KIRIPTO Block Cipher

Eka Novendra Wahyunadi[1], **Ferdy Santoso**[2].

[1,2] Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika (STEI), Institut Teknologi Bandung (ITB), Jalan Ganesha 10, Bandung 40132
E-mail: 13517011@std.stei.itb.ac.id, 13517116@std.stei.itb.ac.id

Abstraksi. Cipher blok telah banyak digunakan dalam kriptografi, cipher blok yang terkenal terdiri dari DES, GOST, AES, dan masih banyak lagi. Masing-masing cipher blok masih dapat ditingkatkan kompleksitasnya agar menjadi lebih aman. Makalah ini berisi hasil desain dan pengembangan cipher blok baru yang bernama KIRIPTO Block Cipher. KIRIPTO block cipher telah mengaplikasikan prinsip-prinsip pengembangan block cipher yakni, *confusion, diffusion, iterated cipher, feistel network,* dan Kotak-S untuk menjamin keamanan cipher blok ini. KIRIPTO Block Cipher telah diuji coba dan mendapatkan hasil yang sangat baik. Makalah ini akan menjelaskan lebih lanjut mengenai detail dan penjelasan algoritma KIRIPTO serta analisis hasilnya. Akan dijelaskan juga ide pengembangan algoritma KIRIPTO untuk kedepannya.
Kata kunci: kiripto, kriptografi, cipher, blok, keamanan, enkripsi.

1. Pendahuluan

1.1. Review Block Cipher GOST

GOST (Gosudarstvenny Standard) seperti yang dijelaskan pada [1] merupakan sebuah algoritma kunci simetris yang dikembangkan oleh *Uni Soviet* pada tahun 1970, algoritma ini dibuat sebagai alternatif terhadap algoritma enkripsi Amerika Serikat DES. Algoritma ini dimulai dengan membagi blok pesan sebanyak 64-bit, lalu masing-masing blok dienkripsi dengan menggunakan jaringan Feistel. Proses enkripsinya adalah dengan mengaplikasikan 3 buah algoritma ke bagian plainteks kanan di jaringan Feistel, yakni penjumlahan modulo 2^{32}, substitusi kotak-s, dan pergeseran sirkuler ke kiri 11 bit. Setelah itu hasil dari cipherteks untuk semua blok akan digabungkan kembali dan algoritma GOST selesai.

1.2. Gagasan Perancangan Block Cipher baru

Kami ingin merancang sebuah *block cipher* baru yang menggunakan prinsip-prinsip pengembangan *block cipher*. Kami memiliki beberapa gagasan yang menurut kami dapat memperkuat *block cipher* kami sehingga menghasilkan *block cipher* yang sebanding atau lebih baik daripada *block cipher* lain yang sudah ada. Gagasan-gagasan tersebut adalah melakukan operasi XOR antara plainteks dengan round-key, menggunakan kotak-s dengan ukuran yang lebih besar, melakukan pembangkitan kotak-s secara acak dengan umpan round-key, dan juga menggunakan rotasi pada *block*. Kami yakin bahwa gagasan-gagasan ini dapat meningkatkan keamanan *block cipher* rancangan kami.

2. Studi Pustaka

2.1. Block Cipher

Cipher Blok atau Block Cipher merupakan sebuah metode kriptografi dengan cara membagi plainteks ke dalam blok-blok yang sama besar. Ukuran blok umumnya ditentukan oleh pengembang cipher blok tersebut, ukuran yang umum digunakan adalah 64-bit. Hasil dari cipher blok ini adalah sebuah cipherteks

yang sama panjangnya dengan plainteks. Ada beberapa metode enkripsi cipher blok, yakni ECB, CBC, CFB, OFB, dan Counter. Untuk makalah ini telah dipilih 3 metode yakni ECB, CBC, dan Counter yang akan dijelaskan lebih lanjut di Subbab 3.1.

2.2. Prinsip-prinsip Pengembangan Block Cipher

Telah banyak algoritma enkripsi yang dapat dengan mudah dipecahkan. Sehingga dalam mengembangkan sebuah cipher blok ada beberapa prinsip yang harus diterapkan agar cipher blok kita memiliki keamanan yang lebih baik. Ada 4 prinsip penting yang harus diterapkan, yakni:

2.2.1. Confusion dan Diffusion

Seperti yang dijelaskan oleh Shannon [2], Confusion merupakan sebuah prinsip yang berguna untuk menyembunyikan apapun yang ada antara plainteks, cipherteks, dan kunci. Prinsip ini berguna untuk menyamarkan pola-pola statistik yang ada pada cipherteks. Contoh pengaplikasian prinsip confusion adalah dengan menggunakan substitusi kotak-s. Diffusion merupakan sebuah prinsip yang bertujuan untuk menyebarkan pengaruh satu bit plainteks atau kunci ke sebanyak mungkin cipherteks. Prinsip ini dapat diterapkan dengan menggunakan mode enkripsi CBC dan dengan menggunakan operasi permutasi.

2.2.2. Iterated Cipher

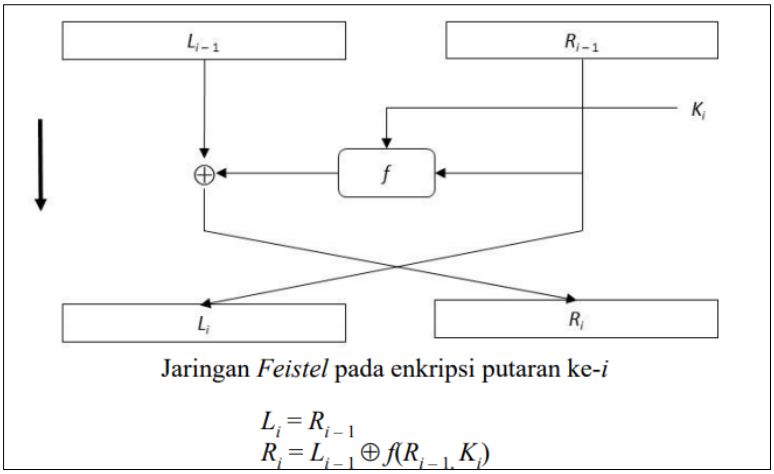
Iterated Cipher atau Cipher berulang merupakan sebuah fungsi transformasi sederhana yang mengubah plainteks menjadi cipherteks berulang kali. Hal ini ditujukan agar hasil enkripsi menjadi lebih aman. Rumus formal iterated cipher adalah sebagai berikut :

$$Ci = f(C_{i-1}, K_i)$$

dengan:
i = 1, 2, …, r (r adalah jumlah putaran).
K_i = upa-kunci (subkey) pada putaran ke-i
f = fungsi transformasi (berisi operasi matematika)

