# Pembuatan Aplikasi Penyimpanan Informasi Rahasia pada *Mobile Device* Android dengan Metode Kriptografi *Serpent Cipher*

Harry Timothy Tumalewa[1], Justinus Andjarwirawan[2], Andreas Handojo[3]
Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 ï 131 Surabaya 60236
Telp. (031) ï 2983455, Fax. (031) - 8417658
E-mail: formulasynchron@gmail.com[1], justin@petra.ac.id[2], handojo@petra.ac.id[3]

**ABSTRAK**: Keamanan informasi merupakan salah satu aspek penting dalam pemanfaatan teknologi informasi. Namun penyimpanan informasi pada tempat yang kurang tepat, misalnya pada sebuah catatan (baik yang merupakan catatan fisik ataupun *software*) seringkali dinilai tidak aman. Karena selama informasi yang disimpan masih dapat terbaca (*human readable*), maka tidak tertutup kemungkinan informasi tersebut suatu saat ditemukan ataupun dibongkar sehingga terbaca.

Aplikasi yang dibuat mampu menyimpan sekaligus mengamankan informasi rahasia dari penggunanya. Selain diamankan oleh *password*, aplikasi juga diamankan oleh *pseudorandom number* melalui fitur *shake*, sehingga keamanannya terjamin.

Berdasarkan hasil pengujian, aplikasi mampu mengimplementasikan fitur untuk meng-*edit* dan menghapus informasi yang tersimpan, mengirim informasi melalui SMS atau *e-mail*, melakukan enkripsi/dekripsi terhadap *file* pada *external storage device*, mengubah *password* yang tersimpan, melakukan *backup* ke Google Drive, dan melakukan *restore*.

**Kata Kunci**: Keamanan, Informasi Rahasia, *Serpent Cipher*, Google Drive, Android

***ABSTRACT:*** *Information security is one of the most important aspect of the use of information technology. However, storing information in a less appropriate place, for example on a note, is often considered as unsafe. Because as long as the stored information is still human readable, then it is still possible that the information will be discovered, or being disassembled one day.*

*This application is made capable of storing as well as securing confidential information of its users. Beside being secured by password, the application is also secured by a pseudorandom number through the shake feature, so safety is assured.*

*Based on test results, the application is able to implement the feature to edit and delete stored information, send information via* SMS *or e-mail, do the encryption / decryption to files on device's external storage, change the stored password, backup to* Google Drive*, and perform restore.*

**Keywords**: *Security*, *Confidential Information*, *Serpent Cipher*, Google Drive, Android

## 1. PENDAHULUAN

Dalam konteks teknologi informasi, keamanan merupakan salah satu elemen penting yang perlu diperhatikan. Salah satu tindak pengamanan yang tergolong paling banyak digunakan saat ini adalah penggunaan *password* untuk melindungi informasi. Saat ini, seorang pengguna internet tidak hanya memiliki dan menyimpan informasi dan hak akses pada satu atau dua situs saja, melainkan lebih. Dan seringkali seorang pengguna internet menggunakan *password* yang berbeda-beda pada setiap situsnya (alasan keamanan). Namun hal ini cukup beresiko, di mana *password* yang kombinasinya mudah memang lebih mudah diingat, tetapi lebih memudahkan *hacker* untuk membongkar *password* dengan menggunakan *dictionary attack*. Sedangkan *password* yang kombinasinya sukar, susah untuk diingat sehingga perlu disimpan.

*Mobile device* adalah salah satu alternatif tempat penyimpanan *password* ataupun informasi rahasia lainnya yang dipercaya cukup aman, karena selalu dibawa, atau setidaknya berada dalam pengawasan pemiliknya, sehingga lebih memudahkan apabila ingin mengakses informasi rahasia yang tersimpan di dalamnya. Namun resiko-resiko seperti adanya serangan *hacker*, orang lain yang meminjam, atau kehilangan itu selalu ada. Untuk itu, diperlukan adanya aplikasi yang dapat mengamankan informasi rahasia pengguna yang tersimpan di dalam *mobile device*.

