APLIKASI PENGAMANAN CHAT DENGAN ALGORITMA CAST-128 BERBASIS WEB PADA PT. TJOKRO BERSAUDARA

Akhmad Affandi[1)], **Ir. Siswanto, M.M**[2)]
[1]Teknik Informatika, Fakultas Teknologi Informasi, Universitas Budi Luhur
[1,2]Jl. Raya Ciledug, Petukangan Utara, Kebayoran Lama, Jakarta Selatan 12260
E-mail : fandyaffan18@gmail.com[1)], siswanto@budiluhur.ac.id[2)]

Abstrak

Aplikasi pengamanan data ini dirancang untuk mengamankan data – data penting pada PT. Tjokro Bersaudara terutama data tagihan biaya corporate dan faktur pajak yang berhubungan dengan tagihan keuangan dari kantor pusat ke kantor cabang. Dokumen tersebut sering kali dikirimkan melalui aplikasi chat, dan sangat mudah dilihat apabila lupa keluar dari aplikasi dan seseorang yang tidak bertanggung jawab memakai komputer yang sama. Keamanan data dibutuhkan karena data tersebut untuk setiap kantor cabang berbeda. Permasalahan tersebut dapat dihadapi dengan membuat aplikasi untuk mencegah pihak lain melihat dokumen rahasia tersebut. Juga menjamin keaslian data sensitif dan penting hanya dapat diterima dan dibaca oleh orang-orang yang berhak mendapatkan data tersebut. Dalam penulisan ini algoritma yang digunakan yaitu algoritma CAST-128 sebagai metode kriptografi. CAST-128 merupakan algortima Block Cipher yang termasuk kedalam jenis sistem kriptografi kunci simetris. Penggunaan sistem kriptografi ini diimplementasikan kedalam aplikasi chatting yang bertujuan untuk mengamankan data rahasia. Bahasa pemrograman yang digunakan dalam membangun aplikasi pengamanan data ini adalah bahasa pemrograman PHP berbasis web. Hasil dari pengujian kriptografi ini, data dapat diamankan untuk menghindari serangan cryptanalysis. Hanya saja rata-rata ukuran data hasil proses enkripsi akan bertambah sekitar 33,3513 persen dari ukuran data asli dan juga program belum bisa melakukan enkripsi pada beberapa tipe file dan dalam waktu yang bersamaan.

Kata kunci : Kriptografi, *Encrypt*, *CAST-128*, Keamanan data, *Chat*

1. PENDAHULUAN

PT. Tjokro Bersaudara merupakan perusahaan bergerak dibidang Fabrikasi dan Machining yang Berdiri sejak 15 Desember 1968 di Surabaya. Saat ini berpusat di Cideng Timur dan memiliki kantor cabang yang tersebar di Indonesia.

Cabang-cabang yang tersebar tersebut tetap megikuti pertauran-peraturan yang ada di pusat, termasuk adanya kewajiban kantor cabang untuk membayar *corporate fee* kepada pusat setiap bulannya. Setiap awal bulan admin finance pusat membuat tagihan kepada masing-masing cabang. Untuk masing-masing kantor cabang mempunyai tagihan dengan jumlah yang berbeda-beda, tergantung kebijakan dari direktur. Oleh karena itu, data tagihan tersebut sangat rahasia. Data tersebut dikirimkan seringkali menggunakan chat kepada controller cabang untuk selanjutnya diberikan ke finance masing-masing cabang. Begitu juga dengan faktur pajak dibuat setelah cabang membayar corporate fee, kemudian dikirim ke controller cabang lalu diberikan ke finance masing-masing cabang. Tentunya dalam pengiriman ke controller cabang tidak ada pengamanan data rahasia tersbut.

Untuk menjaga keamanan data tersebut maka salah satu cara yang harus dilakukan adalah dengan melakukan *encrypt* pada data yang akan dikirim sehingga hanya pihak yang berhak atas data tersebut yang memiliki kunci untuk membuka data.

Salah satu cara yang dapat digunakan untuk menjaga kerahasiaan informasi tersebut adalah dengan menyamarkannya menjadi bentuk tersandi yang tidak bermakna. Hal tersebut dapat dilakukan dalam kriptografi.

Pada saat ini terlihat keamanan data yang masih rendah di PT. Tjokro Bersaudara, maka masalahnya adalah terkadang ada pihak yang tidak bertanggung jawab melihat dengan mudah data rahasia yang dikirimkan melalui chat di PC yang dipakai bersama dan tidak di *logout* dari aplikasi.

Adapun tujuan yang ingin dicapai dalam pembuatan penelitian ini adalah :

a. Membuat aplikasi mencegah pihak yang tidak berhak menerima data bisa membaca isi dokumen penting perusahaan.
b. Menjamin data *file* sensitif dan penting (yang berhubungan dengan tagihan keuangan) hanya dapat diterima dan dibaca oleh orang yang seharusnya.
c. Membuat aplikasi pengamanan data yang bisa mengamankan isi dari berbagai macam format *file* yang digunakan PT. Tjokro Bersaudara.

