Keamanan file dokumen menggunakan algoritme Advanced Encryption Standard pada aplikasi berbasis Android

Tata Fajri Nurrahmi[1], Emy Setyaningsih[*2], and Nuniek Herawati[3]

1 **Jurusan Rekayasa Sistem Komputer, Institut Sains & Teknologi AKPRIND Yogyakarta**
Jl. Kalisahak No 28. Kompl. Balapan Yogyakarta
tatafajri77@gmail.com

2 **Jurusan Rekayasa Sistem Komputer, Institut Sains & Teknologi AKPRIND Yogyakarta**
Jl. Kalisahak No 28. Kompl. Balapan Yogyakarta
emysetyaningsih@akprind.ac.id

3 **Jurusan Rekayasa Sistem Komputer, Institut Sains & Teknologi AKPRIND Yogyakarta**
Jl. Kalisahak No 28. Kompl. Balapan Yogyakarta
nuniekh@akprind.ac.id

Abstrak

Seiring dengan peningkatan kemampuan perangkat *mobile* terutama media penyimpanan yang ukurannya semakin besar, memungkinkan pengguna menyimpan file dokumen penting ke perangkat *mobile*. File dokumen rahasia tersebut menjadi sangat rentan untuk diketahui, diambil atau bahkan dimanipulasi dan disalahgunakan oleh pihak lain yang tidak berhak mengakses perangkat *mobile* tersebut. Oleh karena itu, dibutuhkan aplikasi berbasis Android yang dapat melindungi file dokumen agar tidak dapat dibaca oleh orang lain. Salah satu metode yang dapat digunakan untuk melindungi file dokumen tersebut dari serangan pihak yang tidak bertanggung jawab adalah metode kriptografi. Algoritme kriptografi yang paling sering digunakan untuk mengamankan dokumen dalam bentuk teks adalah algoritme *Advanced Encryption Standard* (AES). Penelitian ini berhasil membangun aplikasi pengamanan file dokumen dengan format penyimpanan *.pdf, *.doc, *.ppt dan *.xls menggunakan algoritme AES berbasis Android. Aplikasi yang dibangun memiliki kelebihan pada penggunaan kunci AES yang selalu berbeda untuk proses enkripsi, sehingga lebih aman terhadap serangan *brute-force*. Penelitian ini juga melakukan perbandingan kinerja dari AES-128, AES-192 dan AES-256 berdasarkan kecepatan proses enkripsi dan dekripsi. Berdasarkan pengujian yang dilakukan, kecepatan waktu enkripsi dan dekripsi tidak dipengaruhi oleh jenis format penyimpanan file, namun dipengaruhi oleh ukuran file dan ukuran kuncinya. Semakin besar ukuran file asli (plainteks) maka semakin besar pula kebutuhan waktu prosesnya. Proses enkripsi dan dekripsi menggunakan panjang kunci 128 bit juga membutuhkan waktu paling cepat dibandingkan menggunakan panjang kunci 256 bit. Hasil pengujian menggunakan uji analisis histogram, memperlihatkan histogram dari cipherteks relatif rata yang menunjukkan algoritme kriptografi AES aman terhadap *statistical attack*. Hasil ini menunjukkan bahwa aplikasi berbasis Android untuk keamanan dokumen menggunakan algoritme AES yang dibangun memiliki keamanan cukup tinggi serta cepat proses enkripsi dan dekripsinya.

Kata Kunci AES, JCA, Android, kriptografi, keamanan dokumen

* Corresponding author.

1 Pendahuluan

Penggunaan perangkat *mobile* sudah menjadi trend di masyarakat dunia. Teknologi perangkat *mobile* berkembang sangat pesat sehingga mempunyai dampak dalam meningkatkan efektifitas dan efisiensi terutama sebagai media komunikasi untuk mendukung setiap pekerjaan. Salah satu sistem operasi *mobile* buatan Google yang banyak digunakan saat ini adalah Android. Hal ini disebabkan kecanggihan perangkat *mobile* dengan sistem operasi Android yang tidak hanya menjadi alat komunikasi, melainkan dapat menjadi *self-assistant*, yaitu untuk mengambil foto, video, suara, membuat dokumen, dan juga menyimpan data.

Media komunikasi memberikan kemudahan akses serta membawa pengaruh terhadap keamanan informasi. Informasi menjadi sangat rentan untuk diketahui, diambil atau bahkan dimanipulasi dan disalahgunakan oleh pihak lain yang tidak berhak. Oleh karena itu, diperlukan suatu metode untuk menjaga keamanan data. Salah satu metode yang dapat dilakukan untuk mengamankan data adalah menggunakan kriptografi. Keamanan dari sebuah algoritme kriptografi diukur dari banyaknya kerja yang dibutuhkan untuk memecahkan chiperteks menjadi plainteks tanpa mengetahui kunci yang digunakan. Kerja ini dapat diekivalenkan dengan waktu, memori, uang, dan lain-lain [1].

