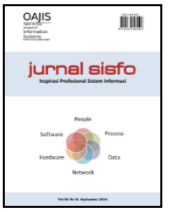
OAJIS
Open Access
Journal of
Information
Systems
is.its.ac.id/pubs/oajis/

jurnal sisfo

Jurnal Sisfo Vol. 07 No. 01 (2017) 15–26



Pengembangan Aplikasi Pengaman Berkas Menggunakan Algoritma *Blowfish*

Aldo Adrian, Silvester Dian Handy Permana*

Program Studi Teknik Informatika Universitas Trilogi

Abstract

Mobile phone provides accessibility of information to its user. Mobile phone with Android operating system have been dominated the market world-wide but is lack of security. The security flaw which was found is that this peticular mobile phone is susceptible against malware and other threat such as hacking. Because of this issue, this research goal is to develop an application that capable of secure any information in form of file which is stored inside the user's mobile phone. This application is using Blowfish algorithm that popular because of its reliability and minimum time-rate consumption in its processes. This peticular application is implement in mobile phone that using Kitkat version of Android operating system and capable of securing any file format which decresing Android's security issue.

Keywords: Android, Android Kitkat, Blowfish Algorithm, Mobile Phone, Security

Abstrak

Ponsel mampu memberikan keleluasaan dalam mengakses informasi kepada penggunanya. Ponsel dengan sistem operasi Android sedang menguasai pasar dengan peminatan terbanyak tetapi memiliki keamanan yang rendah. Celah keamanan yang ditemukan pada ponsel dengan sistem operasi Android adalah kerentannya terhadap malware dan ancaman lain seperti penyadapan. Oleh karena isu keamanan yang terdapat pada ponsel dengan sistem operasi Android, maka penelitian ini dibuat dengan tujuan menciptakan suatu aplikasi yang dapat mengamankan segala informasi dalam bentuk berkas yang tersimpan dalam ponsel pengguna. Aplikasi ini memanfaatkan algoritma Blowfish yang populer dengan keandalan dan kecepatan dalam proses pengamanannya. Aplikasi ini berjalan di ponsel dengan sistem operasi Android versi Kitkat dan mampu mengamankan segala format berkas sehingga meminimalisir isu keamanan Android.

Kata kunci: Algoritma *Blowfish*, Android, Android Kitkat, Keamanan, Ponsel

Histori Artikel : Disubmit 14 Juni 2017; Diterima 20 Juli 2017; Tersedia online 29 September 2017

*Corresponding Author
Email address: handy@trilogi.ac.id (Silvester Dian Handy Permana)

1. Pendahuluan

Teknologi informasi yang didasari oleh proses komputasi adalah awal mula dari PC (*Personal Computer*) dan bentuk-bentuk teknologi lainnya yang ada sekarang, termasuk ponsel. Faktanya, peminatan PC pada tiga tahun terakhir ini disusul oleh peminatan akan perangkat ponsel [1].

Ponsel merangkum kemampuan-kemampuan PC ke dalam bentuk yang lebih kecil sehingga pengguna dapat mengakses informasi tanpa sulit membawa perangkat yang besar. Pengguna diberikan keleluasaan oleh ponsel selain kemudahan akses akan informasi, yaitu mengunggah dan mengunduh informasi yang diinginkan. Hal ini membuat ponsel tampak lebih unggul daripada PC. Namun, ada masalah lain yang ditimbulkan karena kemudahan akses ini, yakni masalah keamanan. "*Using a desktop or laptop PC without security software has become unthinkable. With mobile phones, this sense of respondsibility has not yet reached the majority of users, even though important personal data, personal photos and even company data may be stored on smartphones*" [2].

Ponsel memiliki berbagai macam sistem operasi, sedangkan produk dari vendor Samsung dengan sistem operasi Android menguasai pasaran secara global [3]. Sayangnya, penelitian dari Universitas Cambridge menunjukkan secara statistik tingkat keamanan dari sistem operasi Android yang sangatlah rendah [4].

Penelitian Thomas *et al* [4] melakukan tolok ukur keamanan berdasarkan kerentanan sistem operasi Android dari serangan *malware*. Jika ingin diperhatikan lebih jauh lagi, tidak hanya *malware* yang menjadi ancaman keamanan informasi. Dikutip langsung dari Boyle *et al* [5]; "*The internet has given firms access to billions of customers and other business partners, but it has also given criminals access to hundreds of billions of corporations and individuals*." Kutipan Boyle *et al* [5] menyinggung fakta bahwa para kriminal seperti penyadapan dapat menyerang individu, bahkan pemerintah juga tidak luput dari penyerangan. Maka dari itu, dapat ditarik kesimpulan bahwa dalam konteks individu di atas turut merujuk kepada pengguna perangkat ponsel. Hal ini berarti bahwa kriminal dapat menyerang para pengguna ponsel dengan akses yang dimilikinya, sedangkan sistem operasi Android memiliki keamanan yang sangat rendah [4].

