**Vol: 5, No. 1, Maret 2016** **ISSN: 2302 - 2949**

# ENKRIPSI DAN DEKRIPSI FILE DENGAN ALGORITMA BLOWFISH PADA PERANGKAT MOBILE BERBASIS ANDROID

**Siswo Wardoyo*, Zaldi Imanullah, Rian Fahrizal**
Jurusan Teknik Elektro, FakultasTeknik Universitas Sultan Ageng Tirtayasa
Jl. Jendral Sudirman Km.3 Cilegon – Banten
*Corresponding author, e-mail : siswo@untirta.ac.id

**Abstrak** - Kriptografi merupakan salah satu cara yang digunakan untuk mengamankan data dalam bentuk file dengan mengenkripsi file sehingga orang lain tidak berhak mengetahui file yang sifatnya pribadi dan rahasia. Salah satu metode kriptografi adalah algoritma *Blowfish* yang merupakan algoritma yang menggunakan kunci simetris untuk melakukan enkripsi dan dekripsi. Aplikasi yang dibangun ini dapat melakukan enkripsi file berbentuk gambar, video, dan dokumen. Aplikasi ini dapat berjalan pada ponsel yang minimal memiliki sistem operasi Android versi 2.3. Perangkat lunak yang digunakan untuk membangun aplikasi ini adalah Eclipse. Hasil dari penelitian ini menunjukkan bahwa aplikasi yang dibangun mampu melakukan enkripsi dan dekripsi dengan baik. Hasil enkripsi file membuat file menjadi tidak diketahui lagi maknanya. Dengan menggunakan kunci berjumlah 72 *bit* atau 9 karakter dibutuhkan waktu $1{,}49 \times 10^8$ tahun untuk membongkarnya dengan kecepatan komputasinya adalah $10^6$ *key/sec*.

***Kata kunci:*** *kriptografi, algoritma Blowfish, kunci simteris, Android, enkripsi, dekripsi.*

**Abstract**—Cryptography is one of the ways used to secure data in the form of a file with encrypt files so that others are not entitled to know the file is private and confidential. One method is the algorithm *Blowfish* Cryptography which is a symmetric key using the algorithm to perform encryption and decryption. Applications that are built can perform file encryption-shaped images, videos, and documents. These applications can be running on a *mobile* phone that has a minimal operating system Android version 2.3. The software used to build these applications is Eclipse. The results of this research indicate that applications built capable of performing encryption and decryption. The results file encryption makes files into another unknown meaning. By using the keys numbered 72 bits or 9 character takes 1,49x108 years to break it with the speed it's computation is 106 key/sec.

***Keywords:*** *cryptography, algorithm Blowfish, symmetric key, Android, encryption, decryption*



## 1. PENDAHULUAN

Pesatnya perkembangan media telekomunikasi dan informasi di berbagai bidang selaras dengan tingginya tingkat aktivitas dan kebutuhan manusia akan informasi dalam kehidupan sehari-hari[1]. Kebutuhan akan layanan telekomunikasi dengan kecepatan transmisi yang tinggi serta fleksibilitas dalam hal penggunaan perangkat layanan mendorong para penyedia jasa layanan dan perangkat telekomunikasi untuk senantiasa mengembangkan kemampuan dari teknologi yang dimilikinya, khususnya layanan berbasis nirkabel[2].

Pada saat ini, penggunaan perangkat *mobile* sudah menjadi *trend* di masyarakat dunia. Perangkat *mobile* yang beredar saat ini sangat menakjubkan, sehingga mempunyai dampak dalam meningkatkan efektifitas dan efisiensi dalam melakukan setiap pekerjaan. Sistem operasi untuk perangkat *mobile* semakin berkembang. Android merupakan salah satu sistem operasi *mobile* buatan *Google* yang kini sangat populer dan banyak digunakan orang-orang. Pengguna Android mempunyai kemudahan untuk mendapatkan aplikasi-aplikasi yang diinginkan[3].

Android juga merupakan sistem operasi yang berbasis perangkat lunak yang dapat dikembangkan secara terbuka (open source) sehingga banyak pengembang yang kini turut ikut mengembangkan aplikasi untuk Android. Perangkat *mobile* yang dijalankan oleh Android tidak hanya menjadi alat komunikasi saja, melainkan dapat menjadi *self-assistant*, dapat

*Received date 2016-02-11, Revised date 2016-02-25, Accepted date 2016-02-29*
*DOI : 10.20449/jnte.v5i1.199*

digunakan untuk gaming, *browsing*, pemutar musik dan video, memotret gambar dan merekam video, media penyimpanan, modem, bahkan sampai *internet banking*[4].