Banyak peneliti yang mengembangkan keamanan data dalam bentuk teks, diantaranya menggunakan algoritme *Advanced Encryption Standard* (AES) [2; 3; 4], *International Data Encryption Standart* (IDEA) [5], *Elliptic Curve* [6; 7], menggabungkan algoritme simetrik dan asimetrik [8], serta dengan menggunakan model yang dikembangkan sendiri [9]. Implementasi kriptografi juga dilakukan untuk beragam tujuan, diantaranya [10] membuat perlindungan web pada login sistem menggunakan algoritme *Rijndael* dan melindungi web dari serangan *SQL Injection*, sedangkan [11] menerapkan aplikasi *chatting* berbasis HTML5 WebSocket. Semua proses enkripsi dan dekripsi dengan algoritme *Rijndael* dilakukan di sisi *client*. Hal ini membuat distribusi data pengguna dan pesan instan pada aplikasi tersebut lebih terjaga keamanannya. Data *nickname* dan nama *room* yang disimpan pada server sebelumnya telah mengalami proses enkripsi dengan algoritme *Rijndael*. Data *password room* yang tersimpan pada server sebelumnya telah mengalami proses *hashing* menggunakan algoritme SHA-3. Hal ini membuat keamanan data yang tersimpan di dalam server lebih terjaga keamanannya. Penelitian [3] juga berhasil mengenkripsi pesan teks kemudian disimpan menjadi sebuah file dokumen dan isi file dokumen tersebut di enkripsi. Selanjutnya, hasil enkripsi dari isi file dokumen tersebut, dikompresi dan disembunyikan pada sebuah file gambar. Hasil pengujian menggunakan software *Application Fres-CAESAS* menunjukkan sistem keamanan tersebut cukup optimal dalam mengamankan teks dan dapat terjaga keamanannya karena dibuat dengan cara berlapis. Selain itu, user dapat melakukan kombinasi mengenai teknik keamanan sesuai keinginan user. Sedangkan penelitian yang dilakukan oleh [5] berhasil mengamankan data dalam bentuk format penyimpanan file dokumen yang ada pada Microsoft office word yaitu DOCX menggunakan algoritme IDEA.

Seiring dengan peningkatan kemampuan perangkat *mobile* menggunakan sistem operasi Android terutama media penyimpanan yang ukurannya semakin besar, memungkinkan pengguna menyimpan dokumen penting. Oleh karena itu, dibutuhkan aplikasi berbasis Android yang dapat melindungi dokumen agar tidak dapat dibaca oleh orang lain menggunakan metode kriptografi. Beberapa peneliti telah mengembangkan keamanan data teks menggunakan algoritme kriptografi berbasis Android [12; 13]. Penelitian yang dilakukan oleh [12] berhasil membuat sistem kunci elektronik menggunakan *handphone* sebagai kunci, dan sebuah *mikrokontroler Arduino* pada kendaraan sebagai penerima kontrol. Komunikasi antara perangkat *mobile* dan *mikrokontroler* menggunakan *bluetooth*. Penelitian ini dibuat

untuk melengkapi aspek keamanan kunci elektronik dengan mengimplementasikan AES dengan panjang kunci 128 bit, 192 bit, dan 256 bit. Hasil pengujian menunjukkan algoritme AES berhasil diimplementasikan untuk mengenkripsi atau dekripsi pesan dari sistem operasi Android dengan *mikrokontroler Arduino* yang ada pada kendaraan. Penelitian yang dilakukan oleh [14] berhasil membangun sebuah aplikasi Android foto dengan menggunakan bahasa java. Aplikasi yang dibangun berhasil menyembunyikan sebuah pesan tersembunyi dengan cara menyisipkan pesan ke dalam bit standar AES. Sedangkan [13] menggunakan algoritme AES untuk mengamankan pesan atau percakapan melalui ponsel berbasis Android.

Berdasarkan penjelasan tersebut, sebagian besar penelitian keamanan data teks berbasis Android menggunakan algoritme AES, hal ini disebabkankan AES merupakan salah satu algoritme cipher blok yang cukup aman untuk melindungi dokumen yang bersifat rahasia. Algoritma AES dapat mengenkripsi dan mendekripsi data dengan panjang kunci yang bervariasi, yaitu 128 bit, 192 bit, dan 256 bit [2]. Selain itu AES merupakan cipher yang berorientasi pada bit, sehingga memungkinkan untuk diimplementasikan secara efisien ke dalam software dan hardware. AES memiliki ketahanan terhadap semua jenis serangan yang diketahui, kesederhanaan rancangan, kekompakan kode yang sederhana dan kecepatan pada berbagai platform [13]. Namun, belum banyak penelitian untuk menyelesaikan permasalahan utama yang timbul akibat berkembangnya media penyimpan data perangkat *mobile* menggunakan sistem operasi Android, yaitu tidak adanya perlindungan khususnya file dokumen. Permasalahan ini dikhawatirkan bocornya informasi penting yang dapat dengan mudah dibaca, atau bahkan disebarluaskan oleh orang yang tidak bertanggung jawab. Oleh karena itu, diperlukan rancang bangun aplikasi pengamanan file dokumen berbasis Android untuk mengamankan file dokumen dengan format penyimpanan *.pdf, *.doc, *.ppt dan *.xls.

Berdasarkan paparan di atas, penelitian ini bertujuan untuk membangun aplikasi pengamanan file dokumen menggunakan algoritme AES pada sistem operasi Android. Aplikasi ini diharapkan dapat mengamankan file dokumen yang dapat diakses dengan mudah dan efisien melalui aplikasi yang ada pada perangkat *mobile*. Selanjutnya, pengujian fungsionalitas aplikasi tersebut dilakukan menggunakan metode *blackbox testing* serta mengukur kecepatan proses enkripsi dan dekripsi. Selain itu, juga dilakukan pengukuran tingkat keamanan algoritme AES untuk mengamankan file dokumen yang diimplementasikan pada aplikasi berbasis Android menggunakan analisis histrogram.

Makalah ini disusun sebagai berikut: Bagian 2, membahas metodologi penelitian. Bagian 3 membahas hasil eksperimen dan analisis aplikasi berbasis Android untuk keamanan dokumen menggunakan algoritme AES. Akhirnya, Bagian 4 menyajikan kesimpulan makalah ini.

