IMPLEMENTASI ALGORITMA KRIPTOGRAFI DENGAN S-BOX DINAMIS BERGANTUNG PADA KUNCI UTAMA BERBASIS *ADVANCED ENCRYPTION STANDARD* (AES)

Herdaya Adiyasa, Putut Sri Wasito dan Satriyo Adhy

Jurusan Ilmu Komputer/Informatika,
Fakultas Sains dan Matematika Universitas Diponegoro
Jl. Prof Soedarto, S.H., Tembalang Semarang - 50275
Email : herdaya.adiyasa@gmail.com

Abstrak

Advanced Encryption Standard (AES) merupakan sebuah algoritma kriptografi yang ditetapkan *National Institute of Standards and Technology* (NIST) Amerika sebagai standar untuk algoritma enkripsi data elektronik. Penelitian tentang pengunaan S-box yang dinamis pada AES telah dilakukan sebelumnya dan disimpulkan dapat meningkatkan kompleksitas algoritma AES. Penelitian ini mengimplementasikan penggunaan S-box dinamis dalam algoritma AES tanpa melakukan perubahan operasi dasar AES. Implementasi dilakukan pada berkas citra BMP dengan menggunakan dua mode operasi. Hasil implementasi pada penelitian ini menunjukkan bahwa perubahan S-box AES menjadi S-box dinamis tidak mempengaruhi efektivitas dari algoritma AES. Hal tersebut dibuktikan dengan uji kemiripan citra asli dengan citra hasil dekripsi yang menunjukkan hasil 100 % pada semua citra uji. Penelitian ini juga membuktikan bahwa penggunaan S-box dinamis pada AES dapat diimplementasikan pada berkas citra. Implementasi pada berkas multimedia jenis lain masih memungkinkan untuk penelitian selanjutnya.

Kata kunci : Kriptografi, AES, Modifikasi S-box, S-box Dinamis, Citra BMP.

1. PENDAHULUAN

Perkembangan teknologi informasi yang pesat memberikan kemudahan kepada penggunanya dalam melakukan pertukaran informasi. Pertukaran informasi yang tidak aman dapat meningkatkan kerentanan terhadap akses suatu informasi yang bersifat pribadi atau rahasia. Kriptografi merupakan solusi untuk menyembunyikan informasi asli dari suatu berkas.

Tahun 2013, Sliman Arrag, dkk melakukan penelitian berjudul, "*Implementation of Stronger AES by Using Dynamic S-box Dependent of Master Key*". Penelitian tersebut mengusulkan sebuah algoritma yang berbasiskan AES yang diimplementasikan pada perangkat FPGA. Penelitian tersebut menggunakan tabel S-box dinamis. S-box dibentuk ulang bergantung pada *Master Key* (pada penelitian ini disebut dengan kunci utama) dalam setiap proses enkripsi atau dekripsi. (Arrag, et al., 2013)

Penelitian ini mengimplementasikan algoritma kriptografi usulan Sliman Arrag, dkk pada berkas citra. Hasil implementasi adalah aplikasi kriptografi citra berbasis *desktop* yang dibangun dengan bahasa pemrograman C#.NET. Berkas citra yang digunakan adalah berkas citra BMP. Implementasi pada berkas citra BMP dilakukan tanpa melakukan proses enkripsi pada *header* dan *property* berkas. Enkripsi hanya dilakukan pada data *bitmap* berkas tersebut. Sehingga berkas citra hasil enkripsi dapat dibuka kembali menggunakan *image viewer* untuk dapat dilihat hasil algoritma yang digunakan saat diterapkan pada citra.

Mode operasi *block cipher* pada aplikasi yang dibangun menggunakan dua jenis mode operasi yaitu *Electronic Code Book* (ECB) dan *Cipher Block Chaining* (CBC). Sedangkan basis AES yang digunakan adalah AES 128 bit, 192 bit dan 256 bit.

2. DASAR TEORI

Penelitian ini menggunakan dua jenis algoritma kriptografi sebagai dasar teori, yaitu algoritma AES dan algoritma dengan S-box Dinamis.

Algoritma AES

Algoritma AES memiliki 4 proses dasar dalam melakukan enkripsi maupun dekripsi, yaitu : melakukan substitusi, pergeseran baris, mengacak data dan melakukan operasi XOR. Algoritma AES mempunyai ukuran blok dan kunci yang tetap sebesar 128, 192, 256 bit (NIST, 2001).