2.2.3. Jaringan Feistel

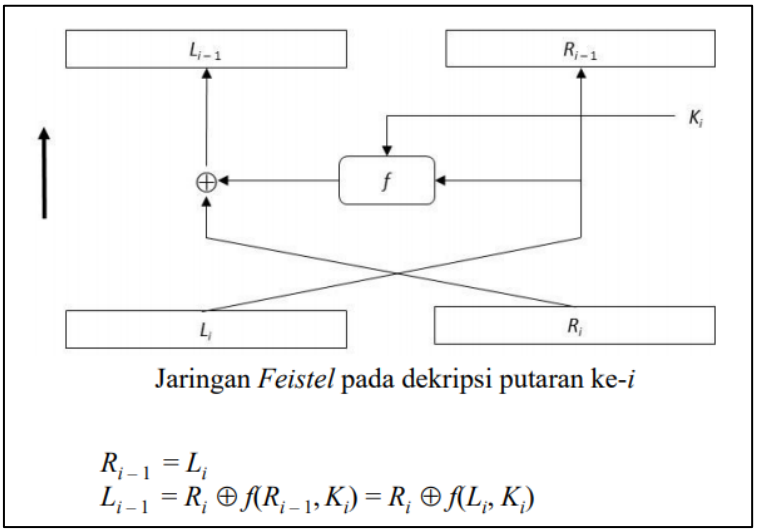
Jaringan Feistel berguna agar proses enkripsi dan dekripsi menjadi bersifat *reversible* sehingga proses enkripsi dan dekripsi hanya membutuhkan 1 buah fungsi saja. Berikut ini adalah diagram serta rumus formal dari jaringan Feistel:



Jaringan *Feistel* pada enkripsi putaran ke-*i*

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

Gambar 2.1. Enkripsi pada Jaringan Feistel



Gambar 2.2. Dekripsi pada Jaringan Feistel

2.2.4. Kotak-S

Kotak-S atau S-Box merupakan sebuah matriks yang berisi substitusi sederhana yang memetakan satu atau lebih bit dengan bit lain. hal ini bertujuan agar algoritma enkripsi menjadi lebih kompleks. berikut ini adalah contoh Kotak-S untuk GOST:

Kotak-S 1:															
4	10	9	2	13	8	0	14	6	11	1	12	7	15	5	3
Kotak-S 2:															
14	11	4	12	6	13	15	10	2	3	8	1	0	7	5	9
Kotak-S 3:															
5	8	1	13	10	3	4	2	14	15	12	7	6	0	9	11
Kotak-S 4:															
7	13	10	1	0	8	9	15	14	4	6	12	11	2	5	3
Kotak-S 5															
6	12	7	1	5	15	13	8	4	10	9	14	0	3	11	2
Kotak-S 6:															
4	11	10	0	7	2	1	13	3	6	8	5	9	12	14	14
Kotak-S 7:															
13	11	4	1	3	15	5	9	0	10	14	7	6	8	2	12
Kotak-S 8:															
1	15	13	0	5	7	10	4	9	2	3	14	6	11	8	12

Gambar 2.3. Kotak-S untuk Algoritma GOST

2.3. Operasi Matematika

2.3.1. Permutasi

Permutasi adalah susunan yang dapat dibentuk dari suatu kumpulan objek yang diambil sebagian atau seluruhnya dengan memperhatikan urutannya.

2.3.2. XOR

XOR adalah operasi biner dimana bit yang dihasilkan adalah satu jika dan hanya jika tepat satu dari bit yang dioperasikan bernilai satu.

2.3.3. Rotasi

Rotasi adalah pemindahan benda dengan melakukan perputaran pada benda tersebut.

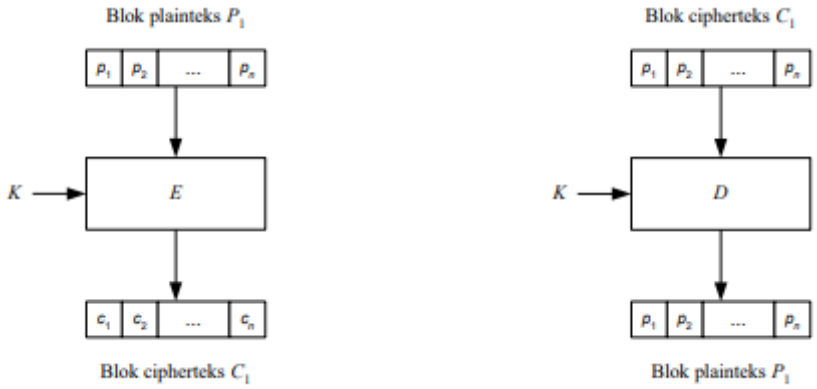
3. KIRIPTO Block Cipher

3.1. Mode enkripsi dan dekripsi

Dalam pengerjaan makalah ini kami menggunakan 3 mode enkripsi dan dekripsi untuk cipher blok kami, yakni ECB, CBC dan Counter.

3.1.1. ECB

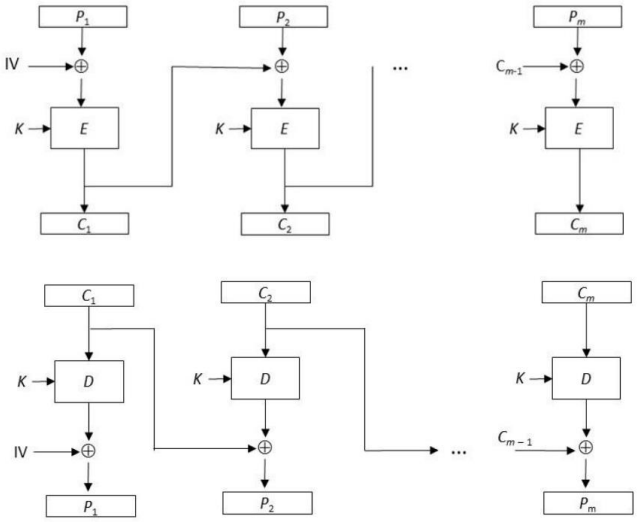
Metode ECB kami berjalan sebagai berikut, dari plainteks yang telah dibagi menjadi blok 64-bit, masing-masing blok plainteks dimasukkan ke dalam algoritma enkripsi yang akan menghasilkan cipherteks. Untuk dekripsi proses yang sama akan dilakukan, hanya saja masukan berupa cipherteks dan keluaran berupa plainteks. Berikut ini adalah diagram algoritma ECB untuk 1 blok plainteks:



