ISSN : 2962 – 5688
Website : jurnal.umj.ac.id/index.php/TripleA
Email : triplea@umj.ac.id

TRIPLE A
Jurnal Pendidikan Teknologi Informasi

IMPLEMENTASI KRIPTOGRAFI *AES* MENGGUNAKAN BAHASA JAVA PROGRAMMING: MENINGKATKAN KEAMANAN DATA MELALUI ENKRIPSI & DEKRIPSI YANG KUAT

Ahmad Azis Iqbal Ramadhan, Ellena Zahrah Rivanti, & Rizky Syahrial Zulva

Teknik Multimedia & Jaringan, Fakultas Teknik Informatika & Komputer, Politeknik Negri Jakarta, f. DR. G.A. Siwabessy, Kukusan, Kecamatan Beji, Kota Depok, Jawa Barat 16425

ahmad.azisiqbalr.tik20@mhsw.pnj.ac.id
ellena.zahrahrivanti.tik20@mhsw.pnj.ac.id
rizky.syahrialzulva.tik20@mhsw.pnj.ac.id

Abstract

Security is a legitimate right for every individual. Every human being deserves proper security. Starting from personal security, identity security, and so on. Likewise, in the world of human technology, humans have the right to have guaranteed security in the internet world. Starting from personal data, history in the internet world, and so on. Likewise, in carrying out messages. The data/message here has the right to be kept confidential with the encryption & decryption process is a mandatory stage to maintain the confidentiality of the data/message. This is the right to protect from leaks. The use of encryption & decryption is one solution to prevent intruders from protecting data / messages that are confidential in nature, using the AES Cryptographic Algorithm is one of the secure algorithms. Used to perform the decryption & encryption process of a data / message. The use of cryptography is expected to keep confidential data secret. This implementation will use a programming language. That is with Java Programming so hopefully it can keep data/messages more secure

Keywords: *AES Cryptography, Java Programming, Data/Message, Encryption & Decryption*

Abstrak

Keamanan merupakan hak yang sah untuk setiap individu. Setiap manusia berhak mendapatkan keamanan yang layak. Mulai dari keamanan diri, keamanan identitas, dan lain sebagainya. Begitu juga di dunia teknologi manusia berhak memiliki keamanan yang terjamin di dunia internet. Mulai dari data diri, Riwayat di dunia internet, dan lain sebagainya. Begitu juga dalam melakukan pesan. Data/pesan disini berhak dijaga kerahasian-nya dengan proses

enkripsi & dekripsi merupakan tahapan yang wajib untuk menjaga kerahasian data/pesan. Ini merupakan hak untuk melindungi dari kebocoran. Penggunaan enkripsi & dekripsi merupakan salah satu solusi untuk mencegah dari para penyusup untuk melindungi data/pesan yang sifat-nya rahasia, dengan menggunakan Algoritma Kriptografi AES merupakan salah satu algoritma yang aman. Digunakan untuk melakukan proses dekripsi & enkripsi suatu data/pesan. Penggunaan Kriptografi ini diharapkan dapat merahasiakan data yang bersifat rahasia. Implementasi ini akan menggunakan bahasa pemrograman. Yaitu dengan Java Programming dengan begitu semoga dapat merahasiakan data/pesan lebih aman

Keywords: *Kriptografi AES, Java Programming, Data/Pesan, Enkripsi & Dekripsi*

PENDAHULUAN

Keamanan data atau pesan yang bersifat informasi secara rahasia merupakan suatu layanan yang dibuat untuk menjaga agar data atau pesan informasi ini tersimpan dengan baik dan aman. Yang nantinya tidak dapat terbaca atau terlihat oleh orang lain atau penyusup. Karena keamanan data adalah hak yang berhak didapatkan oleh semua orang. Oleh karena itu banyak sekali berbagai macam cara untuk mengamankan suatu data atau pesan.