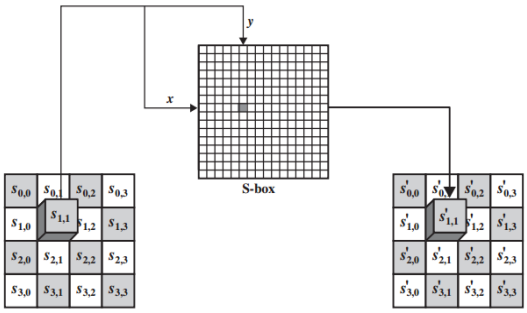
a. Enkripsi pada AES

Secara Garis besar enkripsi algoritma AES 128 bit adalah sebagai berikut (Shirley, 2011) :

– *AddRoundKey* : fungsi XOR dilakukan antara *state byte* awal (plainteks) dengan kunci (*cipher key*).

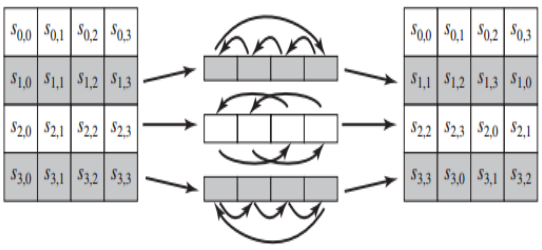
– Putaran sebanyak Nr – 1 kali. Setiap putaran dilakukan proses berikut :

  - *SubBytes* : setiap *state byte* dilakukan substitusi dengan mengganti setiap *byte* yang ada pada *state byte* dengan *byte* yang ada pada tabel substitusi yang disebut S-Box. (Gambar 1)



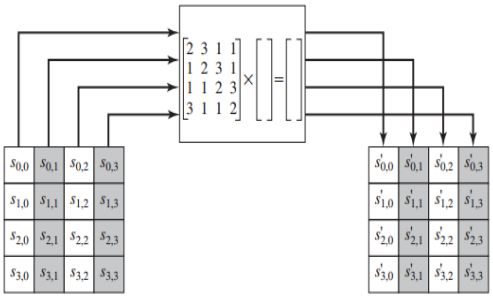
Gambar 1. Proses SubBytes pada AES (Stallings, 2011)

  - ShiftRows : setiap *state byte* dilakukan pergeseran pada masing-masing baris dengan jumlah pergeseran yang berbeda-beda pada setiap baris *state byte* (baris ke-1 diputar 1 kali, baris ke-2 diputar 2 kali dan baris ke-3 diputar 3 kali). (Gambar 2)



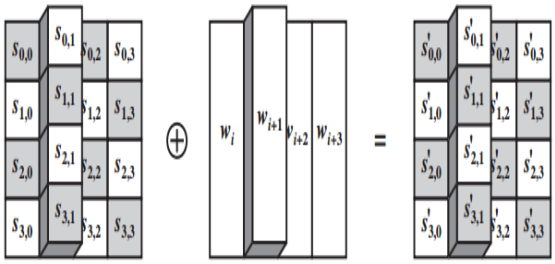
Gambar 2. Proses ShiftRows pada AES (Stallings, 2011)

  - MixColumns : setiap *state byte* dilakukan proses pada setiap kolomnya secara individual. Masing-masing kolom dilakukan perkalian matriks dengan *byte data* yang telah ditetapkan pada AES. (Gambar 3)



Gambar 3. Proses MixColumns pada AES (Stallings, 2011)

  - AddRoundKey : setiap *state byte* dilakukan proses XOR dengan *round key* (dibentuk dengan proses *Key Schedule*). (Gambar 4)



Gambar 4. Proses AddRoundKey (Stallings, 2011)

– *Putaran* terakhir dilakukan hanya tiga proses yaitu :

  - SubBytes
  - ShiftRows
  - AddRoundKey

b. Dekripsi pada AES

Alur proses dekripsi AES menggunakan proses yang sama namun terbalik (*reverse operations*) dan dilakukan dengan urutan terbalik (*reverse orders*). *SubBytes* menjadi *InvSubBytes*, *ShiftRows* menjadi *InvShiftRows* dan *MixColumns* menjadi *InvMixColumns*.

Proses *InvSubBytes* sama dengan proses yang terjadi pada *SubBytes*. Hanya saja pada *InvSubBytes*, tiap elemen pada *state* dipetakan dengan menggunakan tabel *Invers* S-Box. (Gambar 5).

