BEA – A New Block Cipher Algorithm

Luqman A. Siswanto (13513024)[1], Yoga Adrian Saputra (13513030)[2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jalan Ganesha 10-12, Bandung 40132, Indonesia
luqmanarifin@students.itb.ac.id[1], yoga.adrian@students.itb.ac.id[2]

Abstract **–** Kami memperkenalkan algoritma block cipher berjudul BEA, kepanjangan dari *Bonek Encryption Algorithm*. Algoritma ini merupakan gabungan randomisasi teknik enkripsi CBC (*Cipher Block Chaining*) dan *Feistel Network*. Gabungan dari teknik-teknik berikut dengan penggunaan beberapa round, dilengkapi dengan generator random kunci, meningkatkan efek difusi yang dihasilkan pada setiap round. Dengan demikian, keamanan yang diberikan dapat ditingkatkan.

Keywords **–** CBC, dekripsi, enkripsi, *Feistel network,* kriptografi.

I. PENDAHULUAN

Pada masa ini teknologi komputer berkembang dengan sangat cepat. Apalagi pada era informasi ini, perkembangan teknologi komputer berkembang ke arah informasi. Analisis informasi memberikan banyak manfaat. Saat ini dikenal sebuah teknologi yang bernama Big Data. Dalam teknologi tersebut banyak informasi yang disimpan oleh sebuah instansi yang nantinya akan dianalisa menjadi pengetahuan yang sangat berharga. Pengetahuan tersebut akan membantu instansi tersebut meningkatkan performansinya. Masyarakat juga merasakan majunya teknologi dalam arah informasi. Dengan mudah masyarakat mendapatkan informasi akan sesuatu hal yang ingin dicari.

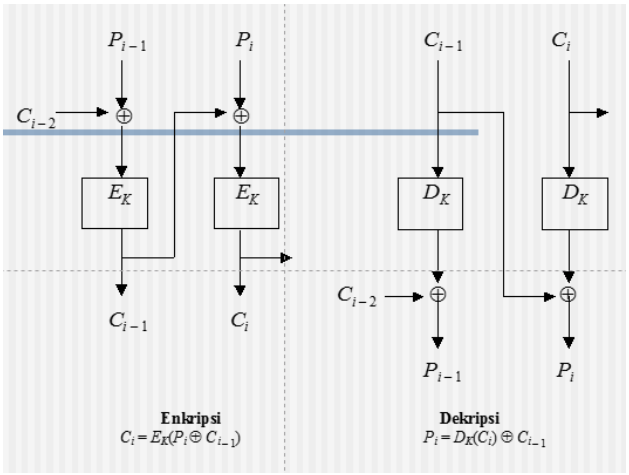
Akan tetapi perkembangan teknologi ke arah informasi juga memberi dampak yang buruk. Banyak teknologi yang dengan mudah mengambil informasi yang bertebaran di jaringan internet. Data privasi seseorang bisa dengan mudah diambil oleh orang lain. Maka dari itu teknologi keamanan informasi juga harus dikembangkan sejajar dengan perkembangan teknologi informasi.

Dalam makalah ini, sebuah algoritma kriptografi yang bernama BEA (*Bonek Encryption Algorithm*) akan dijelaskan. Algoritma kriptografi ini berdasar pada AES (*Advance Encryption Standard*). Algoritma ini berbasis blok-blok bit yang dikumpulkan dengan kunci simetris. Algoritma ini memiliki 2 proses utama dalam pengenkripsian atau pendekripsiannya yaitu CBC dan Feistel.

II. DASAR TEORI

A. *Cipher Block Chaining (CBC)*

Cipher Block Chaining adalah salah satu metode operasi algoritma kriptografi berbasis *block cipher* yang bertujuan untuk mendapatkan ketergantungan tiap blok. Ketergantungan tiap blok ini membuat proses kriptanalisis semakin sulit. Berikut adalah skema umum metode operasi *Cipher Block Chaining*.