Agar penulisan penelitian ini lebih terarah, batasan – batasan masalah yang dimiliki yaitu :

a. Besar *file* optimal yang dapat di proses oleh aplikasi adalah 2 MB.
b. Algoritma yang digunakan adalah algoritma kriptografi *CAST-128.*

c. Browser yang digunakan adalah Mozilla Firefox.
d. Tipe *file* untuk lampiran hanya mendukung .pdf dan .jpg.
e. Besar kunci dipakai hanya 128 bit (16 karakter).
f. Jumlah lampiran *file* yang dapat disisipkan dan di enkripsi adalah 1 *file*.
g. Kantor cabang hanya bisa melakukan chat dengan kantor pusat (tidak ada komunikasi antar cabang).

Aplikasi yang dibangun berupa *chat*. *Chat* adalah komunikasi berbasis teks yang hidup atau *real-time*[3]. Misalnya, ketika berbicara dengan seseorang di *chat*, teks yang diketik langsung diterima oleh peserta lainnya. Sebaliknya, komunikasi berbasis teks lainnya seperti *e-mail* adalah mode korespondensi yang tidak *real-time*.

Keamanan pada aplikasi *chat* ini menggunakan metode kriptografi. Kriptografi merupakan ilmu mengenai teknik enkripsi dimana "naskah asli" (*plaintext*) diacak menggunakan suatu kunci enkripsi menjadi "naskah acak yang sulit dibaca" (*ciphertext*) oleh seseorang yang tidak memiliki kunci dekripsi. Dekripsi menggunakan kunci dekripsi bisa mendapatkan kembali data asli [1].

Algoritma kriptografi yang digunakan adalah *CAST-128* dan *Base64*. Pada *CAST-128* menggunakan 12 atau 16 *round* jaringan *feistel* dengan 64-bit ukuran blok dan ukuran kunci 40 hingga 128 bit [2]. Komponen-komponen termasuk di dalamnya sebuah 8 x 32-bit *S-box* besar yang berdasarkan *bent-function*,rotasi kunci *independent*, *modular addition*, dan substraksi, serta operasi XOR. Algoritma *CAST-128* termasuk diklasifikasikan sebagai algoritma enkripsi yang menggunakan jaringan *feistel*. Berikut ini langkah-langkah secara garis besar dari fungsi enkripsi dalam algoritma *CAST-128* :

Masukkan :

a. *Plaintext* p1…p64 (Blok *plaintext* dengan panjang 64 bit).
b. Kunci K= k1….k128 (Kunci dengan panjang 128 bit).

Keluaran :

Ciphertext c1….c64 (Blok *ciphertext* dengan panjang 64 bit).

Fungsi enkripsi dalam algoritma *CAST-128* [4]:

a. Penjadwalan kunci :
   Menentukan 16 pasang kunci internal dari masukan pengguna.
b. Bagi blok menjadi 2 bagian :
   64 bit blok *plaintext* dibagi menjadi dua bagian yang sama, yaitu masing-masing bagian kiri dan bagian kanan dengan panjang 32 bit.
c. 16 putaran feistel
d. Konkatenasi untuk membentuk text-code : Menukarkan bagian kiri dengan bagian kanan blok di putaran terakhir. Kemudian, kedua bagian digabungkan menjadi satu dan menjadi *text-code*.

Berikut merupakan penjelasan dari model jaringan *feistel* :

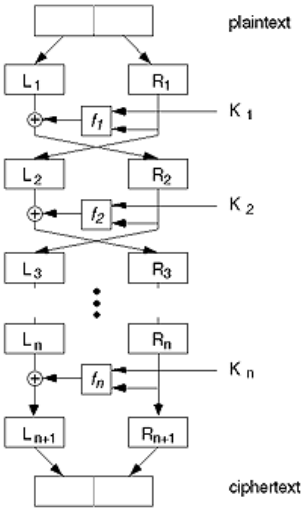
a. Bagi blok dengan panjang n bit menjadi 2 bagian yang sama , kiri (L), dan kanan (R). Dengan syarat panjang n harus genap sehingga panjang L dan R sama yaitu n/2.
b. Definisikan *cipher* blok berulang dimana hasil dari putaran ke-i ditentukan dari hasil putaran sebelumnya, yaitu:
   Li = Ri-1;
   Ri = Li-1 ^ f (Ri-1,Kmi,Kri);
   Dalam hal ini, i adalah bilangan 1, dan r adalah jumlah putaran. Ki adalah kunci internal pada putaran ke-i. Sedangkan f adalah fungsi transformasi. Sebagai catatan, gabungan L0 dan R0 adalah plaintext, sedangkan *ciphertext* didapatkan setelah L dan R hasil putaran terakhir dipertukarkan (Rr, Lr). Jaringan feistel bersifat reversible karena operator XOR mengkombinasikan setengah bagian kiri dengan hasil f, sehingga:
   Li-1 ^f (Ri-1 , Ki) ^ f (Ri-1, Ki) = Li-1
   Sifat reversible ini tidak bergantung pada f, sehingga tidak perlu membuat algoritma baru untuk mendekripsi *ciphertext* [4].