Tingginya persentase isu keamanan Android yang dapat mengancam privasi penggunanya membuat peneliti memutuskan untuk membuat suatu aplikasi keamanan. Aplikasi ini berlaku sebagai pengaman berkas dengan keamanan yang andal. Peneliti memilih untuk membangun aplikasi ini untuk sistem operasi Android versi Kitkat. Pemilihan versi sistem operasi Android Kitkat ini dikarenakan terinstalnya versi ini di pasaran memiliki persentase yang besar [6]. Aplikasi pengaman yang akan dibuat ini menggunakan algoritma pengenkripsian *Blowfish* sebagai pengamannya.

"*Blowfish provides a good encryption rate in software and no effective cryptanalysis of it has been found to date*" [7]. Algoritma *Blowfish* adalah algoritma kriptografi simetris yang cukup dengan menggunakan satu kunci yang sama dalam proses penyandian (enkripsi) dan proses pengembalian (dekripsi). Penggunaan satu kunci yang dibangun oleh fungsi spesifik dapat menyediakan keamanan yang andal sekaligus memakai ruang penyimpanan yang tidak besar dan hanya membutuhkan waktu proses yang singkat sehingga membuat algoritma ini cocok diimplementasikan pada ponsel [8].

Aplikasi yang dibuat mampu mengamankan berkas dalam segala format. Pihak lain yang dapat mencuri berkas yang telah diamankan tidak akan mampu membaca isi berkas tersebut. Berkas yang telah diamankan hanya dapat terbaca jika pengamanan dari berkas tersebut telah dilepaskan oleh aplikasi dari ponsel yang sama. Oleh karena itu, hanya pengguna ponsel yang dapat membuka pengamanan tersebut sehingga permasalahan privasi akan informasi yang dimiliki oleh pengguna ponsel dengan sistem operasi Android akan terpenuhi.

2. Tinjauan Pustaka/Penelitian Sebelumnya

Untuk membuktikan kebaruan penelitian ini, peneliti akan melakukan perbandingan antara penelitian yang dibuat dengan penelitian lain yang sudah ada. Perbandingan ini lebih menekankan kepada persamaan dan perbedaan antara satu penelitian dengan penelitian yang lain, juga mengemukakan alasan pengambilan algoritma yang dianut.

Pemilihan ponsel sebagai perangkat dimana aplikasi pengaman berjalan sudah dijelaskan di Latar Belakang Masalah pada Bab 1, tetapi untuk pemilihan algoritma belum begitu dijelaskan secara rinci. Alasan pengambilan algoritma *Blowfish* sebagai algoritma enkripsi adalah karena algoritma ini hanya membutuhkan penggunaan memori proses yang relatif kecil yaitu minimal sebesar ±5 KB (4.1 KB untuk komponen dasar, belum dengan berkas yang ingin dienkripsi) [10]. Tidak hanya penggunaan memori proses yang kecil, *Blowfish* juga hanya menggunakan operasi-operasi mudah seperti XOR dan modulo yang tidak kompleks. Kedua kelebihan dari *Blowfish* ini membuat algoritma ini cocok diimplementasikan pada ponsel yang memiliki kapabilitas yang lebih minim dibandingkan PC.

Memori proses *Blowfish* dapat dikategorikan rendah dan dengan hal ini membuat *Blowfish* menjadi algoritma simetris tercepat dibandingkan dengan beberapa algoritma simetris lainnya. Tabel 1 adalah hasil dari penelitian Sharma dan Bisht [9] yang membandingkan kecepatan dari beberapa algoritma simetris; Blowfish, Rijndael 128 bit, Rijndael 192 bit, (3DES)DES-XEX3, dan (3DES)DES-EDE3. Dengan besar 256 MB data yang ingin dienkripsi, *Blowfish* jauh lebih cepat daripada algoritma-algoritma lain dengan satuan waktu sebesar 3.976 detik saja.

Tabel 1. Perbandingan Kecepatan Algoritma Simetris [9]

Algorithm	Megabytes Processed	Time Taken
Blowfish	256	3.976
Rijndael (128-bit key)	256	4.196
Rijndael (192-bit key)	256	4.817
(3DES) DES-XEX3	128	6.159
(3DES) DES-EDE3	64	6.499

Dari segi keamanan, *Blowfish* masih belum ditemukan suatu metode khusus untuk diretas [7]. Hal ini membuat peneliti berani mengambil *Blowfish* sebagai algoritma enkripsi karena keamanan, kecepatan, dan minimnya penggunaan memori.

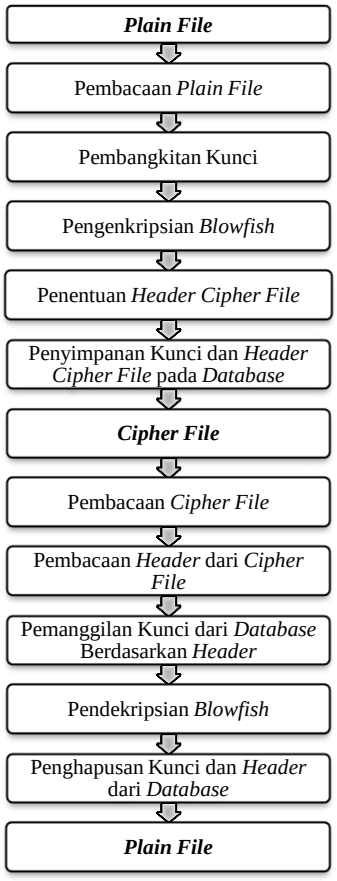
Defni *et al* [11] dan Irawan [12] memiliki kesamaan yang cukup besar dengan penelitian yang dibuat, yaitu bertujuan untuk membuat aplikasi pengaman di ponsel dengan sistem operasi Android. Versi Android yang diambil oleh masing-masing penelitian saling berbeda satu dengan yang lain. Defni menggunakan Android versi 2.2 (*Froyo*), Irawan menggunakan versi 2.3 (*Gingerbread*), sedangkan penelitian yang dibuat menggunakan Android versi lebih baru daripada kedua penelitian di atas, yaitu 4.4 (*Kitkat*).