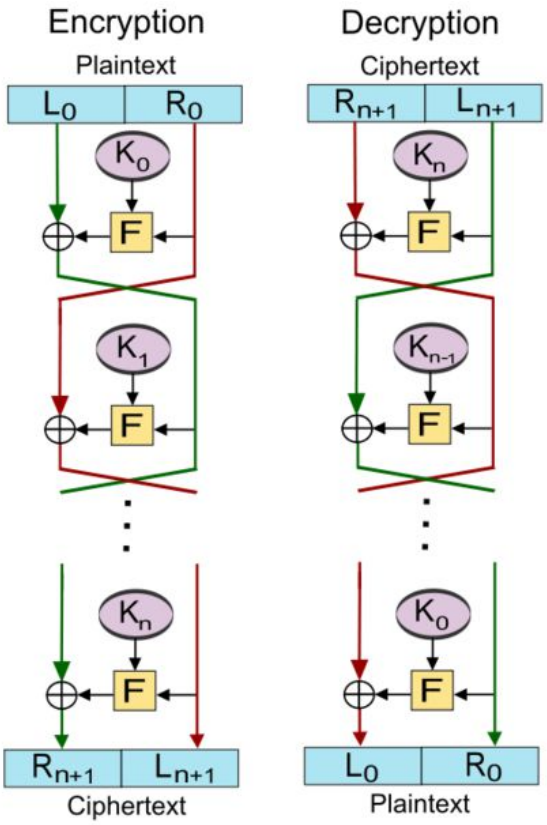


sumber : referensi[3] diakses pada 18/03/2016 17.22

Seperti yang terlihat pada gambar skema enkripsi dan dekripsi di atas, keterhubungan antar blok dilakukan dengan menggunakan hasil enkripsi blok sebelumnya dan dilakukan operasi XOR dengan blok yang ingin di enkripsi saat ini. Karena blok pertama pasti tidak memiliki *cipher text* blok sebelumnya, digunakan *initialization vector* yang berupa bit sepanjang blok untuk dilakukan operasi XOR kepada blok *plain text* pertama. Setelah mendapatkan blok baru hasil operasi XOR, blok tersebut dimasukkan dalam fungsi enkripsi yang bisa berupa apa saja dengan masukan blok tersebut bersama kunci. Fungsi enkripsi itu akan menghasilkan potongan *cipher text* dari 1 blok *plain text*. Hal ini diulangi terus hingga blok akhir. Dengan begitu pada proses enkripsi, keterhubungan blok tidak hanya pada blok sebelumnya melainkan dengan semua blok-blok sebelum blok tersebut.

B. Jaringan Feistel

Jaringan Feistel adalah struktur simetris yang biasa digunakan dalam algoritma kriptografi berbasis *block cipher*. Berikut adalah skema umum pada jaringan Feistel.



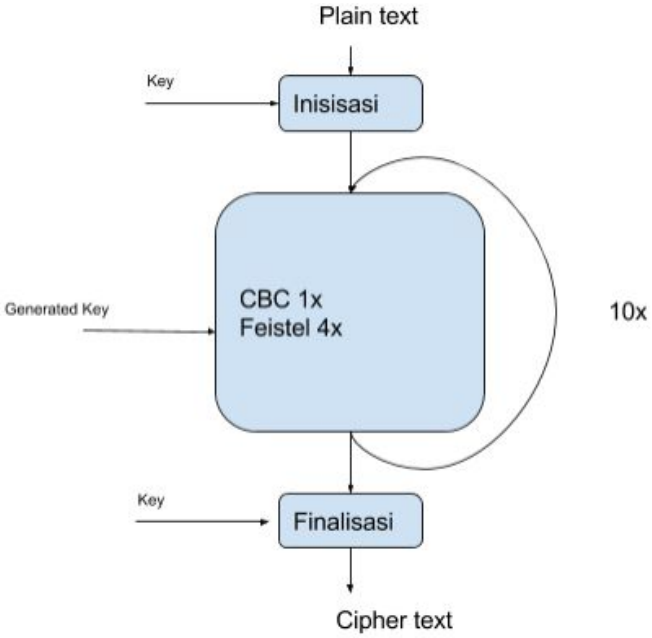
sumber : wikimedia.org diakses pada 18/03/17.26

Dari gambar diatas, setiap blok yang akan dienkripsi atau didekripsi akan dibagi menjadi 2 terlebih dahulu. 2 bagian blok tersebut akan diproses sehingga akan terjadi perubahan bit dalam blok tersebut. Blok bagian kanan (R_0) akan diproses dengan fungsi enkripsi bersama sebuah kunci dengan panjang setengah blok. Fungsi enkripsi itu akan menghasilkan bit-bit sepanjang setengah blok yang akan dienkripsi bersama blok bagian kiri (L_0). Setelah itu, dua bagian blok tersebut akan ditukar posisinya. Proses tersebut diulang sebanyak n buah kali dimana n adalah angka bebas.