Untuk membantu pengguna *mobile device* Android dalam menyimpan dan mengamankan informasi rahasia di dalam *mobile device*-nya, maka dibuatlah aplikasi penyimpanan informasi rahasia. Aplikasi ini dibuat pada *mobile device* Android dengan menggunakan metode kriptografi, yaitu *Serpent cipher*.

Pada aplikasi ini, pengguna diminta menyimpan sebuah *password* yang fungsinya untuk mengakses segala informasi yang tersimpan di dalam aplikasi. *Password* tersebut kemudian diamankan dengan *pseudorandom number* yang dihasilkan melalui *shake*. Aplikasi ini memungkinkan pengguna untuk menyimpan berbagai macam informasi, seperti informasi *login* suatu *website*, akun bank, kartu kredit ataupun catatan sederhana. Pengguna juga dapat mengamankan *file* yang terletak di *external storage* dengan fitur *file encryption*. Aplikasi ini juga memungkinkan pengguna untuk mengubah *password* apabila suatu saat diperlukan.

## 2. LANDASAN TEORI

### 2.1 Kriptografi

Kriptografi adalah ilmu dan seni untuk menjaga kerahasiaan informasi. Ada empat tujuan mendasar dari ilmu kriptografi yang merupakan aspek keamanan informasi, yaitu kerahasiaan, integritas data, autentikasi dan non-repudiasi. Kriptografi terdiri atas 2 proses, yaitu enkripsi dan dekripsi. Enkripsi adalah proses mengubah teks awal atau *plaintext* menjadi teks acak atau *ciphertext*, sedangkan dekripsi adalah proses mengubah *ciphertext* kembali menjadi *plaintext*. Baik proses enkripsi maupun dekripsi, keduanya membutuhkan sebuah kunci atau *key* yang bertindak sebagai pengacak ataupun penyusun kembali teks yang telah dienkripsi.

Berdasarkan alur pengolahan data, skema kriptografi dibedakan menjadi dua kelas, yaitu *block cipher* dan *stream cipher*.

#### 2.1.1 *Stream Cipher*

*Stream cipher* adalah teknik kriptografi di mana enkripsi dilakukan per satuan data (bukan per *block*). Setiap kali enkripsi telah dilakukan terhadap satu satuan data, kunci ikut diproses dalam suatu algoritma, menghasilkan kunci yang baru untuk digunakan dalam mengenkrispi satu satuan data selanjutnya.

#### 2.1.2 *Block Cipher*

*Block cipher* adalah teknik kriptografi di mana teks dibagi-bagi ke dalam beberapa kelompok teks yang disebut *block,* dengan panjang t, di mana tiap *block* dienkripsi dengan menggunakan kunci yang sama. Pada umumnya, *block cipher* memproses *plaintext* dengan *block* yang panjangnya 64 bit atau lebih, untuk mempersulit penyerang (*hacker*) dalam membongkar kunci. Salah satu contoh metode kriptografi yang merupakan *block cipher* adalah *Serpent cipher*.

*Serpent cipher* merupakan *block cipher* dengan ukuran *block* 128 bit yang terdiri atas 4 buah *word* yang masing-masing berukuran 32 bit, dan mempunyai *key* dengan ukuran 256 bit. Algoritma pada *Serpent cipher* dapat berjalan secepat DES (*Data Encryption Standard*), namun hasilnya lebih aman dibanding *tripple* DES. [1]

*Serpent cipher* mengenkripsi 128 bit *plaintext* P menjadi 128 bit *ciphertext* C dalam 32 ronde. [2] Apabila *key*, *plaintext* atau *ciphertext* mempunyai ukuran kurang dari yang seharusnya, maka akan dilakukan *padding* dengan menambahkan bit ñ1ò yang kemudian diteruskan dengan bit ñ0ò sampai jumlah bit terpenuhi.

