Aplikasi Enkripsi Pesan dan File pada *BlackBerry* dengan Menggunakan Mode *Cipher Feedback* 8-bit

Matthew Wangsadiredja – NIM : 13507012
Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
e-mail: matt_jcs@yahoo.com

Abstrak *- BlackBerry bukan sekedar smartphone biasa yang lengkap fitur multimedianya, BlackBerry juga memiliki layanan internet yang sangat baik. Namun layanan yang melalui BlackBerry Internet Service (BIS) ini belum tentu aman, karena dari sekian banyak pesan dan file yang terkirim melalui media BIS harus melalui server yang terletak di Kanada. Oleh karena itu dibutuhkan sebuah perangkat lunak yang dapat melakukan enkripsi dan dekripsi terhadap pesan maupun file, baik yang akan dikirim melalui media internet BlackBerry atau yang hanya disimpan pada perangkat BlackBerry tersebut.*

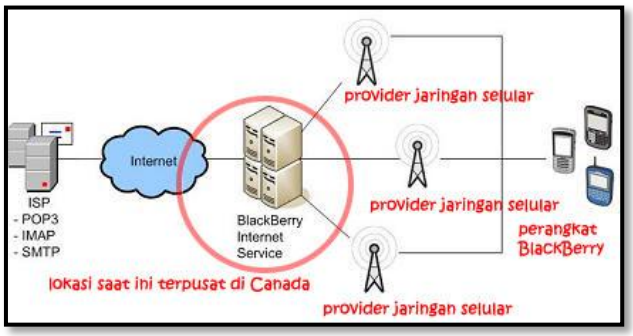
Pada makalah ini digunakan algoritma enkripsi cipher block dengan mode Cipher Feedback (CFB) 8-bit. Mode ini dipilih karena cocok untuk melakukan enkripsi terhadap data pada perangkat BlackBerry. Pada algoritma enkripsinya, digunakan algoritma cipher block dengan ukuran blok sebesar 64 bit. Algoritma ini menggunakan basis jaringan Feistel serta beberapa algoritma enkripsi dekripsi lainnya. Kemudian, algoritma enkripsi cipher block ini diimplementasikan ke sebuah perangkat lunak berbasis Java untuk BlackBerry. Dalam pembangunan perangkat lunak ini, digunakan bantuan IDE Eclipse ganymede serta beberapa plugins lainnya.

Kata kunci: *BlackBerry, enkripsi cipher block, mode Cipher Feedback (CFB) 8-bit.*

I. PENDAHULUAN

Daya tarik *BlackBerry* yang paling utama adalah layanannya yang ekslusif. *RIM* menggunakan *server* milik sendiri sehingga kecepatan akses internetnya menjadi sangat baik. Seluruh aktivitas dalam layanan *BlackBerry* di seluruh dunia langsung berpusat pada *server* milik *RIM* yang berlokasi di Kanada. Hal ini berbeda dengan layanan sejenis dari perangkat dan penyedia layanan lain yang masing-masing memiliki *server* di negara tempat layanan tersebut dioperasikan. Skema kerja jaringan *BlackBerry* dapat dilihat pada Gambar 1, yaitu terpusat di Kanada dan setiap penyedia layanan seluler harus memiliki jalur langsung dari masing-masing negaranya ke Kanada langsung.

Bagi sebagian negara tertentu, kondisi ini malah dianggap berpotensi membahayakan keamanan negara. Isu ini muncul karena seluruh data pengguna *BlackBerry* disimpan di *server* yang berada di luar negeri (Kanada), sehingga hukum lokal negara pengguna menjadi tidak berlaku dan akan sangat menyulitkan analis jika membutuhkan otoritas untuk memeriksa data yang diperlukan. Beberapa negara tersebut akhirnya meminta pihak *RIM* agar menyediakan *server* tersendiri di setiap negara masing-masing, dan seperti yang telah diketahui hingga saat ini pihak *RIM* masih tidak bersedia untuk memenuhi permintaan tersebut [CNN10].



Gambar 1 Diagram jaringan BIS [MAU10]

Seperti yang diketahui, sebagian besar pengguna *BlackBerry* memilih *BlackBerry* karena fitur *Instant Messaging* dan *e-mail* yang sangat baik, yang hingga saat ini belum dimiliki vendor-vendor *smartphone* lainnya. *BlackBerry* memiliki fitur *Instant Messaging* yang eksklusif, yaitu *BBM* (*BlackBerry Messenger)*. Fasilitas aplikasi *BBM* ini hanya dapat digunakan antar sesama pengguna *BlackBerry* saja, karena aplikasi ini menggunakan nomor pin unik yang terintegrasi pada setiap perangkat *BlackBerry* dan tidak dapat dimiliki perangkat *smartphone* lainnya. Berdasarkan jumlah pengguna *BlackBerry* saat ini, dapat dibayangkan seberapa banyaknya pesan teks, gambar, video, dan file dokumen yang dikirim melalui *BBM* ini. Hal inilah yang menjadi perhatian beberapa negara dan juga pemerintah Indonesia, yaitu sangat banyak data masyarakat negaranya, baik pesan teks, gambar, video, dan file

dokumen yang tersimpan di *server* Kanada ketika melakukan pengiriman melalui media *BBM*. Kemudian yang terpenting apakah data tersebut benar-benar dijaga privasinya oleh pihak *RIM*, atau disalahgunakan oleh beberapa pihak tertentu, semuanya itu tidak dapat diketahui oleh pihak negara pengguna yang jelas tidak memiliki akses langsung ke *server RIM* yang berada di Kanada.