Dengan adanya jaringan Feistel ini, bit dalam blok-blok menjadi sangat acak. Selain itu keuntungan dari jaringan feistel ini adalah proses enkripsi dan dekripsi yang mirip. Hanya diperlukan pertukaran urutan kunci yang dimasukkan kedalam jaringan Feistel.

III. Rancangan Block Cipher

Secara umum, rancangan proses algoritmanya adalah sebagai berikut.



Berikut adalah detail rancangan proses algoritma *Bonek Encryption Algorithm*

1. Isi pesan dibagi menjadi blok-blok *cipher* dengan ukuran 128 bit (16 byte).

2. Setiap blok-blok *cipher* akan diinisiasi dengan masukan *key* dari pengguna yang juga berukuran 128 bit. Proses inisiasi tersebut adalah XOR dengan key. Untuk setiap blok, blok tersebut akan di XOR dengan key tersebut sehingga didapatkan blok-blok yang berisi bit baru

3. Blok-blok *cipher* yang baru akan diproses pada proses utama. Proses utama algoritma BEA ini adalah CBC sebanyak 1 kali dan Feistel sebanyak 4 kali. Proses 4 kali Feistel tersebut berarti terjadi 4 kali pertukaran dalam 1 blok *cipher*. Untuk setiap langkah (CBC dan Feistel) terdapat fungsi enkripsi dan *generated key* yang digunakan dalam Algoritma BEA ini. *Generated key* dan fungsi enkripsi yang digunakan akan dijelaskan setelah ini. Proses utama ini diulangi hingga 10 kali sehingga mendapat blok-blok cipher dengan bit yang sangat acak.

4. Blok-blok *cipher* tersebut difinalisasi dengan masukan key awal dari pengguna yang digunakan pada proses inisiasi. Proses finalisasi ini sebenarnya sama dengan proses finalisasi yaitu XOR dengan key. Keluaran dari proses ini yang menjadi *cipher text* dalam algoritma BEA.

A. Fungsi Enkripsi

Fungsi enkripsi adalah fungsi bebas yang digunakan untuk membuat bit dalam setiap blok semakin acak. Dalam CBC, fungsi enkripsi ditunjukkan dengan E_k pada gambar 1 (gambar skema umum CBC). Dalam Feistel, fungsi enkripsi

ditunjukkan dengan F pada gambar (skema struktur Feistel yang digunakan dalam algoritma BEA). *Generated key* yang digunakan pada setiap iterasi berbeda-beda bergantung pada *generated key* sebelumnya.

- Cipher Block Chaining (CBC)

  Fungsi enkripsi pada CBC dilakukan pada setiap blok sekali. Berikut adalah fungsi enkripsinya.

  1. Proses XOR blok *cipher* yang digeser kekanan sebanyak 6 dengan *generated key* yang digeser sebanyak 9 ke kiri.
  2. Bit-bit dalam blok akan diubah ke bentuk matriks. Lalu matriks akan diputar 90 derajat searah jarum jam.

1F	54	AB	D1
17	11	55	66
5C	45	76	33
4A	23	73	A1

  menjadi

4A	5C	17	1F
23	45	11	54
73	76	55	AB
A1	33	66	D1

  3. Penambahan bit *plain* dengan bit *key*
  4. Bit-bit dalam blok akan diubah ke bentuk matriks. Lalu matriks akan diputar 90 derajat ddikelompokkan kedalam *hexa* akan dipermutasi dengan tabel permutasi. Setiap hexa akan dimapping kedalam tabel permutasi, dan diganti dengan *hexa* dalam tabel permutasi. Tabel permutasi yang digunakan akan diberikan di bawah.
  5. Bit-bit dalam blok akan diubah ke bentuk matriks. Lalu matriks akan ditranspose.
  6. Bit-bit dalam blok dilakukan operasi XOR dengan bit key.
  7. Bit-bit dalam blok akan diubah ke bentuk matriks. Lalu matriks akan diputar 180 derajat searah jarum jam.

