus sesuai dengan kamus. Jika lebih dari 3 maka diambil 3 kata terdepan untuk dilakukan pencocokan terhadap kamus. Implementasi algoritma pengecekan kata terhadap kamus akan ditunjukkan pada tabel 5.27.

Tabel 5.27. Implementasi Pengecekan

```
1.  def checker(decrypted):
2.      parameter=0
3.      x=0
4.      while x<len(lib):
5.          if (decrypted==lib[x])is True:
6.              #print ('reached here')
7.              parameter=1
8.              return parameter,x
9.              #break
```

```
10.             if x==len(lib)-1:
11.                 return parameter,0
12.                 #break
13.             x=x+1
```

Sumber : Implementasi

Pencocokan ini menggunakan kamus yang sudah disimpan didalam aplikasi. Adapun setiap pengecekan kata menggunakan pengecekan ini. Pengecekan ini akan mengembalikan nilai 1 jika benar dan mengirimkan indeks kamus yang sesuai dengan kata tersebut. Pada pencocokan panjang kata, syarat utama teks dianggap valid ketika nilai kembalian dari pengecekan dalam kamus secara keseluruhan kata bernilai 1. Jika tidak maka tidak dianggap sebagai kata yangvalid.

BAB VI
PENGUJIAN DAN ANALISIS

Bab ini membahas mengenai tahapan pengujian dan pembahasan sistem hasil implementasi dan pengujian keamanan BREA pada pesan instan. Proses pengujian dilakukan melalui dua macam pengujian yaitu pengujian perangkat lunak dan pengujian keamanan. Pengujian perangkat lunak digunakan untuk menguji pengaruh waktu terhadap pesan instan yang sudah di tanamkan BREA didalamnya. Pengujian keamanan digunakan untuk menguji keamanan BREA.

Proses pembahasan bertujuan untuk mendapatkan kesimpulan dari hasil pengujian hasil implementasi dan pengujian keamanan BREA pada pesan instan. Proses pembahasan mengacu pada dasar teori sesuai dengan hasil pengujian yang didapatkan. Pembahasan dilakukan terhadap hasil pengujian di setiap tahap pengujian. Proses pembahasan yang dilakukan meliputi pembahasan hasil pengujian perangkat lunak dan pengujian keamanan BREA.

6.1. Pengujian Perangkat Lunak

Pada pengujian ini akan dilakukan pengujian terhadap seberapa cepat algoritma ini dapat berjalan. Pengujian ini dibutuhkan untuk mengetahui seberapa besar pengaruh implementasi BREA pada pesan instan terhadap waktu eksekusi. Berikut skenario pengujian :

6.1.1. Pengujian Algoritma

Pengujian algoritma ditujukan untuk mengetahui berapa lama waktu yang dibutuhkan ketika melakukan proses enkripsi dan dekripsi menggunakan BREA saja. Pengujian algoritma akan dilakukan dengan 2 skenario. Berikut skenario yang digunakan :

6.1.1.1. Skenario 1 : Key Sama dan Jumlah Karakter Berbeda

Pengujian ini ditujukan untuk mengetahui berapa lama waktu yang dibutuhkan untuk melakukan enkripsi dan dekripsi BREA. Data uji yang digunakan sebanyak 30 data. Pengujian dilakukan dengan menggunakan key yang sama dan teks yang memiliki jumlah karakter bervariasi. Hasil pengujian akan ditunjukkan pada tabel 6.1.

Tabel 6.1. Hasil Pengujian Key Sama

No	Jumlah Karakter	Key	Waktu Eksekusi	
			Enkripsi (ms)	Dekripsi (ms)
1	5	1001001001001000	6	6
2	6	1001001001001000	9	6
3	6	1001001001001000	7	8
4	10	1001001001001000	6	6
5	12	1001001001001000	5	6
6	14	1001001001001000	6	6
7	14	1001001001001000	6	7
8	16	1001001001001000	6	8
9	18	1001001001001000	7	6
10	18	1001001001001000	6	7
11	24	1001001001001000	7	7
12	25	1001001001001000	7	8
13	25	1001001001001000	8	7
14	32	1001001001001000	7	8
15	33	1001001001001000	6	7
16	37	1001001001001000	6	6
17	38	1001001001001000	8	7
18	41	1001001001001000	7	7
19	45	1001001001001000	7	7
20	45	1001001001001000	6	6
21	45	1001001001001000	7	6
22	48	1001001001001000	7	7
23	49	1001001001001000	7	8
24	51	1001001001001000	8	7
25	52	1001001001001000	8	6
26	53	1001001001001000	7	6
27	68	1001001001001000	8	8
28	80	1001001001001000	7	8
29	89	1001001001001000	8	7

30	93	1001001001001000	7	7
Rata-Rata			6,9	6,866667

Sumber : Pengujian

Berdasarkan hasil pengujian ini, ditunjukkan bahwa pengaruh BREA pada enkripsi dan dekripsi memiliki jarak waktu 6 ms hingga 9 ms.

