Implementasi Enkripsi File dengan Memanfaatkan *Secret Sharing Scheme*

Muhammad Aodyra Khaidir (13513063)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Insitut Teknologi Bandung
Jalan Ganesha 10-12, Bandung 40132, Indonesia
aodyra@gmail.com

***Abstract*—File yang di upload ke *file sharing* atau suatu jaringan memiliki berbagai tingkat keamanan. File tersebut harus dienkripsi menggunakan algoritma enkripsi seperti algoritma *block cipher* AES. Tujuan enkripsi file tersebut adalah menjaga keamanan dan kerahasiaan file. Enkripsi sebuah file menggunakan sebuah kunci simetris. Keamanan kunci simetris dapat ditingkatkan dengan metode *Secret Sharing Scheme*. Metode ini berguna dalam enkripsi dan dekripsi file yang membutuhkan beberapa kunci berbeda.**

Keywords—File, Enkripsi, Dekripsi, Block cipher, Secret Sharing Scheme

I. Pendahuluan

Saat ini *file-file* dikirimkan ke berbagai media transfer dan *sharing*. Media-media tersebut seperti *file sharing*, aplikasi pesan, email, usb, dst. Beberapa Media ini memudahkan kita dalam mengirim berkas-berkas sehingga kita tidak perlu lansung dataang ke tempat tujuan.

Hal diatas menghemat waktu, biaya, dan tenaga yang diperlukan. Teknologi-teknologi yang berkembang saat ini sangat mendukung keberjalanan suatu bisnis. Selain itu teknologi ini juga memberikan manfaat pada bidang lain seperti akademik, organisasi, kenegaraan, informasi kemanan, dst.

Berkas-berkas yang dikirimkan melalui media *online* maupun *offline* mengandung sebuah informasi yang berguna bagi penerimanya. Komunikasi memungkinkan adanya pertukaran informasi dari satu tempat ke tempat lain. Seperti yang disebutkan di atas, media komunikasi yang banyak digunakan untuk mengirim berkas seperti email, aplikasi pesan, dst. Komunikasi-komunikasi tersebut menggunakan jaringan publik atau seluler.

Informasi-informasi yang dibawa dalam sebuah berkas sering merupakan informasi rahasia dan penting. Informasi yang penting menjadi berbahaya apabila orang yang mempunyai hak mengetahui isi informasi. Keamanan informasi menjadi menjadi hal penting sehingga isi yang dikandung tidak diketahui oleh sembarang orang.

Kriptografi adalah ilmu yang mempelajari bagaimana menyembunyikan isi sebuah berkas. Penyembunyian isi sebuah berkas ini bertujuan untuk menjaga kerahasiaan isi berkas dari orang-orang yang tidak berhak dalam membaca pesan. Kriptografi memiliki berbagai konsep untuk melindungi isi berkas. Konsep-konsep tersebut dapat dipadukan sehingga menghasilkan keamanan yang lebih tinggi. Kriptografi juga banyak digunakan dalam berbagai bidang seperti pengiriman surat elektronik, pengiriman data, saluran telfon, dst.

Salah satu konsep kriptografi yang dapat digunakan untuk menjaga keamanan berkas adalah *Secret Sharing Scheme*. Metode ini membagi kunci rahasia ke beberapa kunci. Beberapa kunci yang dihasilkan dapat digabungkan untuk membentuk kunci rahasia. Kunci rahasia dapat digunakan dalam enkripsi dan dekripsi pesan. Salah satu metode *Secret Sharing Scheme* adalah *Shamir's Secret Sharing.* Metode ini menggunakan pemecahan masalah persamaan metematika.

Algoritma *block cipher* AES dapat digunakan untuk enkripsi dan dekripsi sebuah berkas. Algoritma ini menerima masukan berupa kunci untuk enkripsi. Kunci yang sama digunakan untuk proses dekripsi. Algoritma ini melakukan proses enkripsinya dalam orientasi *byte*. Algoritma ini melakukan beberapa putaran dalam prosesn enkripsinya. Algoritma ini menggunakan ukuran block 128 bit.

Dalam makalah ini, akan dilakukan implementasi dan analisis *Secret Sharing Scheme* pada proses enkripsi dan dekripsi *file*. Penggunaan metode ini diharapkan dapat meningkatkan kemanan dan kerahasiaan isi *file*. Hal ini disebabkan persetujuan beberapa orang diperlukan untuk membaca isi berkas. Kunci rahasia awal dibagi ke beberapa

orang. Kunci awal tersebutlah yang menjadi kunci enkripsi dan dekripsi.