Dalam operasinya, algoritma *CAST-128* menggunakan 8 x 32 *S-box*, untuk S1, S2, S3, dan S4 adalah *S-box* fungsi putaran untuk mendekripsi, sedangkan empat *S-box* lainnya yaitu : S5, S6, S7, S8 merupakan *S-box* untuk penjadwalan kunci. Karena *S-box* adalah kotak 8 x 32, kotak ini akan menerima 8 bit masukan dan mengeluarkan 32 bit keluaran.

Karena *CAST-128* menggunakan 16 putaran *feistel*, disetiap putarannya *CAST-128* membutuhkan sepasang kunci internal sebanyak 32 buah. 16 buah pertama(Km1…km16) dan 16 buah berikutnya(Kr1..Kr16). 16 kunci Km1….Km16 digunakan untuk *masking* disetiap putarannya, sedangkan Kr1…Kr16 digunakan untuk kunci rotasi. Kunci internal ini dibangkitkan dari kunci eksternal yang diberikan oleh pengguna. Jadi, dari kunci eksternal yang panjangnya 128 bit, dibentuk 16 pasang kunci internal, yaitu pasangan 32 bit kunci masking dan 5 bit kunci rotasi. Algoritma pembangkitan kunci internal diberikan dengan asumsi kunci eksternal sepanjang 128 bit dibagi menjadi 16 bytes kunci internal, yaitu: x0x1x2x3x4x5x6x7x8x9xAxBxCxDxExF, dimana x0 menunjukkan *most significant byte* (MSB) dan xF menunjukkan *least significant byte* (LSB). Selain itu, digunakan z0..zF yang merupakan temporary byte untuk penyimpanan byte sementara. Digunakan juga Si[ ] yang menunjukan *S-box* ke-i dan "^" melambangkan operasi XOR.

Dari K1..K32 yang dibangkitkan, 16 kunci pertama akan digunakan untuk kunci masking (satu kunci per putaran) dan 16 sisanya digunkana untuk kunci rotasi (satu kunci per putaran). Untuk kunci rotasi, hanya 5 bit dari *least significant byte* yang digunakan. Algoritma penjadwalan kunci diberikan sebagai berikut:

For (i=1; i<=16; i++) { Kmi = Ki; Kri = K16+I; }

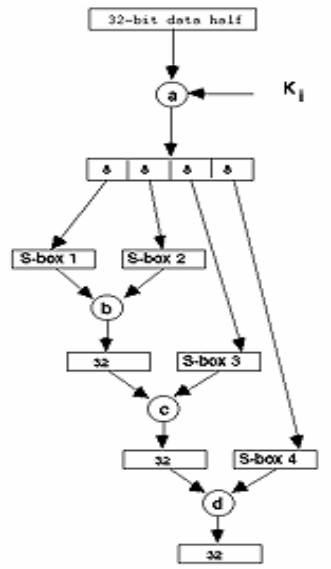


Gambar 1 : Putaran feistel n putaran

Terlihat pada gambar 1, jika n = 16, yang berarti ada 16 putaran *feistel*, akan diperlukan 16 kunci,, yaitu K1..K16. Dalam *CAST-128* ke-16 kunci tersebut diisi dengan ke-16 pasang kunci yang telah dibangkitkan.

Contoh: K1 adalah sepanjang kunci Km1 dan Kr1

Pada fungsi enkripsi, dari 16 putaran *feistel* yang digunakan, *CAST-128* menggunakan 3 jenis putaran yang berbeda. Tiga jenis putaran tersebut dibedakan menurut 3 tipe fungsi enkripsi yang berbeda.