*Serpent cipher* terdiri atas *Initial Permutation* (IP); 32 ronde, yang masing-masing terdiri atas operasi *key mixing*, melalui *S-box*, dan *linear transformation* (untuk setiap ronde selain ronde terakhir). Pada ronde terakhir, *linear transformation* digantikan oleh operasi *key mixing* tambahan; *Final Permutation* (FP). Berikut ini adalah cara kerja pada *Serpent cipher*:

$$B_0 := IP(P) \quad (2.1)$$

$$B_{i+1} := R_i(B_i) \quad (2.2)$$

$$C := FP(B_{32}) \quad (2.3)$$

$$R_i(X) = L(S_i(X \oplus K_i)) \qquad i = 0, \ldots, 30 \quad (2.4)$$

$$R_i(X) = S_i(X \oplus K_i) \oplus K_{32} \qquad i = 31 \quad (2.5)$$

Setiap metode kriptografi membutuhkan minimal sebuah *key* untuk menjalankan proses enkripsi maupun dekripsi. Pada implementasinya, *password* dari pengguna seringkali digunakan sebagai *key*. [3] Namun dibutuhkan sebuah algoritma yang memerlukan *key* kedua (biasanya berupa *pseudorandom number*) untuk mengamankan *password* yang disimpan. Algoritma ini dinamakan *Key Derivation Function* (KDF). Penggunaan KDF pada kriptografi dapat dilihat pada fungsi sebagai berikut:

$$DK = KDF(P, S)$$

dimana DK adalah *derived key*, KDF adalah *Key Derivation Function*, *P* adalah *password*, dan *S* adalah *salt* yang seringkali merupakan *random number*.

### 2.2 Android

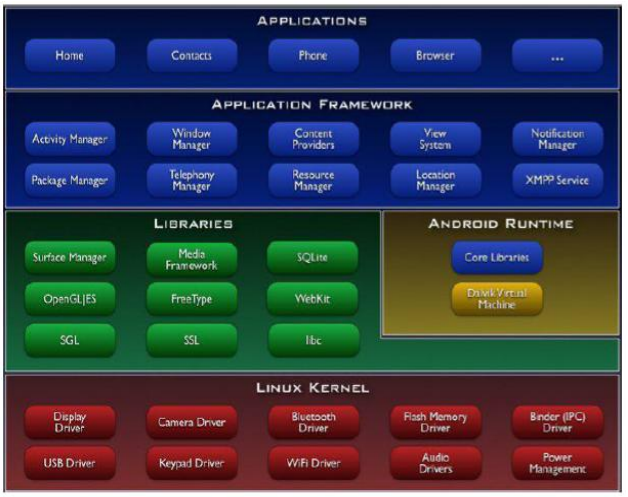
Android adalah sebuah sistem operasi untuk *mobile device* berbasis Linux. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri. [4]

Android memiliki berbagai keunggulan sebagai piranti lunak yang memakai basis kode komputer yang bisa didistribusikan secara terbuka (*open source*) sehingga pengguna bisa membuat aplikasi baru di dalamnya. Android memiliki aplikasi *native Google* yang terintegrasi seperti *pushmail Gmail*, *Google Maps*, dan *Google Calendar*.

Para penggemar *open source* kemudian membangun komunitas yang membangun dan berbagi Android berbasis *firmware* dengan sejumlah penyesuaian dan fitur-fitur tambahan, seperti FLAC *lossless audio* dan kemampuan untuk menyimpan *download* aplikasi pada *microSD card*.

#### 2.2.1 Arsitektur Android

Arsitektur Android terdiri dari: *Applications, Application Framework, Libraries, Android Runtime* dan *Kernel Linux*. Arsitektur lengkap *platform* ini dapat dilihat pada Gambar 1.




**Gambar 1. Arsitektur Android**

### 2.2.2 *The Dalvik Virtual Machine* (DVM)

S alah satu elemen kunci dari Android adalah *Dalvik Virtual Machine* (DVM). Android berjalan di dalam DVM bukan di *Java Virtal Machine* (JVM). DVM adalah ñ*register based*ò sementara JVM adalah ñ*stack based*ò, DVM. *Dalvik Virtual Machine* menggunakan *kernel* Linux untuk menangani fungsionalitas tingkat rendah termasuk keamanan, *threading*, dan proses serta manajemen memori. Semua *hardware* yang berbasis Android dijalankan dengan menggunakan *Virtual Machine* untuk menjalankan aplikasi.

