Implementasi Algoritma AES, ElGamal, dan SHA3 untuk Keamanan *File Digital*

Joshua Allan, Justinus Andjarwirawan, Lily Puspa Dewi
Program Studi Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jl. Siwalankerto 121-131 Surabaya, 60236
Telp. (031) – 2983455, Fax. (031) - 8418658
E-mail : joshuaallan.mx@gmail.com, justin@petra.ac.id, lily@petra.ac.id

ABSTRAK

Di era yang serba modern ini, perkembangan teknologi di dunia berkembang dengan sangat cepat. Kemajuan teknologi yang sangat pesat ini membuat pekerjan dan pertukaran data maupun informasi menjadi sangat cepat, akurat, dan efisien. Tetapi dengan adanya kemudahan tersebut juga muncul celah keamanan baru seperti munculnya tindak kejahatan di dunia maya (*cyber-crime*). Tindakan-tindakan tersebut dapat berupa pencurian data, pembobolan kata sandi (*password*) seseorang, hingga penyebarluasan perangkat lunak (*software*) ataupun data seseorang tanpa ijin. Oleh karena itu, penelitian ini bertujuan untuk mengatasi masalah keamanan tersebut dengan cara mengimplementasikan algoritma AES, algoritma ElGamal, dan fungsi *hash* SHA3 agar dapat digunakan secara bersamaan menjadi suatu sistem enkripsi yang aman.

Pengujian dalam penelitian ini terbagi menjadi tiga tahap, yaitu pengujian kemampuan sistem dalam melakukan proses enkripsi dan dekripsi, pengujian kecepatan sistem, dan pengujian *web platform*. Dari hasil pengujian dapat dibuktikan bahwa program dapat melakukan enkripsi dan dekripsi terhadap semua jenis *file* dengan baik dan dapat mendeteksi keaslian *file* tersebut. Pada pengujian *web*, dapat diketahui bahwa *web* mampu mengirimkan *file* yang terenkripsi kepada penerima dengan baik tanpa ada satupun yang hilang.

Penelitian ini menghasilkan sebuah perangkat lunak yang dapat melakukan enkripsi dan dekripsi terhadap berbagai jenis *file* dan dapat memverifikasi jika terdapat perubahan/modifikasi terhadap *file* tersebut. Penelitian ini juga menghasilkan sebuah *web platform* yang dapat digunakan untuk melakukan pengiriman *file* antar pengguna dengan benar.

Dari hasil pengujian, program dapat melakukan enkripsi pada *file* dengan kecepatan rata-rata yaitu 999.43 bytes/ms dan melakukan dekripsi pada *file* dengan kecepatan rata-rata yaitu 450.38 bytes/ms.

Kata Kunci: algoritma AES, algoritma ElGamal. algoritma SHA3, enkripsi, keamanan, berkas elektronik.

ABSTRACT

In this modern era, technological developments in the world are developing very quickly. This very rapid technological advancement makes work and exchange of data and information very fast, accurate, and efficient. But with this convenience, new security gaps have also emerged, such as the emergence of crimes in cyberspace (cyber-crime). These actions can be in the form of data theft, breaking someone's password, to distributing software or someone's data without permission. Therefore, this study aims to overcome these security problems by implementing the AES algorithm, ElGamal algorithm, and SHA3 hash function so that they can be used simultaneously to become a secure and solid encryption system.

The testing in this study is divided into three stages, namely testing the system's ability to perform the encryption and decryption process, testing the system speed, and testing the web platform. From the test results, it can be proven that the program can encrypt and decrypt all types of files properly and can detect the authenticity of the file. In web testing, it can be seen that the web is able to send encrypted files to recipients properly without missing anything.

This research produces a software that can perform encryption and decryption of various types of files and can verify if there are changes / modifications to the file. This research also produces a web platform that can be used to send files between users correctly.

From the test results, the program can encrypt files with an average speed of 999.43 bytes/ms and decrypt files with an average speed of 450.38 bytes/ms.

Keywords: *AES algorithm, ElGamal algorithm, SHA3 algorithm, encryption, security, file.*

1. PENDAHULUAN

1.1 Latar Belakang

Pada jaman sekarang ini, perkembangan teknologi di dunia berkembang dengan sangat cepat. Hampir setiap aspek dapat digantikan dengan teknologi mulai dari segi komunikasi, cara bertransaksi, hingga cara untuk bertukar informasi. Kemajuan teknologi yang sangat pesat ini membuat pekerjan dan pertukaran data maupun informasi menjadi sangat cepat, akurat, dan efisien.