II. Dasar Teori

A. Secret Sharing Scheme

Secret Sharing Shceme adalah konsep kriptografi yang digunakan untuk membagi rahasia ke beberapa bagian. Rahasia yang dimaksud dalam hal ini adalah sebuah bilangan. Bagian-bagian yang dihasilkan dapat digunakan untuk membangun bilangan awal. Bilangan tersebut merupakan representasi sebuah pesan rahasia yang diudab ke bentuk bilangan atau nilai kunci rahasia. Konsep ini dapat meningkatkan keamanan. Apabila hanya satu kunci yang didapat oleh penyadap atau orang yang tidak berhak maka keamanan berkas dapat tetap terjaga. Pembentukan bilangan awal dapat dilakukan dengan beberapa metode. Salah satu metode yang dapat digunakan adalah *Shamir's Secret Sharing*.

Shamis's Secret Sharing adalah metode pemagian skema rahasia ke dalam beberapa bagian dengan memecahkan permasalahan persamaan matematika. Biasanya metode ini menggunakan notasi (k, n). Nilai n menunjukkan jumlah bagian yang dihasilkan. Selanjutnya nilai k menunjukkan jumlah minimal bagian yang diperlukan untuk rekonstruksi bilangan awal. Biasanya metode ini menggunakan bilangan prima yang lebih besar dari bilangan rahasia dan setiap koefisien persamaan. Jumlah minimal k bilangan untuk rekonstruksi nilai rahasia S menggunakan persamaan matemetika derajat k-1.

Berikut adalah langkah-langkah metode *Shamir's Secret Sharing* untuk membagi bilangan rahasia S_1, S_2, … , S_n ke beberapa bagian bilangan.

- Pilih bilangan prima P, yang lebih besar dari nilai rahasia S dan lebih besar dari setiap koefisien persamaan untuk k partisipan. Komputasi-komputasi dihasilkan dalam modulus P sehingga perhitungan aritmatik dalam bidang terbatas.

- Pilih bilangan acak sebanyak k - 1 angkat yang lebih kecil dari bilangan prima P, misalkan a_1, a_2, …, a_{k-1}. Lalu nyatakan dalam polinomial

$$S(x) \equiv M + a_i x^1 + a_2 x^2 + \ldots + a_{k-1} x^{k-1} \pmod{P}$$

Bentuk polinomial merupakan persamaan rahasia.

- Setiap partisipan ditentukan sebuah nilai x yang berbeda x_1, x_2, … , x_{k-1} (mod P). Selanjutnya setiap partisipan memperoleh *share* masing-masing berdasarkan nilai x_i yang dimasukkan ke persamaan *S(x)*.

$$y_i \equiv S(x_i)$$

Nilai x_i dan y_i yang diapat adalah bagian-bagian untuk n partisipan. Nilai rahasia S semula akan dihasilkan dari gabungan k nilai dari n partisipan. Beberapa cara dapat digunakan untuk menghasilkan nilai rahasia S dari persamaan diatas. Cara-cara yang dapat digunakan adalah Eliminasi Gauss dan Interpolasi Lagrange. Makalah ini menggunakan Interpolasi Lagrange dalam menyelesaikan pesamaan di atas.

B. Lagrange Interpolation

Polinomial Lagrange digunakan untuk interpolasi polimonial. Untuk setiap pasangan x_i dan y_i yang berbeda, polinomial lagrange adalah polinomial derajat terendah yang mengasumsikan setiap nilai x_i bersesuaian dengan nilai y_i. Titik-titik tersebut dimasukkan ke dalam rumus yang disebarluaskan untuk membentuk persamaan lagrange. Persamaan Lagrange yang dimaksud adalah

$$P(x) = \sum_{j=1}^{n} P_j(x),$$

Dimana,

$$P_j(x) = y_j \prod_{\substack{k=1 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}.$$

Persamaan lengkapnya adalah

$$P(x) = \frac{(x-x_2)(x-x_3)\cdots(x-x_n)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_n)} y_1 + \frac{(x-x_1)(x-x_3)\cdots(x-x_n)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_n)} y_2 + \cdots + \frac{(x-x_1)(x-x_2)\cdots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\cdots(x_n-x_{n-1})} y_n.$$

Kalkulasi di atas dalam modulus P. Hal ini disebabkaan agar kalkulasi dalam bidang terbatas. Nilai rahasia S dapat dihitung dengan memasukkan nilai x = 0 pada persamaan *P*(*x*). Nilai *P(0)* bersesuaian dengan nilai rahasia S, karena nilai tersebut merupakan koefisian x^0 pada persamaan

$$S(x) \equiv M + a_i x^1 + a_2 x^2 + \ldots + a_{k-1} x^{k-1} \pmod{P}$$

III. Advanced Encryption Standard

*Advanced Encryption Standard(*AES*)* adalah standar algoritma *cipher block* dengan kunci simetris. AES terdiri dari tiga macam berdasarkan panjang kuncinya AES-128,

AES-192, dan AES-256. Angka-angka tersebut menandakan panjang kunci masing-masing 128 bit, 192 bit dan 256 bit. Nama Algoritma AES adalah *Rijndael* yang merupakan gabungan tiga nama pembuat algoritma tersebut. AES distandarkan pada tanggal 26 November 2001.

