BAB V
PENGUJIAN

Rancangan yang telah diimplementasikan dalam bentuk nyata memerlukan suatu pengujian. Uji coba ini adalah untuk mengetahui hasil dari percobaan yang dilakukan sekaligus sebagai sarana pemunculan ide-ide bagi proses pengembangan selanjutnya. Pengujian yang dilakukan dalam bab ini adalah pengujian kebenaran perangkat lunak, antara lain :

1. Pengujian proses *generate key*
2. Pengujian proses enkripsi.
3. Pengujian proses dekripsi.
4. Analisis *cipher* citra dan *plain* citra.

5.1 Lingkungan Pengujian

Lingkungan pengujian perangkat lunak ini, memiliki spesifikasi yang sama dengan lingkungan implementasi perangkat lunak yang telah dijelaskan pada sub bab 4.4.

5.2 Tujuan Pengujian

Terdapat beberapa hal yang merupakan tujuan dari pengujian perangkat lunak dalam skripsi ini, yaitu:

1. Menguji kebenaran proses citra dapat di enkripsi dan dekripsi.
2. Menganalisa hasil *cipher* citra dan *plain* citra serta membandingkannya.

5.3 Data Pengujian

Pada tabel 5.1 menunjukkan berkas citra yang digunakan untuk menguji perangkat lunak ini.

Table 5.1 Berkas Citra Uji

No.	*Plain* Citra	Ukuran	Kunci Publik Citra g	Ukuran	Dimensi	Format
1	 baboon	192 KB	 goldhill	65 KB	200 x 200	bmp
2	 peppers	768 KB	 boat	257 KB	500 x 500	bmp

5.4 Pelaksanaan dan Hasil Pengujian

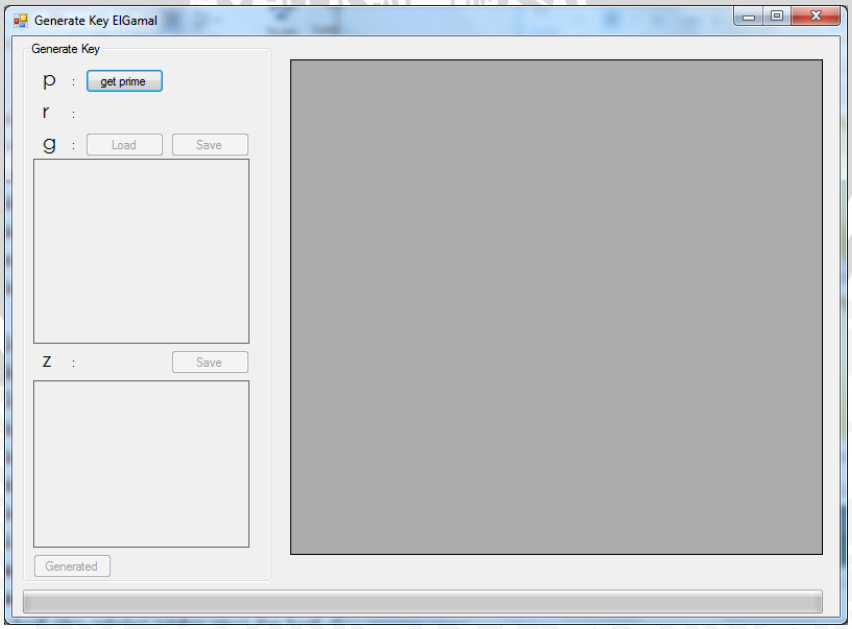
Sub bab ini menjelaskan pelaksanaan dan hasil pengujian perangkat lunak dengan menggunakan data pengujian yang dijelaskan pada tabel 5.1.

5.4.1 Pengujian Proses *Generate key*

Tujuan dari pengujian ini untuk mendapatkan nilai p, r, g, dan z sebagai kunci untuk enkripsi dan dekripsi.