6.1.1.2. Skenario 2: Teks Sama dan Key Berbeda

Pengujian ini ditujukan untuk mengetahui apakah key yang bervariasi dapat mempengaruhi dan memperlambat kinerja BREA dalam melakukan pengamanan pesan teks. Data yang digunakan adalah 3 teks yang sama dengan key yang berbeda. Teks yang digunakan adalah teks sepanjang 52 karakter. Key yang digunakan adalah 20000, 40000 dan 60000 yang dikonversi kedalam biner. Hasil pengujian akan ditunjukkan pada tabel 6.2.

Tabel 6.2. Hasil Pengujian Key Berbeda

No	Jumlah Karakter	Key	Waktu Eksekusi	
			Enkripsi (ms)	Dekripsi (ms)
1	52	10011100010000	7	7
2	52	1001110001000000	8	7
3	52	1110101001100000	6	7

Sumber : Pengujian

Berdasarkan hasil penelitian, terlihat bahwa waktu yang dibutuhkan tetap berkisar antara 6-9 ms.

6.1.2. Perbandingan Pesan Instan biasa dengan Pesan Instan menggunakan BREA

Pengujian ini bertujuan untuk mengetahui apakah BREA ketika di implementasikan kedalam pesan instan akan memperlambat waktu eksekusi program dibandingkan dengan pesan instan yang tidak menggunakan enkripsi. Data yang digunakan sama dengan skenario 1 dengan key yang sama. Pencatatan waktu yang dilakukan pada pengujian ini adalah pencatatan waktu berdasarkan waktu pengiriman dan penerimaan pesan. Untuk enkripsi pada Client 1, waktu dicatat adalah waktu tepat sebelum dilakukan tahap enkripsi dan untuk dekripsi

pada client 2, waktu dicatat adalah waktu tepat setelah hasil dari tahap dekripsi ditampilkan ke layar. Adapun waktu menggunakan jam yang sinkron antar 2 komputer. Waktu hanya diambil digit detik saja sehingga jam dan menit tidak ditampilkan. Pengujian pesan instan dengan BREA menggunakan key yang sama yaitu 111100001011111.

Pengujian dilakukan pada aplikasi pesan instan yang menggunakan BREA dan yang tidak menggunakan enkripsi apapun. Hasil pengujian ditunjukkan oleh tabel 6.3.

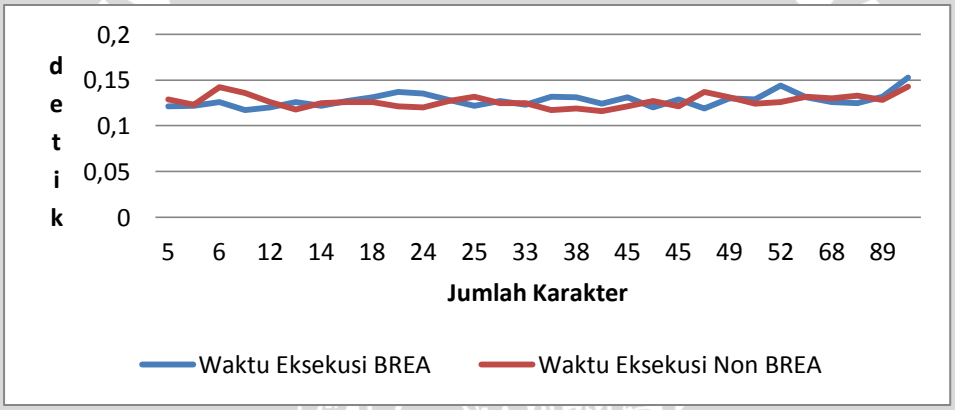
Tabel 6.3. Hasil Pengujian Pesan Instan dengan dan tanpa BREA