Proses *InvShiftRows* merupakan bentuk invers pada proses ShiftRows. Pada proses ShiftRows menggunakan operasi circular shift left, yakni sebuah proses menggeserkan sebuah blok ke arah kiri. Sedangkan proses InvShiftRows menggunakan operasi circular shift right. Operasi circular shift right ini berguna mengembalikan bit yang telah tergeser oleh operasi circular shift left dengan

menggeserkan bit ke arah kanan (Sadikin, 2012).

InvMixColumns adalah kebalikan dari transformasi *MixColumns*. *InvMixColumns* menggunakan perkalian matriks antara matriks dengan nilai yang tetap dengan *state*. Matriks dengan nilai yang tetap yang dipakai pada proses *InvMixColumns* menggunakan *invers* matriks konstan pada transformasi *MixColumns* (Sadikin, 2012).

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Gambar 5. Tabel S-Box AES (NIST, 2001)

Algoritma dengan S-Box Dinamis

Algoritma kriptografi dengan S-Box dinamis adalah algoritma yang telah diteliti oleh Sliman Arrag, dkk. Algoritma tersebut pada dasarnya adalah algoritma AES yang dimodifikasi bagian tabel substitusinya. Tabel S-Box sebagai tabel substitusi yang digunakan pada algoritma AES digantikan dengan tabel baru yang disebut S-Box Dinamis. S-Box dinamis dibentuk ulang dengan bergantung pada masukan kunci utama.

S-Box dinamis disebut dinamis karena dalam proses pembentukannya tabel tersebut bergantung pada kunci utama yang dimasukkan oleh kriptografer. Kunci utama tentu dapat berbeda-beda antara satu kriptografer dan lainnya, juga antara berkas yang diproses satu dan lainnya. Tabel S-Box dinamis mempunyai ukuran yang sama dengan tabel S-Box asli AES. (Arrag, et al., 2013)

Perbandingan antara algoritma yang diteliti Sliman Arrag, dkk dengan algoritma AES dapat dilihat pada tabel 1.

S-Box Dinamis untuk Enkripsi

S-Box dinamis digunakan sebagai pengganti S-Box asli AES pada algoritma enkripsi. Proses pembentukan S-Box dinamis dilakukan dengan membuat tabel baru dengan ukuran sama persis seperti tabel S-Box asli AES. Kunci utama yang merupakan masukan dari kriptografer digunakan untuk pemicu terbuatnya tabel S-Box dinamis. Proses pembentukan S-Box Dinamis dilakukan dalam tahapan sebagai berikut :

a. Pengambilan 1 *byte* pertama dari kunci utama yang akan digunakan dalam enkripsi.
b. Perhitungan XOR antara setiap elemen S-Box asli AES dengan 1 *byte* pertama dari kunci utama yang telah dipilih.
c. Hasil perhitungan XOR dijadikan tabel S-Box baru sebagai pengganti S-Box asli AES.

Tabel 1. Perbandingan AES 128 bit dan Algoritma oleh Sliman Arrag, dkk

	Algoritma AES 128 bit	Algoritma oleh Sliman Arrag, dkk
Panjang *Block*	128 *bit*	128 *bit*
Panjang *Key* (kunci)	128 *bit*	128 *bit*
Jumlah putaran	10 putaran	10 putaran
Fungsi yang digunakan	terdiri dari 4 jenis transformasi dasar : -SubBytes -ShiftRow -MixColumns -AddRoundKey	terdiri dari 4 jenis transformasi dasar : -SubBytes dengan S-Box dinamis -ShiftRow -MixColumns -AddRoundKey
Tabel Substitusi	S-Box (bersifat tetap dari algoritma AES)	S-Box dinamis (bergantung pada kunci utama)
Ekspansi Kunci	menggunakan kunci utama dan S-Box asli dari algoritma AES	menggunakan kunci utama dan S-Box dinamis yang bergantung pada kunci utama

Invers S-Box Dinamis untuk Dekripsi

Invers S-Box dinamis digunakan sebagai pengganti Invers S-Box asli AES pada algoritma dekripsi. Proses pembentukan Invers S-Box dinamis dilakukan dengan membuat tabel baru dengan ukuran sama persis seperti invers S-Box asli AES. Kunci utama yang merupakan masukan dari kriptografer digunakan untuk

pemicu terbuatnya invers S-Box dinamis. Proses pembentukan invers S-Box dinamis dilakukan dalam tahapan sebagai berikut :

a. Pengambilan 1 *byte* pertama dari kunci utama yang akan digunakan dalam dekripsi.
b. Perhitungan XOR antara setiap elemen S-Box asli AES dengan 1 *byte* pertama dari kunci utama yang telah dipilih.
c. Perhitungan *invers* dari hasil XOR antara S-Box asli dengan 1 *byte* kunci utama.
d. Hasil perhitungan *invers* dijadikan tabel baru sebagai pengganti *invers* S-Box asli AES.