Perbedaan lain dapat ditemukan pada obyek pengemanan pada masing-masing penelitian. Defni dan Irawan hanya mengamankan obyek bertipe teks saja, sedangkan penelitian ini mencangkup pengamanan segala format berkas. Algoritma pengaman yang digunakan turut menjadi perbedaan yang kontras. Untuk algoritma pengaman pada Irawan dan penelitian yang dibuat sama-sama menggunakan algoritma kriptografi simetris, sedangkan Defni menggunakan algoritma kriptografi asimetris. Defni menggunakan algoritma Luc yang membutuhkan dua kunci yang berbeda, yaitu kunci privat dan kunci publik. Di lain pihak, Irawan menggunakan algoritma RC6 sedangkan penelitian yang dibuat ini menggunakan algoritma *Blowfish*.

Untuk metode keamanan yang berbeda pada penelitian lain yang juga berjalan di perangkat ponsel dengan sistem operasi Android salah satunya adalah Bucerzan *et al* [13]. Penelitian tersebut tidak menggunakan algoritma dari ranah kriptografi, justru menggunakan algoritma dari ranah steganografi, yaitu *Least Significant Bit* (LSB). Sebenarnya LSB menggunakan sedikit algoritma kriptografi dan menyatukannya dengan fungsi acak, sedangkan algoritma *Blowfish* yang digunakan pada penelitian yang dibuat ini sudah merangkum fungsi acak yang dimaksud oleh LSB. Perbedaan antara LSB dengan *Blowfish* adalah LSB yang digunakan oleh Bucerzan hanya sekadar menyisipkan obyek pengamanan kedalam suatu gambar setelah mengenkripsinya, sedangkan *Blowfish* hanya menggunakan enkripsi acak sekali dengan keamanan yang terjamin tanpa harus melakukan penyembunyian/penyisipan berkas.

3. Metodologi

Berikut adalah metodologi yang digunakan dalam penelitian ini:



Gambar 1 Alur Aplikasi.

Dari Gambar 1 dapat dijabarkan sebagai berikut:

1) *Plain File*. Bentuk awal berkas adalah berkas yang belum dienkripsi dan dipilih untuk memasuki proses enkripsi.