No	Jumlah Karakter	Waktu Eksekusi (s)					
		Dengan BREA			Tanpa Brea		
		Client 1	Client 2	Waktu	Client 1	Client 2	Waktu
1	5	41,988	42,109	0,121	14,699	14,828	0,129
2	6	44,534	44,656	0,122	32,861	32,984	0,123
3	6	47,561	47,687	0,126	34,529	34,671	0,142
4	10	49,351	49,468	0,117	36,598	36,734	0,136
5	12	51,942	52,062	0,12	39,967	40,093	0,126
6	14	54,092	54,218	0,126	42,475	42,593	0,118
7	14	56,206	56,328	0,122	46,625	46,75	0,125
8	16	58,857	58,984	0,127	50,577	50,703	0,126
9	18	6,087	6,218	0,131	56,811	56,937	0,126
10	18	9,878	10,015	0,137	59,441	59,562	0,121
11	24	13,771	13,906	0,135	3,661	3,781	0,12
12	25	20,465	20,593	0,128	17,523	17,65	0,127
13	25	28,987	29,109	0,122	24,977	25,109	0,132
14	32	46,044	46,171	0,127	33,64	33,765	0,125
15	33	52,767	52,89	0,123	54,265	54,39	0,125
16	37	58,899	59,031	0,132	3,32	3,437	0,117
17	38	4,525	4,656	0,131	10,131	10,25	0,119
18	41	10,469	10,593	0,124	24,653	24,769	0,116
19	45	16,228	16,359	0,131	30,05	30,171	0,121
20	45	20,801	20,921	0,12	35,091	35,218	0,127
21	45	25,199	25,328	0,129	45,55	45,671	0,121
22	48	29,006	29,125	0,119	12,55	12,687	0,137
23	49	33,807	33,937	0,13	22,478	22,609	0,131
24	51	39,589	39,718	0,129	32,297	32,421	0,124
25	52	44,731	44,875	0,144	39,374	39,5	0,126
26	53	48,353	48,484	0,131	45,274	45,406	0,132

27	68	52,577	52,703	0,126	0,495	0,625	0,13
28	80	56,671	56,796	0,125	7,757	7,89	0,133
29	89	2,914	3,046	0,132	17,934	18,062	0,128
30	93	6,94	7,093	0,153	31,872	32,015	0,143
Rata-Rata				0,1280			0,1269

Sumber : Pengujian

Dari hasil pengujian, terlihat waktu yang dibutuhkan untuk pesan instan yang menggunakan BREA (0,128 detik) tidak jauh berbeda dengan pesan instan yang tidak menggunakan enkripsi (0,1269 detik). Selisih rata-rata antara pesan instan yang menggunakan BREA dan yang tidak menggunakan BREA adalah 0,0018 detik. Dari data diatas disusun sebuah Grafik hasil pengujian :



Gambar 6.1. Grafik Pengujian Waktu Eksekusi Pesan Instan
Sumber : Pengujian

Dari grafik diatas dapat ditarik kesimpulan bahwa BREA tidak begitu berpengaruh terhadap waktu ketika di implementasikan kedalam pesan instan. Grafik diatas menunjukkan bahwa BREA dan Non BREA memiliki selisih waktu yang sedikit (0,0018 detik).

6.2. Pengujian Keamanan

Pengujian keamanan ditujukan untuk mengetahui Berapa lama waktu yang dibutuhkan untuk proses pendekripsian tanpa key. Pengujian keamanan menggunakan perancangan proses yang digunakan oleh *Cipheranalyst*. Adapun pengujian keamanan adalah sebagai berikut :

6.2.1. Known Plaintext (KP) Attack

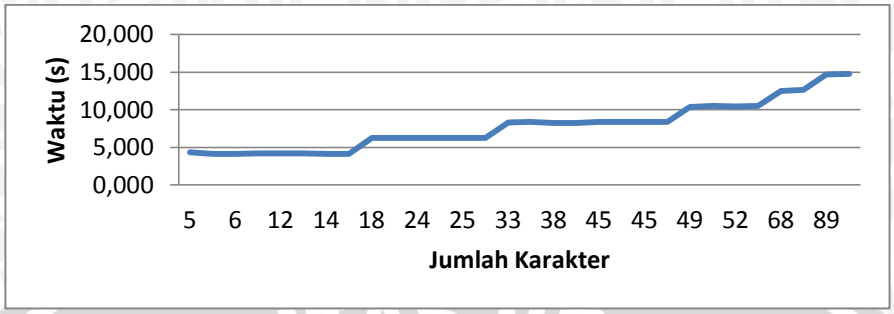
Pada skenario pengujian ini, *Cipheranalyst* mengetahui *plaintext* dan *ciphertext* yang digunakan. Data yang digunakan sama dengan data pengujian algoritma. Hasil pengujian akan ditunjukkan pada tabel 6.4.