Gambar 2 : Fungsi enkripsi CAST-128

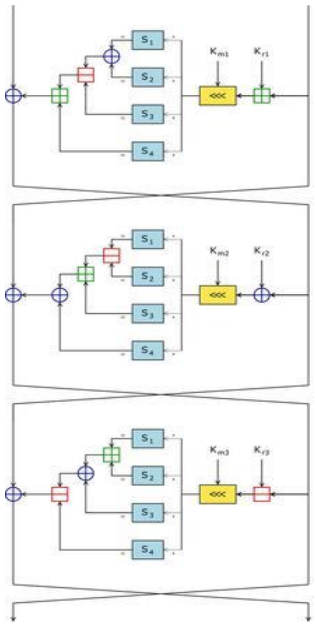
Gambar 2, menunjukkan bahwa fungsi enkripsi *CAST-128* menerima input 32 bit data *half*, yang didapatkan dari 64 bit blok yang telah dibagi dua pada saat masuk jaringan *feistel*. Lalu operasi a dilakukan pada 32 bit data masukan ini. Setelah operasi a dilakukan, hasilnya akan dibagi menjadi 4 bagian dengan panjang yang sama (8 bit). 8 bit pertama akan menjadi input dari *S-box* 1 dan 8 bit kedua akan menjadi input dari *S-box* 2. Keluaran yang dihasilkan 32 bit dari masukan 8 bit, karena *S-box* yang dipakai adalah kotak 8 x 32 bit. Selanjutnya, hasil dari kedua *S-box* tersebut akan digabung dengan hasil dari *S-box* 3 dengan masukan 8 bit ketiga. Kali ini Operasi yang digunakan vadalah operasi c. 8 bit terakhir akan menjadi masukan *S-box* keempat dan hasilnya akan digabung dengan hasil operasi c menggunakan operasi d. Hasil yang diperoleh dari seluruh fungsi ini adalah 32 bit, sesuai dengan 32 bit masukan. 32bit hasil ini akan kembali masuk ke putaran *feistel*.

1) Tipe 1
   a) Operasi a adalah penjumlahan bit modulo 2^{32} dan pergeseran bit ke kiri. Bit- bit masukan akan ditambahkan dengan Kmi (sesuai putaran ke-i) dan akan digeser kekiri sebanyak Kri (sesuai putaran ke-i)
   b) Operasi b adalah XOR
   c) Operasi c adalah pengurangan bit modulo 2^{32}
   d) Operasi d adalah penjumlahan bit modulo 2^{32}
2) Tipe 2
   a) Operasi a adalah XOR dan pergeseran bit ke kiri . Bit-bit masukan akan di-XOR-kan dengan Kmi (sesuai putaran ke-i) dan akan digeser kekiri (sesuai putaran ke-i).
   b) Operasi b adalah pengurangan bit modulo 2^{32}
   c) Operasi c adalah penjumlahan bit modulo 2^{32}
   d) Operasi d adalah XOR
3) Tipe 3
   a) Operasi a adalah pengurangan bit modulo 2^{32} dan pergeseran bit kekiri. Kmi (sesuai putaran ke-i) akan dikurangkan dengan bit-bit masukan dan akan digeser kekiri sebanyak Kri (sesuai putaran ke-i).
   b) Operasi b adalah penjumlahan bit modulo 2^{32}
   c) Operasi c adalah XOR
   d) Operasi d adalah pengurangan bit modulo 2^{32}

Dari ketiga tipe tersebut secara matematis dapat dinyatakan dengan :

Fungsi Tipe 1: I = ((Kmi + D) <<< Kri)
f = ((S1[Ia] ^ S2[Ib]) - S3[Ic]) + S4[Id]
Fungsi Tipe 2: I = ((Kmi ^ D) <<< Kri)
f = ((S1[Ia] - S2[Ib]) + S3[Ic]) ^ S4[Id]
Fungsi Tipe 3: I = ((Kmi - D) <<< Kri)
f = ((S1[Ia] + S2[Ib]) ^ S3[Ic]) - S4[Id]

Dimana D adalah 32 bit data input, I adalah operasi a terhadap D. I dibagi menjadi 4 bagian sepanjang 8 bit Ia, Ib, Ic, dan Id terurut mulai dari *most significant byte* (MSB) sampai *least significant byte* (LSB). f adalah hasil fungsi enkripsi. Sebagai catatan "+" dan "-" adalah penjumlahan dan pengurangan modulo 2^{32}, "^" adalah XOR, dan "<<<" adalah pergeseran bit kekiri.



Gambar 3 : Tiga tipe fungsi CAST-128

Berikut jadwal pemakaian ketiga fungsi tersebut dalam jaringan *feistel* :

1) Putaran 1, 4, 7, 10, 13, dan 16 menggunakan fungsi tipe 1
2) Putaran 2, 5, 8, 11, dan 14 menggunakan fungsi tipe 2
3) Putaran 3, 6, 9, 12,dan 15 menggunakan fungsi tipe 3 [4].

Algoritma kriptografi berikutnya yang digunakan adalah *Base64*. Skema *encode/decode Base64* umumnya digunakan untuk menyandikan data biner yang akan disimpan dan ditransferkan melalui media yang dirancang untuk menangani data tekstual. Hal ini digunakan untuk memastikan data tetap utuh tanpa modifikasi selama proses transformasi. *Base 64* digunakan juga pada layanan *mailing* dan penyimpanan data pada XML. Transformasi *Base64* merupakan salah satu algoritma untuk *Encoding* dan *Decoding* suatu data ke dalam format ASCII, yang didasarkan pada bilangan dasar 64 atau bisa dikatakan sebagai salah satu metoda yang digunakan untuk melakukan encoding (penyandian) terhadap data binary[5]. Karakter yang dihasilkan pada transformasi *Base64* ini terdiri dari A…Z, a…z dan 0…9, serta ditambah dengan 2 karakter terakhir yang bersimbol *plus* (+) dan *slash* (/) serta 1 buah karakter sama dengan (=) yang digunakan untuk menggenapkan data extension atau istilahnya disebut sebagai pengisi pad. Karakter simbol yang akan dihasilkan akan tergantung dari proses algoritma yang berjalan. Base64 banyak digunakan sebagai media data format untuk mengirim data, karena hasil dari base64 merupakan plaintext, maka data akan lebih mudah dikirim dibandingkan format data yang bebentuk *binary*.