- Feistel

  Fungsi enkripsi pada Feistel pada umumnya sama dengan proses enkripsi pada CBC. Perbedaan fungsi enkripsi pada feistel dan CBC adalah ukuran blok yang dienkripsi dan ukuran *generated key*. Selain itu dalam matriksnya, setiap elemen matriks berisi 1 angka hexa. Maka dari itu fungsi enkripsi yang memanipulasi angka *byte* dengan tabel permutasi byte dalam CBC diganti dengan angka hexa dengan tabel permutasi hexa dalam Feistel.

B. Generated Key

Fungsi enkripsi adalah fungsi bebas yang digunakan untuk membuat bit dalam setiap blok semakin acak. Dalam CBC, fungsi enkripsi ditunjukkan dengan E_k pada gambar 1 (gambar skema umum CBC). Dalam Feistel, fungsi enkripsi ditunjukkan dengan F pada gambar (skema struktur Feistel yang digunakan *dalam* algoritma BEA). *Generated key* yang digunakan pada setiap iterasi berbeda-beda bergantung pada *generated key* sebelumnya.

- Tabel Permutasi

  Tabel permutasi yang digunakan ada 2.

  1. Tabel permutasi *byte*

     Tabel permutasi *byte* digunakan untuk membentuk kunci baru yang berukuran 128 bit. Berikut adalah tabel *byte* yang digunakan dalam algoritma ini.

     {202,254,240,205,200,109,147,72,150,146,249,
     63,88,136,81,106,161,44,241,164,16,184,87,19
     3,112,79,133,60,159,125,13,248,189,39,231,21
     2,139,127,55,124,244,107,179,18,35,175,215,1
     74,71,68,208,151,117,116,167,65,235,95,83,13
     4,221,163,27,158,234,252,80,198,91,206,49,54,
     45,110,178,224,246,157,85,182,237,140,21,3,5
     9,43,76,94,176,102,48,0,20,229,50,52,250,228,
     236,181,138,186,238,213,223,185,93,29,253,1
     80,201,100,169,58,77,156,25,22,4,166,141,41,1
     77,56,84,131,137,78,148,104,197,90,103,14,42,
     190,5,24,119,154,7,33,11,145,128,36,168,152,7
     4,64,51,207,171,31,243,233,132,38,10,34,162,4
     6,187,67,2,8,53,165,245,144,114,98,1,9,217,12
     3,61,160,242,130,101,69,219,226,210,135,173,
     23,172,70,122,108,32,199,15,218,247,6,66,183,
     251,214,225,75,255,192,222,26,28,188,82,105,
     155,121,220,47,126,111,30,227,194,113,89,57,
     195,216,99,211,17,239,12,37,86,96,40,19,209,1
     29,120,153,92,203,149,196,142,73,170,232,97,
     230,191,143,115,62,204,118,11,15,10,8,3,1,13,
     6,14,4,12,2,9,7,0,5}

  2. Tabel permutasi *hexa*

     Tabel permutasi *hexa* digunakan untuk membentuk kunci baru yang berukuran 64 bit. Berikut adalah tabel *hexa* yang digunakan dalam algoritma ini.