Tabel 6.4. Hasil Pengujian serangan KP

No	Jumlah Karakter	Waktu Eksekusi (s)
1	5	4,359
2	6	4,156
3	6	4,125
4	10	4,203
5	12	4,203
6	14	4,187
7	14	4,156
8	16	4,172
9	18	6,250
10	18	6,282
11	24	6,234
12	25	6,250
13	25	6,296
14	32	6,266
15	33	8,344
16	37	8,359
17	38	8,281
18	41	8,238
19	45	8,360
20	45	8,390
21	45	8,360
22	48	8,391
23	49	10,391
24	51	10,486
25	52	10,421
26	53	10,500
27	68	12,500
28	80	12,609
29	89	14,657
30	93	14,766

Sumber : Pengujian

Dari data diatas disusun sebuah Grafik hasil pengujian yang diilustrasikan pada gambar 6.2.



Gambar 6.2. Grafik Pengujian Skema KP
Sumber : Pengujian

Pada Grafik terlihat bahwa BREA ketika diserangan menggunakan skenario ini dapat dipecahkan dalam waktu yang bergantung pada panjang teks. Semakin panjang teks yang di enkripsi maka waktu yang dibutuhkan untuk memecahkan juga semakin lama.

6.2.2. Known Ciphertext Only(KC) Attack

Pengujian ini ditujukan untuk mengetahui apakah dengan *ciphertext* saja, teks dapat didekripsi secara Brute Force. Data yang digunakan sama dengan data pengujian aplikasi pesan instan. Tabel 6.5. Menunjukkan hasil pengujian skema *Known Ciphertext*.

Tabel 6.5. Hasil Pengujian Serangan KC

No	Jumlah Karakter	Key Ditemukan?	Sesuai?	Waktu Eksekusi (Jam)
1	16	Tidak	Tidak	2:30:21.132000
2	16	Ya	Ya	1:10:50.831000
3	16	Ya	Ya	1:29:05.502000
4	16	Ya	Ya	1:33:16.235000
5	16	Ya	Tidak	0:12:18.788000
6	16	Ya	Tidak	0:09:18.265000
7	16	Ya	Tidak	1:16:09.075000
8	16	Ya	Ya	1:39:57.557000
9	32	Ya	Tidak	0:02:15.052000
10	32	Tidak	Tidak	2:39:55.556000
11	32	Ya	Ya	1:43:41.057000

12	32	Tidak	Tidak	2:57:57.596000
13	32	Tidak	Tidak	2:53:55.444000
14	32	Ya	Tidak	1:56:34.602000
15	48	Ya	Tidak	0:00:06.348000
16	48	Ya	Ya	1:53:30.331000
17	48	Ya	Tidak	0:43:38.284000
18	48	Ya	Tidak	0:08:21.735000
19	48	Ya	Tidak	0:53:15.013000
20	48	Tidak	Tidak	2:35:23.750000
21	48	Tidak	Tidak	2:58:23.097000
22	48	Ya	Tidak	1:45:04.705000
23	64	Tidak	Tidak	2:54:52.541000
24	64	Ya	Ya	0:42:21.452000
25	64	Ya	Ya	1:49:15.066000
26	64	Ya	Ya	2:03:28.704000
27	80	Ya	Tidak	0:00:41.307000
28	80	Ya	Tidak	0:11:32.156000
29	96	Ya	Tidak	1:22:49.117000
30	96	Ya	Ya	1:35:20.898000

Sumber : Pengujian

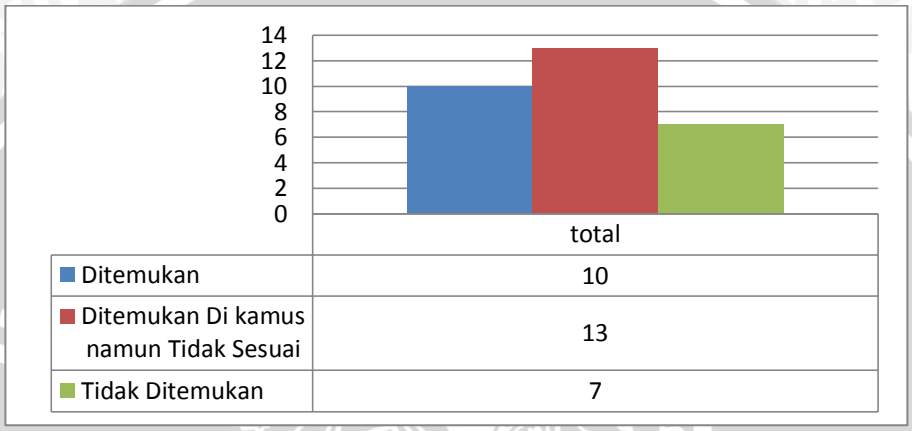
Dari data uji diatas, ada beberapa data yang tidak ditemukan key yang digunakan karena tidak sesuai dengan kamus (data uji nomor 1,10,12,13,20,21,23). Key yang tidak ditemukan memiliki waktu eksekusi yang paling lama karena pengecekan dilakukan keseluruh jangakauan key pada BREA. Pada tabel 6.6. akan ditunjukkan mengenai data yang tidak berhasil didekripsi.

