Implementasi *File-Based Encryption Pada Sebuah Aplikasi*

Authors Name - 13519017 (*Muhammad Galih Raihan Ramadhan*)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail : 13519017@std.stei.itb.ac.id

***Abstract*—File-based encryption* diimplementasikan pada sebuah aplikasi agar semua data yang dibuat dan dibuka oleh aplikasi tersebut memerlukan sebuah *key* untuk mencegah data tersebut diambil oleh pihak yang tidak berwenang.**

***Keywords—AES, kriptogafi kunci simetri,* file-based encryption**

I. Pendahuluan

Keamanan dan privasi sudah lama menjadi 2 buah hal yang semua orang inginkan, dan di dunia teknologi informasi tidak ada yang lebih penting selain keamanan dan data pribadi kita sendiri. Seiring berkembangnya teknologi, semakin sulit bagi setiap orang untuk menjaga keamanan data mereka masing-masing. Bahkan data yang mereka simpan pada perangkat keras yang mereka miliki tidak bisa dijamin keamanannya.

Pada penelitian ini, akan dilakukan eksplorasi mengenai salah satu cara menjaga keamanan data yang kita simpan pada perangkat milik kita sendiri dan sebuah cara untuk mengaplikasikannya pada sebuah program sederhana.

II. Landasan Teori

1. Kriptografi

Kriptografi secara singkat bisa dijelaskan sebagai suatu teknik yang digunakan untuk mencegah pihak yang tidak diinginkan untuk mengakses suatu pesan. Kriptografi yang dimaksud dalam makalah ini adalah kriptografi modern yang dilakukan menggunakan komputer digital. Kriptografi modern pada umumnya dilakukan menggunakan sebuah *key* yang digunakan untuk mengenkripsikan pesan agar tidak bisa dibaca secara langsung serta mendekripsikan pesan agar pesan bisa dibaca kembali.

Teknik kriptografi dirancang agar metode pemecahannya sulit dilakukan, secara teknis masih bisa dilakukan tetapi memerlukan waktu dan/atau kemampuan komputasi yang besar. Kriptografi modern dilakukan pada tingkat bit dari data pesan. Dengan melakukan perubahan pada bit-bit inilah pesan dienkripsikan menjadi pesan lainnya yang tidak bermakna.

Terdapat dua buah kategori khusus pada kriptografi berbasis bit, yaitu ‘cipher alir’ dan ‘cipher blok’. Perbedaan kedua kategori ini terletak pada jumlah bit atau *byte* yang dienkripsi maupun didekripsi pada satu waktu. Cipher alir beroperasi hanya pada satu bit maupun *byte* tunggal, sementara cipher blok beroperasi pada sebuah bit blok, pada umumnya 64 - 256 bit per blok.

Salah satu proses komputasi yang paling sering digunakan pada kriptografi modern adalah operasi *XOR* yang menerima 2 buah input bit dan mengembalikan sebuah output. Berikut adalah tabel yang menggambarkan operasi tersebut.

Tabel 1.1 Tabel Operasi *XOR*

	0	**1**
0	0	1
1	1	0

2. Kriptografi Kunci Simetri

Kriptografi kunci simetri adalah salah satu cabang dari kriptografi modern. Teknik kriptografi ini dikarakterisasikan oleh penggunaan *key* yang sama dalam proses pengenkripsian dan pendekripsian pesan. Metode ini sering digunakan apabila pesan yang rahasia hanya perlu diakses oleh 1 buah pihak.

Konsep yang sering digunakan dalam kriptografi kunci simetri adalah *substitution-box (S-box)* dan *permutation-box (P-box)*. *S-box* berfungsi untuk mensubstitusi bit pada pesan menjadi bit yang berbeda, sementara *P-box* berfungsi untuk mengacak urutan bit pada pesan. *S-box* dan *P-box* yang digunakan pada sebuah cipher biasanya bersifat konstan, akan tetapi ada beberapa buah cipher yang memiliki *S-box* dan *P-box* yang bergantung pada *key* yang digunakan.