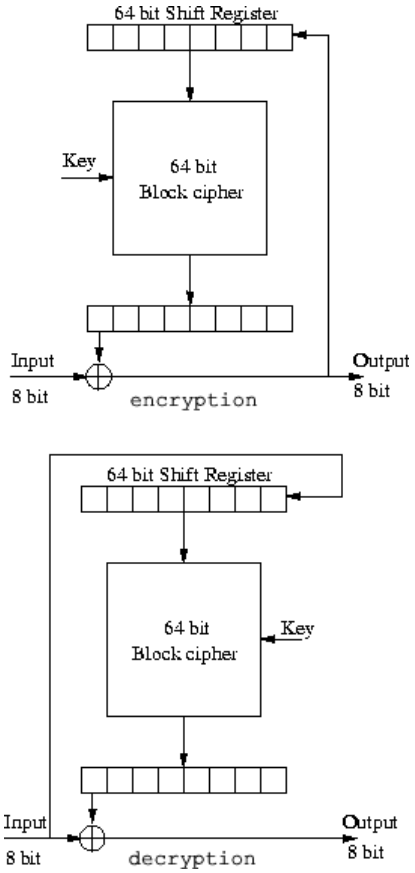
Saat ini sangat dibutuhkan suatu media untuk membantu pihak-pihak yang ingin datanya benar-benar aman disimpan atau saat melakukan pengiriman data. Terlebih juga dibutuhkan suatu media yang dapat meyakinkan pihak pemerintah negara bahwa data penting masyarakat yang tersimpan pada *server RIM* saat terjadi transaksi data antar perangkat *BlackBerry* dapat terjaga dengan aman, sehingga sekalipun *server RIM* tetap terpusat di Kanada (sehingga koneksi antar pengguna *BlackBerry* tetap baik), data yang penting tidak dapat disalahgunakan pihak manapun.

Berdasarkan isu tersebut, melalui makalah ini diangkatlah sebuah tema keamanan pada perangkat *BlackBerry*, kemudian dibuatlah solusi praktisnya melalui sebuah aplikasi yang mampu mengamankan data pengguna *BlackBerry*. Aplikasi yang dibuat akan menerapkan ilmu kriptografi, yaitu dengan menggunakan enkripsi *cipher block* yang merupakan suatu algoritma enkripsi yang membagi bit plainteks dengan panjang yang sama, kemudian melakukan enkripsi terhadap blok demi blok data. Hal ini cukup menjadi masalah, dikarenakan jika ukuran file tidak pas akan terdapat blok yang belum lengkap. Oleh karena itu digunakan mode *Cipher Feedback* (CFB) yang dapat mengatasi permasalahan ini. Dengan menggunakan mode CFB, blok dienkripsi seperti halnya *stream cipher* pada satuan sesuai dengan mode CFB yang digunakan, sehingga ukuran blok plainteks tidak perlu diperbesar .

II. CARA KERJA ALGORITMA CFB 8-BIT

Pada mode *Cipher Feedback*, data dienkripsi ke dalam unit yang lebih kecil dari ukuran blok data *stream*. Dengan mode ini dapat digunakan untuk mengenkripsi sejumlah bit tertentu, c ontoh sebuah bit atau sebuah karakter (byte) [FIP80]. Pada Gambar 2 diberikan contoh algoritma CFB yang menggunakan ukuran blok 64 bit, dan ukuran terkecilnya adalah 8 bit (1 byte).

Pada skema yang ditunjukkan Gambar 2, sebuah byte dapat dienkripsi dan didekripsi menggunakan *cipher block* dengan mode CFB. Pada awalnya *shift register* diinisialisasi dengan *initialization vector* (*IV*) kemudian algoritma enkripsi dilakukan untuk menghasilkan output 64 bit. *IV* sendiri merupakan sebuah vektor binari yang digunakan untuk meninisialisasikan blok input pada mode CFB dan OFB sebagai blok yang nilainya acak[FIP80].



Gambar 2 Enkripsi dan dekripsi CFB 8-bit dengan ukuran blok 64 bit [MUN06]

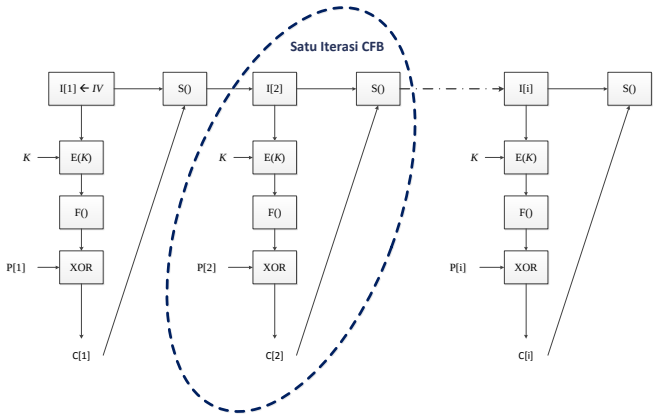
Setelah itu, 8 bit terkiri (*left-most*) dari output diambil untuk di-*XOR* kan dengan *stream* data plainteks. Hasil *XOR* ini yang akan disimpan untuk menjadi cipherteks, kemudian 8 bit hasil *XOR* ini juga dimasukkan ke belakang input blok yang baru untuk, sementara 8 bit terkirinya akan dibuang. Kemudian, algoritma enkripsi ini dilakukan lagi dengan cara yang sama hingga semua data selesai dienkripsi.

Initialization vector yang digunakan pada mode CFB memiliki sifat yang sama dengan *initialization vector* yang digunakan pada CFB. Yaitu *IV* tidak harus bersifat rahasia, namun sebaiknya berbeda untuk setiap pesan yang dienkripsi dengan kunci yang sama.