Tabel 6.6. Hasil Pengujian: Key Tidak Ditemukan

No	Jumlah Karakter	Key Ditemukan?	Sesuai?	Waktu Eksekusi
1	16	Tidak	Tidak	2:30:21
10	32	Tidak	Tidak	2:39:55
12	32	Tidak	Tidak	2:57:58
13	32	Tidak	Tidak	2:53:55
20	48	Tidak	Tidak	2:35:23
21	48	Tidak	Tidak	2:58:23
23	64	Tidak	Tidak	2:54:52
Rata-Rata				2:47:15

Sumber : Pengujian

Dari data hasil pengujian diatas maka waktu rata-rata yang dibutuhkan untuk memecahkan BREA adalah 2 jam 47 menit. Jika dibandingkan dengan waktu pemecahan yang dibutuhkan oleh beberapa algoritma seperti AES ($5x10^{21}$ tahun), 3DES (800 hari), dan DES (400 hari). Maka BREA dapat dikategorikan sebagai algoritma yang tidak aman. Gambar 6.3. menujukkan grafik pengujian untuk skema *Known Ciphertext*.



Gambar 6.3. Grafik Pengujian Brute Force Skema KC
Sumber : Pengujian

Dari grafik diatas menunjukkan bahwa 10 sampel ditemukan, 13 sampel ditemukan dikamus namun tidak sesuai, dan 7 sampel tidak ditemukan. Hal ini terjadi karena beberapa hal yang akan dijelaskan sebagai berikut :

1. Untuk konteks yang ditemukan, kata yang digunakan ditemukan didalam kamus dan sesuai dengan keseluruhan rangkaian kalimat.
2. Untuk konteks yang ditemukan dikamus namun key tidak sesuai, hal ini terjadi karena ada beberapa kata yang cocok ketika dilakukan dekripsi tanpa key, Namun kata tersebut bukanlah kata yang seharusnya digunakan didalam teks yang dimaksud sehingga key yang digunakan tidak mengembalikan *ciphertext* ke bentuk *plaintext* yang sesuai.
3. Untuk konteks yang tidak ditemukan, hal ini terjadi karena beberapa kata didalam *ciphertext* yang telah di dekripsi bukan merupakan kata baku atau kata baku yang tidak umum. Atau isi dari pesan merupakan sebuah link yang tidak ada didalam kamus.

BAB VII
PENUTUP

7.1. Kesimpulan

Berdasarkan hasil perancangan, implementasi dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut :

1. Dari segi kecepatan eksekusi, waktu eksekusi BREA ketika di implemetasikan kedalam pesan instan adalah 0,128 detik. Pesan instan yang tidak menggunakan BREA memiliki waktu eksekusi 0,1269 detik. Selisih dari waktu eksekusi pesan instan yang menggunakan BREA dan tidak menggunakan BREA adalah 0,0018 detik.
2. Dari dari segi keamanan, dapat dikategorikan tidak aman. Berdasarkan hasil penelitian waktu rata-rata yang dibutuhkan untuk memecahkan BREA adalah 2 Jam 47 menit. Hal ini disebabkan penggunaan key 16 bit yang membuat kriptografi dapat dipecahkan dengan waktu rata-rata 2 Jam 47 menit. Jika dibandingkan dengan waktu pemecahan yang dibutuhkan oleh beberapa algoritma seperti AES ($5x10^{21}$ tahun), 3DES (800 hari), dan DES (400 hari). BREA memiliki waktu pemecahan yang lebih singkat dibandingkan dengan AES, 3DES dan DES.