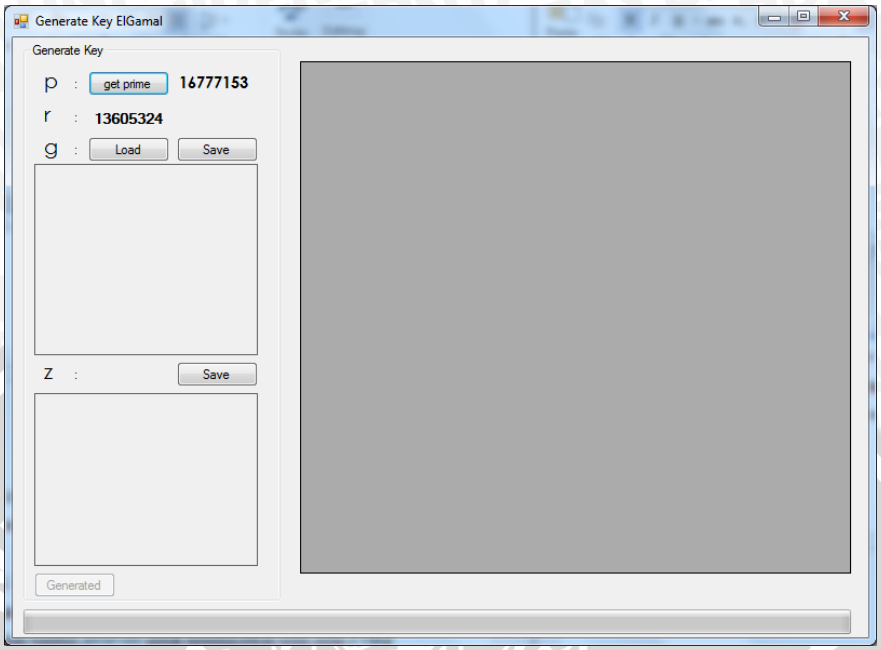
a. Prosedur Pengujian

1. Jalankan program seperti pada gambar 5.1.



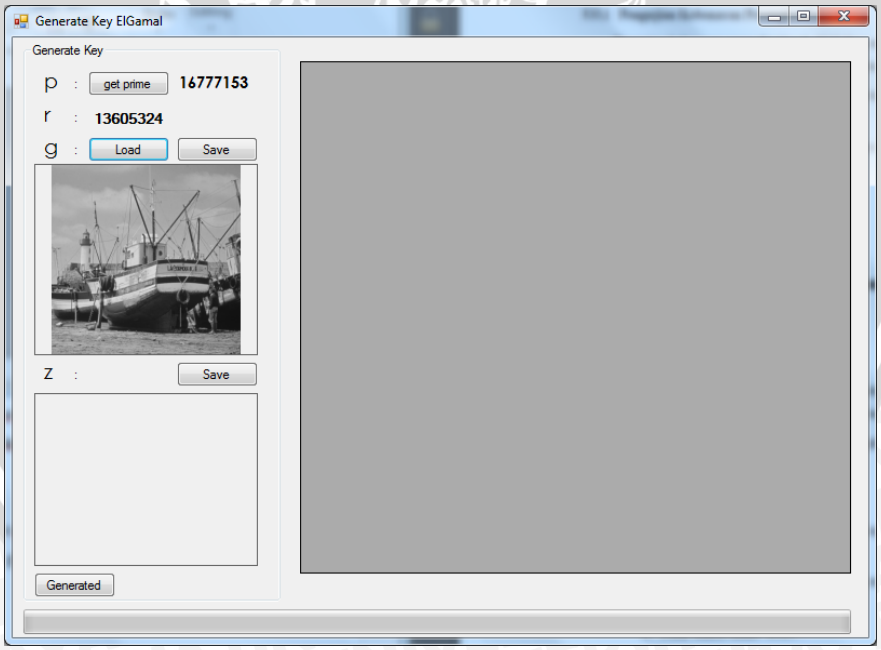
Gambar 5.1 *user interface* program *generate key*

2. Tekan tombol *get prime* untuk mendapatkan bilangan prima dan nilai r (kunci privat) secara acak. Seperti pada gambar 5.2.



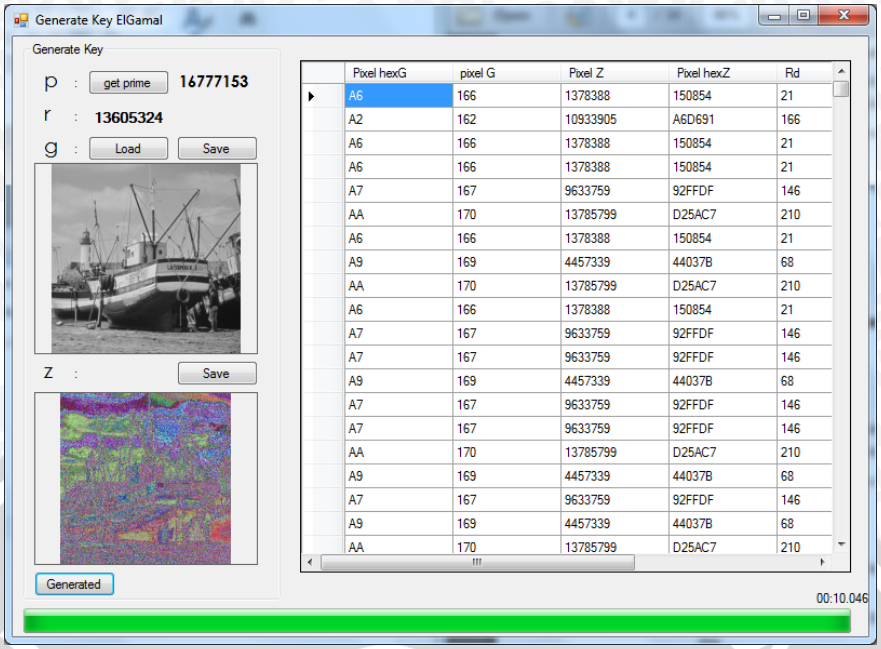
Gambar 5.2 *user interface* mendapatkan bilangan prima dan nilai r

3. Tekan tombol *load* untuk mendapatkan nilai-nilai intensitas citra g. Pada gambar 5.3 citra yang digunakan adalah citra *boat*.



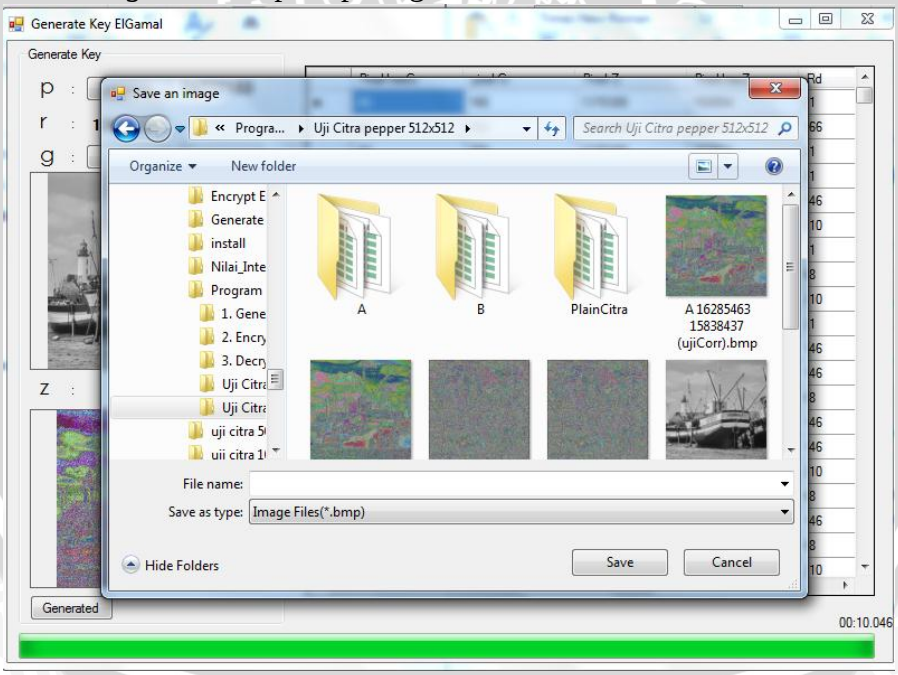
Gambar 5.3 *user interface* mendapatkan nilai intensitas citra g

4. Selanjutnya tekan tombol *generate* untuk mendapatkan nilai-nilai z yang dihitung dengan persamaan $z = g^r \bmod p$ dan diubah menjadi citra. Seperti pada gambar 5.4.