3. AES (Advanced Encryption Standard)

AES adalah sebuah kriptografi kunci simetri berbasis cipher blok. Setiap blok *AES* memiliki panjang 128 bit, sementara kunci yang digunakan adalah antara 128 sampai 256 bit dengan step 32 bit. Proses *AES* juga dilakukan dalam beberapa jumlah putaran, antara 12 sampai 14 kali. Cara kerja *AES* secara singkat adalah sebagai berikut:

a. *AddRoundKey* : Melakukan operasi *XOR* antara *key* dengan pesan.

b. *ExpansionKey* : Pembentukan *round key* menggunakan *key* yang diberikan.
c. *SubBytes* : Substitusi *byte* pesan menggunakan *S-box.*
d. Membagi blok pesan berukuran 128 bit menjadi sebuah *array* berukuran 4x4, lalu dilakukan langkah-langkah berikut.
    i. *ShiftRows* : Pergeseran data pada baris *array.*
    ii. *MixColumns* : Mengacak data pada kolom *array*.
e. *AddRoundKey* : Operasi *XOR* antara keadaan pesan sekarang dengan *round key*.
f. Langkah b sampai dengan e kemudian dilakukan lagi sebanyak jumlah putaran cipher dikurangi 2.
g. Untuk langkah terakhir dilakukan lagi proses *SubBytes, ShiftRows,* dan *AddRoundKey*.

S-box yang digunakan pada AES tidaklah berubah. Berikut adalah *S-box* yang digunakan pada cipher tersebut, tabel telah dibagi menjadi 2 buah bagian agar lebih mudah dibaca.

	00	**01**	**02**	**03**	**04**	**05**	**06**	**07**
00	63	7c	77	7b	f2	6b	6f	c5
10	ca	82	c9	7d	fa	59	47	f0
20	b7	fd	93	26	36	3f	f7	cc
30	04	c7	23	c3	18	96	05	9a
40	09	83	2c	1a	1b	6e	5a	a0
50	53	d1	00	ed	20	fc	b1	5b
60	d0	ef	aa	fb	43	4d	33	85
70	51	a3	40	8f	92	9d	38	f5
80	cd	0c	13	ec	5f	97	44	17
90	60	81	4f	dc	22	2a	90	88
a0	e0	32	3a	0a	49	06	24	5c
b0	e7	c8	37	6d	8d	d5	4e	a9
c0	ba	78	25	2e	1c	a6	b4	c6
d0	70	3e	b5	66	48	03	f6	0e
e0	e1	f8	98	11	69	d9	8e	94
f0	8c	a1	89	0d	bf	e6	42	68

TABEL 2.1 TABEL *S-BOX* BAGIAN PERTAMA

	08	**09**	**0a**	**0b**	**0c**	**0d**	**0e**	**0f**
00	30	01	67	2b	fe	d7	ab	76
10	ad	d4	a2	af	9c	a4	72	c0
20	34	a5	e5	f1	71	d8	31	15
30	07	12	80	e2	eb	27	b2	75
40	52	3b	d6	b3	29	e3	2f	84
50	6a	cb	be	39	4a	4c	58	cf
60	45	f9	02	7f	50	3c	9f	a8
70	bc	b6	da	21	10	ff	f3	d2
80	c4	a7	7e	3d	64	5d	19	73
90	46	ee	b8	14	de	5e	0b	db
a0	c2	d3	ac	62	91	95	e4	79
b0	6c	56	f4	ea	65	7a	ae	08
c0	e8	dd	74	1f	4b	bd	8b	8a
d0	61	35	57	b9	86	c1	1d	9e
e0	9b	1e	87	e9	ce	55	28	df
f0	41	99	2d	0f	b0	54	bb	16

TABEL 2.2 TABEL S-BOX BAGIAN KEDUA

Sementara itu, permutasi pada AES tidak dilakukan dengan menggunakan sebuah *P-box*, seperti kebanyakan kriptografi lainnya. Operasi *ShiftRows* dan *MixColumns* berfungsi untuk melakukan permutasi tersebut dan mereka melakukannya tanpa menggunakan sebuah *P-Box.*

Pada operasi *ShiftRows* setiap bit pada baris *array* digeser secara siklikal ke kiri. Jumlah pergeserannya sesuai dengan nomor baris tersebut. Baris pertama (0) tidak melakukan pergeseran, baris kedua (1) bergeser 1 kali, dan seterusnya.