Kesalahan bit (*bit errors*) pada *cipher block* akan menyebabkan kesalahan bit pada posisi yang sama pada blok plainteks yang didekripsi. Karena itu jika saat menggunakan mode CFB 8-bit terdapat kesalahan pada beberapa bit di blok cipherteks, maka blok plainteks sebesar 64 bit yang berkorespondensi dengan blok cipherteks akan rusak informasinya, sementara blok lainnya tetap akan dapat didekripsi dengan benar.

Untuk memperjelas cara kerja CFB 8-bit dengan

ukuran blok 64 bit, akan dijelaskan melalui skema pada Gambar 3.



Gambar 3 Skema per tahap mode CFB 8-bit dengan ukuran blok 64-bit

Berikut adalah keterangan notasi yang digunakan:

IV: *Initialization Vector*, bersifat acak dan berbeda untuk setiap pesan yang dienkripsi. Panjang *IV* yang digunakan adalah 64 bit.

I[i]: *Input block*, digunakan sebagai blok input yang akan dienkripsi kemudiannya. Panjang I[i] adalah 64 bit. Khusus untuk blok input pertama (I[1]), nilainya adalah *IV*, sedangkan untuk I[2] dan selanjutnya diperoleh dari fungsi S().

E(*K*): Fungsi enkripsi, yaitu fungsi yang mengenkripsi blok 64 bit dengan menggunakan kunci *K* yang dimasukkan pengguna.

F(): Fungsi *filter*, yaitu sebuah fungsi untuk mengambil *left-most* 8 bit dari blok hasil enkripsi yang berukuran 64 bit, yang kemudian akan dilakukan *XOR* dengan blok plainteks.

P[i]: Blok plainteks yang akan dilakukan operasi *XOR* dengan hasil fungsi filter, berukuran 8 bit setiap bloknya.

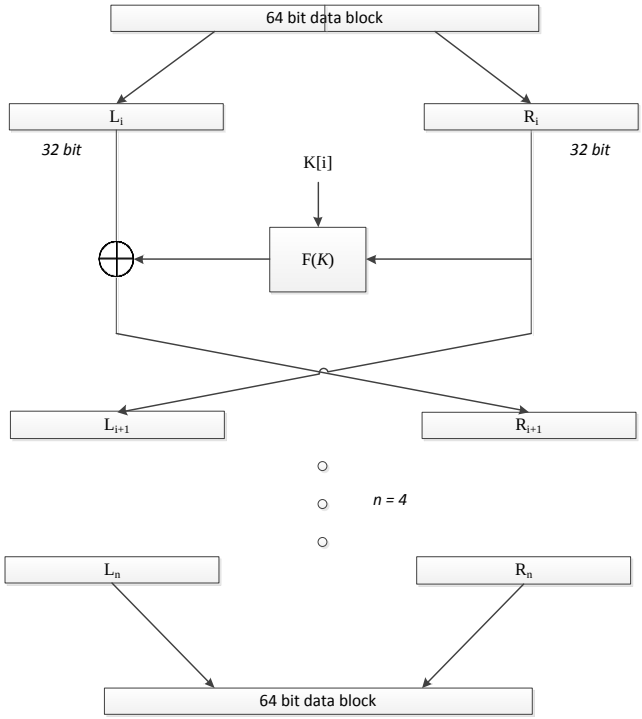
C[i]: Blok cipherteks, yang akan disimpan kemudian digabungkan dengan blok cipherteks lainnya hingga menjadi data cipherteks yang utuh. Setiap bloknya berukuran 8 bit data.

S(): Fungsi *shifting*, yaitu fungsi geser dengan membuang 8 bit terawal blok input, kemudian memasukan blok cipher 8 bit pada bagian akhirnya. Hasil fungsi shifting ini kemudian dijadikan blok input untuk iterasi selanjutnya.

i: Iterasi pada mode CFB. Jumlah iterasi yang dilakukan pada mode CFB 8 bit adalah sebesar (ukuran data dalam bits / 8 bit).

III. ALGORITMA ENKRIPSI E(*K*) DENGAN KUNCI 128-BIT

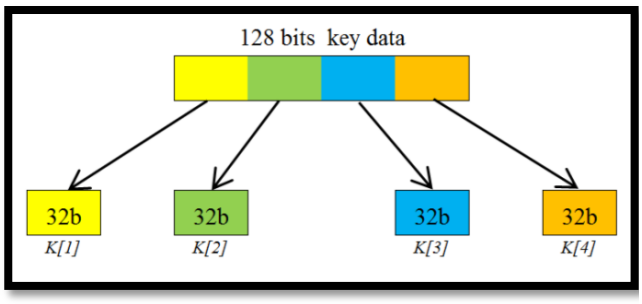
Pada umumnya, mode CFB menggunakan DES sebagai algoritma fungsi enkripsinya[FIP80]. Algoritma DES sendiri menggunakan kunci 128 bit dalam proses enkripsinya. Namun untuk pada makalah ini akan digunakan algoritma enkripsi yang dirancang sendiri. Algoritma enkripsi yang akan digunakan merupakan modifikasi terhadap metode jaringan *Feistel* yang terdiri dari beberapa proses manipulasi data binari dan diiterasi sebanyak empat kali. Sedangkan kunci yang digunakan sepanjang 128 bit, kemudian akan dipecah menjadi empat bagian untuk digunakan pada setiap proses iterasi yang berbeda pada metode jaringan *Feistel*. Cara kerja algoritma enkripsi yang digunakan ditunjukkan pada Gambar 4 di bawah ini.