Gambar 3.1. Mode Enkripsi dan Dekripsi ECB untuk 1 blok plainteks

3.1.2. CBC

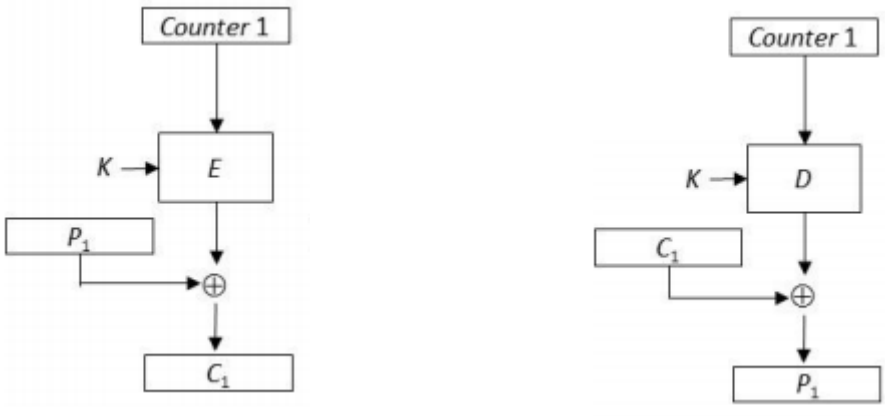
Dari plainteks yang telah dibagi menjadi 64-bit, masing-masing blok plainteks dilakukan operasi XOR terlebih dahulu dengan hasil cipherteks sebelumnya, dan setelah itu dimasukkan ke dalam algoritma enkripsi yang akan menghasilkan cipherteks. Untuk plainteks pertama, digunakan *initialization vector* sebagai pengganti hasil cipherteks sebelumnya yang belum ada. *Initialization vector* dihasilkan secara acak dengan *seed* key. Untuk dekripsi proses yang sama akan dilakukan, hanya saja masukan berupa cipherteks dan keluaran berupa plainteks, serta operasi XOR dilakukan setelah cipherteks diubah menjadi plainteks. Berikut ini adalah diagram algoritma enkripsi dan dekripsi CBC:



Gambar 3.2. Mode Enkripsi dan Dekripsi CBC

3.1.3. Counter

Metode Counter kami berjalan sebagai berikut, pada awalnya kami melakukan pembangkitan blok counter, blok counter kami dibangkitkan secara acak menggunakan umpan kunci enkripsi dan dekripsi. Counter yang telah dibangkitkan akan dimasukkan ke dalam algoritma enkripsi. Hasil enkripsi lalu akan dilakukan operasi XOR dengan plainteks yang telah dibagi menjadi blok 64-bit. Hasil XOR tersebut akan menjadi cipherteks kami. Untuk dekripsi akan dilakukan proses yang sama dengan enkripsi hanya saja masukan berupa cipherteks dan keluaran berupa plainteks. Berikut ini adalah diagram algoritma Counter untuk 1 blok plainteks:



Gambar 3.3. Mode Enkripsi dan Dekripsi Counter untuk 1 blok plainteks

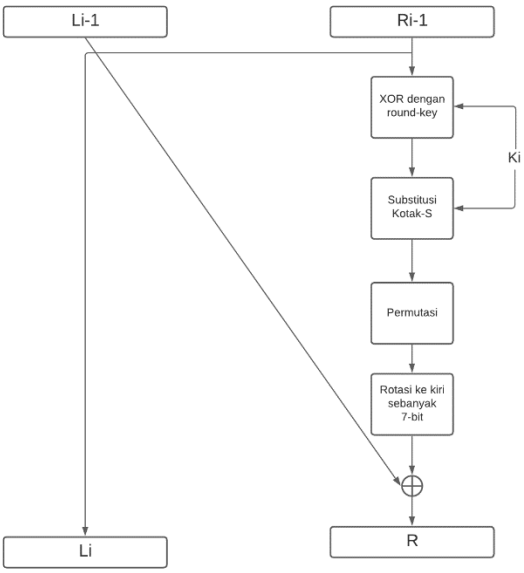
3.2. Algoritma Enkripsi

3.2.1. Preprocessing

Sebelum dilakukan pemrosesan pada plainteks, perlu dilakukan pembagian plainteks menjadi blok-blok yang lebih kecil berukuran 64-bit karena pemrosesan dilakukan pada blok-blok berukuran 64-bit tersebut. Lalu dilakukan pembangkitan round-key untuk seluruh blok. Round-key pertama dibangkitkan dengan cara membagi key 64-bit menjadi 2 bagian 32-bit, lalu kedua bagian tersebut dilakukan operasi XOR. Round-key blok selanjutnya adalah round-key blok sebelumnya yang dilakukan rotasi ke kiri sebanyak 1-bit.

3.2.2. Jaringan Feistel

Kami menggunakan jaringan Feistel dalam algoritma enkripsi kita agar kita hanya perlu membuat 1 fungsi untuk enkripsi dan dekripsi. Jaringan Feistel kami memiliki struktur yang sama dengan struktur jaringan Feistel pada umunya yang telah dijelaskan pada bab 2. Fungsi *f* yang kami gunakan memiliki 4 operasi, yakni, operasi XOR, operasi substitusi Kotak-S, operasi permutasi, dan operasi rotasi. Berikut ini adalah diagram jaringan Feistel algoritma kami bersama dengan isi fungsi f kami:

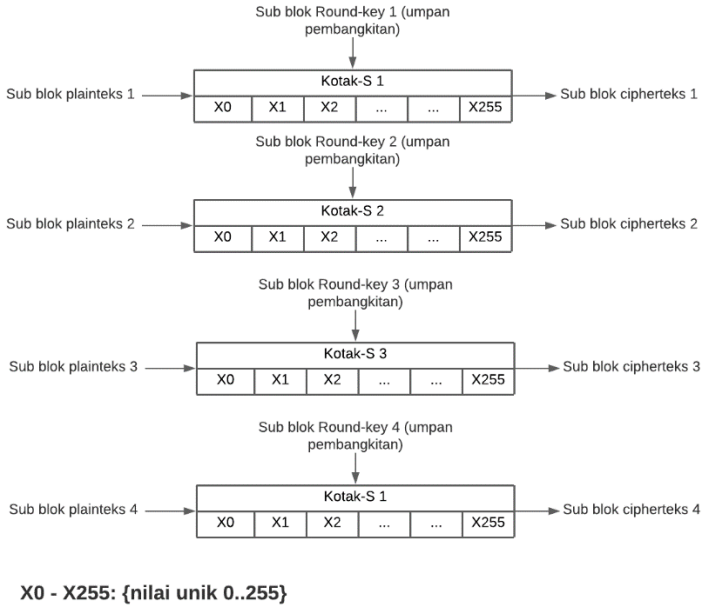


Gambar 3.4. Jaringan Feistel Algoritma Block Cipher KIRIPTO

Masing-masing bagian fungsi *f* tersebut akan dijelaskan pada bagian selanjutnya.