Gambar 5.4 *user interface* mendapatkan nilai citra z

5. Simpan nilai p, r, dan citra z, serta melakukan percobaan dengan *load* citra *goldhill*. Seperti pada gambar 5.5.

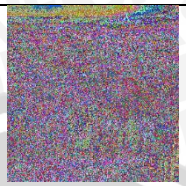
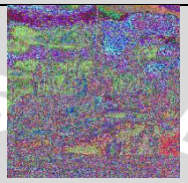


Gambar 5.5 *user interfacesave* citra z

b. Hasil Pengujian

Dari percobaan diatas diperoleh hasil seperti pada tabel 5.2.

Tabel 5.2 Hasil Proses *Generate Key*

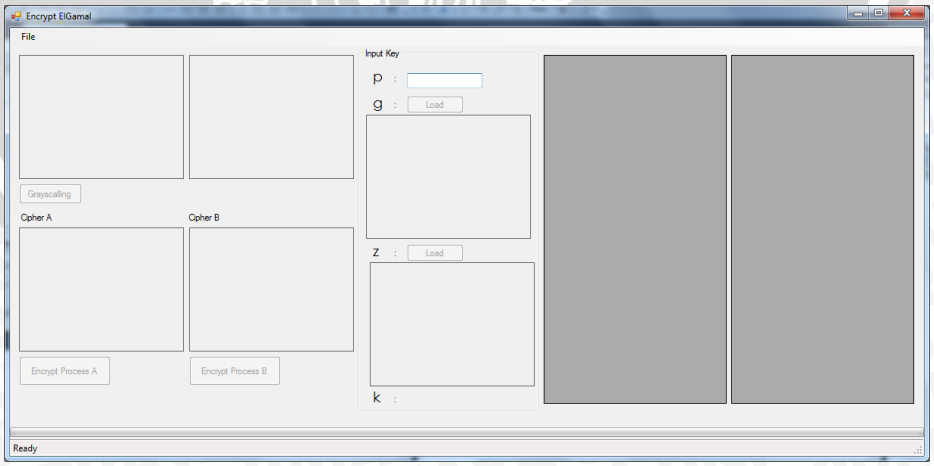
No.	Citra g	Ukuran	Citra z	Ukuran	Dimensi	Format
1	 goldhill	65.0 KB		256 KB	200 x 200	bmp
2	 boat	257 KB		1.00 MB	500 x 500	bmp

Dari hasil pengujian, Citra z yang dihasilkan ukurannya lebih besar dari citra g dan hasil citra z terlihat sedikit sama dengan citra g. Terbukti bahwa perangkat lunak yang sudah dibuat berhasil menjalankan proses *generate key* dengan baik.

5.4.2 Pengujian Proses Enkripsi

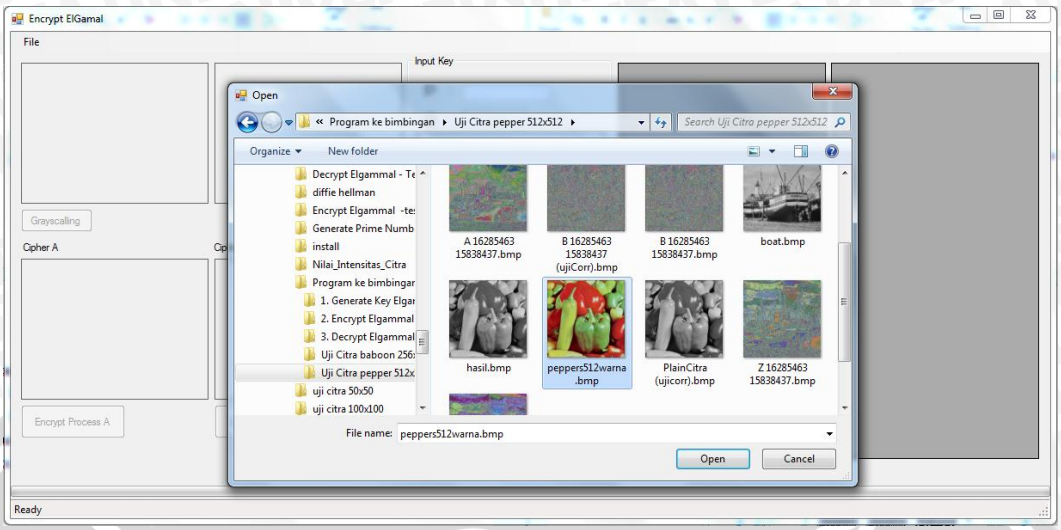
Tujuan dari pengujian ini untuk mendapatkan *cipher* citra a dan b dengan menggunakan bilangan prima dan kunci publik yaitu g dan z.