2. METODE PENELITIAN

Dalam penyusunan laporan penelitian ini telah dilakukan riset untuk memperoleh fakta-fakta mengumpulkan data yang diperlukan. Adapun metode yang digunakan adalah metode *waterfall* dengan langkah-langkah sebagai berikut:

a. Analisa Kebutuhan, menganalisis masalahan, kebutuhan, keperluan, dan penggunaan apa saja yang akan diperlukan untuk pengamanan dokumen di PT. Tjokro Bersaudara.
   1) Perencanaan, mengidentifikasi masalah-masalah keamanan dokumen yang berkaitan dengan tagihan keuangan di PT. Tjokro Bersaudara.
   2) Penelitian lapangan, yaitu melakukan observasi atau praktek lapangan secara langsung di perusahaan terkait guna mendapatkan data yang akurat dan dapat dipertanggung jawabkan keabsahannya.
b. Desain Sistem, persiapan rancang bangun implementasi pada PT. Tjokro Bersaudara yang menggambarkan bagaimana suatu sistem dibentuk yang berupa penggambaran.
c. Ujicoba program, Ujicoba mempresentasikan ketidak normalan yang terjadi pada pengembangan program dengan bahasa pemrograman PHP kepada semua jenis *file* yang digunakan PT. Tjokro Bersaudara, jenis *file* yang digunakan diantaranya .pdf, dan .jpg.
d. Implementasi, tahap dimana semua elemen dan aktivitas sistem disatukan dengan langkah - langkah sebagai berikut :
   1) Menyiapkan fasilitas fisik. Fasilitas - fasilitas fisik yang disiapkan antara lain komputer dan peripheralnya, termasuk keamanan fisik untuk menjaga berlangsungnya peralatan dalam jangka waktu yang lama.
   2) Menyiapkan pengguna, pemakai disiapkan dengan terlebih dahulu yaitu dengan memberikan pelatihan secara prosedural maupun tutorial mengenai program. Tujuannya adalah agar para pengguna mengerti dan mengusai cara kerja program.
   3) Melakukan simulasi, kegiatan simulasi berupa pengujian program secara nyata yang melibatkan personil yang sesungguhnya.

3. HASIL DAN PEMBAHASAN

3.1. Rancangan Menu

Tahapan dalam perancangan aplikasi adalah sebagai berikut :

a. Proses

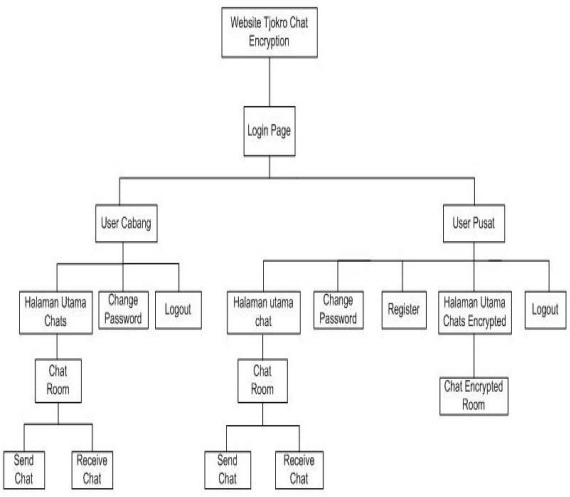
Perancangan proses yang dimaksudkan adalah bagaimana sistem akan bekerja, proses-proses yang digunakan mulai dari *user* mengakses halaman *web*, melakukan *login* lalu menulis pesan *chat* dan *attach file* untuk dienkripsi, kemudian diproses oleh aplikasi