{11,15,10,8,3,1,13,6,14,4,12,2,9,7,0,5}

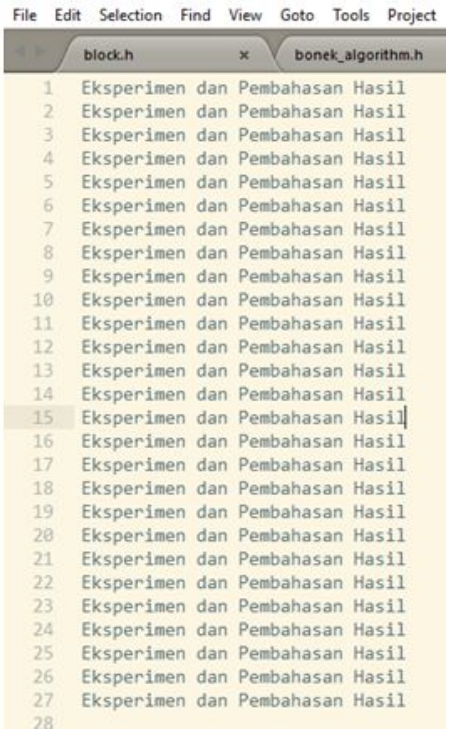
- Urutan Penggunaan Key Masukan dan Generated Key
  1. Key masukan digunakan pada proses inisialisasi.
  2. Untuk CBC, *generated key* dibentuk dari permutasi tabel byte dengan key sebelumnya (Untuk putaran 1 key sebelumnya adalah key masukan. Untuk putaran 2 hingga 10 key sebelumnya adalah hasil *generated key* dari putaran sebelumnya).
  3. Untuk setiap iterasi CBC blok selanjutnya, kunci yang digunakan adalah *generated key* dari *generated key* iterasi blok sebelumnya.
  4. Untuk Feistel, *generated key* dibentuk dari permutasi tabel *byte* dengan *generated key* blok pertama dari CBC. Namun karena kunci yang diperlukan dalam Feistel hanya berukuran 64 bit (setengah blok), maka 1 blok kunci tersebut dibagi menjadi 2 dan digunakan dalam 2 kali Feistel. Untuk melakukan 4 kali Feistel berarti memerlukan 2 *generated key*.
  5. Hasil *generated key* dari Feistel terakhir digunakan kembali oleh CBC dalam putaran 2.
  6. Langkah 2 hingga 5 diulangi hingga sampai putaran 10 kali.
  7. Dalam finalisasi, kunci yang dipakai adalah kunci awal yang merukapan masukan dari *user*.

IV. Eksperimen dan Pembahasan Hasil

Berikut adalah beberapa eksperimen yang dilakukan dalam pengujian BEA.

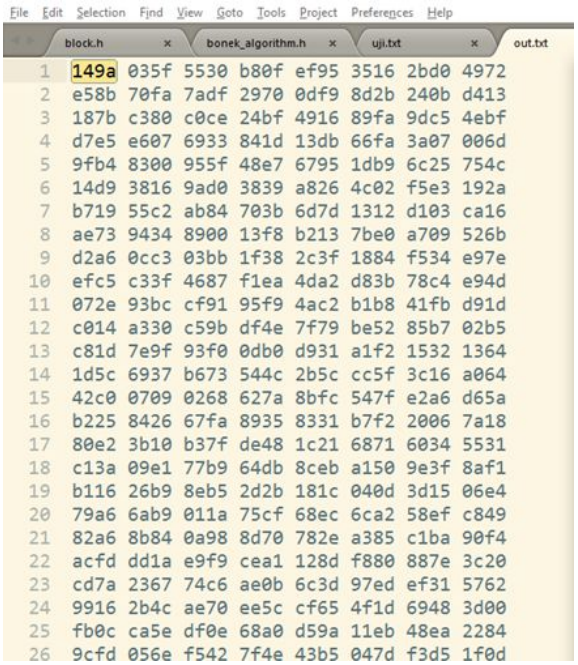
A. Pengujian dengan File Text

File masukkan berekstensi txt dan berukuran 17 KB.