### 2.2.3 Android SDK (*Software Development Kit*)

Android SDK adalah *tools* API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Android merupakan *subset* perangkat lunak untuk ponsel yang meliputi sistem operasi, *middleware* dan aplikasi kunci yang di-*release* oleh Google. Beberapa fitur-fitur Android yang paling penting adalah:

a) *Framework* aplikasi yang mendukung penggantian komponen dan *reusable.*
b) *Virtual Dalvik Machnine* dioptimalkan untuk perangkat *mobile.*
c) *Integrated browser* berdasarkan *engine open source WebKit*.
d) Grafis yang dioptimalkan dan didukung oleh *libraries* grafis 2D, 3D berdasarkan spesifikasi OpenGL 1.0.
e) SQLite untuk penyimpanan data.
f) *Media support* yang mendukung *audio*, *video*, dan gambar (MPEG4, MP3, JPG, PNG, GIF), GSM *Telephony* (tergantung *hardware*).
g) *Bluetooth*, EDGE, 3G, dan WiFi (tergantung *hardware*).
h) Kamera, GPS, kompas, *accelerometer* (tergantung *hardware*).
i) Lingkungan *development* yang lengkap termasuk perangkat *emulator*, *tools* untuk *debugging*, profil dan kinerja memori, dan *plugin* untuk IDE Eclipse. [5]

### 2.2.4 *Android Development Tools* (ADT)

*Android Development Tools* (ADT) adalah *plugin* yang didesain untuk IDE Eclipse yang memberikan kemudahan dalam mengembangkan aplikasi Android dengan menggunakan IDE Eclipse. Dengan menggunakan ADT untuk Eclipse, akan memudahkan dalam membuat aplikasi *project* Android, membuat GUI aplikasi, dan menambahkan komponen-komponen yang lainnya, begitu juga dapat melakukan *running* aplikasi menggunakan Android SDK melalui Eclipse. Selain itu, dengan ADT memungkinkan untuk membuat *package* Android (.apk) yang digunakan untuk mendistribusikan aplikasi Android yang telah dibuat.

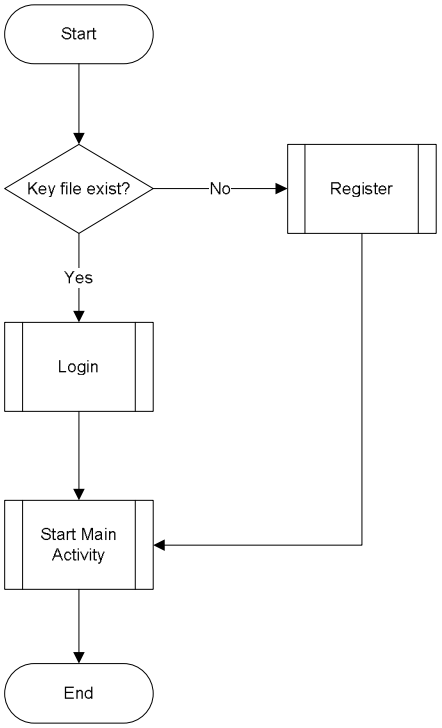
## 3. DESAIN SISTEM

### 3.1 Analisa Sistem

Seperti yang telah dijelaskan pada bab sebelumnya tentang kriptografi, bahwa penggunaan *password* sebagai *key* perlu disertai dengan penggunaan *Key Derivation Function* (KDF). Namun aplikasi yang akan dibuat pada proyek ini menggunakan fungsi enkripsi *Serpent* sebagai KDF. Alasannya adalah karena baik parameter yang dibutuhkan untuk menjalankan fungsi maupun tujuannya adalah sama (menghasilkan *ciphertext*, yaitu *password* yang teracak). *Password* pengguna bertindak sebagai *plaintext*, sedangkan *pseudorandom key* bertindak sebagai *key*. *Pseudorandom key* sendiri didapat dari hasil enkripsi *pseudorandom number* dengan *password* pengguna. Pada aplikasi yang akan dibuat, *pseudorandom number* dihasilkan melalui *event shake* dengan memanfaatkan sensor gerak dari *accelerometer* pada *device* Android. *Key* yang akan digunakan adalah *password* pengguna.