Gambar 4 Proses algoritma jaringan Feistel terhadap blok data 64 bit

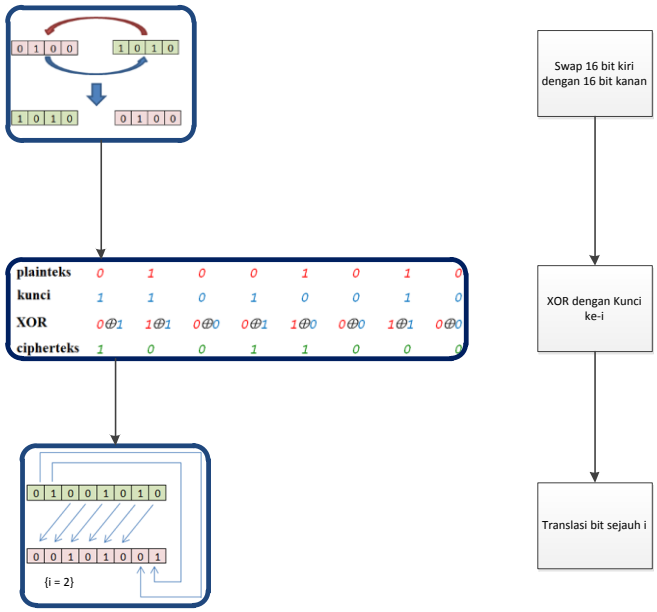
Mula-mula blok data sebesar 64 bit dibagi menjadi 2 bagian, blok kiri (L) dan blok kanan (R), yang

masing-masingnya berukuran 32 bit. Kemudian blok R disimpan untuk blok L selanjutnya, sementara blok R selanjutnya diperoleh dari hasil blok R yang diproses dengan fungsi F dan kunci *K*, kemudian dilakukan operasi *XOR* dengan blok L. Proses ini dilakukan sebanyak 4 kali iterasi, yaitu dengan menggunakan kunci yang berbeda pada fungsi F setiap iterasi. Pembagian kunci 128 bit dilakukan seperti pada Gambar 5, yaitu dengan membaginya menjadi empat bagian.



Gambar 5 Pemecahan kunci 128 bit menjadi empat blok 32 bit

Setelah kunci dibagi menjadi 4 bagian, kemudian kunci ini akan digunakan pada empat buah iterasi fungsi F di dalam jaringan *Feistel*. Fungsi F dilakukan terhadap blok R yang berukuran 32 bit. Fungsi ini akan terdiri dari tiga tahapan seperti yang digambarkan pada Gambar 6, yaitu:

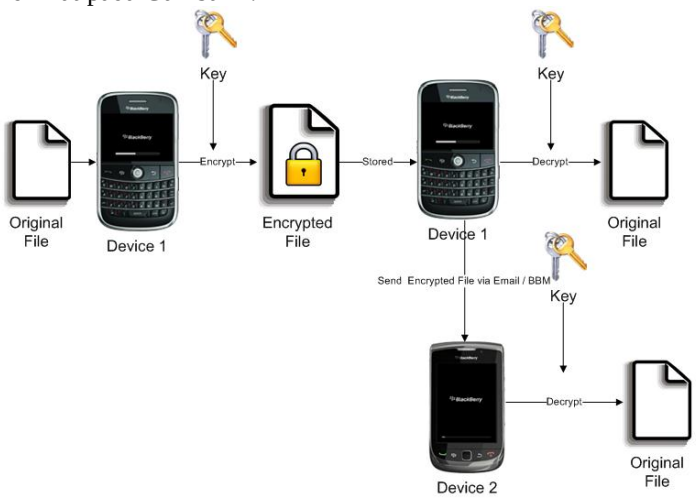


Gambar 6 Tiga tahapan di dalam fungsi F

1. Fungsi *swap*, yaitu menukarkan 16 bit terkiri (*left-most*) dengan 16 bit terkanan (*right-most*) data tersebut.
2. *XOR* dengan kunci ke-*i*, dimana nilai *i* bergantung pada urutan iterasi. Pada iterasi pertama akan dilakukan *XOR* dengan kunci blok 1, begitu pula selanjutnya hingga iterasi ke empat.
3. Translasi sejauh *i*, yaitu menggeser bit sejauh *i*-bit sehingga jarak pergeseran bit akan berbeda setiap iterasinya.

IV. Implementasi Algoritma Enkripsi dengan Mode CFB 8-bit pada Aplikasi BlackBerry

Perangkat lunak yang dibangun dapat melakukan enkripsi dan dekripsi terhadap pesan maupun file dengan menggunakan kunci yang dirahasiakan oleh penggunanya. Perangkat lunak ini berbentuk sebuah aplikasi yang dapat berjalan pada perangkat *BlackBerry* yang memiliki sistem operasi 4.6.0 ke atas. Aplikasi ini melakukan enkripsi dengan mode CFB 8-bit, sedangkan *block encryptor* yang terdapat di dalam mode CFB ini menggunakan algoritma yang dijelaskan pada Bab III. Skema umum dari perangkat lunak yang dibangun dapat dilihat pada Gambar 7.



Gambar 7 Skema umum perangkat lunak

Kebutuhan Fungsional dan Non-Fungsional

Kebutuhan fungsional dari perangkat lunak yang dibangun adalah:

1. Melakukan enkripsi dan dekripsi dengan menggunakan kunci yang dimasukkan pengguna terhadap pesan maupun file yang berada di perangkat *BlackBerry*.
2. Mampu berjalan di semua perangkat *BlackBerry* yang memiliki sistem operasi 4.6.0 ke atas.

3. Menjaga *header* file pada beberapa tipe file yang diperoleh melalui perangkat *BlackBerry*, seperti audio, video, dan gambar.
4. Menyediakan fitur yang dapat mempermudah untuk melakukan enkripsi terhadap pesan teks biasa, sehingga mudah dienkripsi dan mudah disalin.