Dalam suatu media penyimpanan, terdapat suatu data penting atau rahasia yang tidak semua orang boleh mengetahuinya. Data-data penting yang hanya boleh diketahui oleh pemiliknya saja antara lain dokumen, video, foto, akun email, akun jejaring sosial, akun kartu kredit, akun internet banking. Apalagi saat proses pengiriman file melalui media internet maupun saat perangkat *mobile* itu hilang, membuat pemiliknya sangat riskan kehilangan data-data pentingnya[4].

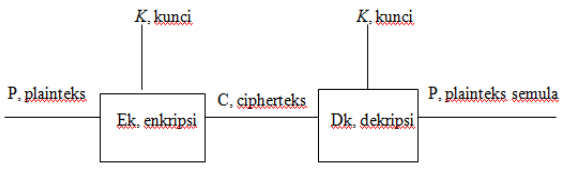
Oleh karena itu, dalam penelitian ini akan coba dibuat sebuah aplikasi pengamanan data berupa dokumen, gambar, dan video dengan menggunakan metode algoritma *Blowfish* untuk mengenkripsi data yang berjalan pada sistem operasi Android sehingga pemilik merasa aman untuk menyimpan datanya.

## 2. TINJAUAN PUSTAKA

### 2.1. Enkripsi dan Dekripsi

Proses menyandikan plainteks menjadi cipherteks[5] disebut enkripsi (*encryption*) atau *enciphering* (standar nama menurut ISO 7498-2). Sedangkan proses mengembalikan cipherteks menjadi plainteks semula dinamakan dekripsi (*decryption*) atau *deciphering* (standar nama menurut ISO 7498-2). Enkripsi dan dekripsi dapat diterapkan baik pada pesan yang dikirim maupun pada pesan tersimpan[6][7].

Istilah *encryption of data in motion* mengacu pada enkripsi pesan yang ditransmisikan melalui saluran komunikasi, sedangkan istilah *encryption of data at-rest* mengacu pada enkripsi dokumen yang disimpan di dalam *storage*. Gambar 1 menunjukkan skema enkripsi dan dekripsi.




Gambar 1. Skema enkripsi dan dekripsi[6][8][9]

### 2.2. Algoritma *Blowfish*

*Blowfish* termasuk dalam enkripsi *block Chiper 64-bit* dengan panjang kunci minimal 32-bit sampai 448-bit[10]. *Blowfish* alias "*OpenPGP.Cipher.4*" merupakan enkripsi yang termasuk dalam golongan *Symmetric Cryptosystem*[5], metode enkripsinya mirip dengan DES (DES *like* Cipher) diciptakan oleh seorang Cryptanalyst bernama *Bruce Schneier* Presiden perusahaan *Counterpane Internet Security*, Inc (Perusahaan konsultan tentang kriptografi dan keamanan komputer) dan dipublikasikan tahun 1994[11]. Dibuat untuk digunakan pada komputer yang mempunyai mikroprosesor besar (32-*bit* keatas dengan *cache* data yang besar).

*Blowfish* dikembangkan untuk memenuhi kriteria desain yang cepat dalam implementasinya dimana pada keadaan optimal dapat mencapai 26 *clock cycle* per *Byte*, kompak dimana dapat berjalan pada memori kurang dari 5 KB, sederhana dalam algoritmanya sehingga mudah diketahui kesalahannya, dan keamanan yang variable panjang kunci bervariasi (minimum 32 *bit*, maksimum 448 *bit*, *multiple* 8 *bit*, *default* 128 *bit*)[4][12].

#### 2.2.1. *Key Expansion*

Berfungsi merubah kunci (minimum 32-*bit*, maksimum 448-*bit*) menjadi beberapa *array* subkunci (*subkey*) dengan total 4168 *Byte* (18x32-*bit* untuk P-*array* dan 4x256x32-*bit* untuk S-*box* sehingga totalnya 33344 *bit* atau 4168 *Byte*)[4]. Kunci disimpan dalam K-*array*:

$$K1, K2, \ldots Kj \; 1 \leq j \leq 14$$

Kunci-kunci ini yang dibangkitkan (*generate*) dengan menggunakan subkunci yang harus dihitung terlebih dahulu sebelum enkripsi atau dekripsi data. Sub-sub kunci yang digunakan terdiri dari:

P-*array* yang terdiri dari 18 buah 32-*bit* subkunci,

$$P1, P2, \ldots, P18$$

S-*box* yang terdiri dari 4 buah 32-*bit*, masing-masing memiliki 256 entri:

$$S_{1,0}, S_{1,1}, S_{1,2}, S_{1,3}, \ldots\ldots\ldots\ldots S_{1,255}$$
$$S_{2,0}, S_{2,1}, S_{2,2}, S_{2,3}, \ldots\ldots\ldots\ldots S_{2,255}$$
$$S_{3,0}, S_{3,1}, S_{3,2}, S_{3,3}, \ldots\ldots\ldots\ldots S_{3,255}$$
$$S_{4,0}, S_{4,1}, S_{4,2}, S_{4,3}, \ldots\ldots\ldots\ldots S_{4,255}$$