AES menggunakan 10, 12, atau 14 putaran dalam proses enkripsi dan dekripsi. Jumlah putaran ini sesuai dengan ukuran kunci pada AES-128, AES-192, dan AES-256. Algoritma ini beroperasi dalam orientasi byte yang berbeda dari algoritma pendahulunya. Setiap putaran menggunakan kunci internal.

AES-128 menggunakan kunci 128 bit dengan ukuran blok 128 bit. Berikut tahap-tahap pada AES

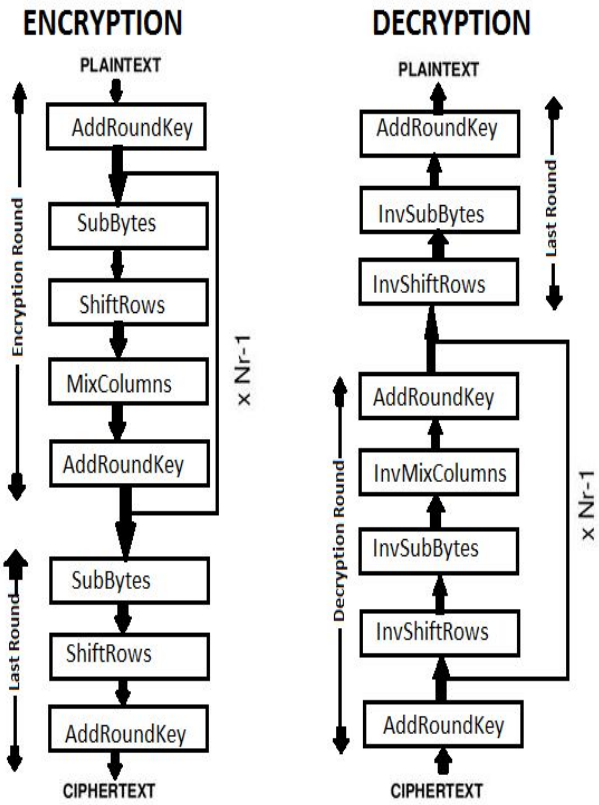
- *Key Expansions*, kunci putaran didapatkan dari kunci awal yang dijadwalkan dengan *Rijndael's Key Schedule*.

- *Initial Round,* tahap ini melakukan operasi *XOR* antara setiap byte *state* awal dengan *cipher key*. Tahap ini disebut juga tahap *AddRoundKey*.

- Putaran, putaran dilakukan Jumlah putaran total - 1. Setiap putaran melakukan operasi
  - *SubBytes*, substitusi setiap byte dengan kotak substitusi S
  - *ShiftRows,* proses pergeseran secara siklik pada tiga baris terakhir dari *array state*.
  - *MixColumns*, proses mengalikan setiap kolom pada *array state* dengan suatu persamaan polinomial dalam modulus $(x^4 + 1)$.
  - *AddRoundKey*, tahap ini melakukan operasi yang serupan dengan *initial round*. Operasi *XOR* dilakukan antara kunci putaran dengan *array state*.

- *Final Round,* Tahap yang mirip dengan tahap putaran namun tanpa proses *MixColumns*. Sehingga proses yang dilakukan pada tahap ini adalah
  - *SubBytes*
  - *ShiftRows*
  - *AddRoundKey*

Berikut adalah gambar proses pada AES



Gambar 1. Skema proses AES

sumber:
http://nevonprojects.com/image-encryption-using-aes-algorithm/

IV. Eksperimen dan Analisis Hasil

Implementasi dilakukan pada lingkungan bahasa pemrograman JAVA. Pengujian dilakukan beberapa tahap. Pertama, kita akan menguji pembagian kunci dengan algoritma yang sudah digunkana. Kedua, kita akan menguji enkripsi dan dekripsi file dengan kunci-kunci yang sudah dibagi. Hasil dari tahap pertama akan ditampilkan kunci awal dan semua kunci pembagiannya. Selanjutnya hasil konstruksi kunci awal juga akan ditampilkan dengan jumlah-jumlah kunci yang dimasukkan beragam.