2 Metodologi

Permasalahan utama media penyimpan data di dalam perangkat *mobile* menggunakan sistem operasi Android adalah tidak adanya perlindungan khusus pada file dokumen. Hal ini dikhawatirkan jika perangkat *mobile* hilang, maka data penting yang tersimpan akan dapat dengan mudah dibaca, atau bahkan disebar luaskan oleh orang yang tidak bertanggung jawab. Oleh karena itu, pada penelitian ini akan membangun aplikasi berbasis Android untuk mengamankan file dokumen dengan metode algoritme AES.

2.1 Data dan Peralatan Penelitian

File dokumen yang digunakan untuk pengujian adalah file dokumen dengan format penyimpanan *.pdf, *.doc, *.ppt dan *.xls seperti ditampilkan pada Tabel 1.

Tabel 1 File dokumen untuk pengujian aplikasi yang dibangun

Jenis File	Nama File	Ukuran File (Kb)
*.doc	AES dalam bhasa java.doc	286
	WORD.doc	1988
	REVISI26LAP KP.doc	7955
*.pdf	TUGAS 3 RESUME revisi.pdf	753
	la-tahzan.pdf	1067
	buku-tahsin-kelas-x.pdf	4931
*.ppt	Representasi Data.pptx	301
	PENGANTAR KOMPUTER DAN IT.pptx	1601
	powerpoint.pptx	4259
*.xls	REALISASI.xlsx	13
	pkm nyata.xlsx	37

Alat yang digunakan pada penelitian ini meliputi *hardware* dan *software* seperti ditampilkan pada Tabel 2 dan 3.

Tabel 2 Spesifikasi *Hardware*

Hardware	Spesifikasi
Laptop	Prosesor: Intel(R) Celeron(R) CPU N3350 @1.10 GHz
	RAM: 6 GB DDR3 L Memory
	Harddisk: 500GB
Smartphone	Type: Meizu Note 3
	RAM: 2 GB
	Versi Android : 5.1 Lollipop

Tabel 3 Spesifikasi *Software*

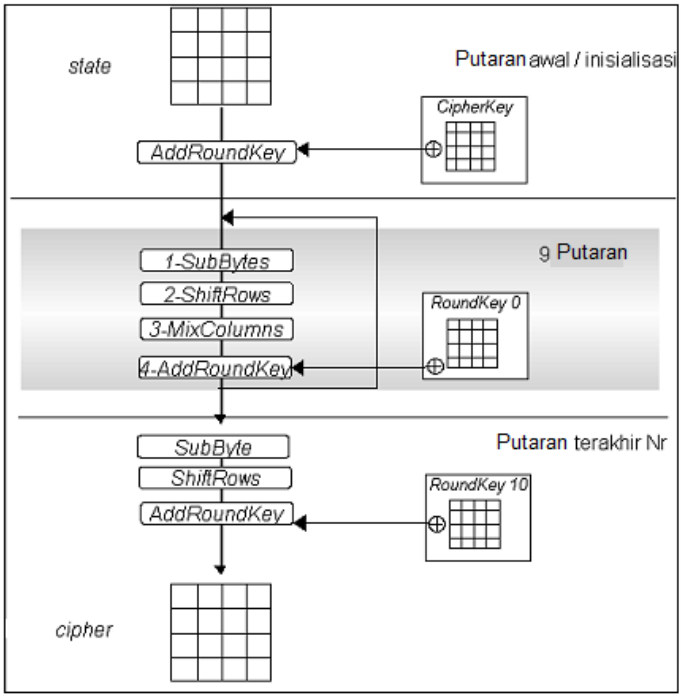
Software	Spesifikasi
Sistem Operasi	Windows 10
UML	The ObjectAid UML Explorer for Eclipse
Design	Android Studio 3.1.1
JDK	Java Development Kit 8.0_11
Browser	Mozilla Firefox
MS Word	Microsoft Word 2016

2.2 Algoritme *Advanced Encryption Standard* (AES)

AES merupakan salah satu algoritme terpopuler yang digunakan dalam kriptografi kunci simetrik. Kunci yang digunakan untuk proses enkripsi dan dekripsi pada AES menggunakan kunci yang sama. AES merupakan algoritme cipher blok yang menggunakan teknik substitusi, permutasi, dan sejumlah putaran pada setiap blok yang akan di enkripsi. Sistem permutasi dan substitusi (S-box) yang digunakan pada AES tidak menggunakan jaringan *Feistel* sebagaimana cipher blok pada umumnya.

2.2.1 Proses Enkripsi Algoritme AES

Proses enkripsi algoritme AES terdiri dari empat jenis transformasi *bytes*, yaitu *SubBytes*, *ShiftRows*, *Mixcolumns*, dan *AddRoundKey* seperti ditampilkan pada Gambar 1 .



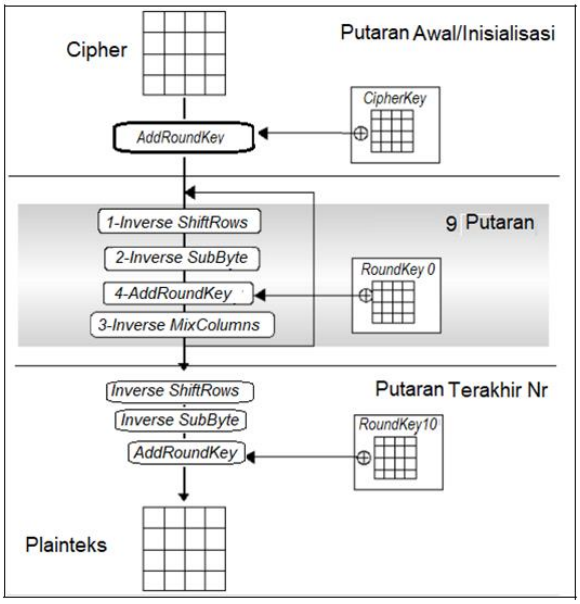
Gambar 1 Proses enkripsi algoritme AES

Proses enkripsi algoritme AES terdiri dari 3 langkah, yaitu:

1. ***Initial round***, yaitu proses mengkopi input ke dalam *state* yang akan mengalami transformasi *AddRoundKey*. Transformasi *AddRoundKey*: melakukan XOR antara *state* awal (plainteks) dengan *cipher key*.
2. **Proses *round function*** sebanyak $Nr - 1$ kali. Proses yang dilakukan pada setiap putaran adalah sebagai berikut :
   - *SubBytes*: substitusi byte dengan tabel substitusi (S-box).
   - *ShiftRows*: pergeseran baris-baris array *state* secara *wrapping*.
   - *MixColumns*: pengacakan data di masing-masing kolom array *state*.
   - *AddRoundKey*: peng-XOR-an antara *state* sekarang dengan *round key*.
3. ***Final round***, yaitu proses untuk putaran terakhir. Putaran yang terakhir agak berbeda dengan putaran sebelumnya, yaitu *state* tidak mengalami transformasi *MixColumns*.

2.2.2 Proses Dekripsi Algoritme AES

Proses dekripsi diimplementasikan dalam arah yang berlawanan dengan enkripsi untuk menghasilkan *inverse cipher*. Transformasi *byte* yang digunakan pada *invers cipher* adalah *Inverse ShiftRows, Inverse SubBytes, Inverse MixColumns*, dan *AddRoundKey*. Urutan proses dekripsi AES tidak merupakan kebalikan dari enkripsi, namun urutannya yang ditukarkan, walaupun penggunaan kuncinya sama. Proses dekripsi algoritme AES terdiri dari 3 langkah seperti pada proses enkripsi seperti ditampilkan pada Gambar 2 .



Gambar 2 Proses dekripsi algoritme AES

2.3 Implementasi Model yang Dikembangkan

2.3.1 Perancangan Aplikasi

Aplikasi pengamanan file dokumen menggunakan algoritme AES berbasis Android dirancang menggunakan *Unified Modeling Language* (UML) sebagai alat bantu analis serta perancangan perangkat lunak berbasis obyek. Selanjutnya, untuk membangun antarmuka aplikasi dilakukan dengan menuliskan kode program menggunakan software Android Studio 3.1 dengan mengimplementasikan *Java Cryptography Architecture* (JCA). JCA merupakan bagian utama dari platform Java yang akan digunakan untuk membangun proses enkripsi atau dekripsi pesan. JCA terdiri dari arsitektur *provider* dan sekumpulan API untuk *digital signatures, message digests (hash)*, sertifikasi dan validasi sertifikat, enkripsi (blok simetri atau asimetri), *key generation* dan manajemen, serta *secure random number generation*. *Packages* dan *classes* yang digunakan untuk enkripsi atau dekripsi file antara lain :

1. *Javax.crypto.Cipher*: digunakan untuk menginisialisasi cipher dan membentuk mode yang kita inginkan *ENCRYPT_MODE* atau *DECRYPT_MODE*.
2. *Javax.crypto.KeyGenerator*: kelas ini menyediakan API publik untuk menghasilkan kunci simetrik.
3. *Javax.crypto.spec.SecretKeySpec*: spesifikasi kunci yang digunakan untuk kunci rahasia dengan ukuran byte.
4. *Java.security.SecureRandom*: kelas ini digunakan ketika sepasang kunci dibuat untuk memilih parameter secara acak.

Alur dari aplikasi pengamanan file dokumen menggunakan algoritme AES berbasis android dibagi menjadi dua tahap, yaitu :

Tahap 1 Proses Pembangkitan Kunci.

Pembangkitan kunci dilakukan secara elektronis dengan menggunakan suatu alat pembangkit rangkaian kunci (*key generator*) yaitu *Random Number Generator* (RNG). RNG

merupakan sejenis perangkat yang dapat menghasilkan rangkaian angka pada interval tertentu sehingga angka yang dihasilkan terlihat tidak dapat diprediksi. Aplikasi ini menggunakan *java.security.SecureRandom* untuk pembuatan kunci secara random dengan menggunakan perintah : *SecureRandom secureRandom =new SecureRandom();*

Tahap 2 **Proses Enkripsi atau Dekripsi**.

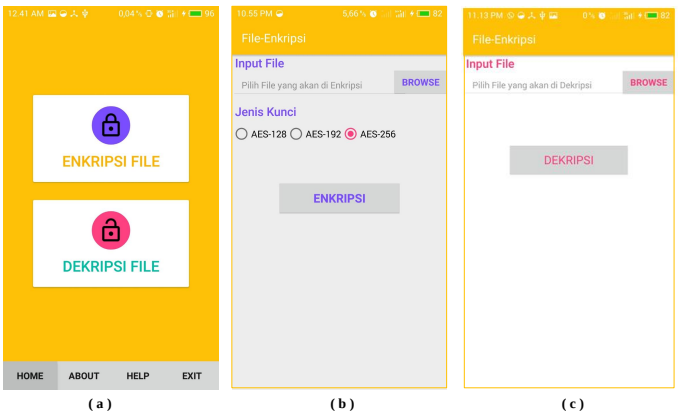
Urutan langkah proses enkripsi adalah sebagai berikut:

1. Membuat *method encrypt* dengan parameter panjang kunci (*outputKeyLength*)
2. Membuat cipherteks dengan menentukan parameter yang digunakan pada algoritme AES
3. Inisialisasi Cipherteks untuk enkripsi dengan menggunakan perintah:
   Cipher cipher = Cipher.getInstance(AES)
4. Enkripsikan File menggunakan perintah:
   Cipher.init(Cipher.ENCRYPT_MODE,key)
5. Menghitung waktu enkripsi.