### 3.2 Garis Besar Aplikasi

Berikut ini adalah *flowchart* yang menggambarkan cara kerja aplikasi secara garis besar:




**Gambar 2. *Flowchart* Cara Kerja Aplikasi**

Aplikasi diawali dengan sebuah pengecekan apakah file *key* ada atau tidak. Apabila ada, berarti aplikasi sudah pernah digunakan dan sudah ada *password* pengguna yang tersimpan, sehingga proses yang dijalankan adalah *Login*. Namun apabila tidak ada, berarti aplikasi belum pernah digunakan atau belum ada *password* pengguna yang tersimpan di dalamnya, sehingga proses yang akan dijalankan adalah *Register*. Baik setelah melakukan *Login* maupun *Register*, aplikasi akan dilanjutkan pada tampilan utama. Sehingga proses yang selanjutnya dijalankan adalah *Start Main Activity*.

## 4. IMPLEMENTASI SISTEM

### 4.1 Implementasi *Serpent*

Dalam implementasinya, baik untuk enkripsi maupun dekripsi, *Serpent* terdiri atas fungsi *ROUND* yang dijalankan sebanyak 32 iterasi. Setiap *ROUND* memerlukan *subkey* untuk menjalankan berbagai fungsi. *Subkey* diperoleh dari *key* yang merupakan input pada fungsi enkripsi dan dekripsi.

Berikut ini adalah penjelasan yang dibagi berdasarkan 2 fungsi, yaitu enkripsi dan dekripsi:

### 4.1.1 Enkripsi

Pada enkripsi, input yang diterima adalah *text* dengan tipe data *byte array* dan sebuah *key* dengan tipe data *string*, sedangkan output-nya adalah *ciphertext* dengan tipe data *byte array*. Fungsi enkripsi dapat dilihat pada *Pseudocode* berikut.

***Pseudocode* 1. Fungsi *encrypt***

```
Function encrypt(text, key)
{
    Prepare key;
    Generate subkeys;
    For i = 0 to text.length-1
    {
        Map text into array of 4
        integer;
        For j = 0 to 30
        {
            Execute ROUND function;
        }
        Execute ROUND_LAST function;
        Map result into array of byte;
    }
    Padding;
    Return ciphertext;
}
```

Proses enkripsi diawali dengan mengubah 256 bit *key* menjadi *subkeys*. Kemudian dilakukan *looping* untuk membaca satu per satu *byte* pada *text*, lalu memetakannya ke dalam *array of 4 integer* (*plaintext*). Setelah itu dilakukan fungsi ROUND sebanyak 31 kali, dan diakhiri dengan mengeksekusi fungsi ROUND_LAST. Kemudian hasilnya dimasukkan ke dalam *array of byte* (*ciphertext*). Setelah itu akan dilakukan proses *padding* untuk mencukupkan jumlah bit yang kurang.

### 4.1.2 Dekripsi

Secara garis besar, fungsi dekripsi hampir sama dengan fungsi enkripsi. Bedanya adalah fungsi *ROUND* pada dekripsi dijalankan secara terbalik (*inverse*). Selain itu, pada dekripsi tidak terdapat tahap *padding* (kecuali untuk *key*), melainkan dilakukan eleminasi terhadap *padding bit*. Fungsi dekripsi dapat dilihat pada *Pseudocode* berikut.

***Pseudocode* 2. Fungsi *decrypt***

```
Function decrypt(text, key)
{
    Prepare key;
    Generate subkeys;
    For i = 0 to text.length-1
    {
        Map text into array of 4
        integer;
        Execute ROUND_FIRST_INVERSE
        function;
        For j = 0 to 30
        {
            Execute ROUND_INVERSE
            function;
        }
        Map result into array of byte;
    }
    Depadding;
    Return plaintext;
}
```

## 4.2 Implementasi *Shake*

Untuk mengimplementasikan *shake* diperlukan sensor gerak pada *accelerometer device* untuk mendeteksi dan menghasilkan nilai dari perubahan posisi. Implementasi *shake* pada aplikasi terbagi atas 3 tahap, yaitu tahap deklarasi, tahap pembuatan *listener* dan tahap *listening*. Pada tahap deklarasi, setiap variabel yang akan digunakan dideklarasi dan di-inisialisasi terlebih dahulu. Tahap pembuatan *listener* berisi suatu algoritma hitungan dan pengecekan hasil dari hitungan tersebut. Apabila hasil hitungan mencapai nilai tertentu, maka dapat dianggap bahwa guncangan (*shake*) telah terjadi. Selanjutnya, *listener* yang telah dibuat dijalankan pada *onResume* pada tahap *listening*.