A. Pengujian Kebenaran Algoritma

Pengujian 1

Kunci Utama: 123456789

Masukan pengguna adalah jumlah pembagian, jumlah minimal bilangan untuk rekonstruksi, kunci utama.

```
masukkan jumlah shares: 5
masukkan jumlah min to construct: 3
masukkan kunci utama: 123456789
```

Keluaran Program

```
Prime number: 225360367
share: 1 key: 88391209
share: 2 key: 88574884
share: 3 key: 124007814
share: 4 key: 194689999
share: 5 key: 75261072
```

Masukan pengguna adalah jumlah kunci bagian yang akan dimasukkan sebesar total jumlah *share*, bilangan prima, angka dan kunci setiap *share*.

```
Masukkan jumlah kunci yang akan dimasukkan: 5
Masukkan bilangan prima: 225360367
pengambilan kunci ke-1
Masukkan angka share: 1
Masukkan angka kunci: 88391209
pengambilan kunci ke-2
Masukkan angka share: 2
Masukkan angka kunci: 88574884
pengambilan kunci ke-3
Masukkan angka share: 3
Masukkan angka kunci: 124007814
pengambilan kunci ke-4
Masukkan angka share: 4
Masukkan angka kunci: 194689999
pengambilan kunci ke-5
Masukkan angka share: 5
Masukkan angka kunci: 75261072
```

Keluaran Program

```
kunci utama: 123456789
```

Masukan pengguna adalah jumlah kunci bagian yang akan dimasukkan sebesar jumlah minimal bilangan rekonstruksi, bilangan prima, angka dan kunci setiap *share*.

```
Masukkan jumlah kunci yang akan dimasukkan: 3
Masukkan bilangan prima: 225360367
pengambilan kunci ke-1
Masukkan angka share: 1
Masukkan angka kunci: 88391209
pengambilan kunci ke-2
Masukkan angka share: 3
Masukkan angka kunci: 124007814
pengambilan kunci ke-3
Masukkan angka share: 5
Masukkan angka kunci: 75261072
```

Keluaran Program

```
kunci utama: 123456789
```

Masukan pengguna adalah jumlah kunci bagian yang akan dimasukkan kurang dari jumlah minimal bilangan rekonstruksi, bilangan prima, angka dan kunci setiap *share*.

```
Masukkan jumlah kunci yang akan dimasukkan: 2
Masukkan bilangan prima: 225360367
pengambilan kunci ke-1
Masukkan angka share: 1
Masukkan angka kunci: 88391209
pengambilan kunci ke-2
Masukkan angka share: 2
Masukkan angka kunci: 88574884
kunci utama: 88207534
```

Keluaran Program

```
kunci utama: 88207534
```

Masukan pengguna adalah jumlah kunci bagian yang akan dimasukkan sebesar jumlah minimal bilangan rekonstruksi, bilangan prima, angka dan kunci setiap *share* dengan salah satunya salah.

```
Masukkan bilangan prima: 225360367
pengambilan kunci ke-1
Masukkan angka share: 1
Masukkan angka kunci: 88391209
pengambilan kunci ke-2
Masukkan angka share: 2
Masukkan angka kunci: 88574884
pengambilan kunci ke-3
Masukkan angka share: 3
Masukkan angka kunci: 812361982
```