Urutan langkah proses dekripsi dilakukan dengan cara membalikkan prosedur enkripsi. Jika untuk *encrypt* kita menggunakan *ENCRYPT_MODE* maka untuk decrypt kita menggunakan *DECRYPT_MODE*, sehingga perintah yang digunakan adalah:
Cipher.init(Cipher.DECRYPT_MODE,key)

2.3.2 Hasil Program

Aplikasi berbasis Android untuk keamanan dokumen menggunakan algoritme AES terdiri dari 3 antarmuka, yaitu menu utama/home, proses enkripsi, dan dekripsi seperti diperlihatkan pada Gambar 3.



Gambar 3 Antarmuka (a) Menu Utama/Home (b) Enkripsi (c) Dekripsi

1. **Antarmuka Menu Utama/Home**

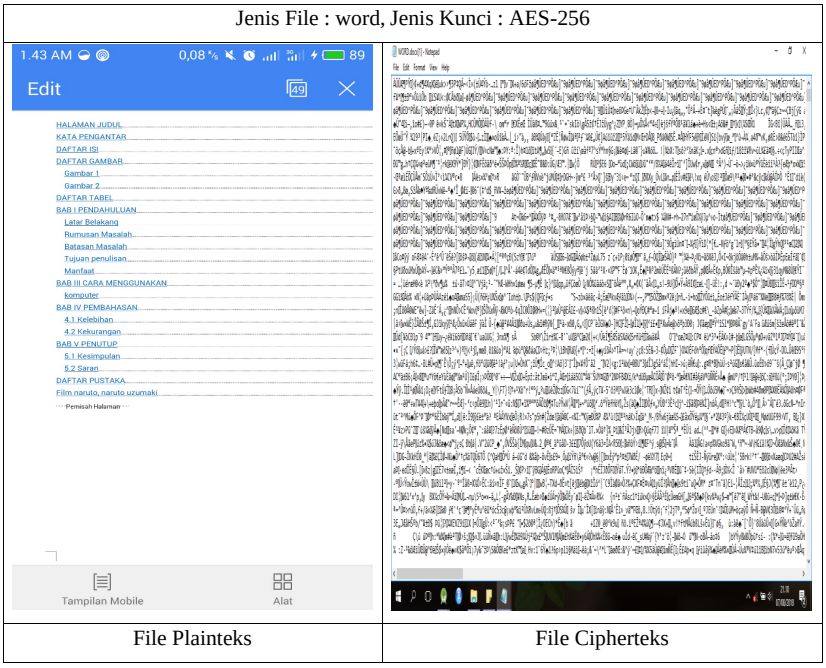
   Saat awal aplikasi dijalankan pada perangkat *mobile* berbasis Android akan tampil menu utama/home seperti ditampilkan pada Gambar 3 (a).

   Fungsi dari bagian-bagian tersebut meliputi:

   - Menu Enkripsi File : menu yang berfungsi untuk melakukan enkripsi file.
   - Menu Dekripsi File : menu yang berfungsi untuk melakukan dekripsi file.
   - Tombol Home : tombol navigasi yang berfungsi untuk membantu pengguna menuju halaman utama aplikasi.

- Tombol About : tombol navigasi yang berfungsi untuk membantu pengguna menuju halaman About.
- Tombol Help : tombol navigasi yang berfungsi untuk membantu pengguna menuju halaman Help.
- Tombol Exit : tombol navigasi yang berfungsi untuk keluar/ menutup aplikasi.

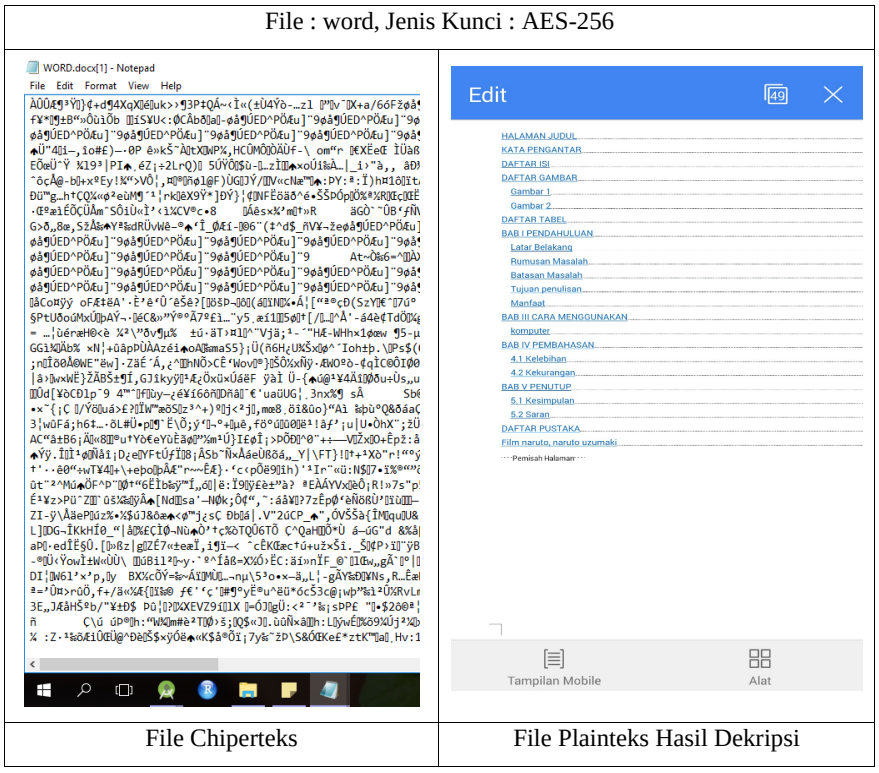
2. **Antar Muka Enkripsi**.
Halaman ini digunakan untuk melakukan proses enkripsi file. Halaman ini akan muncul setelah pengguna memilih menu enkripsi file pada halaman home seperti diperlihatkan pada Gambar 3 (b). Proses enkripsi diawali dengan memilih file yang akan di enkripsi oleh pengguna dengan menekan tombol *browse* untuk memilih file di penyimpanan internal pada perangkat *mobile* berbasis Android. Proses enkripsi selanjutnya dilakukan dengan menggunakan kunci yang dibangkitkan secara random dengan ukuran yang sama dengan algoritme jenis AES yang dipilih oleh user yaitu AES-128, AES-192, atau AES-256. Kunci yang terbentuk secara otomatis akan disimpan ke media penyimpanan menggunakan nama yang sama dengan file yang di enkripsi.
Hasil tampilan dari aplikasi untuk enkripsi apabila dijalankan pada perangkat *mobile* menggunakan sistem operasi Android seperti diperlihatkan pada Gambar 4.