3.2.2.1. XOR
Dilakukan operasi XOR antara plainteks blok 32-bit dengan round-key blok 32-bit.

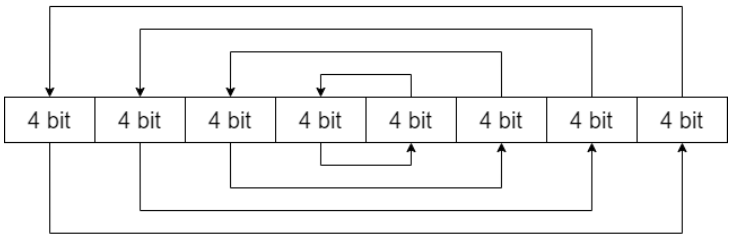
3.2.2.2. Kotak-S
Kami melakukan permutasi menggunakan kotak-s dengan cara membagi plainteks menjadi bagian-bagian 8 bit, sehingga plainteks 32-bit tersebut dipisahkan menjadi 4 bagian yang masing-masing berukuran 8-bit. Blok round-key juga dibagi menjadi bagian 8-bit. Kotak-S berukuran 1 dimensi, akan terdapat 4 kotak-s yang berukuran 8-bit. Sehingga untuk masing-masing kotak-s akan ada pasangan round-key nya yang akan digunakan untuk membangkitkan Kotak-S tersebut. Lalu, Kotak-S dibangkitkan secara acak dengan umpan round-key tersebut. Sehingga untuk masing-masing subblok plainteks akan ada pasangan kotak-s nya masing-masing. Lalu plainteks disubstitusikan dengan cara mengubah nilai plainteks subblok ke-i dengan nilai elemen kotak-s pada indeks nilai plainteks tersebut. Akan diberikan gambar berikut untuk memperjelas algoritma substitusi kotak-s:



Gambar 3.5. Kotak-S Algoritma Block Cipher KIRIPTO

3.2.2.3. Permutasi

Dilakukan permutasi pada blok 32-bit dengan cara membaginya menjadi blok – blok kecil dengan ukuran 4-bit dan disusun ulang dengan urutan terbalik. Berikut ini adalah ilustrasi yang menunjukkan permutasi yang dilakukan:



Gambar 3.6 Penyusunan ulang blok-blok kecil 4-bit

3.2.2.4. Rotasi

Dilakukan rotasi pada blok 32 bit, dengan cara melakukan pergeseran sejauh 7-bit ke kiri secara siklik.

3.2.3. Postprocessing

Setelah semua blok selesai diproses, blok-blok tersebut digabungkan kembali untuk membentuk rangkaian bit penyusun cipherteks.

4. Eksperimen dan Analisis Hasil

4.1. Eksperimen

4.1.1. Pesan Teks Dengan Mode ECB

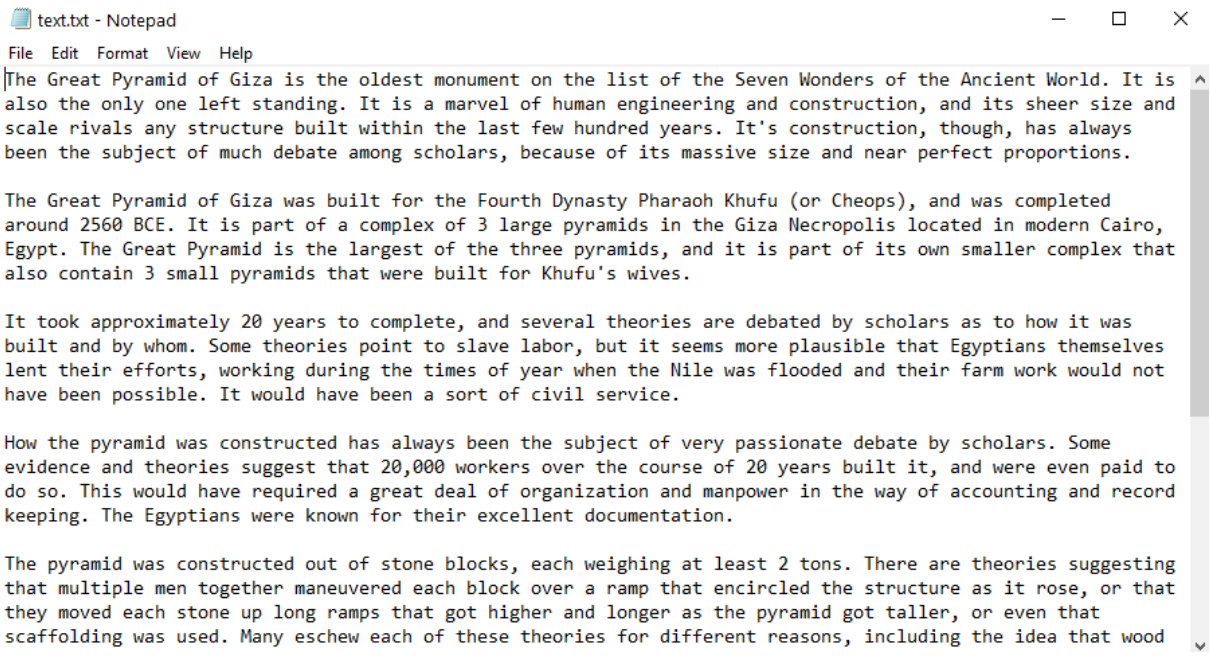
Dilakukan eksperimen dengan pesan teks berukuran 4KB dengan mode ECB menggunakan kunci “haloduni”. Didapatkan waktu pemrosesan 7.27 detik untuk menghasilkan cipherteks dari pesan teks tersebut dan waktu pemrosesan 7.11 detik untuk menghasilkan pesan teks asal dari cipherteks. Berikut ini adalah tangkapan layar dari hasil eksperimen:

```
PS C:\Users\ferdy\Documents\ITB Jurusan IF\Tahun Keempat\Kriptografi\Tugas Makalah\src\Kiripto-Block-Cipher> py main.py
Input mode ('ecb', 'cbc', 'counter'):
>> ecb
Input filename:
>> text.txt
Input key (8 characters):
>> haloduni
Input option ('encrypt' or 'decrypt'):
>> encrypt

Processing...
Process Completed.

Time Needed : 7.271273136138916 second
Output filename:
>> cipherteks.txt
PS C:\Users\ferdy\Documents\ITB Jurusan IF\Tahun Keempat\Kriptografi\Tugas Makalah\src\Kiripto-Block-Cipher>
```