1. Prosedur PengujianJalankan program enkripsi. Seperti pada gambar 5.6.



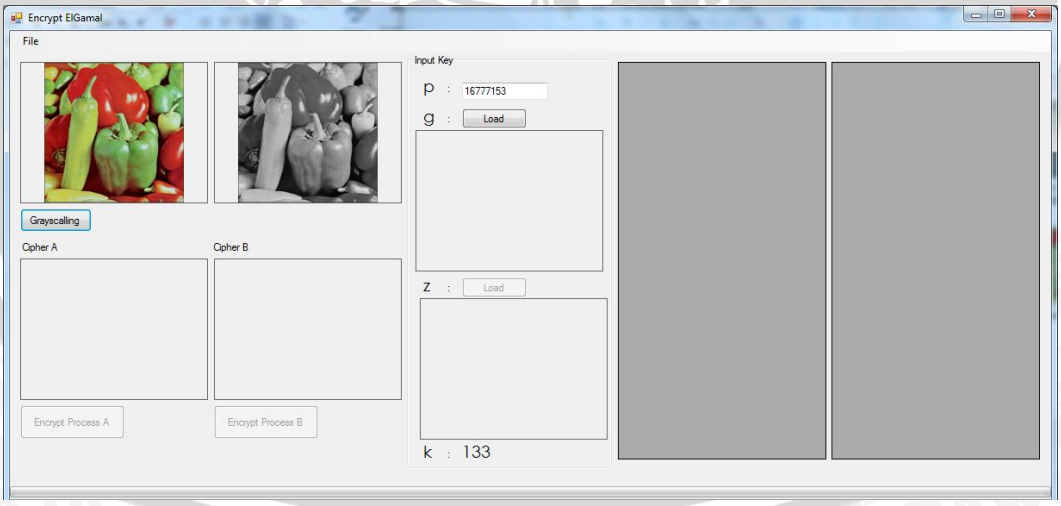
Gambar 5.6 *user interface* program enkripsi

2. Pilih *menu strip file* kemudian pilih pada *tool open* untuk memilih *plain* citra yang akan di enkripsi. Seperti pada gambar 5.7.



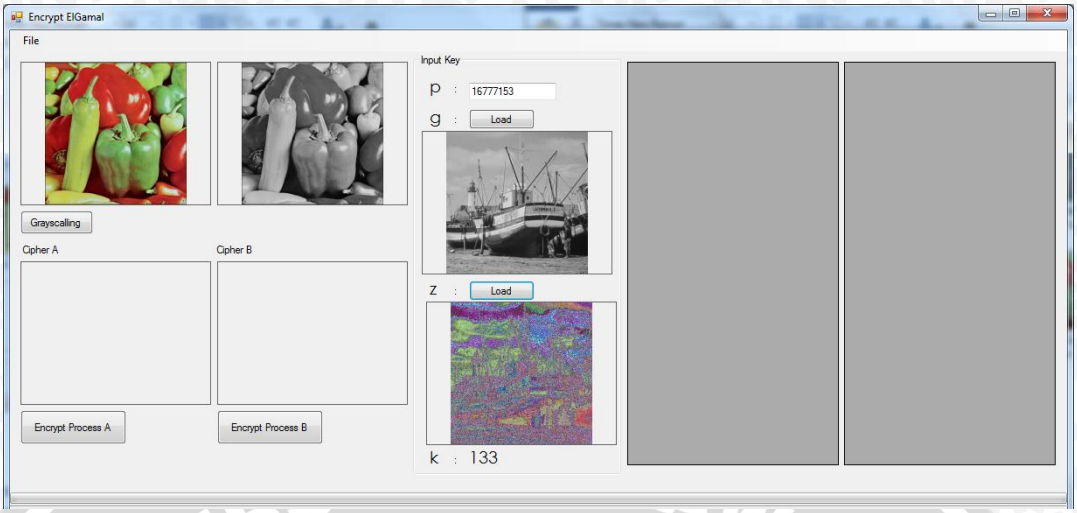
Gambar 5.7 *user interface* saat melakukan proses *open plain* citra

3. *Input* bilangan prima yang didapat dari *generate key*. Dan tekan tombol *Grayscalling* yang akan mengubah citra *true color* menjadi citra *grayscale*. Pada saat setelah *input* bilangan prima secara otomatis nilai k akan dipilih secara acak. Seperti pada gambar 5.8.



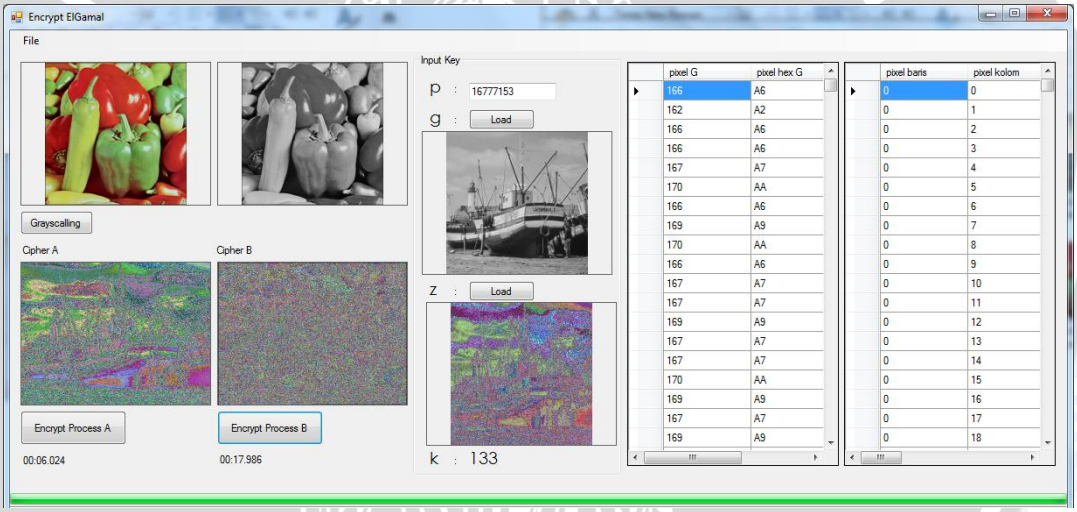
Gambar 5.8 *user interface* saat melakukan *input* bilangan prima dan proses *grayscalling*

4. Tekan tombol *Load* citra kunci publik yaitu citra g dan z yang didapat dari proses *generate key*. seperti pada gambar 5.9