Langkah-langkah perhitungan atau pembangkitan subkunci tersebut adalah sebagai berikut:

1. Inisialisasi P-*array* yang pertama dan juga empat S-*box*, berurutan, dengan *string* yang

telah pasti. *String* tersebut terdiri dari digit-digit heksadesimal dari phi, tidak termasuk angka tiga di awal.
2. XOR-kan P1 dengan 32-*bit* awal kunci, XOR-kan P2 dengan 32-*bit* berikutnya dari kunci, dan seterusnya untuk semua *bit* kunci. Ulangi siklus seluruh *bit* kunci secara berurutan sampai seluruh P-*array* ter-XOR-kan dengan *bit*-*bit* kunci. Atau jika disimbolkan : P1 = P1 ⊕ K1, P2 = P2 ⊕ K2, P3 = P3 ⊕ K3, . . . P14 = P14 ⊕ K14, P15 = P15 ⊕ K1, . . . P18 = P18 ⊕ K4. Keterangan : ⊕ adalah simbol untuk XOR.
3. Enkripsikan *string* yang seluruhnya nol dengan algoritma *Blowfish*, menggunakan subkunci yang telah dideskripsikan pada langkah 1 dan 2.
4. Gantikan P1 dan P2 dengan *output* dari langkah 3.
5. Enkripsikan keluaran langkah 3 menggunakan algoritma *Blowfish* dengan subkunci yang telah dimodifikasi.
6. Gantikan P3 dan P4 dengan *output* dari langkah 5.
7. Lanjutkan langkah-langkah *a-f*, gantikan seluruh elemen P-*array* dan kemudian keempat S-*box* secara berurutan, dengan hasil keluaran algoritma *Blowfish* yang terus-menerus berubah.

Total terdapat 521 iterasi untuk menghasilkan subkunci dan membutuhkan memori sebesar 4KB.

### 2.2.2. Enkripsi Data

Terdiri dari iterasi fungsi sederhana (*Feistel Network*) sebanyak 16 kali putaran (iterasi), masukannya adalah 64 *bit* elemen data X. Setiap putaran terdiri dari permutasi kunci *dependent* dan substitusi kunci dan data *dependent*. Semua operasi adalah penambahan (*addition*) dan XOR pada variabel 32-*bit*. Operasi tambahan lainnya hanyalah empat penelusuran tabel *array* berindeks untuk setiap putaran. Langkahnya adalah seperti berikut:

1. Bagi X menjadi dua bagian yang masing-masing terdiri dari 32-*bit*: XL, XR.
2. Lakukan langkah berikut
   For i = 1 to 16:
   XL = XL ⊕ Pi
   XR = F(XL) ⊕ XR
3. Tukar XL dan XR
4. Setelah iterasi ke-16, tukar XL dan XR lagi untuk melakukan membatalkan pertukaran terakhir.
5. Lalu lakukan
   XR = XR ⊕ P17
   XL = XL ⊕ P18
6. Terakhir, gabungkan kembali XL dan XR untuk mendapatkan cipherteks.

Dekripsi sama persis dengan enkripsi, kecuali bahwa P1, P2,…, P18 digunakan pada urutan yang berbalik (*reverse*). Algoritmanya dapat dinyatakan sebagai berikut:

for i = 1 to 16 do
XRi = XLi-1 ⊕ P19-i;
XLi = F[XRi] ⊕ XRi-1;
XL17 = XR16 ⊕ P1;
XR17 = XL16 ⊕ P2;

## 3. METODOLOGI PENELITIAN

### 3.1. Perancangan Antarmuka (*Interface*)

Perancangan *interface* adalah bagian yang penting dalam aplikasi karena yang pertama kali dilihat ketika aplikasi dijalankan adalah *interface* aplikasi. Perancangan antarmuka sendiri terdiri dari perancangan antarmuka menu utama, perancangan antarmuka *about* dan perancangan antarmuka *help*. Perancangan antarmuka sendiri menggunakan bahasa XML.

1. Perancangan Antarmuka Menu Utama
   Perancangan antarmuka menu utama adalah tampilan yang terdiri dari bagian *input file*, *browse*, *password*, *encrypt*, *decrypt*, *home*, *about*, *help*.
2. Perancangan Antarmuka *About*
   Perancangan antarmuka *about* adalah tampilan yang berisi mengenai penjelasan tentang aplikasi yang telah dibuat.
3. Perancangan Antarmuka *Help*
   Perancangan antarmuka *help* adalah tampilan yang berisi mengenai penjelasan tentang cara melakukan enkripsi dan dekripsi file.