Sedangkan kebutuhan non-fungsional dari perangkat lunak yang akan dibangun adalah:

1. Menyediakan tingkat keamanan enkripsi yang tinggi sehingga pesan maupun file yang dienkripsi dapat terjamin keamanannya.
2. Melakukan proses enkripsi dan dekripsi dengan cepat.
3. Menyediakan perangkat lunak dengan *interface* yang mudah dan nyaman untuk digunakan pada perangkat *BlackBerry*.
4. Menyediakan tautan untuk mengunduh perangkat lunak secara *online*, sehingga mudah diakses oleh semua pihak.

Implementasi Kelas

Perangkat lunak dibangun dengan menggunakan bahasa *Java* untuk *BlackBerry* dengan menggunakan *IDE Eclipse ganymede*. Kelas-kelas yang digunakan pada pengimplementasian dapat dilihat pada Tabel 1 di bawah ini.

Tabel 1 Pengimplementasian kelas-kelas

No	Package	Kelas
1	crypto	BitSetHelper
2	crypto	CFB
3	crypto	Feistel
4	filepicker	FilePicker
5	filepicker	FilePickerDirListFiel
6	filepicker	FilePickerFileHolder
7	filepicker	FilePickerFileListFie
8	filepicker	FilePickerListener
9	helper	Email
10	helper	HexByte
11	pv	HomeScreen
12	pv	Pv
13	component	Colors
14	component	CustomButton
15	component	CustomEditField
16	component	CustomePasswordField
17	component	FileIO
18	component	Fonts
19	component	ImageField
20	component	TitleField

Implementasi Antarmuka

Implementasi antarmuka dari perangkat lunak ini dibuat dalam sebuah halaman utama. Halaman utama ini memiliki 9 buah *field* yang ditunjukkan pada Gambar 8

Gambar 8 Tampilan halaman utama aplikasi

1. *Header* berupa judul aplikasi "*my private file and message*".
2. Tombol "*Choose file"*, akan menampilkan modul *file picker* jika ditekan.
3. Kolom teks yang menunjukkan path file yang dipilih melalui *file picker.*
4. Kolom untuk menulis pesan atau cipherteks yang hendak dienkripsi atau didekripsi.
5. Kolom untuk menuliskan kunci yang akan digunakan untuk proses enkripsi atau dekripsi.
6. Tombol "*Encrypt!*", dengan menekan tombol ini aplikasi akan melakukan proses enkripsi dengan menggunakan kunci yang telah dimasukkan terhadap file yang dipilih atau pesan yang ditulis.
7. Tombol "*Decrypt!*", dengan menekan tombol ini aplikasi akan melakukan proses dekripsi dengan menggunakan kunci yang telah dimasukkan terhadap file yang dipilih atau cipherteks yang ditulis.
8. Kolom gambar yang akan menampilkan gambar berupa jenis file yang dipilih melalui *file picker*.
9. Kolom teks yang akan menampilkan hasil enkripsi atau dekripsi terhadap pesan teks.

V. PENGUJIAN

Tujuan Pengujian

Tujuan dari tahap pengujian ini adalah untuk memenuhi kebutuhan fungsional dan non-fungsional perangkat lunak seperti yang telah disebutkan pada bab sebelumnya mengenai fitur utama perangkat lunak. Tujuan tersebut adalah:

1. Menguji kompatibilitas perangkat lunak pada beberapa perangkat *BlackBerry* yang berbeda.
2. Menguji penanganan perangkat lunak terhadap enkripsi file yang telah dikenali.

3. Menguji penanganan perangkat lunak terhadap enkripsi pesan sehingga mudah digunakan.
4. Menguji kebenaran proses enkripsi dan dekripsi yang dilakukan oleh perangkat lunak.
5. Menguji tingkat keamanan aplikasi berdasar tiga aspek / *goals* suatu aplikasi berbasis *security*, yaitu *confidentiality*, *integrity*, dan *availability*.
6. Menunjukkan kemudahan dalam melakukan pengunduhan aplikasi.

Batasan Pengujian

Perangkat *BlackBerry* yang digunakan dalam pengujian memiliki sistem operasi 4.6.0 ke atas, baik berupa simulator maupun perangkat yang sebenarnya. Pengujian penanganan beberapa jenis file merupakan file yang dihasilkan perangkat *BlackBerry*, seperti gambar, audio, dan video. Sedangkan file gambar, audio, dan video yang diperoleh dari media lain tidak diujicobakan penanganan terhadap *header*-nya.

Data Uji

File yang digunakan pada pengujian dapat dilihat pada Tabel 2.

Tabel 2 File yang digunakan dalam tahap pengujian

No	Nama File	Ukuran	Keterangan
1	audio.amr	5 KB	File audio yang diperoleh dari rekaman *microphone* perangkat *BlackBerry*
2	document.doc	22 KB	Dokumen *Microsoft Word* yang berasal dari sumber lain
3	image.jpg	35 KB	File gambar yang diperoleh dari kamera perangkat *BlackBerry*
4	text.txt	1 KB	File polos, berisi teks yang dibuat dengan teks editor standar
5	video.3gp	35 KB	File video yang diperoleh dari rekaman video perangkat *BlackBerry*

Hasil dan Analisis Hasil Uji

Berikut ini merupakan masing-masing hasil pengujian berdasarkan enam buah kasus uji.