Mengamankan suatu data memiliki banyak cara, seperti dengan penggunaan kriptografi yang digunakan untuk menjaga keamanan pesan atau data yang dikirim dari satu tempat ke tempat lainnya, konsep yang digunakan adalah dengan melakukan teknik enkrip atau dekrip. Nantinya pesan atau data ini akan berisi *plainText* kemudian nantinya akan diubah yang biasa disebut proses enkripsi dengan suatu *key*. Hasil enkrip ini tidak tentu (acak) dan jika sudah dienkripsi disebut dengan *cipherText.* Untuk menjadikan *ciphertext* kembali ke *plainText* proses ini biasa disebut dengan dekripsi. Proses ini akan mengembalikan *cipherText* ke *plainText,* dengan begitu penerima nantinya dapat membaca pesan yang diberikan. Tentunya dengan *key* yang sudah dibuat.

Inti dari *key* disini adalah suatu kunci yang digunakan untuk membuka atau mengunci suatu pesan atau data yang ingin dienkripsi atau dekripsi. Dengan cara ini orang lain atau penysup yang tidak memiliki hak untuk melihatnya tidak dapat melihat isi dari pesan atau data yang sebenarnya. Pada implementasi ini penulis menggunakan kunci simetri *(Simetric-key Algorithm)* yaitu *key* yang sama pada saat melakukan proses enkripsi dan dekripsi, lalu penulis menggunakan perlindungan data/pesan informasi dengan algoritma *AES (Advanced Ecryption Standard).* Nantinya penulis bisa menggunakan ini untuk men *encrypt/decrypt* pesan yang bersifat rahasia, tentunya dengan *Key*. Lalu pada percobaan ini penulis menggunakan bahasa Pemrograman *Java* dengan *Text Editor Visual Studio Code.*

Cryptography AES (Advanced Encryption Standard)

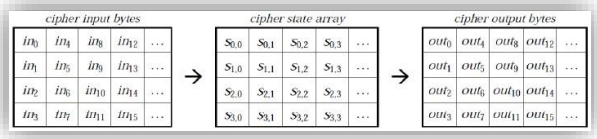
Kriptografi adalah proses mengubah pesan untuk merahasiakan dari yang sebenanrnya. Itu disebut dengan kriptografi yaitu menyembunyikan pesan atau adata

agar informasi bersifat rahasia. Yang diubah itu tidak bisa terbaca, jadi orang lain tidak akan mengetahuinya dengan begitu informasi pesan/data menjadi aman, tujuan-tujuan dari kriptografi ini ada empat *confentiality, non-repudiation, authentication, and integrity.* Pada percobaan kali ini penulis menggunakan Algoritma *kriptografi AES (Advanced Encryption System),* merupakan salah satu algoritma *block chipper* dan menggunakan kunci simetri pada saat proses melakukan enkripsi & dekripsi pesan. *Algoritma AES* memiliki berbagai macama *key* yang bervariasi yaitu *128-bit, 192-bit, dan 256-bit*. Perbedaan dari ketiga *key* tersebut ada pada panjang kunci-nya dan jumlah round-nya.

Tipe	Jumlah Key (Nk)	Besar Blok (Nb)	Jumlah Round(Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Tabel 1. *Jumlah Round AES*

Kemudian pada dasar-nya operasi *AES* dilakukan terhadap *array of byte* dua dimensi yang disebut dengan *state*. *State* mempunyai ukuran *Nrows X Ncols*. Ketika penulis melakukan proses enkripsi, data yang dimasukkan *berupa in0, in2, in3, in4, in5, in6, in7, in8, in9, in10, in11, in12, in13, in14, in15* akan disalin ke dalam *array state*.



Lalu hasil *output*-nya ini yang nantinya akan dilakukan operasi enkripsi & dekripsi. Berikut ini ada gambar mengilustrasikan proses tersebut.

Gambar 1. Proses *Input & Output Bytes*

Java Programming

Java Programming merupakan bahasa pemrograman yang dapat dijalankan diberbagai perangkat komputer. Dan biasanya *java* ini juga untuk membuat & menjalankan perangkat lunak pada komputer secara *standalone* ataupun pada lingkungan jaringan. Kemudian biasanya juga digunakan untuk mengembangkan bagian *back-end* dari *software*, aplikasi *android*, serta *website-website.*

METODE PENELITIAN

Metode yang digunakan adalah metode Tindakan yang bertujuan untuk mengembangkan kemampuan untuk mengimplementasikan dan analisis pada kriptografi *AES* dan melakukan Tindakan yang berkelanjukan untuk akselerasi pemahaman tentang algoritma ini. Langkah-langkah implementasi dilakukan secara repetisi dan terencana.