### 3.2. Pembuatan Kelas Encryption

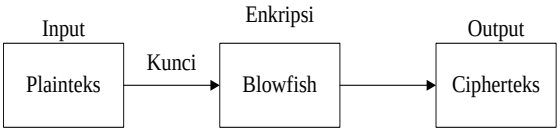
Kelas *encryption* ini adalah kelas yang digunakan untuk melakukan proses enkripsi dan dekripsi file. Pada pembuatan program ini, digunakan *software Eclipse* dengan menggunakan Java. Tampilan pada aplikasi OS Android diatur oleh file XML yang terdapat pada folder res/layout. Tabel 1 merupakan daftar tampilan *layout* yang telah dibuat.

Tabel 1. Nama File *Layout* Beserta Fungsinya

| Nama File | Fungsi |
|---|---|
| about.xml | Mengatur tampilan pada halaman *about* |
| file_view.xml | Mengatur tampilan *browse* file |
| footer.xml | Mengatur tampilan menu di bawah |
| header.xml | Mengatur tampilan menu di atas |
| help.xml | Mengatur tampilan pada halaman *help* |
| main.xml | Mengatur tampilan pada halaman awal |

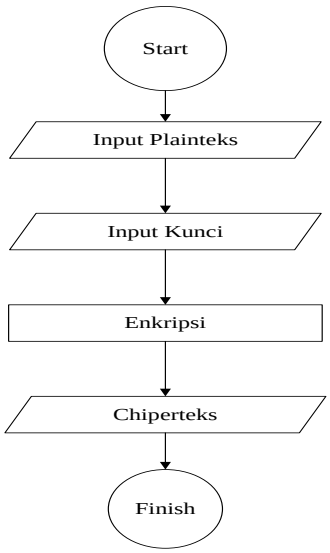
### 3.3. Proses Enkripsi File

Pada tahap ini merancang program untuk mengenkripsi file dengan algoritma *Blowfish*. Diagram blok untuk proses enkripsi file seperti pada Gambar 2.




Gambar 2. Diagram Blok Proses Enkripsi File

Gambar 3 memperlihatkan *flowchart* proses enkripsi file secara keseluruhan. Untuk melakukan proses enkripsi file hal pertama yang dilakukan adalah input plainteks berupa file yang telah ditentukan ukuran dan format filenya.

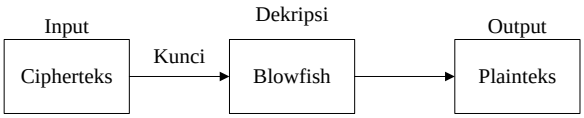



Gambar 3. *Flowchart* Proses Enkripsi File

Kunci yang digunakan untuk proses enkripsi file bisa berupa gabungan angka, huruf dan karakter khusus sesuai keinginan dari penggunanya. *Blowfish* sendiri menggunakan kunci simetris dimana kunci untuk enkripsi dan dekripsi sama. Setelah proses enkripsi file berhasil maka hasil *output*nya berupa cipherteks yang sudah tidak dapat dimengerti maknanya.

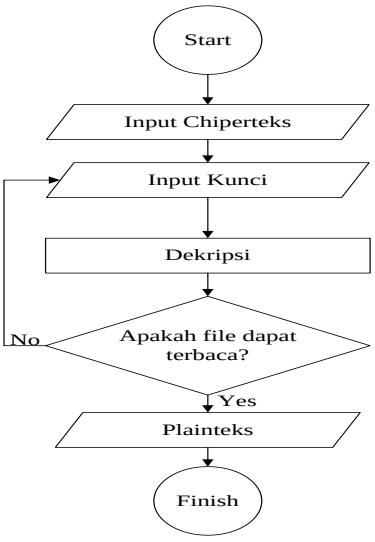
### 3.4. Proses Dekripsi File

Pada tahap ini merancang program untuk mendekripsi file menggunakan algoritma *Blowfish*. Diagram blok sistem untuk proses dekripsi file diperlihatkan pada Gambar 4.




Gambar 4. Diagram Blok Proses Dekripsi File

Gambar 5 memperlihatkan *flowchart* proses enkripsi file secara keseluruhan. Untuk melakukan proses dekripsi file hal pertama yang dilakukan adalah input cipherteks berupa file yang telah ditentukan ukuran dan format filenya.




Gambar 5. *Flowchart* Proses Dekripsi File

Kunci yang digunakan untuk proses enkripsi file bisa berupa gabungan angka, huruf dan karakter khusus sesuai keinginan dari penggunanya. *Blowfish* sendiri menggunakan kunci simetris dimana kunci untuk enkripsi dan dekripsi sama. Setelah proses dekripsi file

berhasil maka hasil *output*nya berupa plainteks yang bisa dimengerti maknanya.

## 4. HASIL DAN PEMBAHASAN

### 4.1. Pengujian Program Aplikasi

Pada bagian ini dilakukan pengujian aplikasi untuk mengenkripsi file dan setelah proses enkripsi file selesai dilakukan akan dilihat hasilnya kemudian dilakukan pengujian dekripsi file untuk mengembalikan file seperti semula. Pengujian dilakukan dengan format file yang berbeda-beda yang umum terdapat pada perangkat Android. Untuk tingkat keamanannya dilakukan terhadap serangan *brute force*.