Pada operasi *MixColumns* setiap bit pada kolom *array* dilakukan sebuah transformasi linear. Transformasi dilakukan sebagai berikut, dengan b sebagai hasil dari operasi dan a sebagai input dari operasi.

$$\begin{bmatrix} b0 \\ b1 \\ b2 \\ b3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a0 \\ a1 \\ a2 \\ a3 \end{bmatrix}$$

Setelah itu, hasil perkalian di modulo-kan dengan polinomial $x^8 + x^4 + x^3 + x + 1$. Hal ini dilakukan karena proses enkripsi dan dekripsi dilakukan pada *galois field* GF (2^8). Sifat ini dari kriptografi tidak akan dijelaskan lebih lanjut pada makalah ini.

4. Filesystem Encryption/File-Based Encryption

Filesystem encryption atau nama lainnya *file-based encryption* adalah sebuah teknik enkripsi data yang beroperasi pada tingkatan *filesystem*. Biasanya enkripsi dilakukan per direktori atau *file* pada *filesystem* tersebut. Semua direktori atau *file* yang disimpan pada *filesystem* ini tidak bermakna sebelum dilakukan enkripsi, mencegah pihak yang tidak memiliki *key* untuk membaca data yang sebenarnya.

5. Salt

Salt adalah sebutan untuk sebuah tambahan pada pesan yang berfungsi untuk menambah keacakan *(randomness)* suatu pesan. Fungsi utama *salt* adalah untuk mencegah 2 buah pesan yang sama dienkripsikan menjadi cipher yang sama pula.

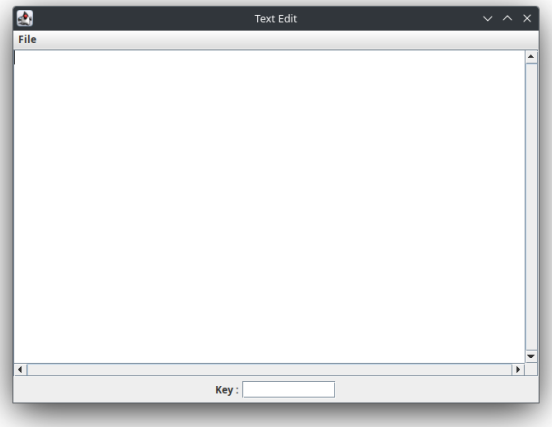
III. Pembahasan

1. *Implementasi*

Untuk laporan ini, aplikasi yang digunakan untuk mendemonstrasikan *file-based encryption* adalah sebuah aplikasi *text editor* sederhana yang dibuat menggunakan *framework swing* dari *Java*.

File-based encryption diaplikasikan secara *hard-coded* pada aplikasi tersebut. Pada *text editor* akan ada berupa *field key* agar pengguna dapat memasukkan *key* yang mereka inginkan, kemudian ketika proses menyimpan *file* dilakukan, *file* akan dienkripsikan menggunakan cipher AES dan *key* yang diberikan, begitu pula ketika *file* akan dibuka, maka *file* akan didekripsikan sebelum hasilnya ditunjukkan kepada pengguna.

Di luar hal-hal tersebut aplikasi berfungsi seperti *text editor* pada umumnya.



Gambar 2.3 *GUI Text editor*

2. *Kode Program*

Kode program dibuat dalam bahasa *Java* seperti yang dijelaskan sebelumnya. *File-file* pada program tersebut juga sudah dibagi mengikuti konvensi pada bahasa pemrograman *Java*.

Berikut adalah beberapa jumlah kode dari bagian *text editor* :

a. Fungsi *open*

```
//OPEN
      if (ae.equals("Open")) {
            returnValue =
jfc.showOpenDialog(null);
            if (returnValue ==
JFileChooser.APPROVE_OPTION) {
            File f = new
File(jfc.getSelectedFile().getAbsolutePath(
));
            try{
                  FileReader read = new
FileReader(f);
                  Scanner scan = new
Scanner(read);

while(scan.hasNextLine()){
                        String line =
scan.nextLine();// + "\n";
                        ingest = ingest +
line;
                  }
                  if (tf.getText() != "")
{
                        AES aes = new
AES();
                        String plain =
aes.decrypt(ingest, tf.getText());
                        if (plain ==
null) {
                        plain = "Error
decrypting text.";
                        }

area.setText(plain);
                  }
                  else {

area.setText(ingest);
                  }

            }
            catch ( FileNotFoundException
ex) { ex.printStackTrace(); }
            }
      }
```

b. Fungsi *save*

```
// SAVE
	else if (ae.equals("Save")) {
		returnValue =
jfc.showSaveDialog(null);
		try {
		File f = new
File(jfc.getSelectedFile().getAbsolutePath(
));
		FileWriter out = new
FileWriter(f);
		AES aes = new AES();
		String cipher =
aes.encrypt(area.getText(), tf.getText());
		out.write(cipher);
		out.close();
		}
		catch (FileNotFoundException
ex) {
		Component f = null;

JOptionPane.showMessageDialog(f,"File not
found.");
		}
		catch (IOException ex) {
		Component f = null;

JOptionPane.showMessageDialog(f,"Error.");
		}
	}
```