Keluaran Program

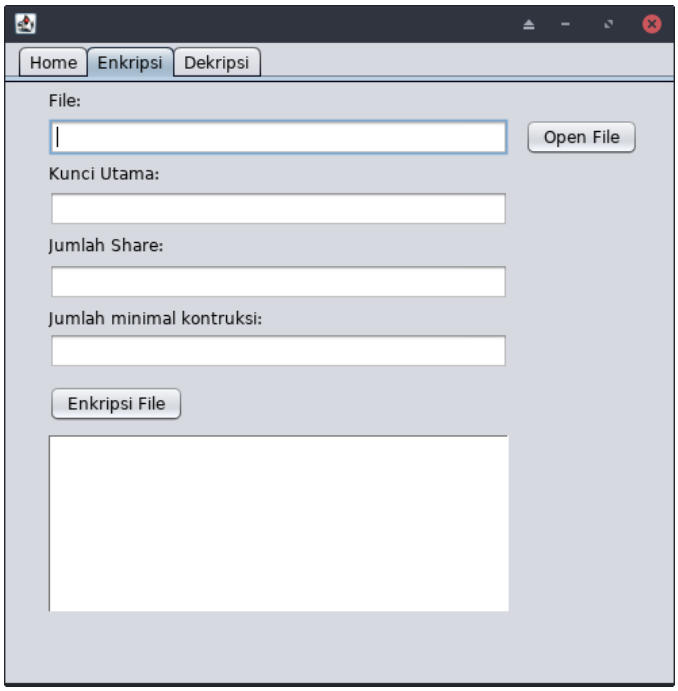
```
kunci utama: 135729856
```

B. Pengujian Kebenaran Aplikasi

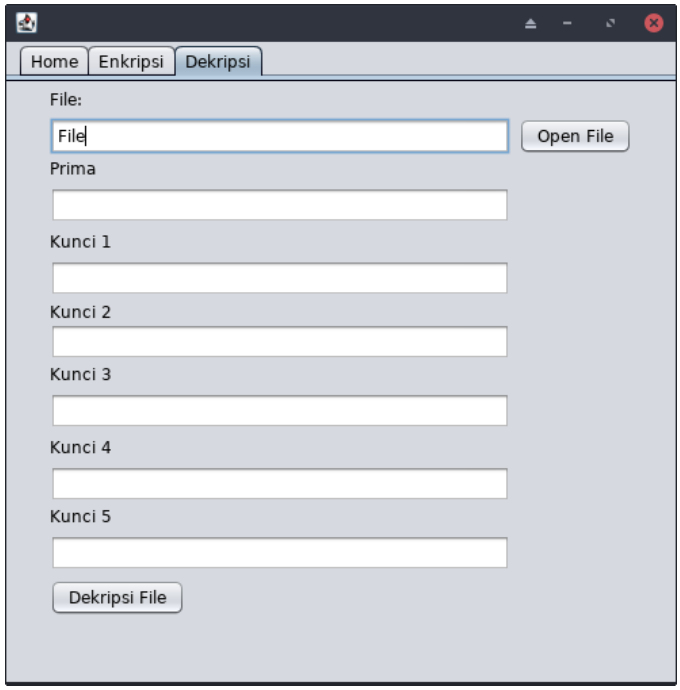
Pengujian 2

Pengujian menggunakan aplikasi dengan antarmuka pengguna untuk enkripsi dan dekripsi berkas.

Gambar 2. Tampilan Awal aplikasi.

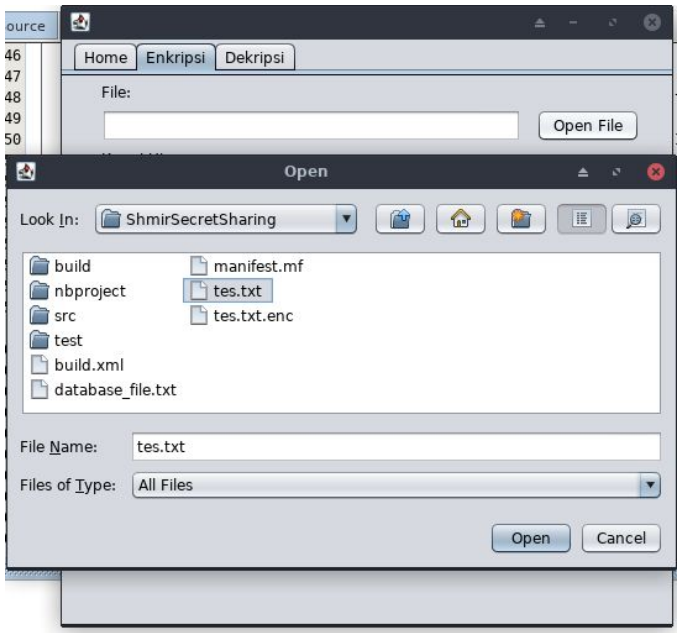


Gambar 3. Tampilan tahapan enkripsi pada aplikasi.



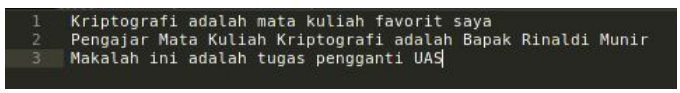
Gambar 4. Tampilan tahapan dekripsi pada aplikasi.

Pertama, kita akan menggunakan *file* "tes.txt".



Gambar 5. Tampilan tahapan pemilihan file untuk enkripsi.