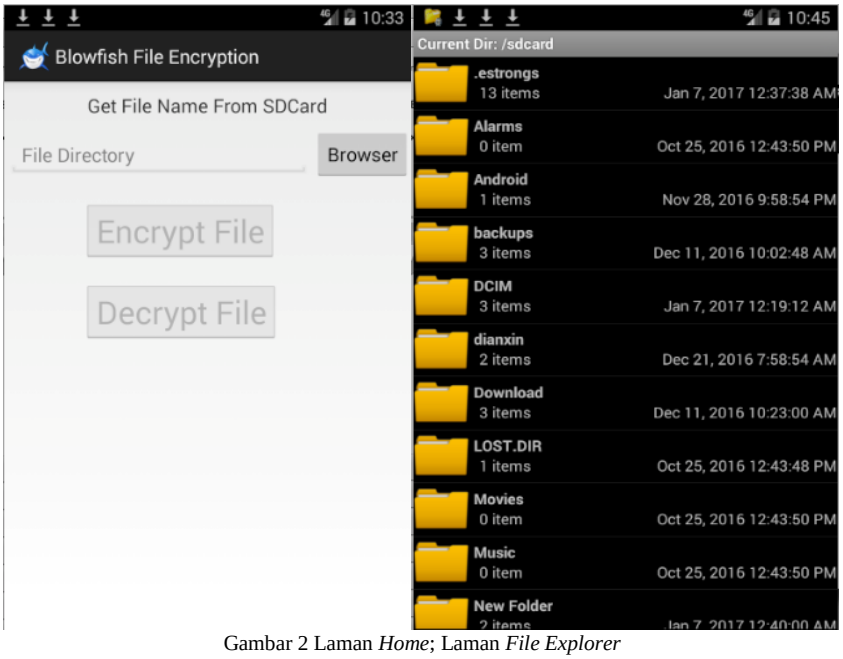
2) *Pembacaan Plain File*. Langkah ini memerlukan masukan berupa direktori dari *Plain File* yang hendak dienkripsi. Dari direktori tersebut, berkas akan dibaca dalam bentuk byte dan siap untuk diolah.
3) *Pembangkitan Kunci*. Tahap ini adalah tahap dimana kunci awalnya dibangkitkan dengan deret yang diambil secara acak. Dari deret kunci acak tersebut baru akan diproses untuk menjadi kunci dalam proses enkripsi algoritma *Blowfish* nantinya[14]. Kunci yang diperoleh akan berbeda pada setiap *Cipher File* (berkas yang terenkripsi). Panjang kunci maksimal adalah 72 byte.
4) *Pengenkripsian Blowfish*. Setelah Kunci dan *Plain File* tersedia dan dalam kondisi yang memenuhi syarat berikutnya akan dilakukan proses enkripsi algoritma *Blowfish*[14] dengan keluaran berupa *Cipher File*.
5) *Penentuan Header Cipher File*. Dari deret byte yang ada pada *Cipher File* akan ditetapkan 5 byte awal sebagai Header. Fungsi *Header* ini dapat dilihat saat proses dekripsi yang membutuhkan kunci. Untuk menandai *Cipher File* mana yang memiliki Kunci spesifik dalam *Database*, dibutuhkan pengecekan spesifik pada *Cipher File*. Oleh karena itu, *Header* ini dibuat agar menjadi obyek pengecekan di dalam *Database*.
6) *Penyimpanan Kunci dan Header Cipher File pada Database*. Kunci dan *Header* dari *Cipher File* akan disimpan kedalam *Database*. Langkah ini adalah langkah terakhir di dalam proses enkripsi.
7) *Cipher File*. Ini adalah berkas hasil dari proses enkripsi yang sudah dilewati dan berkas ini akan menggantikan berkas mentah (*Plain File*).
8) *Pembacaan Cipher File*. Ini adalah awal proses dekripsi. Sama seperti langkah nomor 1, masukan untuk tahap ini adalah direktori dari *Cipher File*. Berikutnya isi dari *Cipher File* akan dibaca oleh program dalam format byte.
9) *Pembacaan Header dari Cipher File*. Deret byte *Cipher File* akan dibaca mulai 5 byte pertama sebagai *Header*-nya.
10) *Pemanggilan Kunci dari Database Berdasarkan Header*. Pada tahap ini akan dilakukan pencarian pada *Database* untuk mencari Kunci dengan *Header* yang sesuai dengan *Header* dari *Cipher File* yang baru didapatkan. Hasil dari proses ini adalah Kunci yang akan digunakan pada proses berikutnya.
11) *Pendekripsian Blowfish*. Proses dekripsi ini membutuhkan Kunci dan *Cipher File* untuk diproses[14]. Hasil dari proses ini adalah *Plain File*.
12) *Penghapusan Kunci dan Header dari Database*. Setelah *Plain File* didapatkan, maka Kunci dan *Header* yang tersimpan pada *Database* sudah tidak lagi dibutuhkan dan akan dihapus agar menghemat memori yang digunakan *Database*. Langkah ini adalah langkah terakhir dari proses dekripsi.
13) *Plain File*. Berkas yang telah didekripsi kembali menjadi berkas mentah yang kembali dapat memasuki proses enkripsi jika diinginkan. Berkas ini dapat kembali dibuka dan memiliki pesan awal yang tidak berubah dan dengan begini proses enkripsi dan dekripsi berjalan dengan lancar.

Aplikasi membutuhkan suatu *Database* untuk menyimpan kunci dan indeks untuk kunci tersebut. *Database* yang dibuat memiliki suatu tabel yang bernama *Keys* dengan isi tabel seperti yang ditunjukkan oleh Tabel 2.

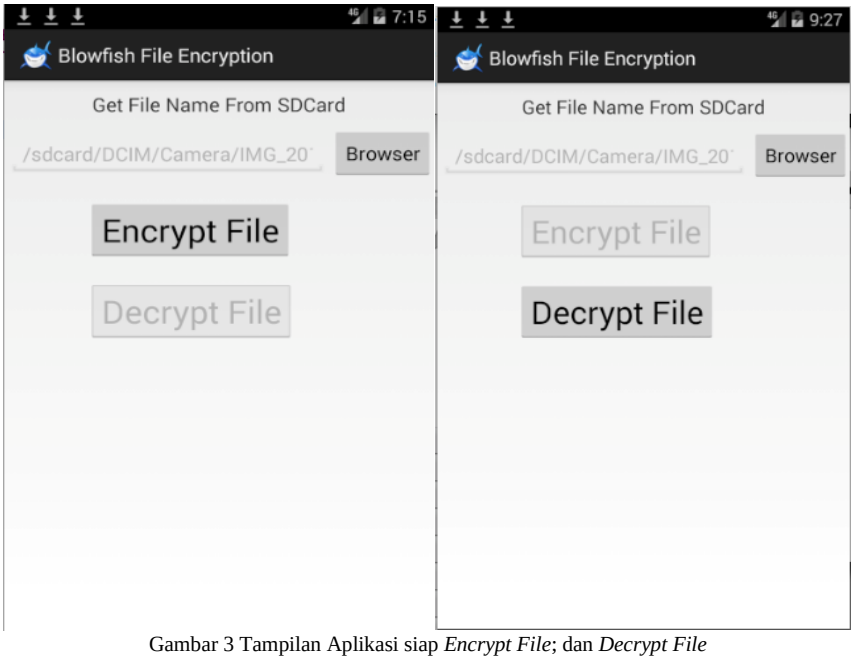
Tabel 1. Tabel *Keys* pada *Database*

Nama Field	Tipe Data	Keterangan
_index	*Text*	Kolom ini adalah *Primary Key* dan berisikan *Header* dari berkas yang sudah terenkripsi (*Cipher File*)
_key	*Blob*	Kunci dari berkas terenkripsi tersebut.