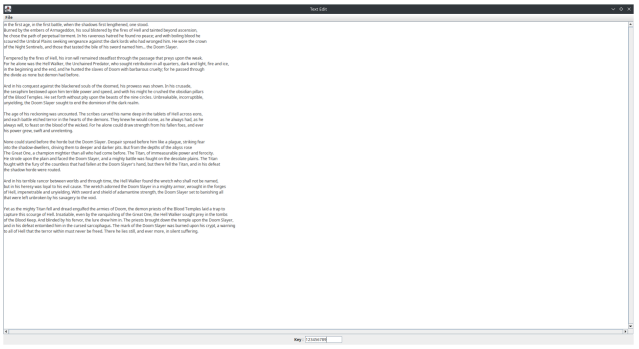
Untuk bagian implementasi AES, digunakan *library crypto* dan *security* yang tersedia di *Java.* Berikut adalah kodenya :

```
public class AES
{
	/* Private variable declaration */
	//private static final String
SECRET_KEY = "123456789";
	private static final String SALTVALUE
= "abcdefg";

	/* Encryption Method */
	public static String encrypt(String
strToEncrypt, String SECRET_KEY)
	{
	try
	{
	/* Declare a byte array. */
	byte[] iv = {0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0};
	IvParameterSpec ivspec = new
IvParameterSpec(iv);
	/* Create factory for secret keys. */
	SecretKeyFactory factory =
SecretKeyFactory.getInstance("PBKDF2WithHma
cSHA256");
	/* PBEKeySpec class implements
KeySpec interface. */
	KeySpec spec = new
PBEKeySpec(SECRET_KEY.toCharArray(),
SALTVALUE.getBytes(), 65536, 256);
	SecretKey tmp =
factory.generateSecret(spec);
	SecretKeySpec secretKey = new
SecretKeySpec(tmp.getEncoded(), "AES");
	Cipher cipher =
Cipher.getInstance("AES/CBC/PKCS5Padding");
	cipher.init(Cipher.ENCRYPT_MODE,
secretKey, ivspec);
	/* Retruns encrypted value. */
	return Base64.getEncoder()
.encodeToString(cipher.doFinal(strToEncrypt
.getBytes(StandardCharsets.UTF_8)));
	}
	catch
(InvalidAlgorithmParameterException |
InvalidKeyException |
NoSuchAlgorithmException |
InvalidKeySpecException |
BadPaddingException |
IllegalBlockSizeException |
NoSuchPaddingException e)
	{
	System.out.println("Error occured
during encryption: " + e.toString());
	}
	return null;
	}

	/* Decryption Method */
	public static String decrypt(String
strToDecrypt, String SECRET_KEY)
	{
	try
	{
```

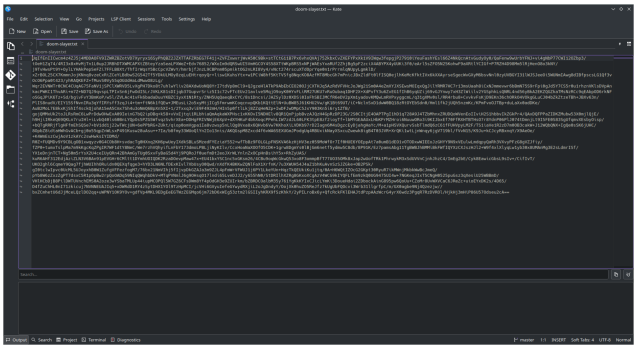
```
        /* Declare a byte array. */
        byte[] iv = {0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0};
        IvParameterSpec ivspec = new
IvParameterSpec(iv);
        /* Create factory for secret keys. */
        SecretKeyFactory factory =
SecretKeyFactory.getInstance("PBKDF2WithHma
cSHA256");
        /* PBEKeySpec class implements
KeySpec interface. */
        KeySpec spec = new
PBEKeySpec(SECRET_KEY.toCharArray(),
SALTVALUE.getBytes(), 65536, 256);
        SecretKey tmp =
factory.generateSecret(spec);
        SecretKeySpec secretKey = new
SecretKeySpec(tmp.getEncoded(), "AES");
        Cipher cipher =
Cipher.getInstance("AES/CBC/PKCS5PADDING");
        cipher.init(Cipher.DECRYPT_MODE,
secretKey, ivspec);
        /* Retruns decrypted value. */
        return new
String(cipher.doFinal(Base64.getDecoder().d
ecode(strToDecrypt)));
        }
        catch
(InvalidAlgorithmParameterException |
InvalidKeyException |
NoSuchAlgorithmException |
InvalidKeySpecException |
BadPaddingException |
IllegalBlockSizeException |
NoSuchPaddingException e)
        {
        System.out.println("Error occured
during decryption: " + e.toString());
        }
        return null;
        }
}
```