Gambar 4 Cipherteks dari jenis file word hasil enkripsi menggunakan AES-256

3. **Antar Muka Dekripsi**.
Halaman ini digunakan untuk melakukan proses dekripsi file. Halaman ini akan muncul setelah pengguna memilih menu dekripsi file pada halaman home seperti diperlihatkan pada Gambar 3 (c). Proses dekripsi dilakukan dengan cara memilih file yang akan di dekripsi. Sedangkan kunci yang digunakan untuk proses dekripsi didapatkan secara otomatis oleh aplikasi dengan cara mencari kunci yang sesuai seperti nama file yang di pilih oleh user.

Contoh hasil tampilan dari aplikasi untuk proses dekripsi apabila dijalankan pada perangkat *mobile* menggunakan sistem operasi Android seperti diperlihatkan pada Gambar 5.



Gambar 5 Plainteks dari jenis file word hasil dekripsi menggunakan AES-256

3 Hasil dan pembahasan

3.1 Pengujian Waktu Proses Enkripsi dan Dekripsi

Pengujian ini dilakukan untuk mengukur kecepatan proses enkripsi dan dekripsi dari ketiga jenis ukuran kunci AES yaitu 128 bit, 192 bit, dan 256 bit. Pengujian dilakukan terhadap empat jenis file dokumen dengan format penyimpanan *.pdf, *.doc, *.ppt dan *.xls dengan ukuran yang berbeda. File hasil enkripsi menghasilkan file dengan ekstensi *.txt. Pengujian ini dilakukan dengan menggunakan masing-masing jenis file yang diuji dengan tiga jenis kunci AES yang berbeda. Hasil pengujian waktu proses enkripsi dan dekripsi ditunjukkan pada Tabel 4 dan 5

Berdasarkan Tabel 4 dan 5 kecepatan waktu enkripsi dan dekripsi tidak dipengaruhi oleh jenis file, namun berdasakan ukuran filenya. Semakin besar ukuran file asli (plainteks) maka semakin lama waktu prosesnya. Selain itu semakin besar ukuran kuncinya, maka waktu untuk proses enkripsi dan dekripsi juga semakin lama. Rata-rata waktu proses untuk enkripsi AES-128, AES-192, dan AES-256 berturut-turut adalah 8,94 detik, 8,95 detik, dan 9,06 detik. Sedangkan rata-rata waktu proses untuk dekripsi AES-128, AES-192, dan AES-256

Tabel 4 Analisis waktu enkripsi berdasarkan jenis file dokumen

Jenis File	Nama File	Ukuran File (Kb)	Waktu Enkripsi (detik)		
			AES -128	AES-192	AES-256
*.doc	AES dalam bhasa java.doc	286	1,32	1,16	1,26
	WORD.doc	1988	8,34	8,39	9,05
	REVISI26LAP KP.doc	7955	33,30	33,30	33,90
*.pdf	TUGAS 3 RESUME revisi.pdf	753	3,13	3,44	3,25
	la-tahzan.pdf	1067	4,53	4,55	4,67
	buku-tahsin-kelas-x.pdf	4931	21,20	21,01	20,60
*.ppt	Representasi Data.pptx	301	1,35	1,27	1,29
	PENGANTAR KOMPUTER DAN IT.pptx	1601	6,86	7,00	6,84
	powerpoint.pptx	4259	18,10	18,05	18,50
*.xls	REALISASI.xlsx	13	0,08	0,09	0,13
	pkm nyata.xlsx	37	0,15	0,17	0,20
		Rata-rata	8,94	8,95	9,06

Tabel 5 Analisis waktu dekripsi berdasarkan jenis file dokumen

Jenis File	Nama File	Ukuran File (Kb)	Waktu Dekripsi (detik)		
			AES -128	AES-192	AES-256
*.doc	AES dalam bhasa java.doc	286	1,78	1,75	1,78
	WORD.doc	1988	11,70	12,00	12,47
	REVISI26LAP KP.doc	7955	46,10	47,30	48,20
*.pdf	TUGAS 3 RESUME revisi.pdf	753	4,42	4,44	4,65
	la-tahzan.pdf	1067	6,27	6,53	6,45
	buku-tahsin-kelas-x.pdf	4931	28,30	29,40	30,60
*.ppt	Representasi Data.pptx	301	1,80	1,67	1,80
	PENGANTAR KOMPUTER DAN IT.pptx	1601	9,52	9,65	9,50
	powerpoint.pptx	4259	24,70	25,00	25,70
*.xls	REALISASI.xlsx	13	0,08	0,09	0,13
	pkm nyata.xlsx	37	0,26	0,29	0,28
		Rata-rata	12,27	12,56	12,87