3. IMPLEMENTASI

Aplikasi kriptografi citra yang merupakan hasil implementasi pada penelitian ini dibangun dengan dua jenis mode operasi *block cipher* yaitu ECB dan CBC. Mode operasi akan mebagi berkas utuh menjadi seukuran 128 *bit* atau 16 *byte*. Sehingga dapat dilakukan proses menggunakan algoritma kriptografi yang mengharuskan ukuran data berukuran 16 *byte*.

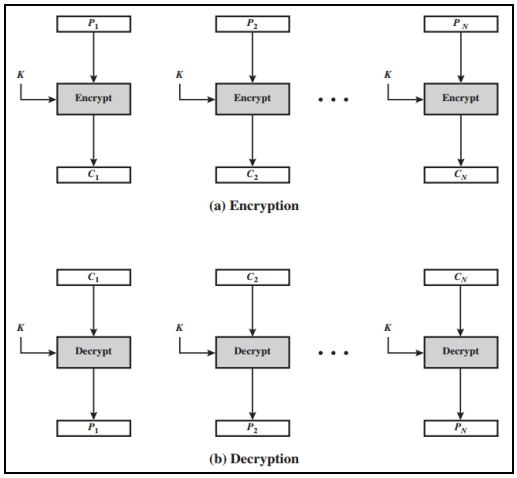
Pembagian blok yang mengharuskan seukuran 16 *byte* untuk proses kriptografi mengakibatkan perlunya proses penambahan ukuran jika terjadi kekurangan *byte* pada blok terakhir. Misalnya terdapat 40 *byte* berkas utuh yang diproses oleh mode operasi, maka akan ada kekurangan 8 *byte* pada blok ketiga atau blok terakhir hasil pembagiannya. Kekurangan tersebut diatasi dengan melakukan *padding.* Algoritma *padding* pada aplikasi ini dapat dilihat pada tabel 2.

Tabel 2. Algoritma *Padding*

```
lebih : int
imageArray : array of byte
tambahan : array [lebih] of byte
imageArrayTemp : array[imageArray.Length
+ lebih] of byte
pjg : int

pjg = imageArray.Length;
lebih = pjg % 16;
lebih = 16 - lebih;
if (lebih > 0)
{
  for (int x = 0; x < lebih; x++)
  { tambahan[x] = 0; }
  BlockCopy(imageArray,0,imageArrayTem
  p,0,pjg);
  BlockCopy(tambahan,0,imageArrayTemp,
  pjg,tambahan.Length);
  imageArray = imageArrayTemp;
}
output imageArray;
```

Electronic Code Books

ECB beroperasi dengan memecah *plaintext* menjadi blok dengan ukuran sesuai dengan blok sistem penyandian. Kemudian masing-masing blok disandi dengan kunci dan algoritma enkripsi yang sama. Dekripsi untuk ECB dilakukan hal yang sama, hanya saja algoritma enkripsi digantikan dengan algoritma dekripsi. ECB diilustrasikan pada gambar 6



Gambar 6. Alur Proses Electronic Code Book (Stallings, 2011)

Pada Tabel 3 dan 4 ditampilkan algoritma ECB yang diimplementasikan pada penelitian ini.