3. Contoh kerja aplikasi

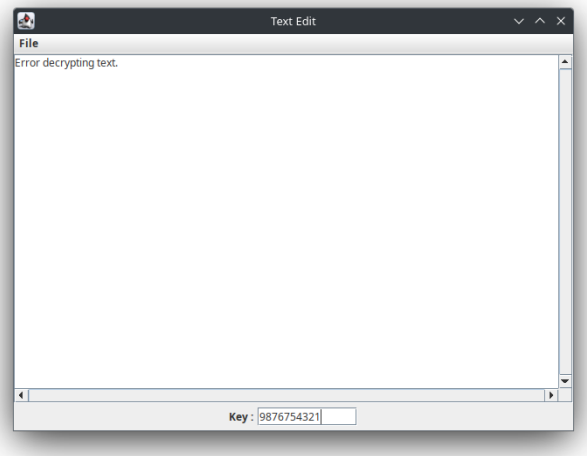
Berikut adalah sebuah contoh jalannya proses enkripsi dan enkripsi sebuah *file*.



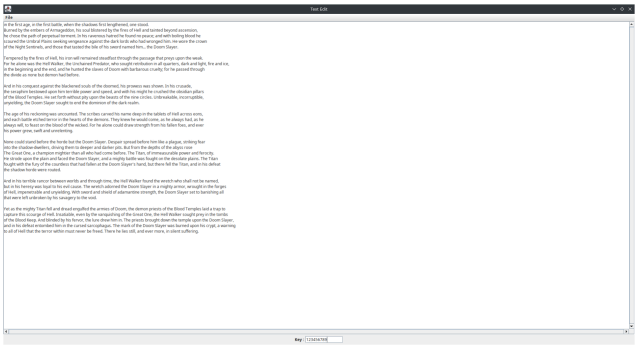
GAMBAR 3.1 TEKS SEBELUM DISIMPAN



GAMBAR 3.2 TEKS SETELAH DISIMPAN DAN DIBUKA PADA APLIKASI LAIN



GAMBAR 3.3.A PROSES DIBUKANYA TEXT DENGAN KEY YANG TIDAK TEPAT



GAMBAR 3.3.B PROSES DIBUKANYA TEKS DENGAN KEY YANG BENAR

IV. KESIMPULAN DAN SARAN

Aplikasi yang telah dibuat dan konsepnya sangatlah sederhana, akan tetapi keefektifannya dalam menangani keamanan data pribadi sangatlah baik. Keamanan data dapat dipastikan selama *key* yang digunakan dalam enkripsi maupun dekripsi tidak disimpan pada di sebuah lokasi yang dapat diakses oleh pihak yang tidak berwenang.

Aplikasi ini juga dibuat agar menambahkan cipher yang baru mudah dilakukan. Pada *source code* aplikasi terdapat beberapa file yang berhubungan dengan cipher *Hanzo* yang berupa cipher yang penulis buat, akan tetapi karena keterbatasannya waktu cipher tersebut tidak diimplementasikan.

Saran lainnya juga bisa berupa sebuah cipher yang memiliki tingkat komputasi yang ringan seperti *SPECK* atau *SIMON*. *Source code* aplikasi dapat diakses pada link berikut.
https://github.com/HagliHagli/if4020-texteditor

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah ini saya tulis sendiri dan bukanlah hasil kerja orang lain, maupun hasil terjemahan pekerjaan lainnya dan bukanlah sebuah hasil plagiarisme.

REFERENCES

[1] Munir, Renaldi. Slide kuliah mengenai kriptografi kunci simetri