Metode penelitian ini menggabungkan observasi dan studi literatur untuk mendapatkan pemahaman yang komprehensif mengenai implementasi kriptografi *AES* dalam bahasa pemrograman *Java*. Melalu observasi, peneliti melakukan pengamatan terhadap proses enkripsi dan dekripsi, penggunaan kunci simetris, dan penggunaan algoritma *AES*. Sementara itu, dengan studi literatur, peneliti mengumpulkan informasi teoritis mengenai konsep dasar kriptografi, algoritma *AES* dan fitur-fitur yang relevan dalam bahasa pemrograman *Java*. Kedua pendekatan ini menghasilkan pemahaman yang holistic dan

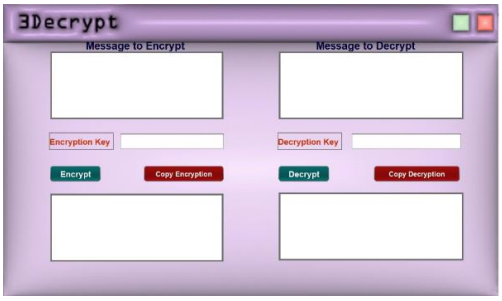
mendalam mengenai implementasi kriptografi *AES*.

Perancangan Perangkat lunak

Analisis kebutuhan dalam implementasi kriptografi *AES* dalam bahasa pemrograman *Java* meliputi kebutuhan fungsional dan keamanan. Secara fungsional, sistem mampu mengimplementasikan fungsi enkripsi dan dekripsi menggunakan algoritma *AES*, dengan kunci simetsris. Secara keamanan, kekuatan enkripsi menjadi penting, di mana algoritma *AES* dengan panjang kunci simetris yang memadai harus diimplementasikan untuk mencapai tingkat keamanan yang sepadan.

HASIL DAN PEMBAHASAN

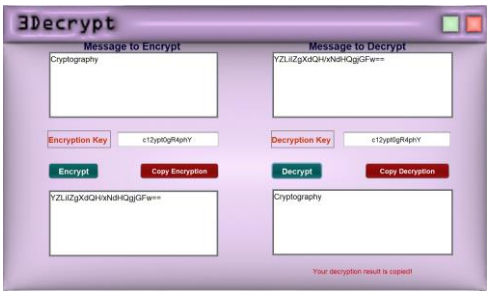
1. **Implementasi Tampilan Menu Utama**
   Pada proses diawal, akan menampilkan menu utama yang menunjukan menu *Massage to Encrypt* dan *Massage to Decrypt* sebagai *input*-an *file* yang akan di enkripsi dan di deskripsi.



Gambar 2. Tampilan menu utama.

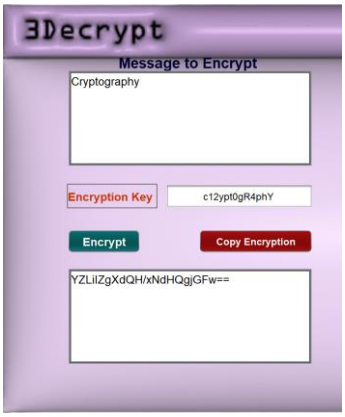
Pada Gambar 2 di atas menunjukan tentang tampilan awal. Dimana terdapat *text field* untuk memasukan kata yang akan di enkripsi dan di deskripsi, masukan kata kunci yang akan dijadikan *key*.

2. **Implementasi Tampilan *Input***
   Pada proses ini pengguna memasukan kata kedalam *text field* yang akan dienskripsi dan dideskripsi disertai dengan memasukan *key* lalu *click Encrypt/Decrypt.*



Gambar 3. Tampilan enkripsi/deskripsi *input*, *key*.

3. **Implementasi Tampilan *Output* dari *Encrypt/Decrypt***
   Hasil yang keluar ketika meng-click *Encrypt/Decrypt* menampilkan kata yang sudah di *Encrypt/Decrypt* oleh program.