Gambar 4.1. Enkripsi teks dengan mode ECB

```
PS C:\Users\ferdy\Documents\ITB Jurusan IF\Tahun Keempat\Kriptografi\Tugas Makalah\src\Kiripto-Block-Cipher> py main.py
                        Input mode ('ecb', 'cbc', 'counter'):
>> ecb
Input filename:
>> cipherteks.txt
Input key (8 characters):
>> haloduni
Input option ('encrypt' or 'decrypt'):
>> decrypt

Processing...
Process Completed.

Time Needed : 7.1093199253082275 second
Output filename:
>> hasil.txt
PS C:\Users\ferdy\Documents\ITB Jurusan IF\Tahun Keempat\Kriptografi\Tugas Makalah\src\Kiripto-Block-Cipher>
```

Gambar 4.2. Dekripsi teks dengan mode ECB



Gambar 4.3. File *teks.txt*

4.1.2. Pesan Teks Dengan Mode CBC

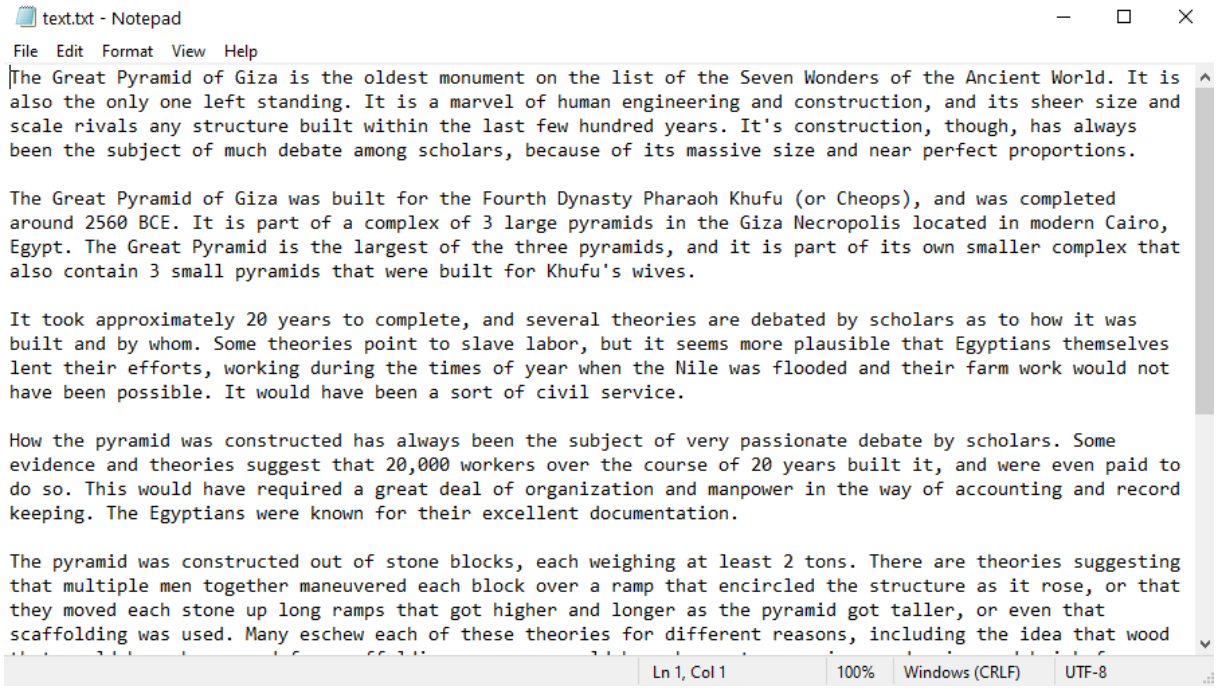
Dilakukan eksperimen dengan pesan teks berukuran 4KB dengan mode CBC menggunakan kunci "haloduni". Didapatkan waktu pemrosesan 7.25 detik untuk menghasilkan cipherteks dari pesan teks tersebut dan waktu pemrosesan 7.53 detik untuk menghasilkan pesan teks asal dari cipherteks. Berikut ini adalah tangkapan layar dari hasil eksperimen:

```
PS C:\Users\ferdy\Documents\ITB Jurusan IF\Tahun Keempat\Kriptografi\Tugas Makalah\src\Kiripto-Block-Cipher> py main.py
Input mode ('ecb', 'cbc', 'counter'):
>> cbc
Input filename:
>> text.txt
Input key (8 characters):
>> haloduni
Input option ('encrypt' or 'decrypt'):
>> encrypt

Processing...
Process Completed.

Time Needed : 7.251927375793457 second
Output filename:
>> cipherteks.txt
```

Gambar 4.4. Enkripsi teks dengan mode CBC

```
PS C:\Users\ferdy\Documents\ITB Jurusan IF\Tahun Keempat\Kriptografi\Tugas Makalah\src\Kiripto-Block-Cipher> py main.py
Input mode ('ecb', 'cbc', 'counter'):
>> cbc
Input filename:
>> cipherteks.txt
Input key (8 characters):
>> haloduni
Input option ('encrypt' or 'decrypt'):
>> decrypt

Processing...
Process Completed.

Time Needed : 7.527095794677734 second
Output filename:
>> hasil.txt
```

Gambar 4.5. Dekripsi teks dengan mode CBC



Gambar 4.6. File *teks.txt*

4.1.3. Pesan Gambar Dengan Mode ECB

Dilakukan eksperimen dengan pesan gambar berukuran 14KB dengan mode ECB menggunakan kunci “sasageyo”. Didapatkan waktu pemrosesan 51 detik untuk menghasilkan cipherteks dari pesan gambar tersebut dan waktu pemrosesan 31.62 detik untuk menghasilkan pesan gambar asal dari cipherteks. Berikut ini adalah tangkapan layar dari hasil eksperimen:

```
PS C:\Users\ferdy\Documents\ITB Jurusan IF\Tahun Keempat\Kriptografi\Tugas Makalah\src\Kiripto-Block-Cipher> py main.py
Input mode ('ecb', 'cbc', 'counter'):
>> ecb
Input filename:
>> katana.jpg
Input key (8 characters):
>> sasageyo
Input option ('encrypt' or 'decrypt'):
>> encrypt

Processing...
Process Completed.

Time Needed : 51.00058650970459 second
Output filename:
>> cipherteks.jpg
```