# 5. PENGUJIAN SISTEM

## 5.1 Spesifikasi *Device*

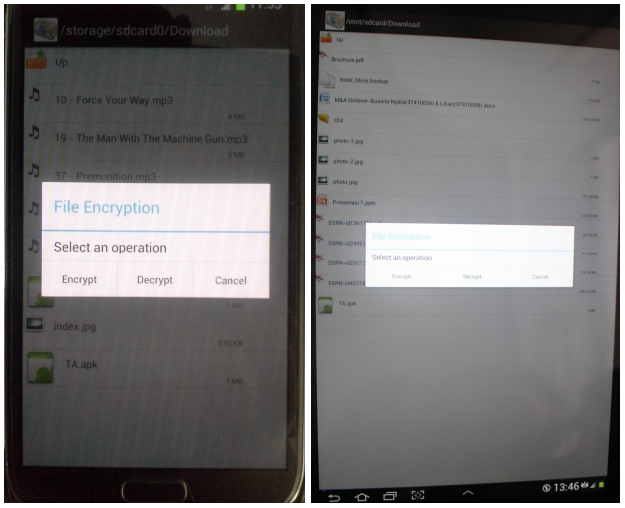
Berikut ini adalah tabel yang memperlihatkan spesifikasi *Device* yang digunakan dalan pengujian sistem.

**Tabel 1. Tabel Spesifikasi *Device***

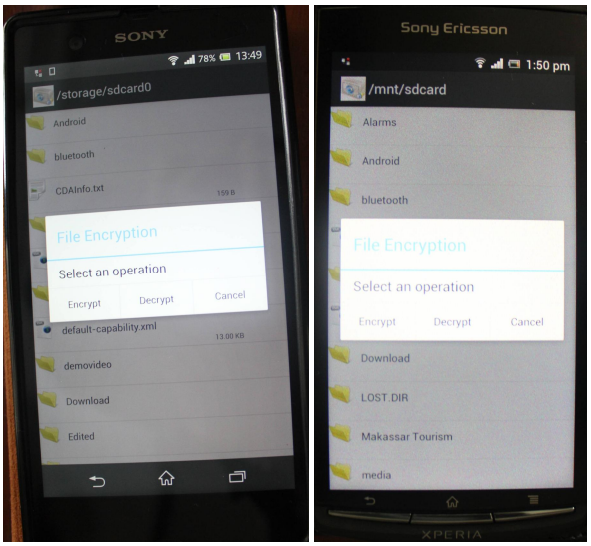
| *Device* | *Operating System* | *Display size* | CPU | *Memory Internal* |
|---|---|---|---|---|
| Samsung GT-N7100 | Android 4.1.2 (Jelly Bean) | 720 x 1280 *pixels*, 5.5 *inches* (~267 ppi *pixel density*) | *Quad Core* - 1,6 GHz | 16 GB *storage*, 2 GB RAM |
| Samsung GT-N8000 | Android 4.0.4 (Ice Cream Sandwich) | 800 x 1280 *pixels*, 10.1 *inches* (~149 ppi *pixel density*) | *Quad Core* - 1,4 GHz | 16 GB *storage*, 2 GB RAM |
| Sony Xperia C6602 | Android 4.1.2 (Jelly Bean) | 1080 x 1920 *pixels*, 5.0 *iches* (~441 ppi *pixel density*) | *Quad Core* - 1.5 GHz | 16 GB *storage*, 2 GB RAM |
| Sony Xperia LT18i | Android 4.0.4 (Ice Cream Sandwich) | 480 x 854 *pixels*, 4.2 *inches* (~233 ppi *pixel density*) | *Single Core* - 1.4 GHz | 1 GB *storage*, 512 MB RAM |