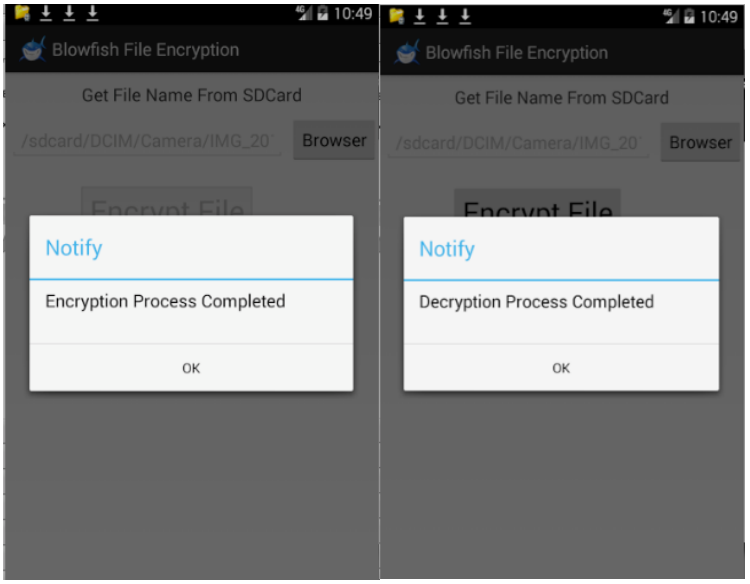
Terdapat dua laman pada Aplikasi, yaitu Laman *Home* dan Laman *File Explorer* (Gambar 2). Saat tombol *Browser* pada Laman *Home* ditekan maka Laman *File Explorer* akan terbuka dan *User* dapat memilih berkas yang hendak diproses. Setelah berkas terpilih maka Laman *File Explorer* akan tertutup dan kembali menampilkan Laman *Home* yang siap untuk melakukan proses enkripsi atau dekripsi (Gambar 3). Setelah proses enkripsi atau dekripsi selesai dilakukan, aplikasi akan menunjukkan pemberitahuan seperti pada Gambar 4.



Gambar 2 Laman *Home*; Laman *File Explorer*



Gambar 3 Tampilan Aplikasi siap *Encrypt File*; dan *Decrypt File*



Gambar 4 Pemberitahuan setelah selesai *Encrypt File* dan *Decrypt File*

Rencana pengujian aplikasi dibagi berdasarkan lamannya. Rencana pengujian tersebut dapat dilihat pada Tabel 3.

Tabel 2. Rencana Pengujian Aplikasi

Laman yang Diuji	Keterangan	Jenis Uji
Home	*User* dapat melakukan *Encrypt File*, *Decrypt File*, dan membuka laman *File Explorer*	*Black Box*
File Explorer	*User* mencari File yang hendak diproses	*Black Box*

Dari rencana pada Tabel 3, dapat dipecah menjadi dua tabel, yaitu untuk pengujian Laman *Home* dan *File Explorer*. Pengujian Laman *Home* padat dilihat pada Tabel 4, sedangkan pengujian Laman *File Explorer* dapat dilihat pada Tabel 5.

Tabel 3. Pengujian Laman Home

Hasil Uji		
Masukan	**Hasil yang diharapkan**	**Kesimpulan uji**
Tombol *Browser*	Buka laman *File Explorer*	Andal
Tombol *Encrypt File*	Melakukan proses enkripsi berkas	Andal
Tombol *Decrypt File*	Melakukan proses dekripsi berkas	Andal

Tabel 4. Pengujian Laman File Explorer

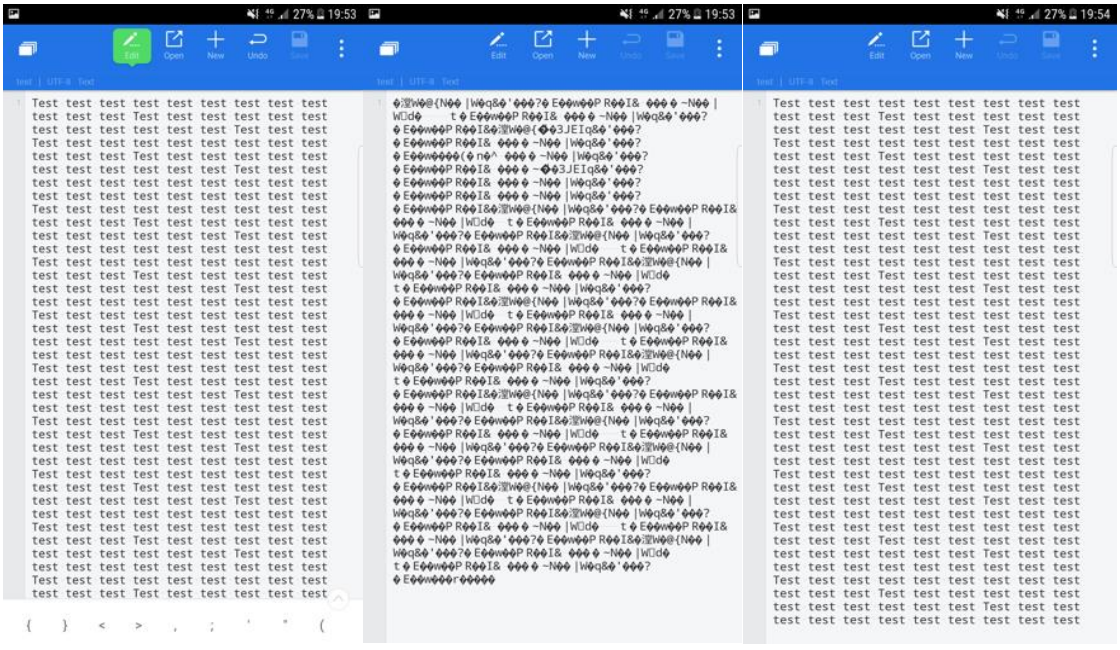
Hasil Uji		
Masukan	**Hasil yang diharapkan**	**Kesimpulan uji**
Pilih item *Folder*	Buka *Folder* dan menampilkan isinya	Andal
Pilih item *File*	Tutup Laman *File Explorer* dan mengisi teks obyek *Edit Text* pada Laman *Home* dengan direktori dari berkas yang dipilih	Andal