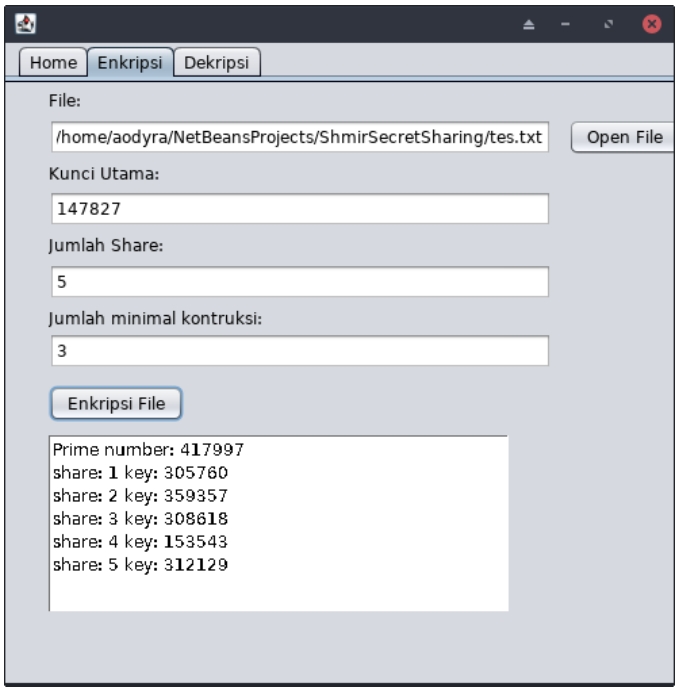
File tersebut berisi tiga baris kalimat.



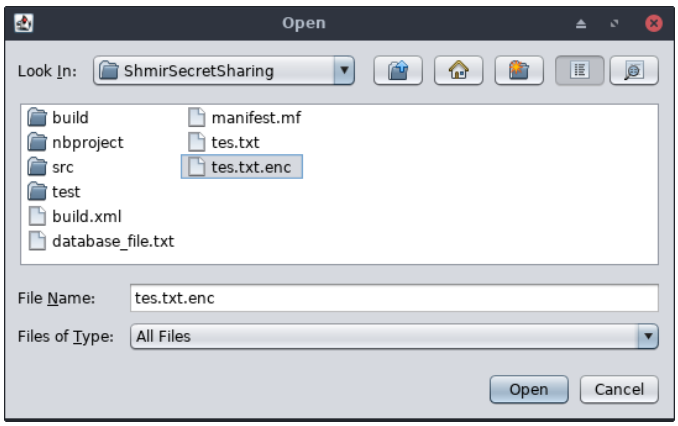
Gambar 6. Tampilan isi file untuk enkripsi.

Selanjutnya, kita masukan bebera parameter sebagai berikut. Hasil dari enkripsi *file* adalah tampilan yang berisi bilangan prima dan *share* dengan jumlah sesuai Jumlah Share.

Kunci Utama: 147827
Jumlah Share: 5
Jumlah minimal Konstruksi: 3

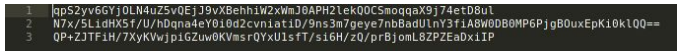


Gambar 7. Tampilan hasil proses enkripsi berkas.



Gambar 8. Tampilan berkas hasil proses enkripsi.

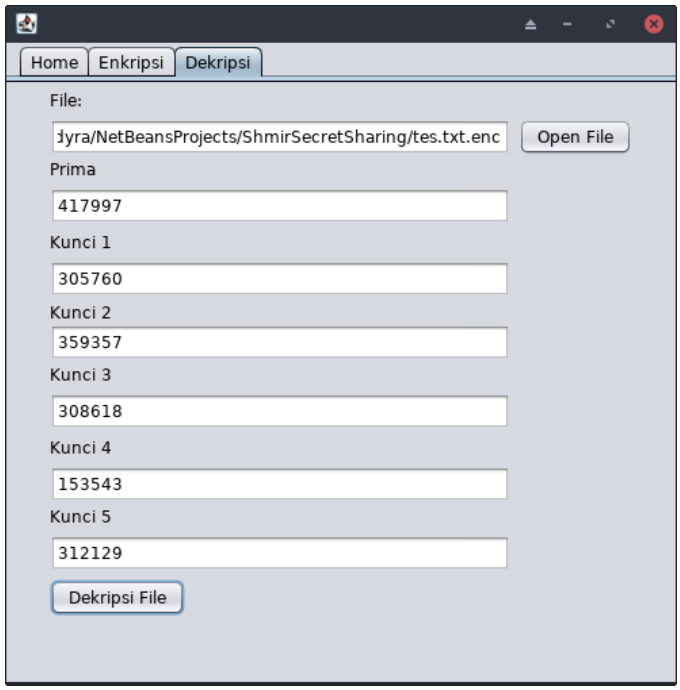
Gambar di atas menandakan *file* hasil enkripsi dan gambar selanjutnya adalah isi file tersebut.



Gambar 9. Tampilan isi berkas hasil enkripsi.