#### 4.1.1. Pengujian Terhadap Ukuran File

Hasil pengujian terhadap file-file yang telah ditentukan sebelumnya dari berbagai macam format file dapat dilihat pada Tabel 2 yang menunjukkan hasil ukuran file setelah dilakukan proses enkripsi file dan Tabel 3 yang menunjukkan hasil ukuran file setelah dilakukan proses dekripsi file.

Setelah file didekripsi kembali ukuran file akan sama seperti semula. Hal ini menunjukan aplikasi ini dapat bekerja dengan baik.

Tabel 2. Hasil Proses Enkripsi beberapa format file yang berbeda

| No | Nama file plainteks | Ukuran file plainteks (*Byte*) | Ukuran file ciphertek s (*Byte*) |
|---|---|---|---|
| 1 | Algoritma dan Flowchart.pptx | 1528716 | 1528720 |
| 2 | Berapa Lama Waktu Bongkar Password.doc | 117248 | 117256 |
| 3 | Cara Merubah Background Pas Foto.mp4 | 4910554 | 4910560 |
| 4 | Fellaini.jpg | 334713 | 334720 |
| 5 | LIST SILABUS CCNA EXPLORATION.txt | 1114 | 1120 |
| 6 | MAC Random Access 10.pptx | 442168 | 442176 |
| 7 | Matematika Teknik 2 excel 97-2003.xls | 151552 | 151560 |
| 8 | Motorola Charm.jpg | 201285 | 201288 |
| 9 | Paper Aes.pdf | 673435 | 673440 |
| 10 | Sejarah Adobe Photoshop.pdf | 1130348 | 1130352 |

Ukuran file setelah dienkripsi lebih besar dari ukuran asli disebabkan karena adanya proses *padding* yaitu untuk memperoleh blok terakhir sebesar 64 *bit*. Panjang *password*/*key* untuk melakukan pengujian ini adalah 9 karakter yang terdiri dari kombinasi huruf dan angka. *Key* yang digunakan untuk ujicoba adalah *qwerty123*.

Tabel 3. Hasil Proses Dekripsi beberapa format file yang berbeda

| No | Nama file plainteks | Ukuran file ciphertek s (*Byte*) | Ukuran file plainteks (*Byte*) |
|---|---|---|---|
| 1 | Algoritma dan Flowchart.pptx | 1528720 | 1528716 |
| 2 | Berapa Lama Waktu Bongkar Password.doc | 117256 | 117248 |
| 3 | Cara Merubah Background Pas Foto.mp4 | 4910560 | 4910554 |
| 4 | Fellaini.jpg | 334720 | 334713 |
| 5 | LIST SILABUS CCNA EXPLORATION.txt | 1120 | 1114 |
| 6 | MAC Random Access 10.pptx | 442176 | 442168 |
| 7 | Matematika Teknik 2 excel 97-2003.xls | 151560 | 151552 |
| 8 | Motorola Charm.jpg | 201288 | 201285 |
| 9 | Paper Aes.pdf | 673440 | 673435 |
| 10 | Sejarah Adobe Photoshop.pdf | 1130352 | 1130348 |

#### 4.1.2. Pengujian Enkripsi Terhadap Beberapa Tipe File

Ujicoba enkripsi dan dekripsi yang telah dilakukan pada file yang berformat (.pptx), (.doc), (.mp4), (.jpg), (.txt), (.xls) dan (.pdf).

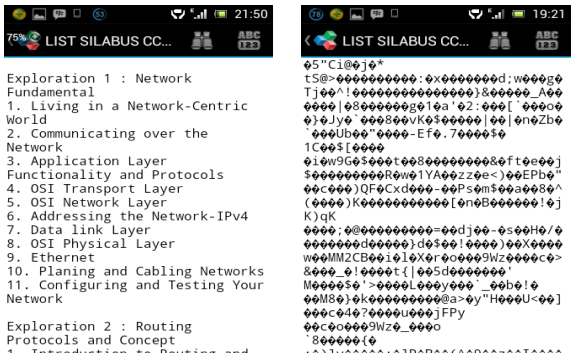
1. File .txt

Untuk pengujian pada file (.txt) dengan nama file LIST SILABUS CCNA EXPLORATION.txt dengan menggunakan *password* "qwerty123" untuk enkripsi dan dekripsinya. Plainteksnya ditunjukkan pada Gambar 6 (a) dengan ukuran file sebesar 1114 *Byte*. Hasil enkripsinya ditunjukkan pada Gambar 6 (b) dengan ukuran file sebesar 1120 *Byte* atau lebih besar 6 *Byte* dari ukuran aslinya.