Gambar 4.7. Enkripsi gambar dengan mode ECB

```
PS C:\Users\ferdy\Documents\ITB Jurusan IF\Tahun Keempat\Kriptografi\Tugas Makalah\src\Kiripto-Block-Cipher> py main.py
Input mode ('ecb', 'cbc', 'counter'):
>> ecb
Input filename:
>> cipherteks.jpg
Input key (8 characters):
>> sasageyo
Input option ('encrypt' or 'decrypt'):
>> decrypt

Processing...
Process Completed.

Time Needed : 31.62018632888794 second
Output filename:
>> hasil.jpg
```

Gambar 4.8. Dekripsi gambar dengan mode ECB

Gambar 4.9. File *katana.jpg*

4.1.4. Pesan Gambar Dengan Mode Counter

Dilakukan eksperimen dengan pesan gambar berukuran 14KB dengan mode Counter menggunakan kunci “sasageyo”. Didapatkan waktu pemrosesan 30.97 detik untuk menghasilkan cipherteks dari pesan gambar tersebut dan waktu pemrosesan 31.88 detik untuk menghasilkan pesan gambar asal dari cipherteks. Berikut ini adalah tangkapan layar dari hasil eksperimen:

```
PS C:\Users\ferdy\Documents\ITB Jurusan IF\Tahun Keempat\Kriptografi\Tugas Makalah\src\Kiripto-Block-Cipher> py main.py
Input mode ('ecb', 'cbc', 'counter'):
>> counter
Input filename:
>> katana.jpg
Input key (8 characters):
>> sasageyo
Input option ('encrypt' or 'decrypt'):
>> encrypt

Processing...
Process Completed.

Time Needed : 30.967087745666504 second
Output filename:
>> cipherteks.jpg
```

Gambar 4.10. Enkripsi gambar dengan mode Counter

```
PS C:\Users\ferdy\Documents\ITB Jurusan IF\Tahun Keempat\Kriptografi\Tugas Makalah\src\Kiripto-Block-Cipher> py main.py
Input mode ('ecb', 'cbc', 'counter'):
>> counter
Input filename:
>> cipherteks.jpg
Input key (8 characters):
>> sasageyo
Input option ('encrypt' or 'decrypt'):
>> decrypt

Processing...
Process Completed.

Time Needed : 31.878237009048462 second
Output filename:
>> hasil.jpg
```

Gambar 4.11. Dekripsi gambar dengan mode Counter

Gambar 4.12. File *katana.jpg*

4.1.5. Pengubahan 1-byte kunci

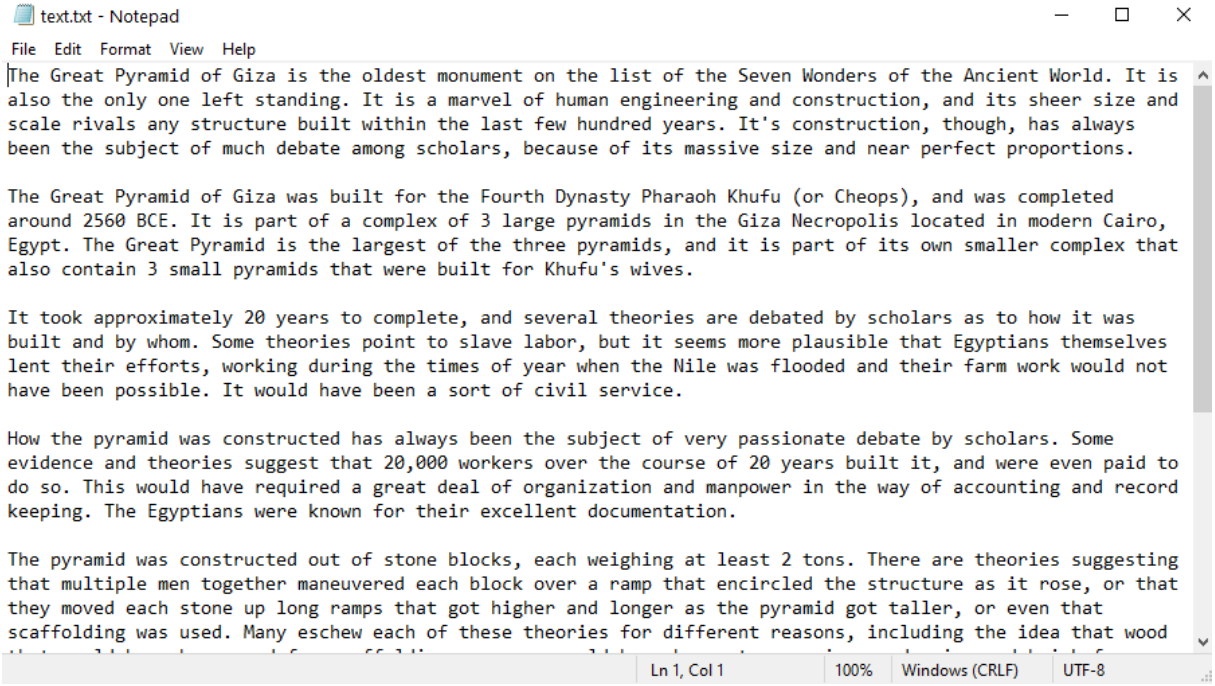
Dilakukan eksperimen dengan pesan teks berukuran 4KB dengan mode ECB menggunakan kunci "haloduni" untuk enkripsi dan "maloduni" untuk dekripsi. Didapatkan waktu pemrosesan 7.51 detik untuk menghasilkan cipherteks dari pesan teks tersebut dan waktu pemrosesan 10.16 detik untuk menghasilkan pesan teks asal dari cipherteks. Berikut ini adalah tangkapan layar dari hasil eksperimen:

```
PS C:\Users\ferdy\Documents\ITB Jurusan IF\Tahun Keempat\Kriptografi\Tugas Makalah\src\Kiripto-Block-Cipher> py main.py
Input mode ('ecb', 'cbc', 'counter'):
>> ecb
Input filename:
>> text.txt
Input key (8 characters):
>> haloduni
Input option ('encrypt' or 'decrypt'):
>> encrypt

Processing...
Process Completed.

Time Needed : 7.506096363067627 second
Output filename:
>> cipherteks.txt
```