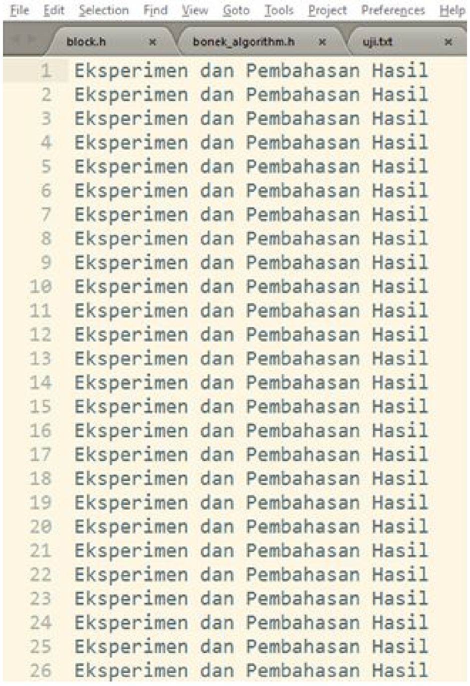
Dienkripsi dengan kunci

00000aaaaa00000aaaaa00000aaaaafc



Didekripsi dengan kunci

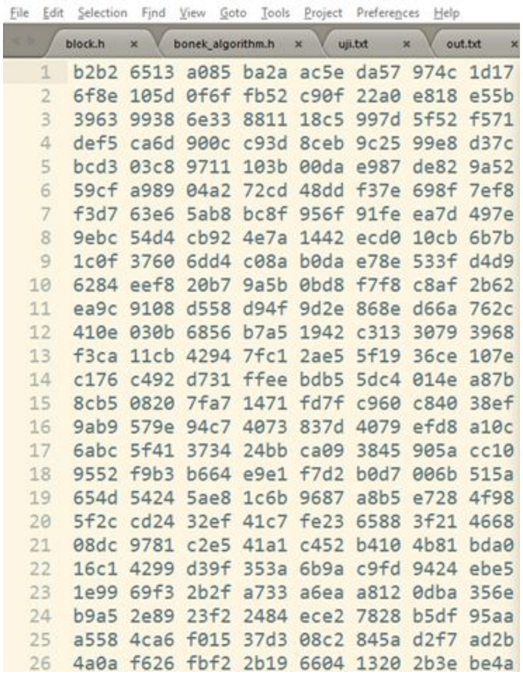
00000aaaaa00000aaaaa00000aaaaafc




```
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
Eksperimen dan Pembahasan Hasil
```

Didekripsi dengan kunci yang salah

00100aabaa00200afaaa00c00a3a7afc



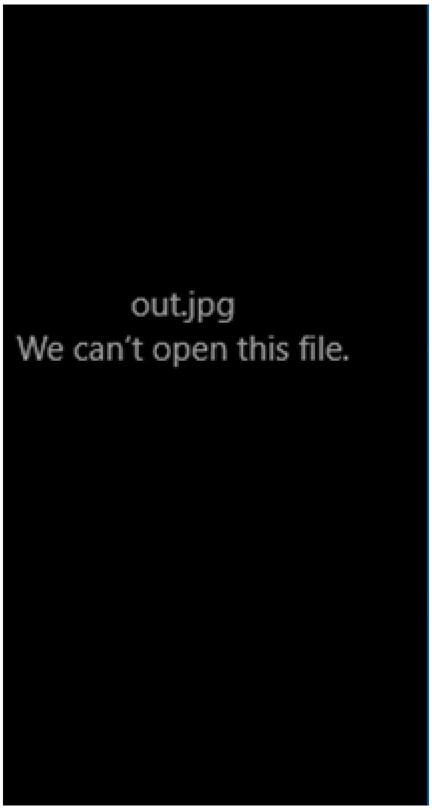

```
b2b2 6513 a085 ba2a ac5e da57 974c 1d17
6f8e 105d 0f6f fb52 c90f 22a0 e818 e55b
3963 9938 6e33 8811 18c5 997d 5f52 f571
def5 ca6d 900c c93d 8ceb 9c25 99e8 d37c
bcd3 03c8 9711 103b 00da e987 de82 9a52
59cf a989 04a2 72cd 48dd f37e 698f 7ef8
f3d7 63e6 5ab8 bc8f 956f 91fe ea7d 497e
9ebc 54d4 cb92 4e7a 1442 ecd0 10cb 6b7b
1c0f 3760 6dd4 c08a b0da e78e 533f d4d9
6284 eef8 20b7 9a5b 0bd8 f7f8 c8af 2b62
ea9c 9108 d558 d94f 9d2e 868e d66a 762c
410e 030b 6856 b7a5 1942 c313 3079 3968
f3ca 11cb 4294 7fc1 2ae5 5f19 36ce 107e
c176 c492 d731 ffee bdb5 5dc4 014e a87b
8cb5 0820 7fa7 1471 fd7f c960 c840 38ef
9ab9 579e 94c7 4073 837d 4079 efd8 a10c
6abc 5f41 3734 24bb ca09 3845 905a cc10
9552 f9b3 b664 e9e1 f7d2 b0d7 006b 515a
654d 5424 5ae8 1c6b 9687 a8b5 e728 4f98
5f2c cd24 32ef 41c7 fe23 6588 3f21 4668
08dc 9781 c2e5 41a1 c452 b410 4b81 bda0
16c1 4299 d39f 353a 6b9a c9fd 9424 ebe5
1e99 69f3 2b2f a733 a6ea a812 0dba 356e
b9a5 2e89 23f2 2484 ece2 7828 b5df 95aa
a558 4ca6 f015 37d3 08c2 845a d2f7 ad2b
4a0a f626 fbf2 2b19 6604 1320 2b3e be4a
```

B. Pengujian dengan File JPG (biner)

File masukkan berekstensi jpg dan berukuran 14 KB.