1. Kasus Uji 1

Pada pengujian ini, pertama-tama dilakukan penginstalasian aplikasi ke empat perangkat *BlackBerry* Pada keempat perangkat tersebut aplikasi diuji untuk di-*running*, dan hasilnya aplikasi dapat berjalan pada semua perangkat tersebut. Setelah itu, dilakukan peninjauan secara objektif akan tampilan antarmuka di keempat perangkat. Spesifikasi ukuran layar dan sistem operasi masing-masing perangkat ditunjukkan pada Tabel 4.

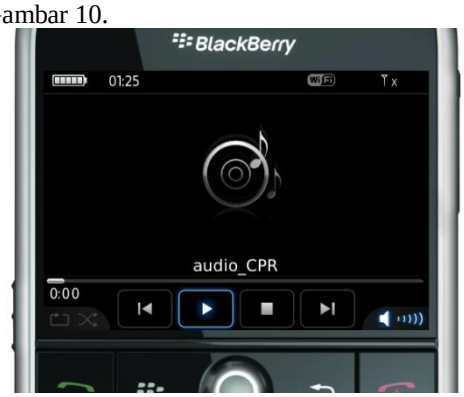
Tabel 3 Perangkat yang digunakan dalam pengujian kompatibilitas

No	Model	Jenis	Resolusi	Sistem Operasi
1	*BlackBerry 9000 Bold*	Simulator	480 x 320 *pixel*	v4.6.0.92
2	*BlackBerry 9800 Torch*	Simulator	360 x 480 *pixel*	v6.0.0.141
3	*BlackBerry 8520 Gemini*	Perangkat	320 x 240 *pixel*	v4.6.1.314
4	*BlackBerry 9700 Onyx*	Perangkat	480 x 360 *pixel*	v5.0.0.405

Melalui pengujian ini, disimpulkan kompatibilitas perangkat lunak sudah baik, karena perangkat lunak dapat dijalankan pada beberapa perangkat *BlackBerry* yang berbeda sistem operasinya, yaitu sistem operasi 4.6, 5.0, dan 6.0. Selain itu tampilan antarmuka perangkat lunak juga baik, karena secara objektif tampilan sudah nyaman dalam beberapa jenis perangkat *BlackBerry* yang berbeda resolusi layar.

2. Kasus Uji 2

Untuk menguji apakah *header* file berhasil dijaga atau tidak, maka perlu dilakukan enkripsi terlebih dahulu kepada tiga jenis file tersebut. Pertama-tama, disiapkan file gambar, audio, dan video yaitu file "*audio.amr*", "*image.jpg*", dan "*video.3gp*". Setelah itu ketiga file dienkripsi dengan kunci "*1234*", sehingga menghasilkan file "*audio_CPR.amr*", "*image_CPR.jpg", "video_CPR.3gp"*. Kemudian ketiga jenis file dibuka melalui *media explorer* milik *BlackBerry*, yang hasilnya ditunjukkan pada gambar-gambar di bawah ini.Tampilan cipherteks audio jika dibuka oleh *explorer BlackBerry* akan dikenali sebagai file audio biasa, seperti ditunjukkan pada Gambar 9. Namun, durasi dari audio tidak dapat terbaca. Selain itu jika tombol *play* ditekan, tidak ada audio yang terputar. Untuk cipherteks gambar, file dapat dibuka seperti gambar lainnya, namun isi dari gambar saja yang rusak. Tampilan file cipherteks gambar ditunjukkan pada Gambar 10.



Gambar 9 Tampilan cipherteks audio dibuka oleh explorer BlackBerry

Gambar 10 Tampilan cipherteks gambar dibuka oleh explorer BlackBerry

Gambar 11 Tampilan cipherteks video dibuka oleh explorer BlackBerry

File cipherteks yang berasal dari file video tidak dapat dikenali sebagai file video oleh *explorer* perangkat *BlackBerry*. Hal ini ditunjukkan pada Gambar 11, yaitu muncul notifikasi "*The media being played is of an unsupported format"*, yaitu file berupa format yang tidak dikenali.
Melalui pengujian ini, disimpulkan penanganan perangkat lunak terhadap file gambar sudah baik, yaitu gambar tetap dikenali dan dapat dibuka sebagai file gambar. Sedangkan penanganan terhadap file audio cukup baik, karena meski file audio dapat dikenali sebagai file audio, namun file audio tidak dapat didengar sama sekali dan tidak ada keterangan durasi audio tersebut. Sedangkan penanganan terhadap file video masih kurang baik, hal ini ditunjukkan dengan tidak dikenalinya file video yang terenkripsi, meskipun header file video tersebut sudah dijaga.

3. Kasus Uji 3

Untuk menguji apakah hasil cipherteks dari pengenkripsian teks / pesan sudah baik penanganannya, maka akan dilakukan proses pengenkripsian terhadap teks tertentu.
Melalui pengujian ini, disimpulkan penanganan perangkat lunak terhadap enkripsi pesan sudah baik. Hal ini ditunjukkan dengan hasil enkripsi pesan yang berupa bilangan heksa yang mudah di-*copy* atau bahkan diketik ulang. Selain itu juga telah disediakannya fitur mengirim melalui media *e-mail* dan fitur menyalin ke *clipboard* yang memudahkan pengguna untuk mengirim cipherteks pesan.

4. Kasus Uji 4

Untuk menunjukkan proses enkripsi dan dekripsi telah berjalan dengan baik dan benar, maka akan dilakukan pengujian proses enkripsi dan dekripsi dengan menggunakan perangkat yang sebenarnya, yaitu perangkat pertama *BlackBerry 8520* dan perangkat kedua *BlackBerry 9700*.
Melalui pengujian ini, disimpulkan proses enkripsi dan dekripsi oleh perangkat lunak sudah benar. Hal ini ditunjukkan dengan dapat dilakukannya enkripsi dan dekripsi oleh perangkat yang berbeda, dan pesan plainteks yang asli dapat hanya dapat diperoleh jika menggunakan kunci yang sama ketika melakukan dekripsi.