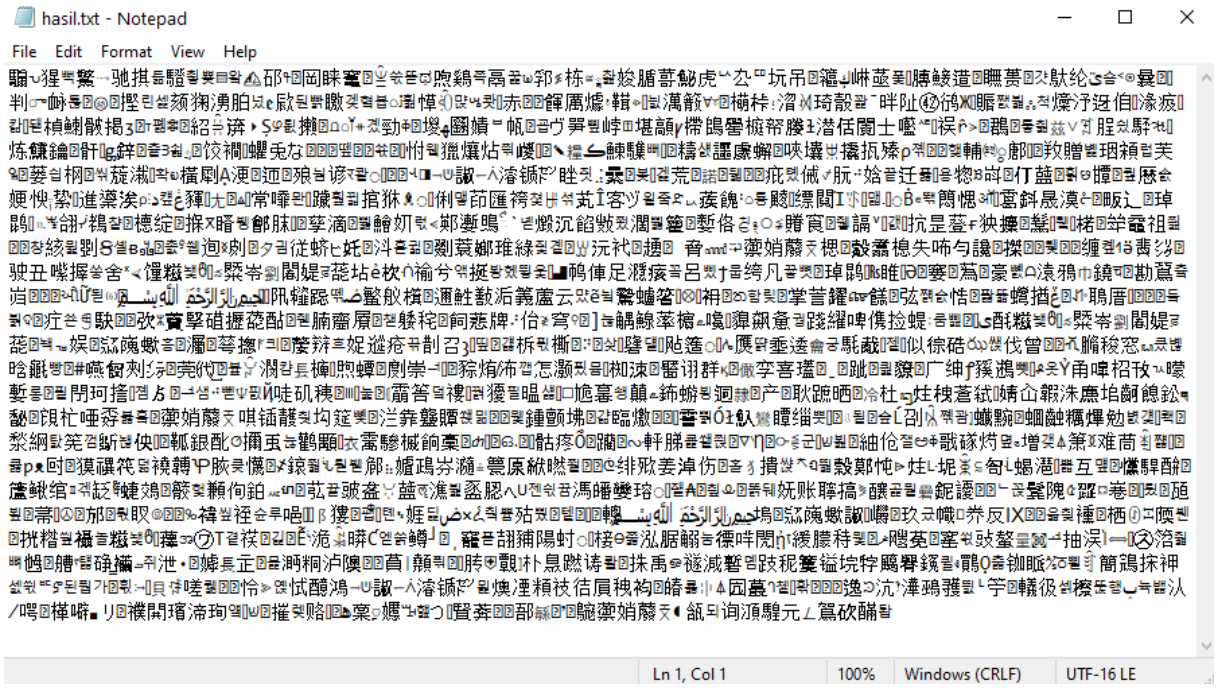
Gambar 4.13. Enkripsi teks dengan mode ECB

```
PS C:\Users\ferdy\Documents\ITB Jurusan IF\Tahun Keempat\Kriptografi\Tugas Makalah\src\Kiripto-Block-Cipher> py main.py
Input mode ('ecb', 'cbc', 'counter'):
>> ecb
Input filename:
>> cipherteks.txt
Input key (8 characters):
>> maloduni
Input option ('encrypt' or 'decrypt'):
>> decrypt

Processing...
Process Completed.

Time Needed : 10.158752679824829 second
Output filename:
>> hasil.txt
```

Gambar 4.14. Dekripsi teks dengan mode ECB dengan penggantian kunci 1-byte di depan



Gambar 4.15. File *teks.txt* (sebelum enkripsi)



Gambar 4.16. File *hasil.txt* (setelah dekripsi)

4.1.6. Pengubahan 1-byte cipherteks

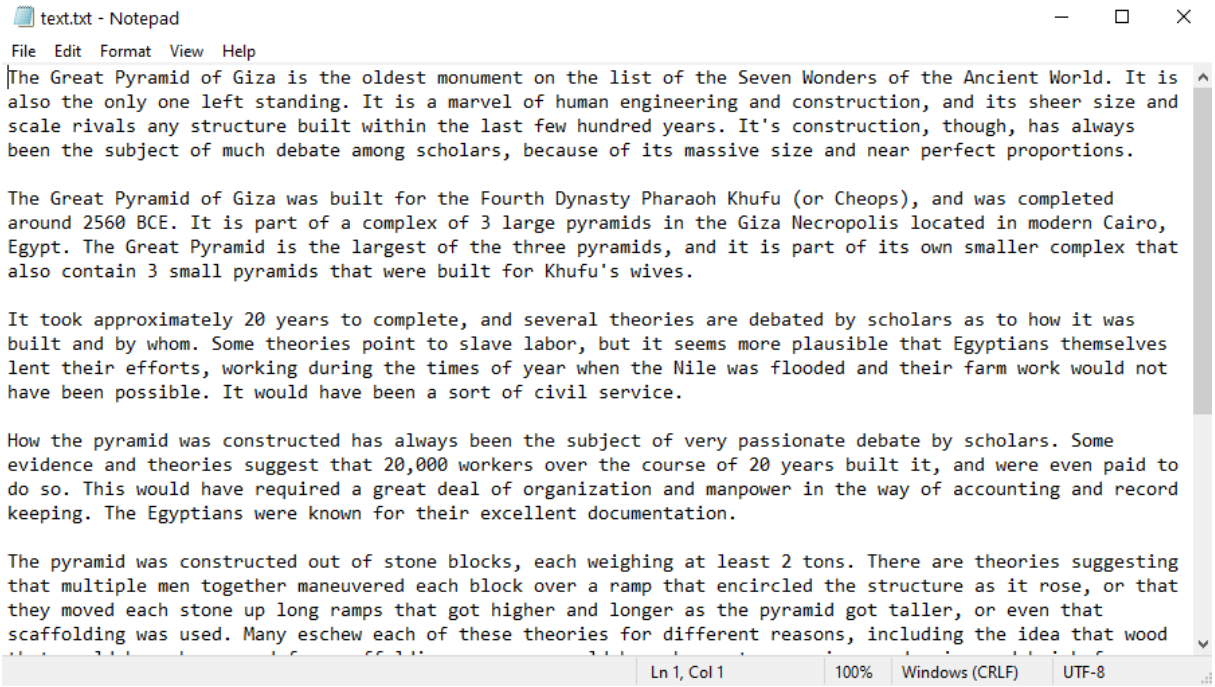
Dilakukan eksperimen dengan pesan teks berukuran 4KB dengan mode CBC menggunakan kunci "haloduni". Dilakukan pengubahan pada cipherteks sebelum di dekripsi. Didapatkan waktu pemrosesan 7.53 detik untuk menghasilkan cipherteks dari pesan teks tersebut dan waktu pemrosesan 13.17 detik untuk menghasilkan pesan teks asal dari cipherteks. Berikut ini adalah tangkapan layar dari hasil eksperimen:

```
PS C:\Users\ferdy\Documents\ITB Jurusan IF\Tahun Keempat\Kriptografi\Tugas Makalah\src\Kiripto-Block-Cipher> py main.py
Input mode ('ecb', 'cbc', 'counter'):
>> cbc
Input filename:
>> text.txt
Input key (8 characters):
>> haloduni
Input option ('encrypt' or 'decrypt'):
>> encrypt

Processing...
Process Completed.

Time Needed : 7.528507232666016 second
Output filename:
>> cipherteks.txt
```