1.	Mulai
2.	*Plaintext* = 64 bit
3.	*Key* = 128 bit
4.	i = 0
5.	i = i + 1
6.	Kmi = Ki
7.	Kri = K16 + i
8.	*If* i = 16, *then*
9.	*Plaintext* dibagi 2 = L0(32 bit) & R0 (32bit)
10.	r = 0
11.	r = r + 1
12.	I = ((Kmi ^ D)<<<Kri)
13.	F = ((S1[Ia]-S2[Ib])+S3[Ic])^S4[Id])
14.	r = r + 1
15.	I = ((Kmi + D)<<<Kri)
16.	F = ((S1[Ia]^S2[Ib])-S3[Ic])+S4[Id])
17.	r = r + 1
18.	I = ((Kmi - D)<<<Kri)
19.	F = ((S1[Ia]+S2[Ib])^S3[Ic])-S4[Id])
20.	*If* r = 16, *then*
21.	Tukar posisi L16 dengan R16
22.	Satukan L16 dengan R16
23.	*Ciphertext* = 64 bit
24.	*Else*
25.	Kembali ke baris 11
26.	*Else*
27.	Kembali ke baris 5
28.	*End If*

sehingga dapat mengeluarkan *output* berupa hasil *encrypt* pesan tersebut.

b. Struktur menu

Struktur menu apikasi dibuat sederhana agar setiap pemakai dapat menggunakannya tanpa kesulitan. Bentuk rancangan menu dapat dilihat pada gambar dibawah ini:



Gambar 4 : Struktur menu aplikasi keamanan chat

3.2. Algoritma

Algoritma yang dipakai pada aplikasi pengaman chat ini yaitu *CAST-128* dan *Base64*, berikut merupakan alur enkripsi dan dekripsi dari masing-masing algoritma.

A. *CAST-128*

Berikut alur enkripsi pada *cast-128:*

Berikut alur dekripsi pada *cast-128:*

1.	Mulai
2.	*Ciphertext* = 64 bit
3.	Key = 128 bit
4.	i = 0
5.	i = i + 1
6.	Kmi = Ki
7.	Kri = K16 + i
8.	*If* i = 16, *then*
9.	*Ciphertext* dibagi 2=L0(32 bit)&R0 (32bit)
10.	Tukar posisi L0 dengan R0
11.	r = 0
12.	r = r + 1
13.	I = ((Kmi ^ D)<<<Kri)
14.	F = ((S1[Ia]-S2[Ib])+S3[Ic])^S4[Id])
15.	r = r + 1
16.	I = ((Kmi - D)<<<Kri)
17.	F = ((S1[Ia]+S2[Ib])^S3[Ic])-S4[Id])
18.	r = r + 1
19.	I = ((Kmi + D)<<<Kri)
20.	F = ((S1[Ia]^S2[Ib])-S3[Ic])+S4[Id])
21.	*If* r = 16, *then*
22.	Satukan L16 dengan R16
23.	*Plaintext* = 64 bit
24.	*Else*
25.	Kembali ke baris 12
26.	*Else*
27.	Kembali ke baris 5
28.	*End If*

B. Base64

Berikut alur enkripsi pada *base64:*

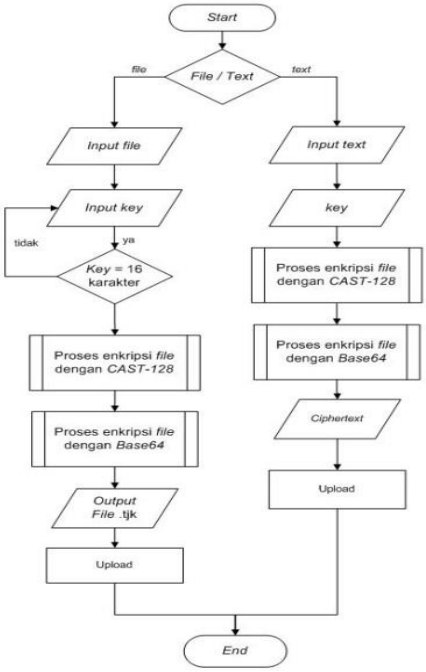
1.	Mulai
2.	Masukkan *Plaintext*
3.	Pecah *String per* 3 karakter
4.	Karakter dijadikan bilangan biner (sesuaikan dengan tabel ASCII)
5.	3 karakter berjumlah 24 bit
6.	Bagi 24 bit menjadi 4 bagian, perbagian masing-masing 6 bit
7.	*If* jumlah karakter != 3, *then*
8.	Tambahkan karakter "=" (sama dengan) agar menjadi 3 karakter
9.	Ubah biner menjadi *decimal*
10.	Konversikan *decimal* sebagai *index* untuk memilih karakter pada tabel *base64*
11.	*Printf encrypt*

Berikut alur dekripsi pada *base64:*

7.	Bagi 24 bit menjadi 3 bagian, perbagian masing-masing 8 bit
8.	Ubah biner menjadi *decimal*
9.	Konversikan *decimal* sebagai *index* untuk memilih karakter pada tabel ACII
10.	*Printf decrypt*