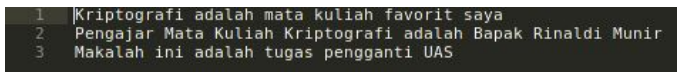
Proses dekripsi dilakukan dengan memasukkan beberapa parameter. Pengujian ini menggunakan semua *share key* untuk rekonstruksi bilangan rahasia.

Prima: 417997
Kunci 1: 305760
Kunci 2: 359357
Kunci 3: 308618
Kunci 4: 153543
Kunci 5: 312129



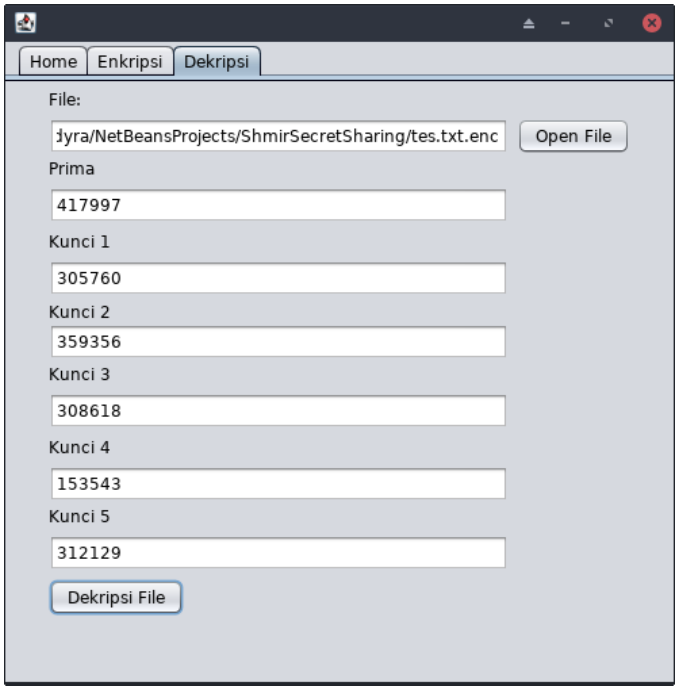
Gambar 10. Tampilan proses dekripsi.

Berikut adalah gambar hasil proses dekripsi.



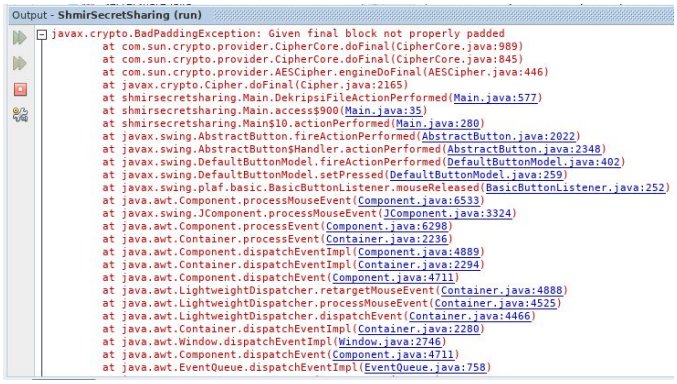
Gambar 11. Tampilan isi berkas hasil dekripsi.

Selanjutnya, kita akan mengubah salah satu kunci. Kunci yang diubah adalah kunci 2 dari 359357 menjadi 359356.



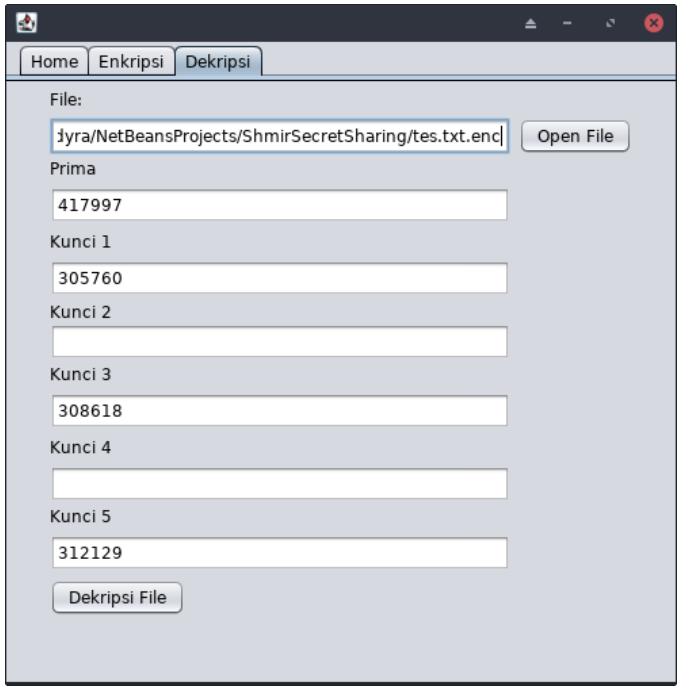
Gambar 12. Tampilan perubahan parameter kunci 2.