2. File .jpg

Untuk pengujian pada file (.jpg) dengan nama file Motorola Charm.jpg dengan menggunakan

*password* "qwerty123" untuk enkripsi dan dekripsinya. Plainteksnya ditunjukkan pada Gambar 7 (a) dengan ukuran file sebesar 201285 *Byte*. Hasil enkripsinya ditunjukkan pada Gambar 7 (b) dengan ukuran file sebesar 201288 *Byte* atau lebih besar 3 *Byte* dari ukuran aslinya.




(a) (b)

Gambar 6. Plainteks (a) dan Cipherteks (b) dari file berformat .txt

(a) (b)

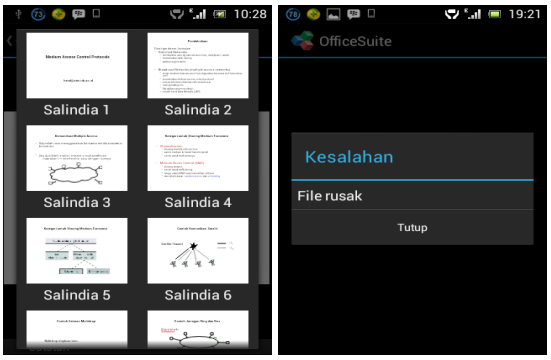
Gambar 7. Plainteks (a) dan Cipherteks (b) dari file berformat .jpg

3. File.pptx

Untuk pengujian pada file (.pptx) dengan nama file MAC Random Access 10.pptx dengan menggunakan *password* "qwerty123" untuk enkripsi dan dekripsinya. Plainteksnya ditunjukkan pada Gambar 8 (a) dengan ukuran file sebesar 442168 *Byte*. Hasil enkripsinya ditunjukkan pada Gambar 8 (b) dengan ukuran file sebesar 442176 *Byte* atau lebih besar 8 *Byte* dari ukuran aslinya. Hasil enkripsinya berupa file yang tidak bisa dibaca kembali.

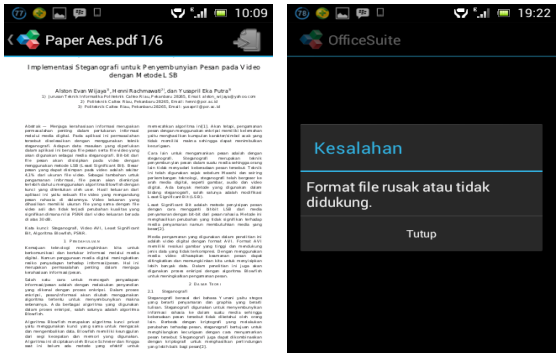
4. File.pdf

Untuk pengujian pada file (.pdf) dengan nama file Paper Aes.pdf dengan menggunakan *password* "qwerty123" untuk enkripsi dan dekripsinya. Plainteksnya ditunjukkan pada Gambar 9 (a) dengan ukuran file sebesar 673435 *Byte*. Hasil enkripsinya ditunjukkan pada Gambar 9 (b) dengan ukuran file sebesar 673440 *Byte* atau lebih besar 5 *Byte* dari ukuran aslinya. Hasil enkripsinya berupa file yang tidak bisa dibaca kembali.




(a) (b)

Gambar 8. Plainteks (a) dan Cipherteks (b) dari file berformat .pptx

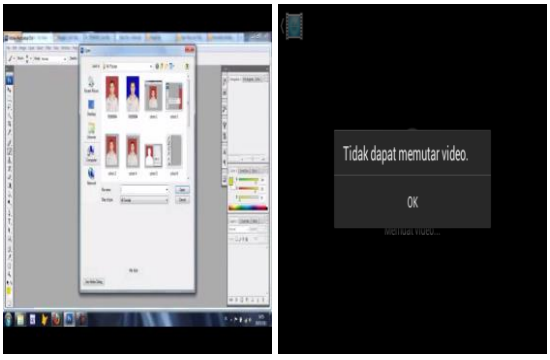



(a) (b)

Gambar 9. Plainteks (a) dan Cipherteks (b) dari file berformat .pdf

5. File.mp4

Untuk pengujian pada file (.mp4) dengan nama file Cara Merubah Background Pas Foto.mp4 dengan menggunakan *password* "qwerty123" untuk enkripsi dan dekripsinya. Plainteksnya ditunjukkan pada Gambar 10 (a) dengan ukuran file sebesar 4910554 *Byte*. Hasil enkripsinya ditunjukkan pada Gambar 10 (b) dengan ukuran file sebesar 4910560 *Byte* atau lebih besar 6 *Byte* dari ukuran aslinya.