Tabel 3. Algoritma ECB untuk Enkripsi

```
imageArray : array[] of byte {berkas input
utuh}
inputByte : array[16] of byte {blok 16 byte
hasil bagi untuk input}
outputByte : array[16] of byte {blok 16
byte output}
joinByte : array[] of byte {berkas output
utuh}
posisi : int

posisi = 0
for (int i = 0; i < jumlah_ulang; i++)
{
  BlockCopy(imageArray, posisi,
inputByte,
  0, 16);
  Cipher(inputByte, outputByte);
  // Cipher() adalah fungsi untuk
  mengenkripsi blok 16 byte //
```

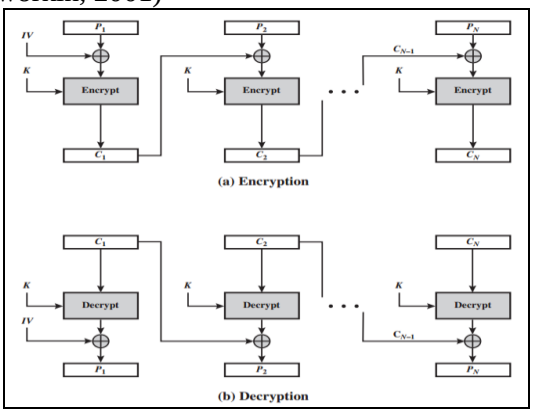
```
  BlockCopy(outputByte, 0, joinByte,
  posisi, 16);
  posisi += 16;
}
output joinByte;
```

Tabel 4. Algoritma ECB untuk Dekripsi

```
imageArray : array[] of byte {berkas input
utuh}
inputByte : array[16] of byte {blok 16 byte
hasil bagi untuk input}
outputByte : array[16] of byte {blok 16
byte output}
joinByte : array[] of byte {berkas output
utuh}
posisi : int

posisi = 0
for (int i = 0; i < jumlah_ulang; i++)
{
  BlockCopy(imageArray, posisi,
  inputByte, 0, 16);
  InvCipher(inputByte, outputByte);
  //InvCipher() adalah fungsi untuk
  mendekripsi blok 16 byte //
  BlockCopy(outputByte, 0, joinByte,
  posisi, 16);
  posisi += 16;
}
output joinByte;
```

Cipher Block Chaining

Mode *Cipher Block Chaining* (CBC) merupakan operasi *block cipher* dimana nilai blok *ciphertext* bergantung pada nilai blok *plaintext*-nya dan seluruh blok *plaintext* sebelumnya. Hasil enkripsi blok sebelumnya di-umpan-balikkan ke dalam enkripsi blok *current* (blok selanjutnya yang akan diproses). (Dworkin, 2001)



Gambar 7. Alur Proses Cipher Block Chaining (Stallings, 2011)

Mode CBC memerlukan blok semu yang disebut IV (*initialization vector*) untuk umpan awal pada proses XOR enkripsi blok pertama. Proses dekripsi CBC menggunakan IV untuk mendapatkan blok *plaintext* dengan cara meng XOR-kan IV dengan hasil dekripsi blok *ciphertext* pertama. CBC diilustrasikan pada gambar 7. Pada tabel 5 dan tabel 6 ditampilkan algoritma CBC yang diimplementasikan dalam penelitian ini.

Tabel 5. Algoritma CBC untuk Enkripsi

```
imageArray : array[] of byte {berkas input
utuh}
inputByte : array[16] of byte {blok 16 byte
hasil bagi untuk input}
outputByte : array[16] of byte {blok 16 byte
output}
joinByte : array[] of byte {berkas output utuh}
chainVector : array[16] of byte {blok 16 byte
hasil XOR di CBC}
posisi : int

posisi = 0
for (int i = 0; i < jumlah_ulang; i++)

{
  BlockCopy(imageArray, posisi, inputByte,
  0, 16);
  for (int x = 0; x < inputByte.Length; ++x)

  {
    chainVector[x] = inputByte[x] XOR
    initVector[x];
  }

  Cipher(chainVector, outputByte);
  //Cipher() adalah fungsi untuk mengenkripsi
  blok 16 byte//

  for (int x = 0; x < outputByte.Length; ++x)

  {
    initVector[x] = outputByte[x];
  }

  BlockCopy(outputByte, 0, joinByte, posisi,
  16);
  posisi += 16;

}
output joinByte;
```

Tabel 6. Algoritma CBC untuk Dekripsi

```
imageArray : array[] of byte {berkas input utuh}
inputByte : array[16] of byte {blok 16 byte hasil bagi untuk input}
outputByte : array[16] of byte {blok 16 byte output}
joinByte : array[] of byte {berkas output utuh}
chainVector : array[16] of byte {blok 16 byte hasil XOR di CBC}
posisi : int

posisi = 0
for (int i = 0; i < jumlah_ulang; i++)

{

  BlockCopy(imageArray, posisi, inputByte, 0, 16);

  InvCipher(inputByte, chainVector);
  //InvCipher() adalah fungsi untuk mendekripsi blok 16 byte//
  for (int x = 0; x < chainVector.Length; ++x)

  {
    outputByte[x]=chainVector[x] XOR initVector[x]);
  }

  for (int x = 0; x < inputByte.Length; ++x)
  {
    initVector[x] = inputByte[x];
  }

  BlockCopy(outputByte, 0, joinByte, posisi, 16);
  posisi += 16;
}
output joinByte;
```