5. Kasus Uji 5

Confidentiality: pengujian dilakukan dengan menggunakan *debug mode* pada *IDE Eclipse Ganymede*, kemudian memeriksa beberapa variabel penting, apakah nilainya dapat disadap atau tidak melalui *debug mode* ini. *Debug mode* ini disetarakan dengan aplikasi lain yang mencoba mencuri data dari memori perangkat ketika proses enkripsi atau dekripsi dilakukan.
Melalui pengujian perangkat lunak ini dapat lolos dari pengujian *confidentiality* dengan baik, yaitu data rahasia (kunci dan plainteks) tidak mudah jatuh ke tangan pihak lain atau dicuri aplikasi lain.

Integrity: pengujian dilakukan dengan melakukan manipulasi pada berkas plainteks, cipherteks, maupun kunci yang digunakan dalam proses enkripsi dan dekripsi. Pada pengujian ini dilakukan melalui enkripsi terhadap *string*:

```
Ini adalah pesan rahasia.
```

dengan menggunakan kunci:

```
12345678
```

Setelah itu, dilakukan proses enkripsi secara berulang sebanyak tiga kali, kemudian hasilnya adalah sebagai berikut:

Enkripsi-1
401EFA37C93BC0DA6C048E7B55D2286F116BDBF421E71D7E35E073E9322B9A6F54
Enkripsi-2
2472779C27B554ED2799143AA0FA192877AE5F76F4532196EC8BC5B03CF1A3055C
Enkripsi-3
69EBE95F311E2282C940BD177C58168E916209BAFF14B5733399D2A8B94467430B

Ketiga cipherteks berbeda dikarenakan setiap kali proses enkripsi, digunakan *Initialization Vector* (*IV*) yang berbeda. Sehingga, meski dengan kunci yang sama, hasil enkripsi terhadap pesan yang sama akan selalu berubah. Setelah itu, dilakukan beberapa ujicoba pengubahan pada

kunci dan cipherteks dan bagaimana dampaknya pada plainteks yang didekripsi. Cipherteks yang digunakan adalah hasil enkripsi pertama pada pengujian sebelumnya yaitu "*401EFA37C93BC0DA6C048E7B55D2286F116BDBF421E71D7E35E073E9322B9A6F54*" dan menggunakan kunci yang sama yaitu "*12345678*". Hasil pengujian ini dapat dilihat pada Tabel 5.

Tabel 4 Tabel hasil pengujian integrity *aplikasi*

Cipherteks	Kunci	Hasil Dekripsi
401EFA37C93BC0DA6C048E7B55D2286F116BDBF421E71D7E35E073E9322B9A6F54	**12345678**	**Ini adalah pesan rahasia.**
?01EFA37C93BC0DA6C048E7B55D2286F116BDBF421E71D7E35E073E9322B9A6F54	**12345678**	?i adaah pesan rahasia.
401EFA37C93BC0DA6C048E7B55D2286F116BDBF421E71D7E35E073E9322B9A6F5?	**12345678**	?Ni ad`ah pesan rahasia.
401EFA37C93?C0DA6C048E7B55D2286F116BDBF421E71D7E35E073E9322B9A6F54	**12345678**	Ini af?c,ah p?³an rahasia.
?01EFA37C93BC0DA6C048E7B55D2286F116BDBF421E71D7E35E073E9322B9A6F54	**12345678**	°Bs 9½½ì Y¼ =¬-©
?401EFA37C93BC0DA6C048E7B55D2286F116BDBF421E71D7E35E073E9322B9A6F54	**12345678**	¿e])½ ýõ} 9½½ì Y¼ =¬-©
401EFA37C93BC0DA6C048E7B55D2286F116BDBF421E71D7E35E073E9322B9A6F5?	**12345678**	_Ò³Ìah pesan rahasia
401EFA37C93BC0DA6C048E7B55D2286F116BDBF421E71D7E35E073E9322B9A6F54	?23456 78	Z}z3rwrr{3cv` r}3ar{r`zr=
401EFA37C93BC0DA6C048E7B55D2286F116BDBF421E71D7E35E073E9322B9A6F54	223456 78	ôÕÒÚßÚ×ÚÓËÞÈÚ ÕÉÚÓÚÈÒÚ
401EFA37C93BC0DA6C048E7B55D2286F116BDBF421E71D7E35E073E9322B9A6F54	?23456 78	·°ù¸½¸µ¸±ù©¼ª ¸·ù«¸±¸ª°¸÷
401EFA37C93BC0DA6C048E7B55D2286F116BDBF421E71D7E35E073E9322B9A6F54	?12345 678	4]]]S
401EFA37C93BC0DA6C048E7B55D2286F116BDBF421E71D7E35E073E9322B9A6F54	123456 789	Ââå¬íèíàíä¬üé ÿíâ¬þíäíÿåí¢
401EFA37C93BC0DA6C048E7B55D2286F116BDBF421E71D7E35E073E9322B9A6F54	?12345 6789	ªÃÃÃÍ

Melalui pengujian *integrity*, disimpulkan perangkat lunak juga sudah menangani kasus integrity dengan baik, yaitu memberikan perubahan yang besar pada hasil dekripsi jika terdapat ada sedikit saja kunci atau blok cipherteks yang diubah. Selain itu juga digunakan *IV* yang menyebabkan proses enkripsi selalu berubah-rubah meski menggunakan plainteks dan kunci yang sama.