Perubahan parameter salah satu kunci menjadi salah, menyebabkan *error* sehingga program tidak akan menghasilkan file hasil dekripsi.

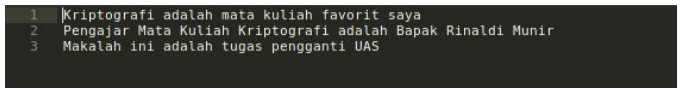


Gambar 13. Tampilan hasil perubahan parameter kunci 2.

Berikut adalah perubahan jumlah parameter kunci menjadi sebanyak jumlah minimal rekonstruksi dengan kunci yang benar untuk setiap *share*. Proses dibawah menghasilkan *file* hasil dekripsi.



Gambar 14. Tampilan proses dekripsi dengan jumlah kunci 3.



Gambar 15. Tampilan hasil proses dekripsi dengan jumlah kunci 3.

C. *Pembahasan Hasil*

Dari hasil pengujian, program *Shamir's Secret Sharing* berjalan sesuai harapan. Program dapat membagi bilangan rahasia S ke dalam beberapa bagian bilangan. Pembagian bilangan-bilangan dapat merekonstruksi bilangan rahasia S. Bilangan rahasia S dapat direkonstruksi dengan syarat

- Jumlah kunci yang dimasukkan sesuai dengan jumlah minimal bilangan rekonstruksi.
- Semua kunci yang dimasukkan benar.

Hasil pengujian Aplikasi berjalan sesuai harapan. Aplikasi dapat melakukan enkripsi dan dekripsi *file*. Aplikasi dapat menampilkan kunci-kunci pembagian dan bilangan prima yang dipakai untuk enkripsi. Aplikasi menerima kunci-kunci pembagian dan bilangan prima untuk melakukan dekripsi *file*. *File-file* hasil enkripsi dapat dilakukan dekripsi dengan syarat yang sama dengan program *Shamir's Secret Sharing.*

D. *Analisis Keamanan*

Metode *secret sharing* menambah keamanan kunci enkripsi dan dekripsi. Hal ini dikarenakan kunci rahasia utama hanya bisa didapatkan jika dilakukan rekonstruksi dengan jumlah yang sesuai dengan jumlah minimal. Setiap Kunci yang dimasukkan juga harus benar untuk melakukan dekripsi

file. *File-file* enkripsi akan tetap terjaga keamanannya dengan syarat

- Jumlah kunci benar yang diketahui orang yang tidak berhak kurang dari jumlah minimal kunci rekonstruksi.
- Bilangan prima yang digunakan dalam persamaan tidak diketahui oleh pihak luar.
-

E. Kesimpulan dan Saran

Aplikasi ini dapat mengatasi enkripsi dan dekripsi *file* dengan jumlah *share* adalah 5. Aplikasi ini berhasil membagi bilangan rahasia S. Aplikasi ini juga berhasil merekonstruksi bilangan rahasia S dari hasil kunci-kunci pembagian.

Aplikasi ini dapat dikembangkan dalam jumlah pembagian kunci. Pengembangan ini dapat menambah keamanan sebuah *file* karena jumlah pembagian dan minimal untuk merekonstruksi bilangan rahasia dapat lebih banyak. Selain itu, fitur enkripsi dan dekripsi teks dapat ditambahkan untuk mempermudah proses enkripsi dan dekripsi teks sederhana.

Referensi

[1] Munir, Rinaldi. *Advanced Encryption Standard*. Oktober 2015. Presentasi PowerPoint

[2] Munir, Rinaldi. *Skema Pembagian Data Rahasia (Secret Sharing Schemes)*. 2015. Presentasi PowerPoint

[3] http://nevonprojects.com/image-encryption-using-aes-algorithm/. Tanggal akses: 19 Desember 2016, 12.00 WIB

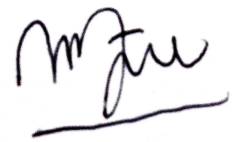
[4] http://nevonprojects.com/image-encryption-using-aes-algorithm/. Tanggal akses: 19 Desember 2016, 12.00 WIB

[5] http://mathworld.wolfram.com/LagrangeInterpolatingPolynomial.html. Tanggal akses: 19 Desember 2016, 12.00 WIB.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Desember 2016



Muhammad Aodyra Khaidir