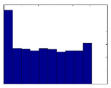
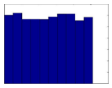
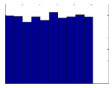
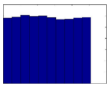
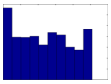
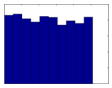
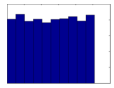
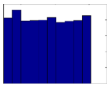
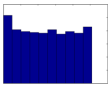
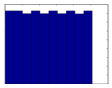
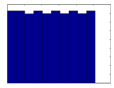
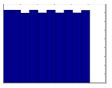
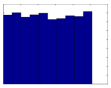
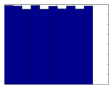
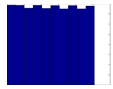
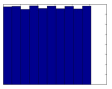
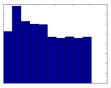
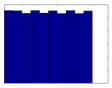
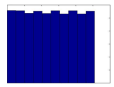
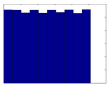
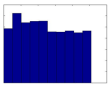
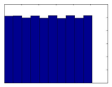
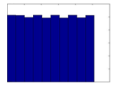
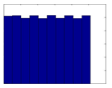
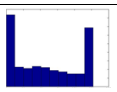
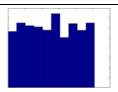
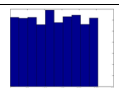
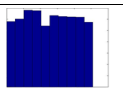
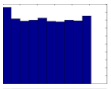
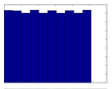
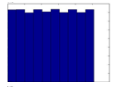
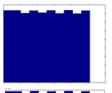
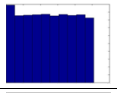
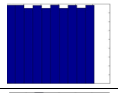
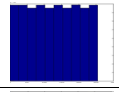
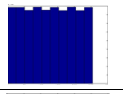
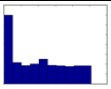
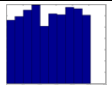
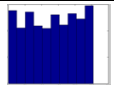
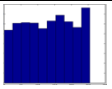
adalah berturut-tuurut adalah 12,27 detik, 12,56 detik, dan 12,87 detik. Berdasarkan waktu proses enkripsi dan dekripsi dapat disimpulkan AES-128 adalah yang paling cepat, sedangkan AES-256 adalah yang paling lambat. Hal ini disebabkan panjang kunci untuk mengamankan data sangat berpengaruh pada waktu proses.

3.2 Analisis Histogram

Pengujian menggunakan analisis histogram digunakan untuk melihat seberapa aman algoritme yang diimplementasikan untuk enkripsi file dokumen terhadap *statistical attack*. Serangan ini menggunakan informasi yang berasal dari histogram cipherteks berdasarkan frekuensi kemunculan karakter pada file dokumen. Jika nilai histogram cipherteks memiliki distribusi kemunculan karakter yang hampir seragam, serta memiliki perbedaan yang signifikan terhadap histogram plainteks, maka dapat dikatakan cipherteks cukup aman [15]. Hal ini

disebabkan tidak terdapat karakter pada cipherteks yang menonjol, Sehingga, *cryptanalyst* tidak akan mendapatkan petunjuk dari histogram cipherteks untuk melakukan serangan menggunakan metode *statistical attack*.

Berikut hasil uji histogram dari data file dokumen yang diujikan dengan panjang kunci AES seperti ditampilkan pada Gambar 6.

Nama File	Format Penyimpanan File	Plainteks	Cipherteks		
			AES -128	AES-192	AES-256
AES dalam bhasa java.doc	*.doc				
WORD.docx					
REVISI26LAP KP.doc					
TUGAS 3 RESUME revisi.pdf	*.pdf				
la-tahzan.pdf					
buku-tahsin-kelas-x.pdf					
Representasi Data.pptx	*.ppt				
PENGANTAR KOMPUTER DAN IT.pptx					
powerpoint.pptx					
REALISASI.xlsx	*.xls				
pkm nyata.xlsx					

Gambar 6 Histogram plainteks dan cipherteks

Berdasarkan Gambar 6 terlihat histogram cipherteks yang dihasilkan dari algoritme AES-128, AES-192, dan AES-256 terlihat rata. Hal ini menunjukkan bahwa algoritme AES dengan menggunakan panjang kunci yang terpendek yaitu 128 byte masih cukup aman terhadap *statistical attack*, terlihat histogram untuk setiap data uji yang relatif rata.

3.3 Pengujian Sistem

Tahap ini digunakan untuk menguji aplikasi berbasis Android untuk keamanan dokumen menggunakan algoritme AES yang telah dibangun. Pengujian dilakukan dengan menggu-

nakan metode *blackbox testing*, yaitu pengujian perangkat lunak yang berfokus pada sisi fungsionalitas aplikasi yang dikembangkan khususnya pada input dan output aplikasi.

Berdasarkan hasil pengujian fungsionalitas aplikasi yang dibangun dapat disimpulkan kelebihan dan kekurangan dari aplikasi tersebut, yaitu :

1. Kelebihan aplikasi yang dikembangkan:
   a. Proses enkripsi dan dekripsi tidak membutuhkan waktu lama. Hasil pengujian terjadap waktu proses enkripsi rata-rata kurang dari 1 menit dan waktu dekripsi juga tidak jauh berbeda walaupun lebih lama dari proses enkripsi.
   b. Kunci atau *password* yang digunakan tidak melibatkan user yaitu dibuat secara random agar lebih aman. Sehingga user tidak perlu mengingat *password* dan memasukkan kunci pada saat enkripsi maupun dekripsi. Kunci akan di *generate* pada saat memilih jenis kunci pada saat enkripsi file.
   c. Kunci hanya digunakan untuk satu kali proses enkripsi, sehingga lebih aman.
2. Kekurangan aplikasi yang dikembangkan adalah aplikasi hanya bisa digunakan untuk mengenkripsi file dokumen dan tidak dapat digunakan untuk data file gambar.