## 5.2 Pengujian *File Encryption*

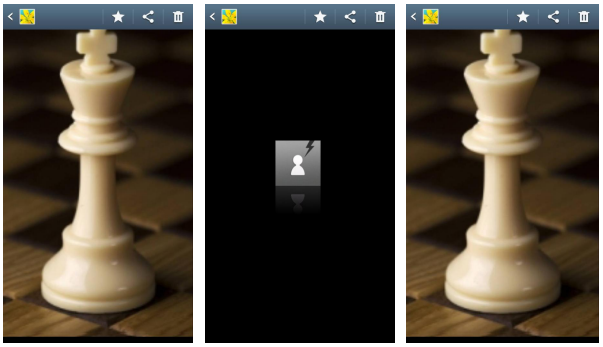
Enkripsi *file* dapat dilakukan dengan meng-klik tombol *file encryption* pada halaman *menu*. Setelah itu akan ditampilkan halaman *file encryption* yang berisi *list* dari *file* dan *folder* yang terdapat pada *external storage*. Pada tampilan ini pengguna dapat melakukan pencarian *file*, kemudian meng-klik *file* yang ingin di-enkripsi atau dekripsi. Pengujian terhadap enkripsi dan dekripsi *file* dapat dilihat pada gambar berikut.




Samsung GT-N7100 Samsung GT-N8000




Sony Xperia C6602 Sony Xperia LT18i



Sebelum enkripsi Setelah enkripsi Setelah dekripsi


Sebelum enkripsi Setelah enkripsi Setelah dekripsi

**Gambar 3. Pengujian *File Encryption***

## 5.3 Pengujian Kecepatan

Untuk mengetahui hubungan antara *device* yang digunakan, ukuran dan tipe *file*, serta kecepatan enkripsi dan dekripsi, maka dilakukan pengujian kecepatan. Berikut ini adalah tabel yang menunjukkan hasil pengujian kecepatan.

**Tabel 2. Tabel Hasil Pengujian Kecepatan**

| *Device* | *Size (byte)* | *Type* | *Speed (second)* | |
|---|---|---|---|---|
| | | | *Encryption* | *Decryption* |
| Samsung GT-N7100 | 1 M | *Text* | 2,363 | 2,328 |
| | | *Image* | 2,355 | 2,317 |
| | | *Sound* | 2,505 | 2,417 |
| | 5 M | *Text* | 10,586 | 10,198 |
| | | *Image* | 10,414 | 9,95 |
| | | *Sound* | 10,538 | 10,036 |
| | 10 M | *Text* | 20,65 | 20,094 |
| | | *Image* | 20,581 | 20,28 |
| | | *Sound* | 20,772 | 19,818 |
| Samsung GT-N8000 | 1 M | *Text* | 2,768 | 2,891 |
| | | *Image* | 2,74 | 2,853 |
| | | *Sound* | 2,936 | 2,867 |
| | 5 M | *Text* | 11,919 | 12,146 |
| | | *Image* | 11,66 | 11,55 |
| | | *Sound* | 12,179 | 11,942 |
| | 10 M | *Text* | 22,924 | 23,043 |
| | | *Image* | 23,528 | 23,811 |
| | | *Sound* | 23,08 | 23,584 |
| Sony Xperia C6602 | 1 M | *Text* | 2,97 | 2,932 |
| | | *Image* | 2,951 | 2,898 |
| | | *Sound* | 3,032 | 2,911 |
| | 5 M | *Text* | 12,993 | 13,076 |
| | | *Image* | 12,969 | 12,762 |
| | | *Sound* | 12,987 | 12,722 |
| | 10 M | *Text* | 25,314 | 25,645 |
| | | *Image* | 25,644 | 25,818 |
| | | *Sound* | 25,702 | 25,779 |

| | | | | |
|---|---|---|---|---|
| | 1 M | *Text* | 3,799 | 3,847 |
| | | *Image* | 3,799 | 3,679 |
| | | *Sound* | 3,841 | 3,809 |
| Sony Xperia LT18i | 5 M | *Text* | 15,601 | 15,516 |
| | | *Image* | 15,029 | 15,248 |
| | | *Sound* | 15,665 | 15,461 |
| | 10 M | *Text* | 30,705 | 29,962 |
| | | *Image* | 31,339 | 30,293 |
| | | *Sound* | 31,33 | 30,333 |

Berdasarkan tabel hasil pengujian di atas, dapat dilihat bahwa ukuran *file* berpengaruh pada kecepatan enkripsi dan dekripsi. Semakin besar ukuran *file*, maka waktu eksekusi yang dibutuhkan semakin lama. Tipe *file* juga memiliki pengaruh terhadap kecepatan enkripsi dan dekripsi, di mana *file image* yang paling cepat dibanding jenis *file* lainnya, dan *file sound* yang paling lambat.