3.3. Proses Enkrip dan Dekrip

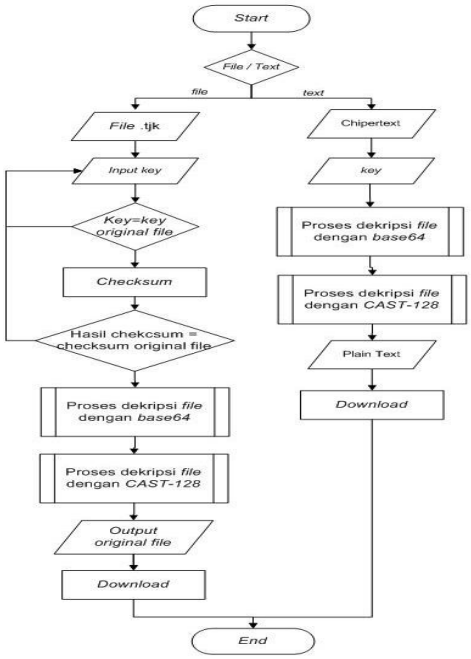
Proses *encrypt* dimulai dari proses *input*, untuk *input* ini dapat berupa *input text* dan *input file*, jika *input file* perlu memasukkan *secret key* sepanjang 16 karakter, kemudian melakukan proses *encrypt (encrypt CAST-128 dan encode Base64)*, setelah selesai proses *encrypt*, pesan atau *file* disimpan dalam *database*.



Gambar 5 : Flowchart proses enkripsi

Untuk proses *decrypt file* dilakukan proses pengecekan *secret key* yang akan dimasukkan sesuai dengan *secret key* yang dimasukkan pada proses *encrypt*, jika salah memasukkan *key* maka akan muncul pesan "*Key Not Valid*" dan setelah itu melakukan *checksum* dimana pada saat proses ini isi *file* (*plaintext)* yang sebelumnya dirubah kedalam bentuk md5 akan dicek. Jika isi *file* yang akan di *decrypt* sama dengan bentuk md5 yang tersimpan didalam *file*, maka proses *decrypt* akan dilanjutkan. Jika berbeda maka proses tidak akan berjalan dan akan muncul pesan error *"Content Changed"*. Untuk pesan teks, akan secara otomatis didekripsi tanpa memasukkan *key* secara manual.

1.	Mulai
2.	Masukkan *Ciphertext*
3.	Pecah *String per* 4 karakter
4.	Karakter dijadikan bilangan biner (sesuaikan dengan tabel *index Base64*)
5.	Hilangkan karakter "=" (sama dengan)
6.	4 karakter berjumlah 24 bit



Gambar 6 : Flowchart proses dekripsi

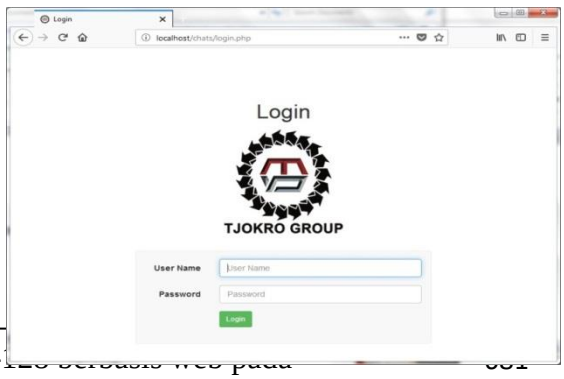
3.4. Spesifikasi Hardware dan Software

Berikut informasi hardware dan software dalam proses pengujian:

1) Spesifikasi *Hardware*
   a) Intel Core i3-2330M 2.20 GHz
   b) RAM 2 GB
   c) Harddisk 500 GB
   d) VGA NVIDIA GeForce GT 520M 2GB
2) Spesifikasi *Software*
   a) Windows 7 Ultimate 64-bit
   b) XAMPP Control Panel Versi 3.2.2
   c) Software Notepad++ v7.5.2
   d) *Browser* Mozilla Firefox v57.0.2

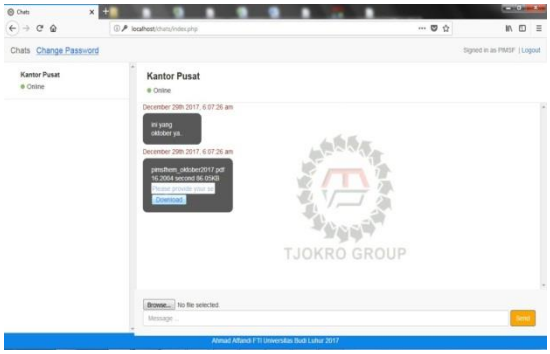
3.5. Tampilan Menu Program

a. Tampilan petama saat membuka aplikasi ini adalah tampilan login.



Gambar 7 : Tampilan halaman login

b. Halaman *chat room* pada user cabang, dimana *user* cabang hanya bisa berkomunikasi dengan *user* pusat. Untuk melihat isi lampiran yang dikirimkan *user* pusat, *user* cabang harus mengisi *key* pada kolom "*insert key*" (*key* yang dimasukkan harus sama dengan *key* yang dimasukkan oleh *user* pusat).