Gambar 4. Tampilan kata yang sudah di *Encrypt*

Gambar 5. Tampilan kata yang sudah di *Decrypt.*

Pada Gambar 4 dan 5 di atas menunjukan tentang informasi yang didapat oleh pengguna sebagai hasil akhir dari proses *Encrypt/Decrypt.*

a. *Code Encrypt*

```
private void encryptActionPerformed(java.awt.event.ActionEvent evt) {
    String strToEncrypt;
    String secret;
    try {
        strToEncrypt = text1.getText();
        secret = msg1.getText();
        setKey(secret);
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);

text2.setText(Base64.getEncoder().encodeToString(cipher.doFinal(strToE
ncrypt.getBytes("UTF-8"))));
    } catch (Exception e) {
        text2.setText("Please fill up the right secret key");
    }
}
```

Gambar 6. Tampilan *Code Encrypt*

Pada Gambar 6 diatas Variabel *'strEncrypt'* akan menyimpan teks yang akan dienkripsi, sedangkan variabel *'secret'* akan menyimpan kunci rahasia yang digunakan untuk enkripsi. *'setKey'* dipanggil dengan argumen *secret*. Metode ini bertanggung jawab untuk mengubah teks kunci menjadi sebuah kunci yang dapat digunakan oleh algoritma enkripsi. Mengenkripsi teks tersebut menggunakan algoritma AES dengan kunci rahasia yang diberikan, dan menampilkan hasil enkripsi dalam bentuk *string Base64*.

b. *Code Decrypt*

```
private void decryptActionPerformed(java.awt.event.ActionEvent evt) {
    String secret;
    String strToDecrypt;
    try {
        secret = msg2.getText();
        strToDecrypt = text3.getText();
        setKey(secret);
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
        cipher.init(Cipher.DECRYPT_MODE, secretKey);
        text4.setText(new
String(cipher.doFinal(Base64.getDecoder().decode(strToDecrypt))));
    } catch (Exception e) {
        text4.setText("Please fill up the right secret key");
    }
}
```

Gambar 7. Tampilan *Code Decrypt*

Pada Gambar 7 diatas Variabel *'secret'* akan menyimpan kunci rahasia yang digunakan untuk dekripsi, sedangkan variabel *'strToDecrypt'* akan menyimpan teks yang akan didekripsi. Metode *'setKey'* dipanggil dengan argumen *secret*. Metode ini bertanggung jawab untuk mengubah teks kunci menjadi sebuah kunci yang dapat digunakan oleh algoritma dekripsi. Mendekripsi teks tersebut menggunakan algoritma AES dengan kunci rahasia yang diberikan, dan menampilkan hasil dekripsi dalam bentuk teks yang dapat dibaca.

c. *Code* Perintah *key*

```
public static void setKey(String myKey)
  {
    MessageDigest sha = null;
    try {
      key = myKey.getBytes("UTF-8");
      sha = MessageDigest.getInstance("SHA-1");
      key = sha.digest(key);
      key = Arrays.copyOf(key, 16);
      secretKey = new SecretKeySpec(key, "AES");
    }
    catch (NoSuchAlgorithmException e) {
      e.printStackTrace();
    }
    catch (UnsupportedEncodingException e) {
      e.printStackTrace();
    }
  }
```

Gambar 8. Tampilan *Code Key*

Pada Gambar 8 diatas *'setKey'* adalah metode statis, yang berarti metode ini dapat dipanggil tanpa menginstansiasi objek dari kelas yang mengandungnya. Metode ini menerima sebuah parameter dengan nama *'myKey'*, yang merupakan teks kunci yang akan diubah menjadi kunci yang valid untuk algoritma AES. Jika terjadi pengecualian *'NoSuchAlgorithmException'* atau *'UnsupportedEncodingException'* selama proses, pengecualian tersebut akan dicetak menggunakan metode *'printStackTrace'.*

Teks kunci yang diberikan diubah menjadi kunci yang valid dengan menggunakan *fungsi hash SHA-1* dan menghasilkan *array byte* dengan panjang yang sesuai dengan kebutuhan algoritma *AES*. Kunci yang dihasilkan disimpan dalam objek *'SecretKeySpec'* yang akan digunakan dalam proses enkripsi atau dekripsi.

SIMPULAN

Sistem otentikasi berbasis *One Time Password (OTP)* merupakan metode keamanan yang melibatkan penggunaan kata sandi sekali pakai yang berlaku hanya untuk satu sesi otentikasi. Algoritma *RSA (Rivest-Shamir-Adleman)* merupakan algoritma kriptografi asimetris yang sering digunakan untuk keperluan enkripsi, dekripsi, dan tanda tangan digital.