4 Kesimpulan dan saran

Penelitian yang dilakukan untuk mengamankan file dokumen menggunakan algoritme AES pada aplikasi berbasis android, dapat ditarik kesimpulan sebagai berikut :

1. Berdasarkan uji *black box* aplikasi yang dirancang sudah dapat berjalan dengan baik sesuai dengan rancangan yang dikehendaki.
2. Aplikasi yang dibangun dapat dimanfaatkan untuk proses enkripsi dan dekripsi file dokumen dengan format penyimpanan *.pdf, *.doc, *.ppt dan *.xls, menggunakan algoritme AES-128, AES-192 dan AES-256, sehingga file asli (plainteks) tidak dapat dibaca atau dimengerti lagi.
3. Analisis histogram hasil cipherteks menampilkan histogram yang relatif rata untuk algoritme AES-128, AES-192 dan AES-256, sehingga terbukti metode AES aman terhadap serangan *statistical attack*.
4. Kecepatan waktu proses enkripsi dan dekripsi dipengaruhi oleh ukuran file dokumen dan ukuran panjang kunci AES.
5. Waktu rata-rata proses enkripsi dan dekripsi paling cepat menggunakan panjang kunci AES-128 bit, yaitu $8,94$ detik dan $12,27$ detik, sedangkan rata-rata proses enkripsi dan dekripsi menggunakan panjang kunci AES-256 bit paling lama, yaitu $9,06$ dan $12,87$ detik.
6. Kecepatan proses untuk enkripsi dan dekripsi file dokumen dengan ukuran file terbesar dan kunci AES terpanjang rata-rata cukup cepat yaitu kurang dari 1 menit, sehingga aplikasi ini dapat diimplementasikan pada perangkat *mobile*.

Pustaka

1 E. Setyaningsih and R. Wardoyo, "Review of Image Compression and Encryption Techniques," *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 2, pp. 83–94, 2017.

2 R. Primartha, "Penerapan Enkripsi dan Dekripsi File menggunakan Algoritma Advanced Encryption Standard (AES)," *Journal of Research in Computer Science and Applications*, vol. 2, no. 1, pp. 13–18, 2013.

3 F. N. Pabokory, I. F. Astuti, and A. H. Kridalaksana, "Implementasi Kriptografi Pengamanan Data Pada Pesan Teks, Isi File Dokumen, Dan File Dokumen Menggunakan Algoritma Advanced Encryption Standard," *Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer*, vol. 10, no. 1, p. 20, jun 2016.

4 F. A. Sianturi, "Perancangan Aplikasi Pengamanan Data Dengan Kriptografi Advanced Encryption Standard (AES)," *Pelita Informatika Budi Darma*, vol. 4, no. 1, pp. 42–46, 2013.

5 O. L. Suryo, F. Mahbub, and M. Mirawati, "Pengamanan File DOCX Menerapkan Algoritma International Data Encryption Standart," in *Seminar Nasional Teknologi Komputer & Sains (SAINTEKS)*, 2020, pp. 438–446.

6 L. D. Singh and K. M. Singh, "Implementation of Text Encryption using Elliptic Curve Cryptography," *Procedia - Procedia Computer Science*, vol. 54, no. 1, pp. 73–82, 2015.

7 K. Mahmood, H. Naqvi, S. A. Chaudhry, and S. Kumari, "An elliptic curve cryptography based lightweight authentication scheme for smart grid communication," *Future Generation Computer Systems*, vol. 81, pp. 557–565, 2018.

8 K. Shafinah and M. M. Ikram, "File Security based on Pretty Good Privacy (PGP) Concept," *Computer and Information Science*, vol. 4, no. 4, pp. 10–28, 2014.

9 E. Agrawal and P. R. Pal, "A Secure and Fast Approach for Encryption and Decryption of Message Communication," *International Journal of Engineering Science and Computing*, vol. 7, no. 5, pp. 11 481–11 485, 2017.

10 M. Dharmawan, Eka Adhitya , Erni Yudaningtyas, "Perlindungan Web pada Login Sistem Menggunakan Algoritma Rijndael," *Eeccis*, vol. 7, no. 1, pp. 77–84, 2013.

11 E. Sularso, W. S. Rahardjo, and Y. Lukito, "Implementasi Algoritma Rijndael 128 pada Aplikasi Chatting Berbasis HTML5 websocket," *Jurnal Informatika*, vol. 10, no. 2, pp. 66–79, jan 2015.

12 A. F. Ramdhansya, E. Ariyanto, and H. H. Nuha, "Implementasi Advanced Encryption Standard (Aes) Pada Sistem Kunci Elektronik Kendaraan Berbasis Sistem Operasi Android Dan Mikrokontroler Arduino," in *Seminar Nasional Informatika*, 2014, pp. 92–98.

13 S. Saefudin and S. Syamsudin, "Aplikasi Enkripsi Pesan Teks Dengan Metode Advanced Encryption Standard Pada Ponsel Berbasis Android," *JSiI (Jurnal Sistem Informasi)*, vol. 4, pp. 29–31, oct 2017.

14 A. Muzakir, "Implementasi Teknik Steganografi dengan Kriptografi Kunci Private AES untuk Keamanan File Gambar Berbasis Android," in *Seminar Nasional Teknologi Informasi dan Multimedia 2016*. Yogyakarta: STIMIK AMIKOM Yogyakarta, 2016, pp. 43–48.

15 R. Munir, "Analisis Keamanan Algoritma Enkripsi Citra Digital menggunakan Kombinasi Dua Chaos Map dan Penerapan Teknik Selektif," *JUTI: Jurnal Ilmiah Teknologi Informasi*, vol. 10, no. 2, p. 89, jul 2012.