7.2. Saran

Saran yang dapat diberikan untuk pengembangan perangkat lunak ini antara lain :

1. Untuk pengembangan lebih lanjut, key BREA harus menggunakan tingkat kerumitan key yang lebih tinggi. Dengan demikian waktu yang dibutuhkan untuk pemecahan key secara Brute Force dapat menjadi lebih lama dan lebih aman.
2. Untuk pengembangan yang mengikuti prosedur penggunaan key yang sesuai dengan BREA, dibutuhkan *key distribution* yang mantap sehingga tidak dapat terbaca ketika dilakukan *sniffing* pada jaringan.
3. Untuk pengembangan perangkat pesan instan lebih lanjut dapat menambahkan beberapa fitur tambahan seperti kirim data, video, dan sebagainya.

4. Untuk Pengembangan teknik Pencocokan teks hasil dekripsi tanpa key dengan kamus, dapat dilakukan pengecekan yang lebih spesifik dalam melakukan pemisahan kata dengan tanda baca atau menggunakan sistem Artificial Intelligent (AI) untuk memecahkan permasalahan pemisahan kata dengan tanda baca agar hasil pendekripsian tanpa key dapat menjadi lebih akurat.
5. Untuk penelitian keamanan lebih lanjut, maka dapat dilakukan proses *Sniffing* yang sebenarnya dengan menggunakan 2 komputer sebagai client dan 1 komputer sebagai penyerang (*sniffer*).

DAFTAR PUSTAKA

[BSU-12] Bathi, Sunita.2012, *A New Approach towards Encryption Schemes: Byte – Rotation Encryption Algorithm*. WCECS.

[ZEF-11] Zam,Efvy.2011, Buku Sakti Hacker. Mediakita, Halaman 89.

[WAP-12] Wicaksana, Anata Pandu.2012, Teknik Kriptografi dengan Menggunakan Algoritma Gabungan LCG-ROT, Institut Teknologi Bandung

[MRP-04] Munir, Rinaldi.2004, Pengantar Kriptografi, Institut Teknologi Bandung.

[MRT-04] Munir, Rinaldi.2004, Tipe dan Mode Algoritma Simetri, Institut Teknologi Bandung.

[MRF-04] Munir, Rinaldi.2004, Fungsi Hash Satu-Arah dan Algoritma MD5, Institut Teknologi Bandung.

[RDA-00] Roylance, David.2000, *Matriks and Index Notation*. Massachusetts Institue of Technology.

[MAT-13] http://www.maths.manchester.ac.uk/kd/ma2m1/matrices.pdf diakses pada 30 Juni 2013

[KSE-10] Kromodimoeljo, Sentot.2010. Teori dan Aplikasi Kriptografi. SPK IT Consulting, Halaman 5.

[ECO-13] http://www.giac.org/cissp-papers/57.pdf diakses pada 30 Juni 2013

[AHO-10] Alahazi, Hamdan.2010. *New Comparative Study Between DES, 3DES and AES within Nine Factors*. Journal Of Computing.

LAMPIRAN

Lampiran 1 : Data Uji

No	Kata
1	bacod
2	masbro
3	thx ya
4	tanya dong
5	apa kabar lo
6	mib minta dunk
7	hai semua nya
8	aku salah apa ??
9	apa ada yg kurang?
10	mandi sana bauuuuu
11	selamat subuh warga gogs
12	bagi link dwg parser donk
13	backgroundnnya galak euy
14	bisa ganti warna tulisan ID gak?
15	di coba aja id nya,udh bsa kok ^^
16	jadi beneran itu sms gituan ? awkoawk
17	tapi kenapa di japank gak ada list nya
18	ada yg tau ga.. ganti background warcraft
19	ape lo sal ? ngeluh2.. piket sono wakkakaka
20	jepang menang kok.. udah gw tanya eyang subur
21	dih , mengada2 ini dante- yaoi = orang pintar
22	bentar yo ham . baru di umum banyak yg tenggelam
23	gan mau tanya, patch dota gogs yang 1.24b kan ya?
24	ane mau donasi nih uda sms ke nomer gada yang bales
25	udh gw request di kolom request tp lom ada tanggepan
26	kalo dikatain anak Centa , yg ngatain kena flame gak?
27	di urus ya gan http://forum.gogamers.us/showthread....978#post429978
28	gan mau nanya ini emang op nya ga ol ato apa kenapa case2 ngga diselesaikan? thx
29	laporan aku buat leaver di ID forum Hippocrates sudah 3 hari belom ada tanggapan nih
30	kk masak belom ada yang bisa parser ini http://forum.gogamers.us/showthread....dlynnekaramina