Kriptografi pada Berkas Citra BMP

Penggunaan berkas citra BMP sebagai berkas yang diproses dalam kriptografi enkripsi dan dekripsi mengakibatkan perlu adanya proses khusus. Proses khusus tersebut dilakukan agar berkas citra dapat dienkripsi dan didekripsi tanpa merusak struktur asli dari berkas citra itu sendiri.

Berkas citra yang digunakan adalah berkas citra BMP versi Windows. Berkas citra BMP sebagai input akan dipisahkan bagian header berkasnya. Kemudian setelah dilakukan proses kriptografi maka akan digabungkan kembali bagian header berkasnya. Proses pemisahan header dan penggabungan header berkas citra BMP terlihat seperti pada tabel 7.

Tabel 7. Algoritma penanganan khusus berkas BMP

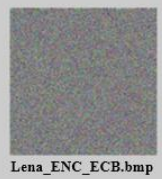
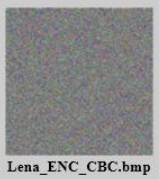
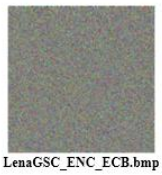
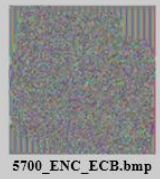
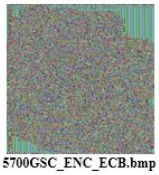
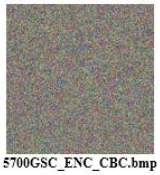
```
filein : array[] of byte
header : array[54] of byte
imagearray : array[] of byte
joinByte : array[] of byte {blok utuh hasil enkripsi / dekripsi}
fileout : array[] of byte

header[] = file[0 .. 53]
imagearray[] = file[54 .. n]
//n adalah index max untuk array berkas

..
.. proses enkripsi / dekripsi ..
..

BlockCopy(header, 0, fileout, 0, header.Length);
BlockCopy(joinByte, 0, fileout, header.Length, joinByte.Length);
```

4. PENGUJIAN DAN ANALISIS HASIL

Pengujian

Pengujian aplikasi dilakukan dalam dua bagian pada 10 variasi berkas citra. Pengujian pertama disebut dengan pengujian kriptografi citra. Pengujian ini dilakukan untuk mengukur performa algoritma untuk berkas citra dengan cara mencatat ukuran citra, lama waktu proses dan jenis citra. Pengujian pertama menggunakan kunci dan *initialization vector* CBC yang sama yaitu **"informatika09084"**. Beberapa contoh hasil pengujian pertama ditampilkan dalam gambar 8.

Pengujian kedua disebut dengan pengujian kemiripan citra hasil. Pengujian ini dilakukan dengan membandingkan setiap nilai *pixel* berkas citra asli (*plain image*) atau citra sebelum dienkripsi dengan citra hasil dekripsi kembali. Setiap nilai *pixel* dilakukan pembandingan untuk melihat perubahan yang terjadi. Kemudian setiap *pixel* yang berbeda dihitung untuk dibuat persentase dengan total *pixel* gambar Metode ini disebut dengan *Measurement Based on Value Changing* (Jolfaei & Mirghadri, 2010). Gambar 9 menampilkan hasil pengujian kemiripan citra hasil.