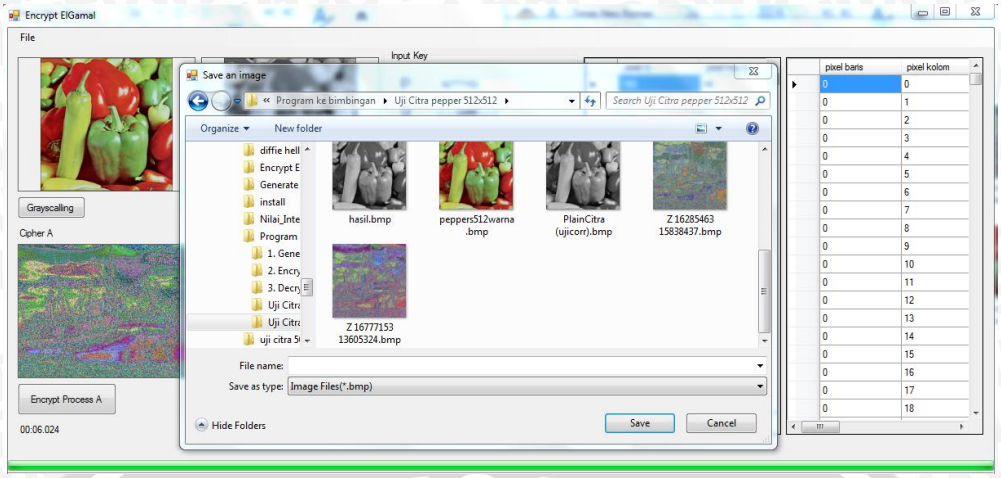
Gambar 5.9 *user interface* saat melakukan *load* citra g dan z

5. Kemudian tekan tombol *Encrypt Process A*dan *Encrypt Process B*. Seperti pada gambar 5.10.



Gambar 5.10 *user interface* hasil *Cipher A*dan *Cipher B*

6. Setelah mendapatkan *cipher* citra Adan *cipher* citra B seperti pada gambar 5.10. Lakukan percobaan untuk *input plain* citra *baboon*, kunci citra g dan z. kemudian simpan masing-masing *cipher* citra pada *save toolmenu strip*. Seperti pada gambar 5.11.

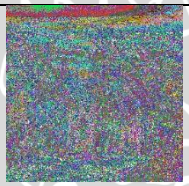
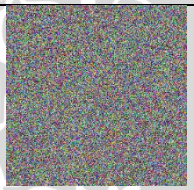
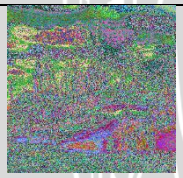
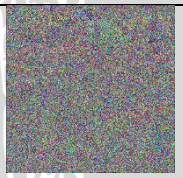


Gambar 5.11 *user interface* saat melakukan *save cipher* citra A dan B

a. Hasil Pengujian

Dari percobaan diatas diperoleh hasil seperti pada tabel 5.3.

Tabel 5.3 Hasil Proses Enkripsi

No.	*Plain* Citra	*Cipher* A	Ukuran	*Cipher* B	Ukuran	Dimensi
1.	 baboon		256 KB		256 KB	200 x 200
2.	 peppers		1.00 MB		1.00 MB	500 x 500

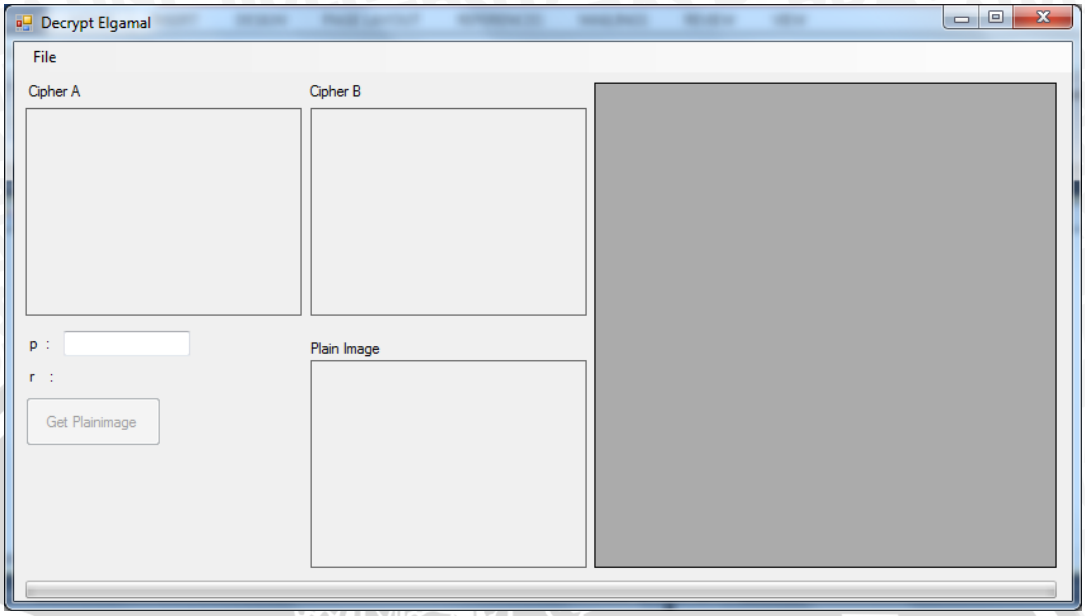
Dari hasil uji tabel 3, hasil *cipher* citra mempunyai ukuran 2 kali dari *plain* citra. Dan untuk *cipher*A masih terlihat sedikit sama dengan citra z. sedangkan *cipher* B terlihat lebih acak.