Dari hasil uji coba yang dilakukan, fungsionalitas dari Aplikasi Pengaman Berkas Menggunakan Algoritma Blowfish telah lolos uji. Aplikasi ini menjalankan fungsionalitasnya sesuai dengan hasil yang diharapkan, sehingga aplikasi dapat dinyatakan sudah layak pakai.

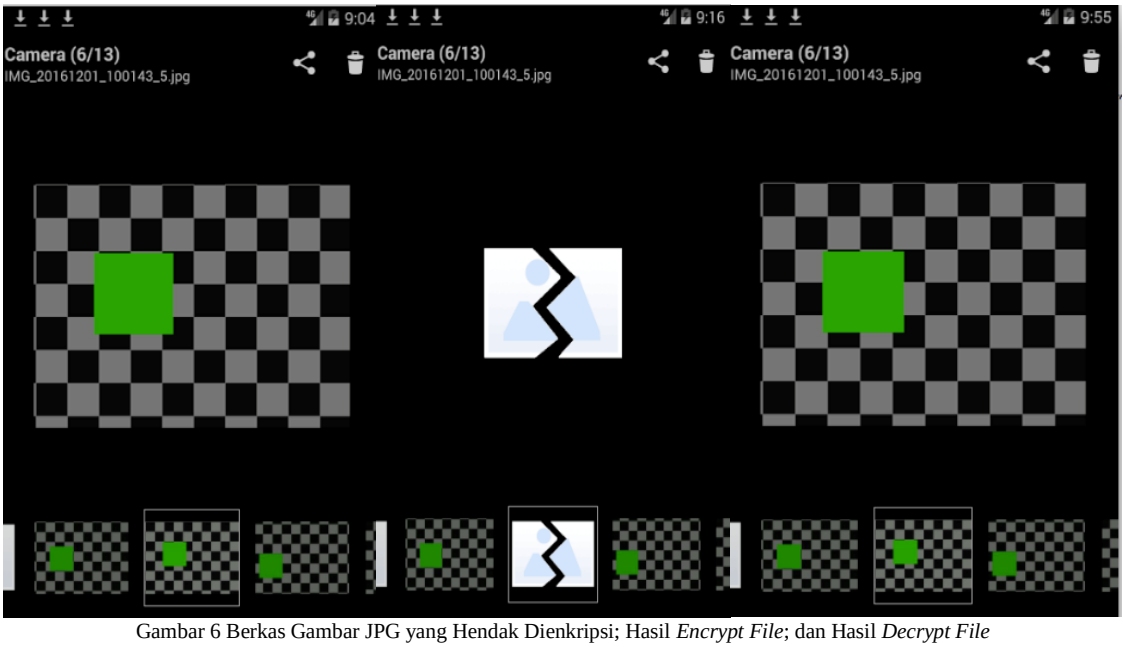
Aplikasi yang telah dibuat selanjutnya akan digunakan untuk mengamankan berbagai jenis format berkas. Pada Gambar 5 (kiri), aplikasi dijalankan untuk mengenkripsi berkas teks berformat .txt dengan menekan

tombol *Encrypt File* pada Laman *Home* setelah berkas dipilih terlebih dahulu dari Laman *File Explorer*. Setelah aplikasi selesai mengenkripsi berkas teks tersebut, maka berkas akan kembali dibuka untuk melihat apakah isi berkas telah berhasil diamankan atau tidak. Gambar 5 (tengah) menunjukkan bahwa isi berkas berhasil terenkripsi dan tidak dapat terbaca seperti sebelum dilakukannya proses enkripsi. Berkas yang telah terenkripsi tersebut dinyatakan aman dan tidak akan bermakna apa pun jika dicuri. Hal ini dikarenakan berkas tidak lagi berisi informasi yang diinginkan. Jika *User* ingin mengembalikan isi berkas agar dapat dibaca, maka kembali buka aplikasi, pilih berkas yang sama, lalu tekan tombol *Decrypt File*. Setelah proses dekripsi selesai, berkas .txt yang dipilih dapat kembali dibuka dan dibaca seperti yang ditunjukkan oleh Gambar 5 (kanan).