Availability: pengujian dilakukan dengan menggunakan melakukan mencoba kemampuan aplikasi dalam menangani beberapa kasus kesalahan yang tidak diharapkan. Yang pertama adalah kesalahan input pada cipherteks pesan yang hendak didekripsi, yaitu bukan merupakan bilangan heksa. Hal ini ditangani dengan menampilkan sebuah *alert box* bertulisan "*Wrong ciphertext format!"* seperti yang ditunjukkan pada Gambar 12.

Gambar 12 Pesan saat memasukan cipherteks selain bilangan heksa

Kondisi yang kedua adalah apabila belum memilih file atau memasukkan pesan yang hendak dienkripsi atau didekripsi. Pada kondisi ini, maka akan ditampilan juga sebuah *alert box* bertulisan "*Please select file or write messages to encrypt!"*.
Melalui penguijian tersebut, disimpulkan aplikasi sudah cukup menangani beberapa kasus kesalahan input. Namun aplikasi masih belum cukup baik dalam menangani kesalahan yang diakibatkan ukuran file terlalu besar, sehingga membutuhkan bantuan sistem operasi untuk memberhentikan aplikasi.

6. Kasus Uji 6

Pada pengujian kali ini, dilakukan pengunggahan aplikasi ke situs web *online.* Seperti yang telah dijelaskan pada subbab II.1.2.3, penginstallasian aplikasi ke perangkat *BlackBerry* dapat dilakukan secara *OTA* (*Over the Air*). Seletah aplikasi diunggah, yaitu file *executable* (file dengan ekstensi *.cod*) dan file keterangan (file dengan ekstensi *.jad*) diunggah, maka cukup memanggil file berekstensi .jad dari *browser* perangkat *BlackBerry*. Setelah masuk ke alamat file *.jad* berada, maka akan

ditampilkan menu *download*, kemudian cukup menekan tombol *download* untuk mengunduh aplikasi seperti pada Gambar 14.

Gambar 13 Proses pengunduhan aplikasi secara OTA

Melalui pengujian ini, disimpulkan aplikasi *BlackBerry* ini mudah diunduh dan dilakukan pengsintalasian. Hal ini ditunjukkan melalui pengujian yang hanya membutuhkan beberapa langkah sederhana untuk mengunduh aplikasi sekaligus melakukan instalasi aplikasi.

VI. KESIMPULAN

Kesimpulan yang dapat diambil dari pengerjaan makalah ini adalah:

1. Secara umum, algoritma blokcipher pada mode CFB 8-bit dapat diimplementasikan dan berhasil berjalan dengan baik pada perangkat *BlackBerry*. Melalui pengujian ditunjukkan bahwa aplikasi dapat berjalan dengan baik pada berbagai jenis perangkat *BlackBerry*.
2. Proses enkripsi dan dekripsi yang dihasilkan aplikasi sudah baik, yaitu hanya dapat melakukan dekripsi jika dan hanya jika kunci yang digunakan sama dengan kunci ketika melakukan proses enkripsinya.
3. Penanganan terhadap pesan, file audio, serta file gambar berhasil ditangani dengan menjaga *header* file serta membuat cipherteks pesan menjadi bilangan heksa. Namun untuk jenis file video, penanganan untuk menjaga *header* masih belum sempurna karena cipherteks file video tidak dikenali sebagai file video biasanya oleh perangkat *BlackBerry*.
4. Tingkat keamanan aplikasi sudah cukup baik karena memenuhi tiga aspek aplikasi berbasis *security* secara umum, yaitu *confidentiality*, *integrity*, dan *availability*.

VII. SARAN

Untuk pengembangan di masa mendatang, saran-saran yang diberikan pada makalah ini adalah sebagai berikut:

1. Penanganan pada file video diperbaiki, selain itu juga dibuat penanganan khusus untuk jenis-jenis file lainnya, seperti file dokumen, file audio musik, file gambar dengan format lain, dan jenis-jenis file umum lainnya.
2. Memperbaiki design algoritma enkripsi agar menjadi lebih baik kemudian mendokumentasikannya. Setelah itu mem-*publish* dokumentasi dari algoritma tersebut sehingga kekuatan algoritma tersebut dapat dianalisis oleh pihak-pihak lain yang ahli dibidang kriptografi ini.
3. Melakukan peningkatan performansi pada algoritma sehingga dapat melakukan enkripsi pada file yang berukuran relatif lebih besar dari yang saat ini mampu dienkripsi.
4. Menambahkan fitur penintegrasian aplikasi ke sistem operasi *BlackBerry*, sehingga fitur enkripsi terhadap pesan mudah diakses, baik melalui media *SMS*, *IM*, maupun *BBM*.

VIII. REFERENSI

[CNN10] Saudis to block *BlackBerry* service. URL: http://edition.cnn.com/2010/WORLD/meast/08/03/saudi.arabia.*BlackBerry*/index.html#fbid=aIVo2cKvO5r&wom=false . Waktu Akses: 5 November 2010

[FIP80] Federal Information Processing Standards Publication 81. 1980 December 2, *Announcing the* ***Standard for DES MODES OF OPERATION*****. URL:** http://www.itl.nist.gov/fipspubs/fip81.htm. Waktu Akses: 5 Januari 2011.

[MAU10] *BlackBerry* Internet Service 3.0 Up And Running For North America. URL: http://*BlackBerry*rocks.com/2010/03/29/*BlackBerry*-internet-service-3-0-running-north-america-features-enhancements-news/. Waktu akses: 20 Oktober 2010.

[MUN06] Munir, Rinaldi. *Slide Kuliah IF5054 – Algoritma Kriptografi Modern*. Departemen Teknik Informatika, Institut Teknologi Bandung. 2006.