5.4.3 Pengujian Proses Dekripsi

Tujuan dari pengujian ini untuk mendapatkan kembali *plain* citra *grayscale.*

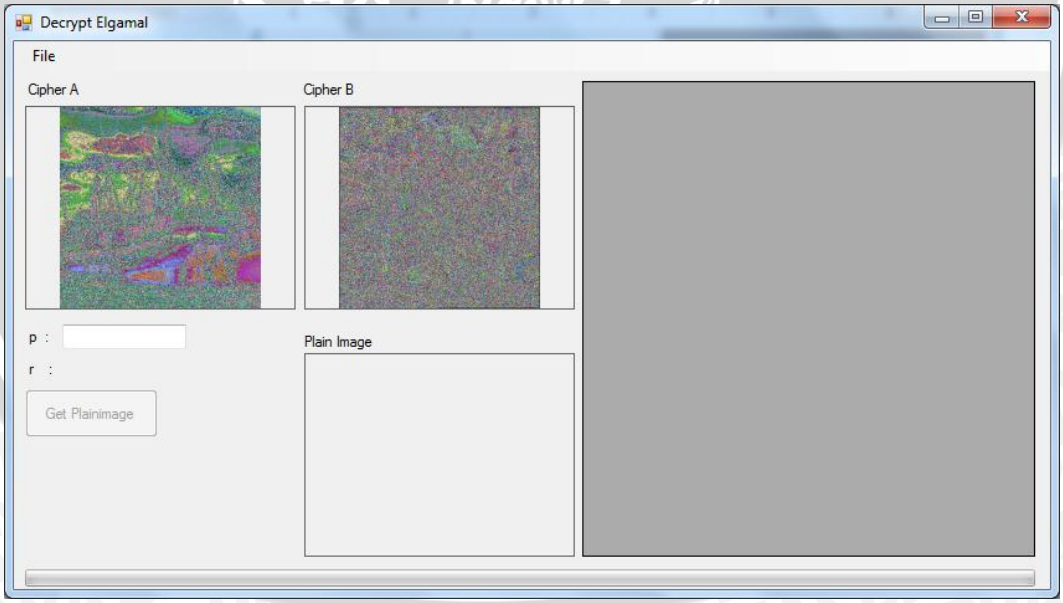
a. Prosedur Pengujian

1. Jalankan program proses dekripsi. Seperti pada gambar 5.12.



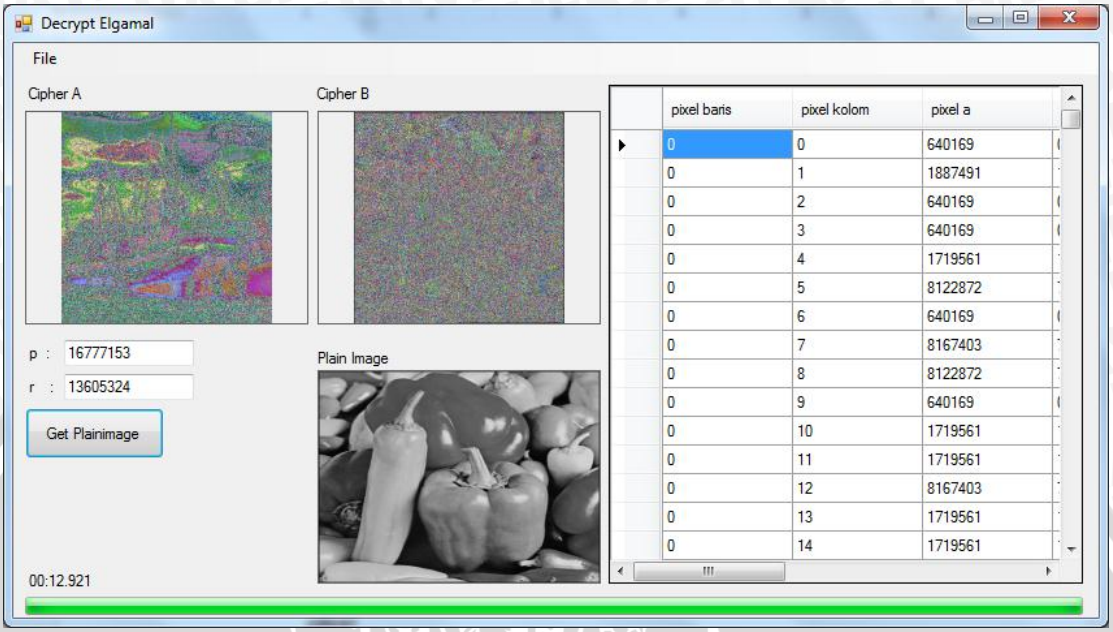
Gambar 5.12 *User Interface* Program dekripsi

2. Pilih *menu tool strip* kemudian pilih *load cipher A* dan *load cipher B*. Pilih *cipher* citra A dan B yang didapat dari hasil proses enkripsi. Seperti pada gambar 5.13.



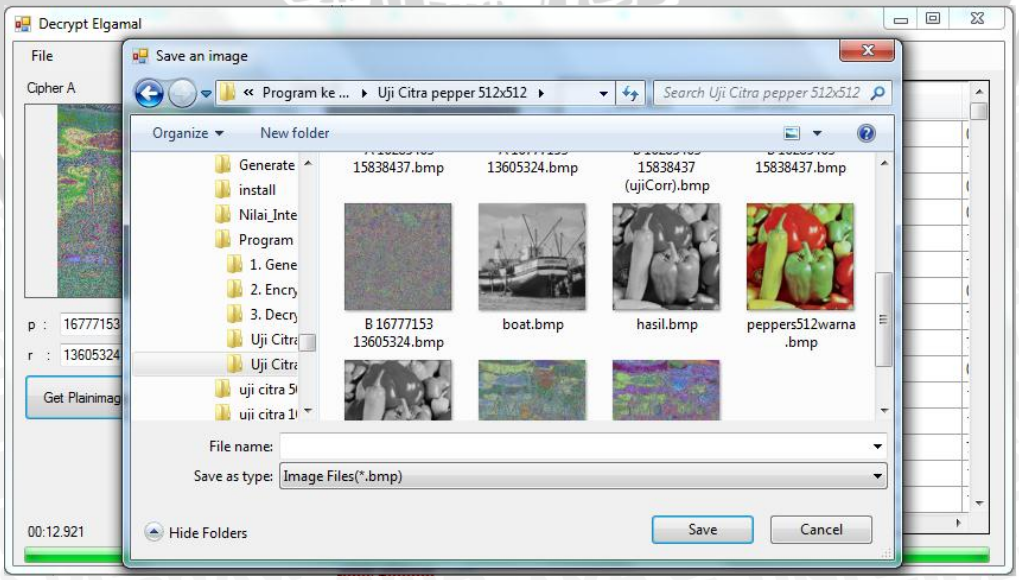
Gambar 5.13 *User Interface load cipher A*dan *cipher B*

3. *Input* bilangan prima p dan nilai kunci privat r yang didapat dari proses *generate key*. Kemudian tekan tombol *Get Plainimage*. Seperti pada gambar 5.14.