Gambar 4.17. Enkripsi teks dengan mode CBC

```
PS C:\Users\ferdy\Documents\ITB Jurusan IF\Tahun Keempat\Kriptografi\Tugas Makalah\src\Kiripto-Block-Cipher> py main.py
Input mode ('ecb', 'cbc', 'counter'):
>> cbc
Input filename:
>> cipherteks.txt
Input key (8 characters):
>> haloduni
Input option ('encrypt' or 'decrypt'):
>> decrypt

Processing...
Process Completed.

Time Needed : 13.171130657196045 second
Output filename:
>> hasil.txt
```

Gambar 4.18. Dekripsi teks dengan mode CBC dengan penggantian cipherteks 1-byte di depan



Gambar 4.19. File *teks.txt* (sebelum enkripsi)

Gambar 4.20. File *hasil.txt* (setelah dekripsi)

4.2. Analisis Hasil

Untuk enkripsi teks, dengan ukuran file teks 4KB untuk mode ECB dan CBC didapatkan hasil teks yang sama di akhir algoritma. Waktu eksekusi untuk mode ECB dan CBC relatif mirip namun mode ECB sedikit lebih cepat dengan waktu enkripsi 7.27 detik dan waktu dekripsi 7.11 detik dibandingkan dengan mode CBC dengan waktu enkripsi 7.25 detik dan waktu dekripsi 7.53 detik. Untuk enkripsi gambar, dengan ukuran file gambar 14KB untuk mode ECB dan Counter didapatkan hasil gambar yang sama di akhir algoritma. Namun, mode ECB lebih lambat dengan waktu enkripsi 51 detik dan waktu dekripsi 31.62 detik dibandingkan dengan mode Counter dengan waktu enkripsi 30.97 detik dan waktu dekripsi 31.88 detik. Ukuran dari file sangat mempengaruhi waktu eksekusi, file teks dengan ukuran 4KB akan diproses dengan waktu yang jauh lebih cepat dibandingkan dengan file gambar dengan ukuran 14KB. Hasil perubahan kunci 1-byte untuk mode ECB sangat mempengaruhi hasil dekripsi, hasil dekripsi jauh berbeda dengan teks awal sebelum enkripsi. Hasil perubahan cipherteks 1-byte untuk mode CBC sangat mempengaruhi hasil dekripsi, hasil dekripsi jauh berbeda dengan teks awal sebelum enkripsi.

4.3. Analisis Keamanan KIRIPTO Block Cipher

Dari hasil dekripsi dan enkripsi yang melibatkan perubahan 1-byte kunci dan 1-byte cipherteks dapat diukur keamanan dari cipher blok kami. Untuk hasil dekripsi dengan perubahan 1-byte kunci hasilnya jauh berbeda dengan teks awal, hal itu menandakan algoritma mode ECB aman jika kunci diubah walaupun hanya sedikit. Untuk hasil dekripsi dengan perubahan 1-byte cipherteks hasilnya jauh berbeda dengan teks awal, hal itu menandakan algoritma mode CBC aman jika cipherteks diubah walaupun hanya sedikit, hal itu dikarenakan blok cipherteks satu dan lainnya saling menyambung, sehingga jika ada perbedaan sedikit di cipherteks blok pertama, cipherteks blok selanjutnya akan berubah juga. Sehingga prinsip *confusion* dan *diffusion* sudah teraplikasikan dengan baik.

5. Kesimpulan dan Saran Pengembangan

KIRIPTO adalah sebuah block cipher baru dengan ukuran blok 64-bit dan ukuran kunci 64-bit. KIRIPTO adalah sebuah block cipher yang dapat dioperasikan dalam 3 mode, yaitu ECB, CBC, dan Counter. KIRIPTO menggunakan perputaran dalam struktur jaringan Feistel dengan fungsi f yang terdiri dari beberapa operasi, yaitu XOR, kotak-S, permutasi dan rotasi. KIRIPTO block cipher menerapkan prinsip *diffusion* dan *confusion* dengan bantuan kotak-S, permutasi, rotasi, dan mode CBC. Kedepannya, KIRIPTO masih dapat dikembangkan lebih lanjut, seperti penambahan mode CFB dan OFB, peningkatan kompleksitas permutasi agar lebih kuat, atau peningkatan kinerja dengan operasi-operasi yang lebih efisien, selain itu GUI (*Graphical User Interface*) juga dapat ditambahkan agar pengguna lebih nyaman dalam menggunakan program.

6. Referensi

[1] V. Dolmatov, "GOST 28147-89: Encryption, Decryption, and Message Authentication Code (MAC) Algorithms," RFC Editor, Mar. 2010.

[2] C. E. Shannon, "Communication theory of secrecy systems. 1945.," *MD. Comput.*, vol. 15, no. 1, pp. 57–64, 1998.

Acknowledgments

Puji syukur ke hadirat Tuhan yang Maha Esa, karena atas berkat dan rahmat-Nya penulis memiliki kesempatan untuk menyelesaikan makalah ini. Penulis juga mengucapkan terima kasih kepada semua pihak yang telah membantu dalam proses desain dan implementasi dari KIRIPTO block cipher, khususnya Bapak Rinaldi Munir yang telah memberikan wawasan dan pengetahuan mengenai kriptografi, dan orang tua serta teman-teman penulis yang senantiasa memberikan saran dan dukungan sehingga penulis dapat menyelesaikan makalah ini.