Gambar 8 : Tampilan halaman chat room pada user cabang

3.6. Hasil Pengujian

Dari beberapa hasil pengujian *encrypt file* yang dilakukan sebanyak 5 kali dengan ukuran *file* kurang dari 2 MB, maka didapatkan hasil sebagai berikut :

Tabel 1. Tabel pengujian *encrypt file*

No	Nama *File*	Sebelum *Encrypt*		Ukuran *file encrypt* (bytes)	Lama Proses *Encrypt* (S)	Tambahan Ukuran *File* (%)
		Format *File*	Ukuran *File* (bytes)			
1	Cilegon_desember	pdf	414102	552211	16.255	33.351
2	balikapapan_oktober	pdf	88418	117971	3.525	33.424
3	Management_charge_TPP_sby_desember	jpg	431850	575879	16.988	33.352
4	MT_oktober	pdf	419190	558995	16.473	33.351
5	TL-WN822N-User-Guide	pdf	1648039	2197459	66.924	33.338
Rata-rata penambahan ukuran *file* setelah dilakukan *encrypt* =						33.3513

Setelah percobaan proses *encrypt*, dilakukan percobaan proses *decrypt* pada sepuluh *file* yang telah ter*encrypt* pada percobaan sebelumnya, dan didapat hasil seperti pada tabel berikut:

Tabel 2. Tabel pengujian *decrypt file*

No	Nama *File*	Ukuran *File* (bytes)	Setelah *Decrypt*		Lama Proses *Decrypt* (S)	Tambahan Ukuran *File* (%)
			Format *File*	Ukuran *File* (bytes)		
1	Cilegon_desember	552211	pdf	414104	16.73	-25.010
2	balikapapan_oktober	117971	pdf	88424	4.87	-25.046
3	Management_charge_TPP_sby_desember	575879	jpg	431856	17.72	-25.001
4	MT_oktober	558995	pdf	419192	17.54	-25.010
5	TL-WN822N-User-Guide	2197459	pdf	1648040	67.69	-25.002
Rata-rata penambahan ukuran *file* setelah dilakukan *decrypt* =						-25.0086

4. KESIMPULAN

Berdasarkan hasil penelitian dan pembahasan serta uji coba sistem dapat disimpulkan sebagai berikut :

a. Pegamanan data pesan *chat* dapat diamankan dengan algortima kriptografi *CAST-128.*
b. Data *file* yang dilampirkan tidak dapat dibuka oleh pihak yang tidak berhak yang tidak memiliki *secret key*.
c. Program sistem keamanan dengan sistem kriptografi algoritma *CAST-128* telah diuji coba, sehingga program dinyatakan sudah sesuai.
d. Ukuran *file* setelah melalui proses *encrypt* akan bertambah besar dibandingkan dengan ukuran *file* awal sebelum dilakukan proses *encrypt*.
e. Ukuran *file* untuk dilampirkan yang paling optimal untuk aplikasi ini adalah kurang dari 2 MB. Jika ukuran diatas 2 MB tidak bisa diproses.

f. Rata-rata ukuran *file* yang telah melalui proses *encrypt* bertambah sekitar 33.3513 persen dari ukuran asli *file* sebelum melalui proses *encrypt*.
g. Rata-rata perubahan ukuran *file* yang telah melalui proses *decrypt* akan bertambah sebesar -25.0086 persen (berkurang 33.3513 persen).
h. Aplikasi memiliki proses autentikasi menggunakan *checksum* dengan metode md5, sehingga *file* lampiran yang terenkripsi dapat terjaga keasliannya.
i. Proses pengiriman *chat* encrypt tergantung dengan koneksi dari server ke client dan *server performance* yang tersedia, sehingga proses pengiriman *chat encrypt* bisa cepat atau lama.

DAFTAR PUSTAKA

[1] Andika, Dwiky, 2016, Pnegertian dan Sejarah Kriptografi, Diakses 15 Januari 2018, <https://www.it-jurnal.com/pengertian-dan-sejarah-kriptografi/>.
[2] As'ad, Nabila, 2007, Studi Analisis Algoritma CAST dan Implementasinya dalam PGP, Bandung: Makalah Teknik Informatika.
[3] Computerhope, 2017, *Chat*, Diakses 15 Januari 2018, <https://www.computerhope.com/jargon/c/chat.htm>.
[4] Gunawan, Ade, 2006, Studi Mengenai Algoritma Simetri CAST-128 dan Aplikasinya, Bandung: Institut Teknologi Bandung, diakses pada 28 Oktober 2017, <http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2006-2007/Makalah1/Makalah1-025.pdf>.
[5] Sholeh, Ahmad T, Erwin Gunadhi, Asep Deddy S, 2013, Mengamankan Skrip Pada Bahasa Pemograman PHP Dengan Menggunakan Kriptografi Base64, Jurnal Algoritma Sekolah Tinggi Teknologi Garut, ISSN : 2302-7339, Vol. 10 No 1.