Gambar 5.14 *User Interface* hasil proses dekripsi

4. Dari gambar 5.14 didapat hasil *plain* citra. Lakukan percobaan seperti diatas dengan menggunakan *cipher* citra A dan *cipher* citra B untuk *plain* citra *baboon*. Kemudian simpan dengan menggunakan *save menu tool strip* dan pilih *save plain image*. Seperti pada gambar 5.15.

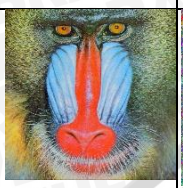
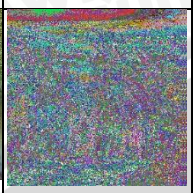
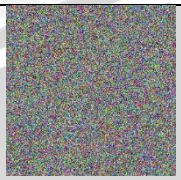
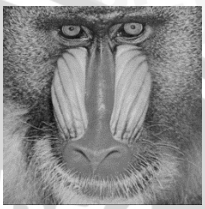
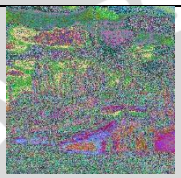
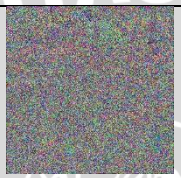
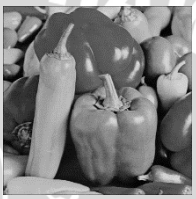


Gambar 5.15 *User Interface save menu tool strip* hasil proses dekripsi

b. Hasil Pengujian

Dari percobaan diatas didapat hasil proses dekripsi. Seperti pada tabel 5.4.

Tabel 5.4 Hasil Proses Dekripsi

No.	*Plain* Citra	*Cipher* A	*Cipher* B	Hasil *Plain* Citra	Ukuran Hasil *Plain* Citra	Dimensi
1.	 baboon				256 KB	200 x 200
2.	 peppers				1.00 MB	500 x 500

Dari hasil uji tabel 4, terlihat hasil *plain* citra dapat ditemukan kembali. Dan ukuran hasil *plain* citra lebih besar dari ukuran *plain* citra awal.

5.4.4 Analisis hasil *cipher* citra dan *plain* citra

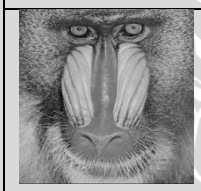
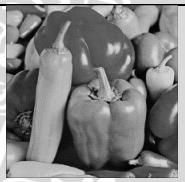
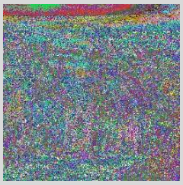
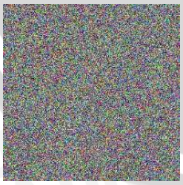
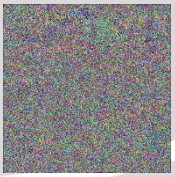
Pada bagian ini dilakukan 2 metode analisis, yaitu analisis korelasi dan analisis histogram.

5.4.4.1 Analisis Korelasi

Di dalam natural-*image, pixel-pixel* yang bertetangga memiliki hubungan linier yang kuat. Ini ditandai oleh koefisien korelasinya yang tinggi (mendekati +1 atau -1). Di dalam citra acak, korelasi antar *pixel* bertetangga tidak ada atau koefisien korelasinya mendekati nol. Enkripsi citra bertujuan membuat korelasi *pixel-pixel* yang bertetangga di dalam *cipher-image* menjadi lemah atau dengan kata lain membuat koefisien korelasinya mendekati nol. Untuk mengetahui korelasi *pixel-pixel* di

dalam *plain-image* maupun *cipher-image*, maka dihitung koefisien korelasi antara dua *pixel* bertetangga secara horizontal [f(i,j) dan f(i, j+1)], dua *pixel* bertetangga secara vertkal [f(i,j) dan f(i+1, j)], dan dua *pixel* bertetangga secara diagonal f(i,j) dan f(i+1, j+1)]. Dengan mengambil 1000 pasang *pixel* bertetangga pada setiap arah (vertikal, horizontal, dan diagonal), masing-masing pada citra *plainimage* dan *cipher-image*. Tanpa kehilangan generalisasi, analisis korelasi dilakukan pada citra *grayscale* saja. Dalam hal ini x dan y adalah nilai keabuan dari dua *pixel* bertetangga. [7] Berikut hasil perhitungan korelasi pada tabel 5.5.