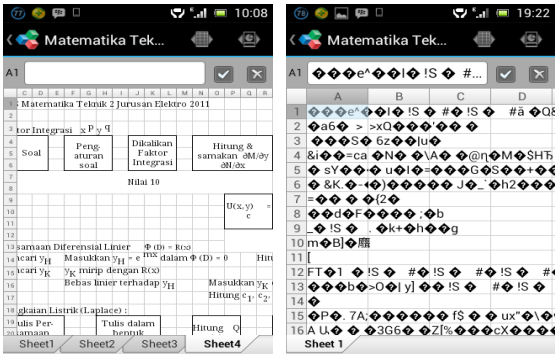



(a) (b)

Gambar 10. Plainteks (a) dan Cipherteks (b) dari file berformat .mp4

6. File .xls

Untuk pengujian pada file (.xls) dengan nama file Matematika Teknik 2 excel 97-2003.xls dengan menggunakan *password* "qwerty123" untuk enkripsi dan dekripsinya. Plainteksnya ditunjukkan pada Gambar 11 (a) dengan ukuran file sebesar 151552 *Byte*. Hasil enkripsinya ditunjukkan pada Gambar 11 (b) dengan ukuran file sebesar 151560 *Byte* atau lebih besar 8 *Byte* dari ukuran aslinya.

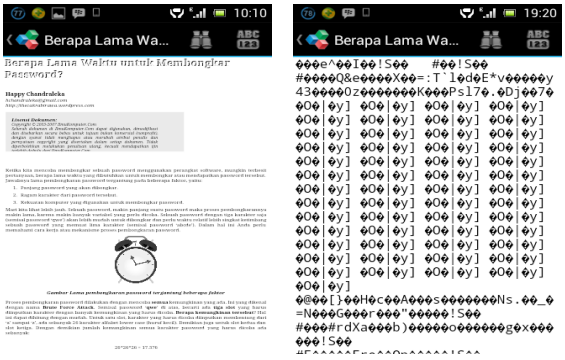



(a) (b)

Gambar 11. Plainteks (a) dan Cipherteks (b) dari file berformat .xls

7. File.doc

Untuk pengujian pada file (.doc) dengan nama file Berapa Lama Waktu Bongkar *Password*.doc dengan menggunakan *password* "qwerty123" untuk enkripsi dan dekripsinya. Plainteksnya ditunjukkan pada Gambar 12 (a) dengan ukuran file sebesar 117248 *Byte*. Hasil enkripsinya ditunjukkan pada Gambar 12 (b) dengan ukuran file sebesar 117256 *Byte* atau lebih besar 8 *Byte* dari ukuran aslinya.




(a) (b)

Gambar 12. Plainteks (a) dan Cipherteks (b) dari file berformat .doc

Dari hasil pengujian aplikasi terhadap file-file yang telah ditentukan format filenya, rancangan proses enkripsi pada file tersebut yang telah dirancang dapat diimplementasikan dan berhasil dijalankan dengan baik. Hal ini dibuktikan dengan aplikasi yang bisa diinstal ke dalam *handphone* Android dan dapat digunakan untuk melakukan proses enkripsi dan dekripsi file. *Password/key* digunakan harus sama karena algoritma *Blowfish* menggunakan kunci simetris untuk melakukan proses enkripsi dan dekripsi file. Proses dekripsi file dengan *password/key* yang salah tidak dapat membuka file kembali seperti keadaan semula. Untuk itu pengguna aplikasi ini disarankan untuk mengingat dengan baik *password/key* yang digunakan. Jika pengguna lupa *password/key* maka file tidak dapat didekripsi lagi. Panjang *password/key* dan kerahasiaan *password/key* mempengaruhi dari tingkat keamanan file itu sendiri.

Untuk tingkat keamanan dilakukan pengujian terhadap serangan *brute force*. Serangan bertipe *brute force* adalah dengan menerapkan percobaan setiap kemungkinan kunci yang ada satu per satu sampai diperoleh plainteks yang diharapkan. Waktu yang diperlukan untuk memperoleh kunci yang diharapkan, selalu berbanding lurus dengan panjang *bit* kunci yang dimiliki oleh algoritma kriptografi, dan makin cepat prosesor yang dipakai, makin cepat waktu yang pula dibutuhkan untuk dapat memperoleh kunci tersebut. Analisa *brute force attack* pada sistem ini adalah dengan melakukan perhitungan matematis dari masing-masing kunci untuk mendapatkan nilai durasi proses memecahkan kemungkinan kunci. Panjang kunci yang digunakan dalam penelitian ini adalah 72 *bit*.

Kemungkinan kunci yang dapat dihasilkan dengan menggunakan panjang 72 *bit* adalah sejumlah 4,7 x $10^{21}$ kunci. Asumsi kecepatan komputasi adalah $10^6$ *key/sec* maka :

$$\frac{4{,}7\times10^{21}}{10^{6}}=\frac{4{,}7\times10^{15}\,\text{sec}}{3{,}1536\times10^{7}}=1{,}49\times10^{8}$$

menghasilkan durasi sejumlah 1,49x$10^8$ tahun untuk memecahkan seluruh kemungkinan kunci.