Apabila dibandingkan hasil pengujian antar *device* yang sejenis, maka faktor penentu kecepatan eksekusi adalah *processor*-nya. Berdasarkan Tabel 1., *processor* yang digunakan oleh Samsung GT-N7100 adalah *Quad Core* 1,6 GHz, sedangkan Samsung GT-N8000 menggunakan *processor Quad Core* 1,4 GHz. Namun pada tabel hasil pengujian kecepatan di atas dapat dilihat bahwa Samsung GT-N7100 mampu mengeksekusi *file* apapun dengan waktu yang lebih singkat dibanding Samsung GT-N8000. Apabila hasil uji kecepatan Sony Xperia C6602 dibandingkan dengan Sony Xperia LT18i, juga terlihat bahwa *processor* yang digunakan oleh *device* sangat berpengaruh terhadap kecepatan eksekusi, di mana Sony Xperia C6602 dengan *processor Quad Core* 1,5 GHz mampu menunjukkan kecepatan eksekusi yang lebih baik dibanding Sony Xperia LT18i dengan *processor Single Core* 1,4 GHz.

Apabila *device* yang berbeda jenis (Samsung dengan Sony) dibandingkan, dapat dilihat bahwa ada pengaruh antara jenis *device* yang digunakan dengan kecepatan eksekusi. Hal ini terlihat pada perbandingan antara Samsung GT-N8000 dengan Sony C6602. Samsung GT-N8000 menggunakan *processor Quad Core* 1,4 GHz, sedangkan Sony Xperia C6602 menggunakan *processor Quad Core* 1,5 GHz yang seharusnya lebih baik secara kualitas. Namun hasil pengujian menunjukkan bahwa waktu yang diperlukan untuk mengeksekusi baik enkripsi maupun dekripsi pada Samsung GT-N8000 lebih singkat dibanding Sony Xperia C6602. Sehingga melalui pengujian kecepatan ini dapat disimpulkan bahwa kecepatan enkripsi maupun dekripsi dipengaruhi oleh kualitas *processor* dan jenis *device* yang digunakan.

# 6. KESIMPULAN

Berdasarkan hasil pengujian fitur, dapat disimpulkan bahwa aplikasi yang telah dibuat dapat menyimpan informasi berupa *text*, melakukan enkripsi dan dekripsi dengan metode *Serpent*, mengimplementasikan *shake*, menampilkan kembali informasi yang telah disimpan, meng-*edit* ataupun menghapus informasi, mengirim informasi melalui SMS dan *e-mail*, melakukan enkripsi/dekripsi terhadap *file* di *external storage*, mengganti *password*, mem-*backup* seluruh informasi yang tersimpan di dalam aplikasi ke Google Drive, dan melakukan *restore*.

Berdasarkan hasil pengujian kecepatan, dapat diambil kesimpulan bahwa kualitas *processor* dan jenis *device* berpengaruh terhadap kecepatan eksekusi dari *Serpent cipher*.

# 7. DAFTAR PUSTAKA


[1] Anderson, Ross (2011). *Serpent: A Candidate Block Cipher for the Advanced Encryption Standard*. http://www.cl.cam.ac.uk/~rja14/serpent.html

[2] Anderson, Ross; Eli Biham; Lars Knudsen (1998). *Serpent: A Proposal for the Advanced Encryption Standard*. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.5585

[3] Dumas, Mel (2011). *eWallet*. http://salutmel.com/27024/285111/portfolio/ewallet

[4] Safaat, Nazruddin. (2012). Pemograman Aplikasi *Mobile Smartphone* dan *Tablet* PC Berbasis Android. Bandung: Penerbit Informatika.

[5] Shodiq, Amri. Tutorial Dasar Pemograman *Google Maps* API.