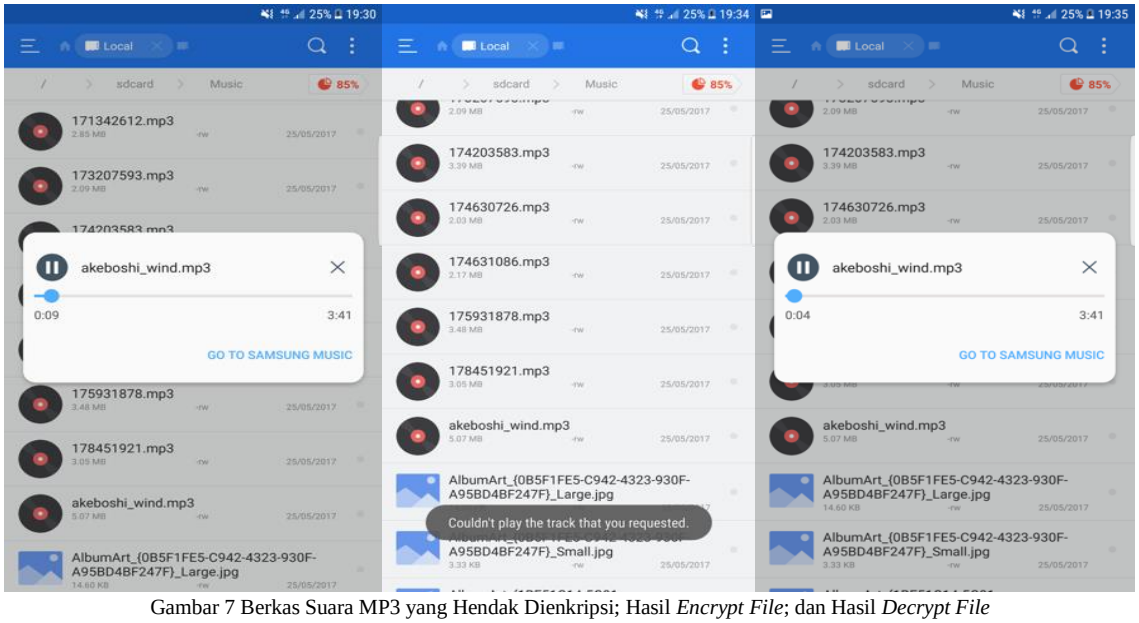
Tidak hanya berkas berformat .txt, aplikasi pengaman berkas ini juga digunakan untuk mengamankan format berkas lainnya. Pada Gambar 6 terdapat berkas berformat .jpg yang berhasil dienkripsi dan didekripsi. Pada Gambar 7 terdapat berkas .mp3. Gambar 8 menunjukkan berkas berformat .mp4 yang akan dienkripsi, hasil enkripsi, dan setelah didekripsi.



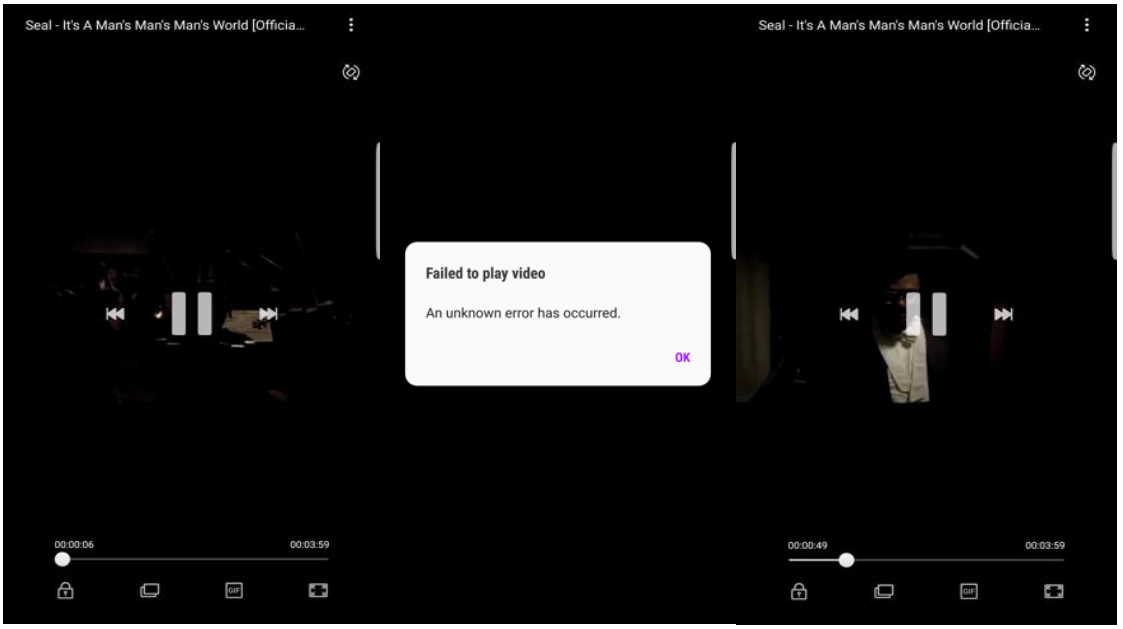
Gambar 5 Berkas Teks TXT yang Hendak Dienkripsi; Hasil *Encrypt File*; dan Hasil *Decrypt File*