Dengan berkembang pesatnya teknologi dalam cara kita bertukar informasi ini juga tentunya membuat celah keamanan baru. Pihak-pihak yang tidak bertanggung jawab bisa saja melakukan tindak kejahatan di dunia maya (*cyber-crime*). Tindakan-tindakan tersebut dapat berupa pencurian data, pembobolan kata sandi (*password*) seseorang, hingga penyebarluasan perangkat lunak (*software*) ataupun data seseorang tanpa ijin atau melanggar hak cipta dari pemilik data tersebut [1]. Celah-celah keamanan ini akan sangat merugikan apabila tidak ada tindak pencegahan dari pihak yang bersangkutan.

Dalam pengiriman dan penyimpanan file digital melalui perangkat elektronik, diperlukan sebuah metode yang dapat menjamin keamanan dan keutuhan dari data yang dikirimkan tersebut. Data tersebut harus tetap terjaga kerahasiannya selama pengiriman dan harus tetap terjaga integritasnya pada saat penerimaan di tujuan. Untuk merealisasikan hal tersebut, perlu dilakukan proses enkripsi yang berguna untuk menjaga kerahasiaan dari isi file tersebut dan

dilakukan pula proses signing untuk menjamin keaslian dan integritas yang memastikan file masih utuh dan tidak ada perubahan pada file tersebut.

Dalam hal keamanan data, terdapat 4 aspek layanan keamanan yaitu kerahasiaan (*confidentiality*), keutuhan (*integrity*), autentikasi (*authentication*) dan nirpenyangkalan (*non-repudiation*) [9]. Banyak algoritma kriptografi telah digunakan untuk melakukan pengamanan terhadap data. Penelitian sejenis juga pernah dilakukan oleh Indra Gunawan pada tahun 2018 yang berjudul "Kombinasi Algoritma Caesar Cipher dan Algoritma RSA untuk Pengamanan File Dokumen dan Pesan Teks" [4]. Pada penelitian ini digunakan metode enkripsi ganda yaitu dengan mengkombinasikan Caesar Cipher dan algoritma RSA yang menghasilkan kesimpulan bahwa dengan mengkombinasikan kedua metode algoritma tersebut dapat membantu meningkatkan keamanan data, jika dibandingkan dengan hanya menggunakan satu metode saja. Tetapi penelitian tersebut masih memiliki kekurangan dikarenakan metode algoritma Caesar Cipher sangatlah mudah untuk diretas dengan cara mem-*bruteforce* algoritma tersebut dan metode kriptografi RSA yang kurang efektif jika digunakan karena memerlukan kunci yang sangat panjang.

Oleh karena latar belakang masalah tersebut penelitian ini dibuat dengan judul "Implementasi Algoritma AES, ElGamal, dan SHA-3 untuk Keamanan *File Digital*".

1.2 Perumusan Masalah

Bagaimana implementasi gabungan dari algoritma AES, ElGamal, dan fungsi *hash* SHA3 dapat digunakan untuk meningkatkan keamanan data.

1.3 Tujuan

Adapun tujuan dari penelitian ini adalah untuk menganalisa dan membangun suatu sistem kriptografi hibrida dengan mengkombinasikan algoritma kunci simetris yaitu AES, algoritma kunci asimetris ElGamal, dan fungsi *hash* SHA3.

2. TINJAUAN PUSTAKA

2.1 Kriptografi

Kriptografi (*cryptography*) berasal dari Bahasa Yunani yaitu *cryptos* berarti rahasia, dan *graphien* berarti tulisan. Jadi berdasarkan bahasa tersebut, kriptografi berarti *secret writing* (tulisan rahasia). Kriptografi sendiri bertujuan untuk memberikan layanan keamanan, layanan keamanan tersebut terdiri dari empat aspek sebagai berikut: Kerahasiaan (*confidentiality*), Integritas data (*data integrity*), Otentikasi (*authentication*), dan Nir-penyangkalan (*non-repudiation*) [8]. Berdasarkan tipe kunci yang digunakan, algoritma kriptografi dibagi menjadi dua yaitu algoritma Simetris dan algoritma Asimetris.

Algoritma simetrik menggunakan sebuah kunci yang sama untuk proses enkripsi dan dekripsi. Dalam Kriptografi simetris, kunci yang digunakan untuk enkripsi mirip dengan kunci yang digunakan dalam dekripsi.

Algoritma kriptografi asimetrik menggunakan dua kunci yang berbeda untuk enkripsi dan dekripsi, kunci yang digunakan untuk mengenkripsi pesan disebut *public key*, sedangkan kunci yang digunakan untuk mendekripsi sandi disebut *private key*.