Dienkripsi dengan kunci

00000aaaaa00000aaaaa00000aaaaafc

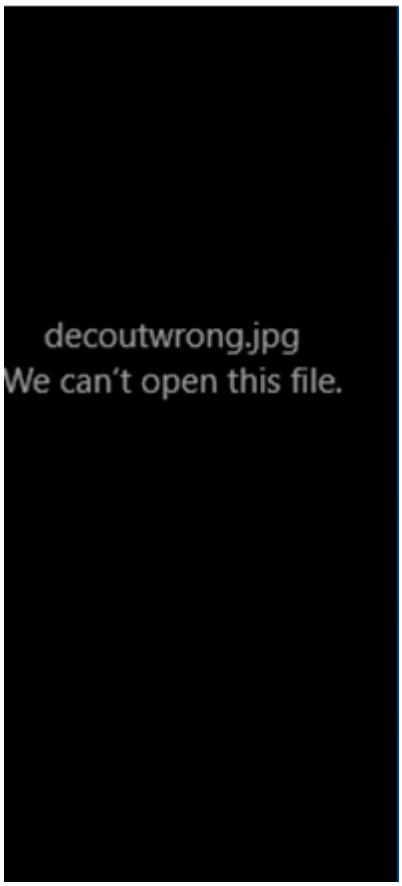


Didekripsi dengan kunci

00000aaaaa00000aaaaa00000aaaaafc

Didekripsi dengan kunci yang salah

00100aabaa00200afaaa00c00a3a7af



C. *Contoh Hasil Running Program*

Proses enkripsi menggunakan BEA. Program berbasis CLI (*Command Line Interface*)

```
D:\OneDrive\tempDocument\Kriptografi\bonek-encryption-algorithm>
main enc layton.jpg out.jpg 00000aaaaa00000aaaaa00000aaaaafc
main enc layton.jpg out.jpg 00000aaaaa00000aaaaa00000aaaaafc
hexa to binary key done
Reading binary done. Original filesize is 13932 bytes
Adding 4 bytes
Encrypting 13936 bytes with key 00000aaaaa00000aaaaa00000aaaaafc

Encrypting success!

D:\OneDrive\tempDocument\Kriptografi\bonek-encryption-algorithm>
```

Proses dekripsi menggunakan BEA.

```
D:\OneDrive\tempDocument\Kriptografi\bonek-encryption-algorithm>
main dec out.jpg decoutright.jpg 00000aaaaa00000aaaaa00000aaaaaf
c
main dec out.jpg decoutright.jpg 00000aaaaa00000aaaaa00000aaaaaf
c
hexa to binary key done
Reading binary done. Original filesize is 13936 bytes
No byte added
Decrypting 13936 bytes with key 00000aaaaa00000aaaaa00000aaaaafc

Decrypting success!

D:\OneDrive\tempDocument\Kriptografi\bonek-encryption-algorithm>
```

D. *Pembahasan Hasil*

Dari eksperimen, Algoritma BEA berhasil mengacak acak isi file teks maupun biner. Dari *file* teks, dapat dilihat bahwa isi teksnya berubah menjadi *byte* lain. Dari file gambar (jpg) dan biner (pdf) dapat dilihat bahwa file keluaran tidak bisa dibuka atau rusak. Hal itu terjadi karena format header file-file tersebut berubah dan tidak bisa diidentifikasi.

Algoritma BEA pun dapat mendekripsi *file* yang telah diacak-acak menjadi *file* semula dengan kunci yang tepat. Bila kunci yang dipakai salah, maka *file* semula tidak akan terbentuk. Tambahan *padding bit* tidak menyebabkan *error* karena *padding bit* yang ditambahkan adalah 00..00. Pada *file* teks, tambahan ASCII 0 atau NULL berarti *end of file*. Karena sebelumnya sudah terdapat NULL, penambahan NULL tidak memberikan efek apapun selain penambahan ukuran. Pada *file binary* karena *default* data adalah NULL, maka penambahan NULL pun tidak akan merusak *end of file* dari *file binary* tersebut.