Dalam konteks diatas, perancangan sistem otentikasi berbasis *OTP* menggunakan algoritma *RSA* sebagai metode autentikasi mungkin melibatkan beberapa langkah dasar berikut:

1. Pembangkitan kunci *RSA*: Sistem akan menghasilkan sepasang kunci *RSA*, yaitu kunci publik dan kunci pribadi. Kunci publik digunakan untuk mengenkripsi *OTP*, sedangkan kunci pribadi digunakan untuk mendekripsi *OTP*.
2. Pembangkitan *OTP*: Setiap kali pengguna ingin melakukan otentikasi, sistem akan menghasilkan *OTP* secara acak.
3. Enkripsi *OTP*: *OTP* yang dihasilkan akan dienkripsi menggunakan kunci publik *RSA*. Hal ini memastikan bahwa hanya pemilik kunci pribadi yang dapat mendekripsi *OTP* tersebut.
4. Pengiriman *OTP*: *OTP* yang telah dienkripsi akan dikirimkan kepada pengguna melalui saluran komunikasi yang aman, misalnya melalui pesan teks atau aplikasi otentikasi.
5. Dekripsi *OTP*: Pengguna menerima *OTP* yang dienkripsi dan menggunakan kunci pribadi *RSA* untuk mendekripsinya.
6. Verifikasi *OTP*: Sistem akan membandingkan *OTP* yang didekripsi dengan *OTP* yang asli yang dihasilkan oleh sistem. Jika kedua *OTP* cocok, pengguna dianggap berhasil melakukan otentikasi.

Keuntungan dari menggunakan *OTP* dengan algoritma *RSA* sebagai metode

autentikasi termasuk keamanan yang tinggi karena penggunaan *OTP* sekali pakai dan kekuatan enkripsi yang dihasilkan oleh algoritma *RSA*.

DAFTAR PUSTAKA

Adm, Z. (2020, july 15). *Pengertian Bahasa Pemrograman Java: Fungsi, Contoh, Kelebihan & Kekurangan*. Diambil kembali dari www.zanoor.com:https://www.zanoor.com/pengertian-java/

Agus, S. M. (2009). Implementasi Perangkat Lunak Penyandian Pesan Menggunakan Algoritma RSA. *Jurnal Informatika Mulawarman*, 13-20.

Anggraeni Eka Putri, A. K. (2021). Implementasi Kriptografi Dengan Algoritma Advanced Encryption Standard (AES) 128 Bit dan Steganografi Menggunakan Metode End Of File (EOF) Berbasis Java Desktop Pada Dinas Pendidikan Kabupaten Tangerang. *Implementasi Kriptografi*, 69-79.

Asri Prameshwari, N. P. (2018). Implementasi Algoritma *Advanced Encryption Standard* (AES) 128 Untuk Enkripsi dan Dekripsi File Dokumen. *JURNAL EKSPLORA INFORMATIKA*, 52-58.

J.Tuturoong, N. (2010). Perbandingan Rasio dan Kecepatan Kompresi Menggunakan Algoritma Huffman, Lzw dan Dmc. *Tekno*, 18-31.

Lusiana, V. (2018). Implementasi Kriptografi Pada File Dokumen Menggunakan Algoritma AES-128. *Kriptografi*, 1-5.

Mhhabib. (2022, Januari 2). *AES-Encryption-and-Decryption-in-java-master*. Diambil kembali dari github.com: https://github.com/mhhabib/AES-Encryption-and-Decryption-in-java

Name, N. (2020, January 10). *Apa itu Cryptography, Pengertian, Tujuan, Jenis & Komponen*. Diambil kembali dari www.bitocto.com: https://bitocto.com/octopedia/apa-itu-cryptography/

Soni Harza Putra, E. S. (2010). Implementasi Algoritma Kriptografi Advanced Endryption Standard (AES) Pada Kompresi Data Teks. *IOT*, 3-14.

Suci Indah Febriani, Safitri Juanita, Mardi Hardjianto. "Implementasi Kriptografi Teks pada SMS Menggunakan Algoritme Multiple Encryption dengan Metode RSA dan 3DES". Jurnal Telematika, vol. 15 no.2, 2020. Institus Teknologi Harapan Bangsa, Bandung.