No.	Citra Asli	Hasil Enkripsi 128 bit dengan ECB	Hasil Dekripsi 128 bit dengan ECB	Hasil Enkripsi 128 bit dengan CBC	Hasil Dekripsi 128 bit dengan CBC
1	 Lena.bmp ukuran file : 768,1 KB dimensi : 512 x 512 px	 Lena_ENC_ECB.bmp waktu proses : 7065 ms	 Lena_DEC_ECB.bmp waktu proses : 25771 ms	 Lena_ENC_CBC.bmp waktu proses : 7130 ms	 Lena_DEC_CBC.bmp waktu proses : 25910 ms
2	 LenaGSC.bmp ukuran file : 257,1 KB dimensi : 512 x 512 px	 LenaGSC_ENC_ECB.bmp waktu proses : 2365 ms	 LenaGSC_DEC_ECB.bmp waktu proses : 8603 ms	 LenaGSC_ENC_CBC.bmp waktu proses : 2415 ms	 LenaGSC_ENC_CBC.bmp waktu proses : 8637 ms
3	 5700.bmp ukuran file : 328,2 KB dimensi : 400 x 280 px	 5700_ENC_ECB.bmp waktu proses : 3023 ms	 5700_DEC_ECB.bmp waktu proses : 10947 ms	 5700_ENC_CBC.bmp waktu proses : 3040 ms	 5700_DEC_CBC.bmp waktu proses : 10972 ms
4	 5700GSC.bmp ukuran file : 110,4 KB dimensi : 400 x 280 px	 5700GSC_ENC_ECB.bmp waktu proses : 1015 ms	 5700GSC_DEC_ECB.bmp waktu proses : 3714 ms	 5700GSC_ENC_CBC.bmp waktu proses : 1034 ms	 5700GSC_DEC_CBC.bmp waktu proses : 3727 ms

Gambar 8. Contoh hasil pengujian kriptografi citra

No.	Nama Citra	persentase kemiripan bila diproses dengan mode operasi ECB	persentase kemiripan bila diproses dengan mode operasi CBC
1	Lena.bmp	100 %	100 %
2	LenaGSC.bmp	100 %	100 %
3	5700.bmp	100 %	100 %
4	5700GSC.bmp	100 %	100 %
5	Moto.bmp	100 %	100 %
6	MotoGSC.bmp	100 %	100 %
7	Sky.bmp	100 %	100 %
8	SkyGSC.bmp	100 %	100 %
9	Tux.bmp	100 %	100 %
10	TuxGSC.bmp	100 %	100 %

Gambar 9. Hasil pengujian kemiripan citra hasil

Analisis Hasil

Berdasarkan 10 variasi citra yang telah diuji pada pengujian dihasilkan simpulan sebagai berikut :

a. Berkas citra yang dienkripsi berubah menjadi citra yang acak dengan tampilan berupa gambar *noise* yang tidak teratur dan dapat dikembalikan seperti semula.

b. Pengujian kemiripan citra hasil menunjukkan bahwa citra dapat didekripsi kembali menjadi seperti semula dengan tingkat kemiripan menunjukkan nilai 100 % pada semua citra uji.

c. Mode operasi mempengaruhi hasil enkripsi. Citra 3, 4, 5, 6, 9 dan 10 menunjukkan bahwa mode ECB tidak sesuai untuk digunakan pada berkas citra dengan latar belakang putih atau citra dengan bagian yang mempunyai kontras tinggi. Penggunaan mode operasi CBC lebih sesuai untuk berbagai jenis berkas citra, termasuk citra dengan latar belakang putih.

d. Semua citra *input* yang diuji menunjukkan bahwa proses dekripsi membutuhkan waktu lebih lama daripada proses enkripsi.

e. Ukuran kunci berpengaruh pada lamanya waktu proses enkripsi dan dekripsi. semakin panjang jenis kunci yang dipilih maka akan semakin lama waktu proses.

f. Ukuran *file* berpengaruh pada lamanya waktu proses enkripsi dan dekripsi. Semakin besar ukuran *file* maka semakin lama proses enkripsi dan dekripsi pada suatu berkas citra.

g. Semua citra *input* berjenis *greyscale* yang telah diuji menunjukkan bahwa citra berwarna membutuhkan waktu proses lebih lama daripada *greyscale*.

5. KESIMPULAN

Berdasarkan hasil pengujian dan analisis yang telah dilakukan terhadap berbagai *input data*, maka dapat diambil kesimpulan sebagai berikut :

a. Rancang bangun untuk implementasi algoritma kriptografi dengan S-box dinamis bergantung pada kunci utama berbasis AES telah berhasil dibangun dengan menggunakan dua jenis mode operasi.

b. Implementasi S-box dinamis tidak mengganggu efektivitas dari basis algoritma AES. Hal ini dibuktikan dengan hasil pengujian kemiripan hasil dekripsi dengan citra asli yang menunjukkan 100 % mirip di semua citra uji.

c. Waktu proses enkripsi dan dekripsi bergantung pada jenis kunci, ukuran gambar dan jenis gambar. Semakin panjang kunci maka akan semakin lama waktu proses. Semakin besar ukuran gambar maka semakin lama waktu proses. Selain itu, citra berwarna membutuhkan waktu lebih lama dibanding *grayscale*.

d. Jika citra yang digunakan mempunyai bagian dengan tingkat kontras yang tinggi maka *cipher* dengan mode ECB masih akan terlihat pola asli gambar. Oleh karena itu, lebih baik menggunakan CBC daripada ECB untuk berkas citra.