Tabel 5.5 Perbandingan Koefisien Korelasi antara dua pixel bertetangga

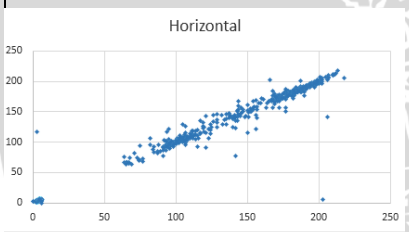
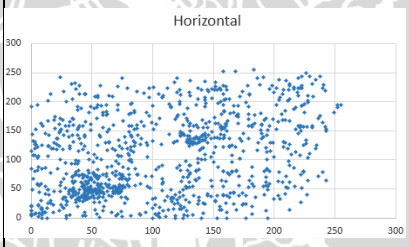
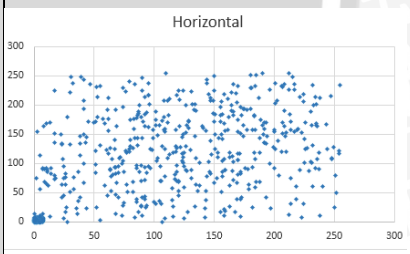
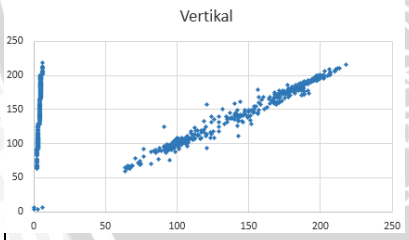
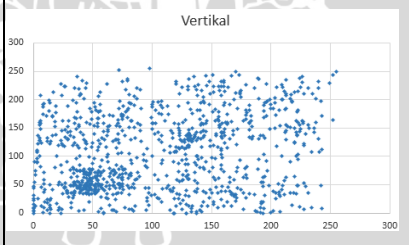
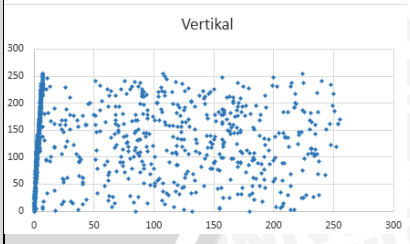
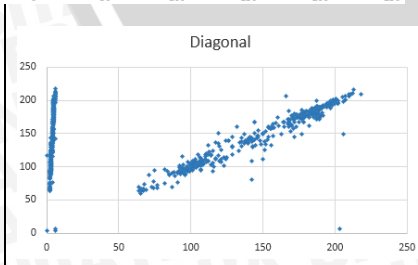
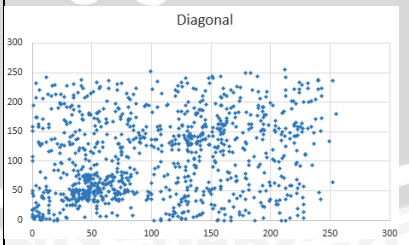
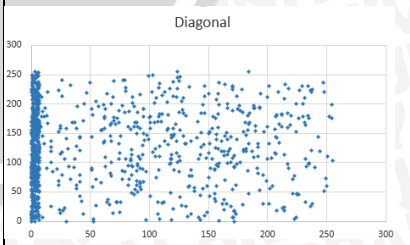
Citra	Korelasi	Citra	Korelasi
 baboon	Horizontal 0.548474 Vertikal 0.533306 Diagonal 0.394014	 peppers	Horizontal 0.99338 Vertikal 0.262402 Diagonal 0.251838
 *cipher* A	Horizontal 0.405652 Vertikal 0.311766 Diagonal 0.206862	 *cipher* A	Horizontal 0.357317 Vertikal 0.248607 Diagonal 0.239924
 *cipher* B	Horizontal 0.131774 Vertikal 0.191518 Diagonal 0.027682	 *cipher* B	Horizontal 0.704049 Vertikal 0.051994 Diagonal 0.029243

Dari tabel 5.5 dapat dilihat bahwa koefisien *plain* citra lebih mendekati 1 daripada koefisien *cipher* citra, yang

mengindikasikan *plain* citra mempunyai korelasi yang lebih kuat dari pada *cipher* citra. Pada koefisien *cipher*A masih terlihat adanya sedikit korelasi. Sedangkan *cipher* B koefisien korelasi mendekati 0 yang berarti pixel-pixel yang bertetangga tidak lagi berkolerasi.

Untuk melihat lebih jelas korelasi antara pixel-pixel bertetangga, maka tabel 5.6 memperlihatkan distribusi korelasi pixel-pixel yang bertetangga. Pada *plain* citra dapat dilihat bahwa pixel-pixel yang bertetangga nilai-nilainya saling berkolerasi. Sedangkan pada *cipher* citra A dan B nilainya tersebar dan sedikit nilai yang saling berkolerasi.

Tabel 5.6 Distribusi Korelasi Pixel-pixel Bertetangga Pada *Plain* Citra dan *Cipher* Citra dari Citra 'peppers'

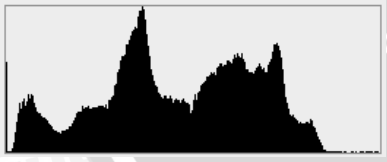
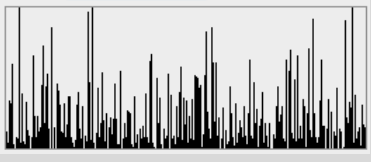
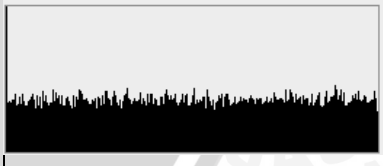
Plain Citra	*Cipher* A	*Cipher* B
		
		
		

5.4.4.2 Analisis Histogram

Di dalam bidang pengolahan citra histogram memperlihatkan distribusi nilai *pixel* di dalam sebuah citra. Histogram digunakan penyerang (*attacker*) untuk melakukan kriptanalisis dengan memanfaatkan frekuensi kemunculan *pixel* di dalam histogram. Penyerang berharap nilai *pixel* yang sering muncul di dalam *plain-image* berkorelasi dengan nilai *pixel* yang sering muncul di dalam *cipher-image*. Dengan menganalisis frekuensi kemunculan nilai *pixel*, penyerang mendeduksi kunci atau *pixel-pixel* di dalam *plain-image*.

Agar penyerang tidak dapat menggunakan histogram untuk melakukan analisis frekuensi, maka histogram *plain-image* dan histogram *cipherimage* seharusnya berbeda secara signifikan atau secara statistik tidak memiliki kemiripan. Oleh karena itu, histogram *cipher-image* seharusnya datar (*flat*) atau secara statistik memiliki distribusi (relatif) *uniform*. Distribusi yang (relatif) *uniform* pada *cipher-image* adalah sebuah indikasi bahwa algoritma enkripsi citra memiliki tingkat keamanan yang bagus. [7]

Tabel 5.7 Histogram Citra 'peppers'

Plain Citra	*Cipher* A	*Cipher* B
		

Pada tabel 5.7 memperlihatkan histogram pada *cipher* B terlihat datar (flat) atau terdistribusi uniform dan berbeda signifikan dengan histogram *plain* citra dan *cipher* A.