Berdasarkan dari hasil pengujian tersebut, maka aplikasi enkripsi dan dekripsi file ini aman untuk melindungi file-file yang sifatnya pribadi dan rahasia yang orang lain tidak berhak untuk mengetahuinya.

## 5. KESIMPULAN

Berdasarkan pengujian dan analisis yang telah dilakukan Aplikasi enkripsi dan dekripsi file dapat berfungsi dengan baik pada *handphone* Android dengan OS Android 2.3, 4.0, dan 4.1. Aplikasi yang dibuat dapat diimplementasikan dengan baik untuk melakukan enkripsi dan dekripsi file karena file yang telah dienkripsi menjadi tidak bisa dimengerti dan pahami lagi isinya. Tingkat keamanan dari aplikasi yang dibuat cukup aman karena algoritma *Blowfish* memiliki panjang kunci yang besar. Dengan menggunakan kunci berjumlah 72 *bit* atau 9 karakter dibutuhkan waktu 1,49x$10^8$ tahun untuk membongkarnya dengan kecepatan komputasinya adalah $10^6$ *key/sec*.

Diharapkan kedepan sistem aplikasi ini bisa dikembangkan menjadi lebih baik lagi seperti menambahkan beberapa fitur diantaranya penambahan pilihan algoritma lain dan kemampuan untuk mengenkripsi folder. Pembuatan fungsi waktu sehingga bisa diketahui seberapa cepat enkripsi dan dekripsi filenya. Untuk mengetahui tingkat keamanan, perlu dilakukan pula uji *sniffing* di jaringan komputer.

## DAFTAR PUSTAKA

[1] Wardoyo, Siswo, Taufik Ryadi, dan Rian Fahrizal. "Analisis Performa File Transport Protocol Pada Perbandingan Metode IPv4 Murni, IPv6 Murni dan Tunneling 6to4 Berbasis Router Mikrotik." Jurnal Nasional Teknik Elektro, Vol 3. no 2, 2014.

[2] Mukti, Prasetiyono Hari, Rizki Aris Yunianto, dan Achmad Affandi. "Evaluasi Kinerja Layanan IPTV pada Jaringan Testbed WiMAX Berbasis Standar IEEE 802.16-2004". Jurnal Nasional Teknik Elektro, Vol 4. no 2, 2015.

[3] D. T. Massandy, "Studi dan Implementasi Cryptography Package pada Sistem Operasi Android," no. 13508051, 2011.

[4] C. A. Sutanto, "Penggunaan Algoritma Blowfish dalam Kriptografi," Bandung, Indonesia, 2010.

[5] E.A.Shanty, "Implementasi Algoritma Kriptografi Blowfish Untuk Keamanan Dokumen Pada Microsoft Office," *J. Chem. Inf. Model.*, vol. 53, no. 9, pp. 1689–1699, 2013.

[6] A. Rahman, T. Maskes, "Implementasi Algoritma Serpent Untuk Enkripsi Dan Dekripsi Data File Pada Ponsel," *J. Jur. Tek. Inform. STMIK GI MDP*, 2013.

[7] I. Rahmayun, D. Jurusan, T. Informasi, and P. Negeri, "Enkripsi SMS ( SHORT MESSAGE SERVICE ) Pada Telepon Selular Berbasis Android Dengan Metode RC6," *J. Momentum*, vol. 16, no. 1, pp. 63–73, 2014.

[8] R. Munir, *Kriptografi*. Bandung, Indonesia: Penerbit Informatika Bandung, 2006.

[9] R. Munir, "Diktat Kuliah IF5054 Kriptografi," Bandung, Indonesia, 2004.

[10] Tetuko, P.N, A. Qoiriah, "Rancang Bangun Aplikasi Enkripsi Database MYSQL Dengan Algoritma BLOWFISH," *J. Manejemen Inform.*, vol. 02, no. 01, pp. 39–44, 2013.

[11] S. Sitinjak and Y. Fauziah, "Aplikasi Kriptografi File Menggunakan," vol. 2010, no. semnasIF, pp. 78–86, 2010.

[12] A. E Pratiwi, "Implementasi Enkripsi Data Dengan Algoritma Blowfish Menggunakan Java Pada Aplikasi Email," *J. Jur. Tek. Komput. Bandung*, 2011.

***Biodata Penulis***

**Siswo Wardoyo**, lahir di Boyolali tanggal 07 Maret 1978 Pendidikan S1 di Jurusan Teknik Elektro Universitas Diponegoro Semarang pada tahun 2002. Pendidikan S2 di Jurusan Teknik

Elektro Universitas Gadjah Mada Yogyakarta dengan Program Studi Teknik Elektro pada tahun 2008. Sekarang menjadi staff pengajar di Jurusan Teknik Elektro, Universitas Sultan Ageng Tirtayasa, Cilegon, Banten, Indonesia.