SARAN

Saran untuk pengembangan lebih lanjut tentang penelitian dengan tema sejenis adalah sebagai berikut:

a. Implementasi kriptografi dapat dilakukan juga pada format berkas citra lain seperti JPEG, TIFF, PNG dan GIF. Selain itu tidak menutup kemungkinan untuk jenis berkas multimedia lainnya, seperti *audio* dan *video*.

b. Jenis mode operasi lain seperti CFB, OFB dan CTR juga dapat sebagai alternatif untuk mode operasi yang digunakan pada kriptografi berkas citra.

6. DAFTAR PUSTAKA

[1] Arlow, J. & Neudtadt, I., 2002. *"UML and The Unified Process : Practical Object Oriented Analysist & Design"*. London: Pearson Education Limited.

[2] Arlow, J. & Neudtadt, I., 2005. *"UML 2 and The Unified Process : Practical Object Oriented Analysist & Design"*. London: Pearson Education Limited.

[3] Arrag, S., Hamdoun, A., Tragha, A. & Khamlich, S. E., 2013. Implementation of Stronger AES By Using Dynamic S-Box Dependent of Master Key. *Journal of Theoritical and Applied IT*, Volume 53, p. 197.

[4] Daemen, J. & Rijmen, V., 2001. *"The Design of Rijndael : AES - The Advanced Encryption Standard"*. Berlin: Springer.

[5] Dworkin, M., 2001. *"Recommendation for Block Cipher Modes of Operation"*. Washington: U.S. Government Printing Office.

[6] Fowler, M. & Scott, K., 2000. *"UML Distilled"*. Canada: Addison Wesley Longman Inc.

[7] H, E. D. & Risal, L., 2011. *"Pemrograman Berorientasi Objek C#"*. Bandung: Penerbit Informatika.

[8] Jolfaei, A. & Mirghadri, A., 2010. A New Approach to Measure Quality of Image Encryption. International Journal of Computer and Network Security, 2(8), pp. 38-43.

[9] Larman, C., 2004. *"Applying UML and Patterns"*. New Jersey: Prentice Hall.

[10] Munir, R., 2004. *"Pengolahan Citra Digital dengan Pendekatan Algoritmik"*. Bandung: Penerbit Informatika.

[11] Munir, R., 2006. *"Kriptografi"*. Bandung: Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, ITB.

[12] NIST, 2001. *Advanced Encryption Standard (AES)*, s.l.: Springfield.

[13] Oestereich, B., 2002. *"Developing Software with UML Object Oriented Analysis and Design in Practice"*. London: Pearson Education Limited.

[14] Paar, C. & Pelzl, J., 2010. *"Understanding Cryptography : A Textbook for Students and Pratitioners"*. New York: Springer.

[15] Rumbaugh, J., Jacobson, I. & Booch, G., 1999. *"The Unified Modeling Language Reference Manual"*. Canada: Addison Wesley Longman.

[16] Sadikin, R., 2012. *"Kriptografi untuk Keamanan Jaringan"*. Yogyakarta: Penerbit Andi.

[17] Shirley, 2011. *"Analisis dan dan Implementasi Algoritma AES dalam Enkripsi Suara"*, Bandung: Makalah Kriptografi IF3058 ITB.

[18] S, R. A. & Shalahuddin, M., 2013. *"Rekayasa Perangkat Lunak : Terstruktur dan Berorientasi Objek"*. Bandung: Penerbit Informatika.

[19] Stallings, W., 2011. *"Cryptography and Network Security Principles and Practice : 5th edition"*. New York: Prentice Hall.

[20] Surian, D., 2006. "Algoritma Kriptografi AES Rijndael". *TESLA Jurnal Teknik Elektro*, 8(2), pp. 97-101.

[21] Widodo, P. P. & Herlawati, 2011. *"Menggunakan UML"*. Bandung: Penerbit Informatika.