Gambar 6 Berkas Gambar JPG yang Hendak Dienkripsi; Hasil *Encrypt File*; dan Hasil *Decrypt File*



Gambar 7 Berkas Suara MP3 yang Hendak Dienkripsi; Hasil *Encrypt File*; dan Hasil *Decrypt File*



Gambar 8 Berkas Video MP4 yang Hendak Dienkripsi; Hasil *Encrypt File*; dan Hasil *Decrypt File*

Format berkas lain yang sudah dicoba dan dapat diamankan oleh aplikasi ini adalah:

- Berkas kompresi .rar, .zip, dan .7z;
- Berkas *Office* .docx, pptx, xlsx, dan vsd;
- Berkas gambar .png dan .gif;
- Berkas multimedia .3gp, .pdf, dan .mkv;
- Berkas lainnya .apk, .java, dan .pkg.

5. Kesimpulan

5.1 Simpulan

Berdasarkan penelitian yang telah dilakukan mengenai pengembangan aplikasi pengaman berkas dapat diambil kesimpulan sebagai berikut:

1) Penelitian ini menghasilkan aplikasi pengaman berkas menggunakan algoritma *Blowfish* yang memiliki kemampuan untuk mengenkripsi segala jenis dan format berkas.
2) Fitur-fitur yang disediakan oleh aplikasi yang dibuat adalah enkripsi (*Encrypt File*) dan dekripsi (*Decrypt File*) menggunakan algoritma *Blowfish* yang sudah lolos tahap pengujian.

5.2 Saran

Dari penelitian yang telah dilakukan, penulis menyarankan beberapa hal untuk pengembangan dari aplikasi ini, yaitu:

1) Penambahan fitur dirasa perlu. Fitur yang disarankan untuk dibuat adalah kemampuan aplikasi untuk mengenkripsi beberapa File sekaligus.

2) Penambahan pengaman aplikasi. Ketika *User* membuka aplikasi ini *User* langsung dapat mengenkripsi dan mendekripsi berkas. Tidak ada jaminan jika *User* yang sedang mengoperasikan aplikasi adalah pemilik ponsel tersebut, maka dibutuhkan pengaman aplikasi yang mengecek otentikasi pengguna.

6. Daftar Rujukan

[1] Chaffey, D., 2016. *Mobile Marketing Statistics Compilation*. [Online]
Available at: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/. [Accessed 8 January 2017]

[2] AV-Comparatives Organization, 2013. *Mobile Security Review*. Unpublished.

[3] Statista, 2016. Vendors' Sales of Mobile Phone Sales to End Users Worldwide from 2010 to 2015 (in Million Units), by Quarter. [Online]
Available at: https://www.statista.com/statistics/263355/global-mobile-device-sales-by-vendor-since-1st-quarter-2008/. [Accessed 8 January 2017]

[4] Thomas, D. R., Beresford, A. R., & Rice, A., 2015. Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices. *Security Metrics for the Android Ecosystem*, New York, pp. 87-98.

[5] Boyle, R. J., & Panko, R. R., 2013. *Corporate Computer Security Third Edition. Pearson*.

[6] Richter, F., 2016. *The Difference Between iOS and Android*. [Online]
Available at: https://www.statista.com/chart/5930/adoption-of-ios-and-android-versions/. [Accessed 8 January 2017]

[7] Patil, S. D., 2013. IJRRE ST: International Journal of Research Review in Engineering Science and Technology. *Passwords Management using Blowfish Algorithm*. Vol. 2, pp. 48-52.

[8] Shrivastava, V. & Singh, G. IJCST. *Computer Trend with Security by RSA, DES and BLOWFISH Algorithm*. Vol. 4, 2013, pp. 618-620.

[9] Sharma, S. & Bisht, J. S., 2015. International Journal of Scientific Research in Network Security and Communication. *Performance Analysis of Data Encryption Algorithms*. Vol. 3, pp. 1-5.

[10] Kumar, S., 2016. *Blowfish Encryption Algorithm: [Explanation and Examples]*. [Online]
Available at: http://www.tips2secure.com/2016/02/Blowfish-encryption.html. [Accessed 8 January 2017]

[11] Defni & Rahmayun, I., 2014. Jurnal Momentum. *Enkripsi SMS (Short Message Service) Pada Telepon Selular Berbasis Android Dengan Metode RC6*. Vol. 16, pp. 63-73.

[12] Irawan, A. F., 2013. *Sistem Keamanan Pesan Pada Android Gingerbread (2.3.4) Dengan Algoritma Luc*. Jember: Unpublished.

[13] Bucerzan, D., Ratiu, C., & Manolescu, M. J., 2013. CCC Publication. *SmartSteg: A New Android Based Steganography Application*, Vol. 8, pp. 681-688.

[14] Valmik, N. K. & Kshirsagar, V. K. 2014. IOSR – Journal of Computer Engineering. *Blowfish Algorithm*. Vol. 16, pp. 80-83.

Halaman ini sengaja dikosongkan