Berikut ini adalah performansi waktu dari *Bonek Encryption Algoritm* (BEA) dijalankan pada platform tertentu untuk tipe berkas tertentu.

No	Performansi			
	Tipe File	***Ukuran***	***Waktu***	***Kecepatan***
1	PDF	100 KB	4,055 sec	24,6 KB/sec
3	MP3	4506 KB	191,118 sec	23,5 KB/sec
4	MP4	4215 KB	177,537 sec	23,7 KB/sec
5	JPG	1606 KB	70,868 sec	22,6 KB/sec
6	DOC	67 KB	2,757 sec	24,3 KB/sec
7	TXT	8 KB	0,334 sec	23,9 KB/sec
Rata-rata				**23,7 KB/sec**

Hasil eksekusi berikut diimplementasi menggunakan bahasa C++11 dengan MinGW GNU C++ pada Intel i5-3317U @1.7 GHz (1 Core, 8 GB RAM, *64-bit architecture*).

Hasil *running* algoritma ini cukup lambat karena belum dilakukan optimasi pada struktur kode C++ implementasi BEA. Dengan kata lain, implementasi di atas masih belum dioptimasi sehingga *computer-friendly.* Implementasi kode tersebut masih banyak melibatkan penciptaan objek C++ sehingga menghasilkan banyak overhead. Akibatnya performansi algoritma belum maksimal.

E. Analisis Keamanan

Karena proses utama dalam algoritma BEA ini adalah Feistel dan CBC dengan standar kriptografi AES, keamanan algoritma ini dapat dikatakan bagus. Menurut hasil ekperimen pada file berekstensi txt, *byte* dalam *file* tidak ada yang berulang padahal isi *file* aslinya adalah teks yang berulang. Hal ini terjadi karena penggunaan metode CBC dengan *generated key* yang selalu berubah-ubah. Dengan begitu kriptanalis pada algoritma ini bisa dikatakan cukup sulit. Selain itu karena metode Feistel dan didukung oleh fungsi enkripsi yang rumit, maka cara kriptanalisis menggunakan tabel frekuensi pun menjadi sangat sulit dilakukan.

V. KESIMPULAN DAN SARAN

Algoritma BEA ini memenuhi AES dan cukup kuat karena ada adanya keterhubungan tiap blok dan pengacakan setiap blok itu sendiri yang cukup kompleks. Meskipun begitu, kecepatan eksekusi BEA masih lamban dibandingkan dengan kecepatan algoritma lain sehingga perlu dikembangkan lebih lanjut.

Sedangkan untuk saran pengembangan berikutnya mungkin penambahan proses pada proses utama sehingga keamanan algoritma BEA ini semakin kuat.

KATA PENUTUP

Algoritma enkripsi ini dikembangan oleh Luqman A. Siswanto dan Yoga Adrian Saputra untuk keperluan memenuhi tugas mata kuliah IF4020 Kriptografi di Institut Teknologi Bandung semester genap tahun ajaran 2015/2016.

Penulis ingin menghaturkan banyak terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T. yang telah mengampu mata kuliah IF4020 Kriptografi selama satu semester ini.

REFERENSI

[1] M. Luby and C. Rackoff. "How to Construct Pseudorandom Permutations and Pseudorandom Functions." In SIAM J. Comput., vol. 17, 1988, pp. 373-386.

[2] Jacques Patarin, Luby-Rackoff: 7 Rounds Are Enough for Security, Lecture Notes in Computer Science, Volume 2729, Oct 2003, Pages 513 – 529.

[3] Rinaldi Munir. Kriptografi Modern, bahan kuliah IF4020 Kriptografi, Teknik Informatika, Institut Teknologi Bandung.

[4] Rinaldi Munir. Advanced Encryption Standard (AES), bahan kuliah IF4020 Kriptografi, Teknik Informatika, Institut Teknologi Bandung.

[5] Iskandar Setiadi, Atsuko Miyaji, Achmad Imam Kistijantoro. Elliptic Curve Cryptography: Algorithms and Implementation Analysis over Coordinate Systems. Institut Teknologi Bandung. Japan Advanced Institute of Science and Technology.

[6] Ronald Rivest, et al. The RC6™ Block Cipher. MIT Laboratory of Computer Science. RSA Laboratories.