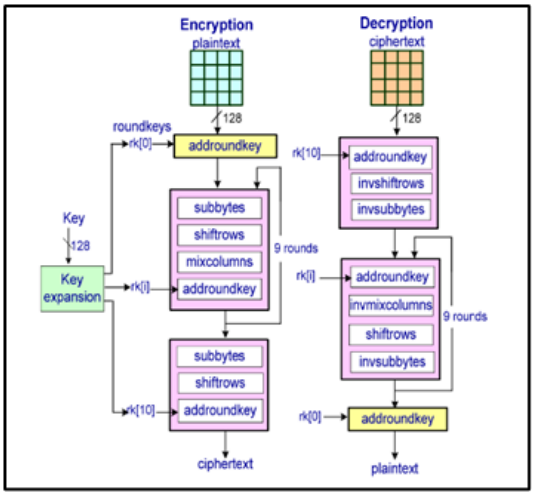
2.2 Hybrid Cryptosystem

Hybrid Cryptosystem adalah sebuah algoritma yang mengkombinasikan algoritma kunci publik (*public-key cryptosystem*) dengan algoritma simetris (*symmetric-key cryptosystem*) dan fungsi *hash* [10]. *Public-Key Cryptosystem* digunakan karena tidak mengharuskan pengirim dan penerima berbagi kunci (*password*) untuk berkomunikasi dengan aman. Namun karena *public-key cryptosystem* menggunakan perhitungan matematis yang sangat rumit untuk mengenkripsi sebuah pesan, maka hal ini akan sangat tidak efektif digunakan untuk mengenkripsi pesan yang sangat panjang karena membutuhkan waktu yang sangat lama. Oleh karena itu, hal ini dapat diatasi oleh sistem *hybrid* dengan menggabungkan algoritma kunci publik dan algoritma simetris. Metode enkripsi pada *hybrid cryptosystem* terbagi menjadi dua tahapan, yaitu [5] :

- *Key Encapsulation Scheme* (Menggunakan *Public-Key Cryptosystem*)
- *Data Encapsulation Scheme* (Menggunakan *Symmetric-Key Cryptosystem*)

2.3 Algoritma AES

Algoritma AES ini menggunkan *block* berukuran 128-*bit input* dan memiliki 3 ukuran *key* yaitu 128-*bit*, 192-*bit*, dan 256-*bit*. *Input* dan *output* dari algoritma AES terdiri dari urutan data yang berukuran sebesar 128-*bit*. Urutan data setiap satu kelompok 128-*bit input* disebut sebagai blok data atau *plaintext* yang kemudian akan dienkripsi menjadi *ciphertext*. Pengelompokkan jenis AES ini adalah berdasarkan panjang kunci yang digunakan. Angka-angka yang terdapat di belakang kata AES menunjukkan panjang kunci yang digunakan dalam satuan *bit*. Berdasarkan ukuran bloknya, AES bekerja pada matriks dengan ukuran 4x4 di mana tiap-tiap sel matriks terdiri atas 1 *byte* (8 bit). Selain itu, yang menjadi perbedaan tipe AES ini yaitu banyaknya round yang dipakai. AES-128 menggunakan 10 *round*, AES-192 menggunakan 12 *round*, sedangkan AES-256 menggunakan 14 *round*.



Gambar 1. Diagram Proses Enkripsi dan Dekripsi Algoritma AES

Gambar 1 merupakan diagram yang menjelaskan proses enkripsi dan dekripsi pada algoritma AES, yang beroperasi pada blok 128-*bit* dengan kunci (*key*) 128-*bit*. Proses-proses yang dilakukan terdiri dari 3 tahap, yaitu sebagai berikut :

Tahap 1 : *AddRoundKey* dimana dalam tahap ini dilakukan dengan melakukan operasi XOR antara *state* dengan *cipher key*. Tahap pertama *AddRoundKey* ini disebut juga sebagai *initial round.*

Tahap 2 : Putaran sebanyak (Nr – 1) kali. Proses yang dilakukan pada setiap 1 putaran adalah sebagai berikut. Pertama, *SubBytes* dimana terjadi pertukaran isi matriks yang ada dengan matriks substitusi yang disebut dengan S-Box. Kemudian tahap *Shift Rows*

dimana dilakukannya pergeseran per-baris pada setiap *block* yang ada dalam sebuah *state*. Baris pertama tidak dilakukan pergeseran, baris berikutnya dilakukan pergeseran sebanyak (*n*-1), dimana *n* merupakan posisi baris pada *blok* elemen tersebut. Ketiga adalah tahap *Mix Column* dimana pada tahap ini dilakukan perkalian pada tiap elemen dari *block cipher* dengan Mix-Column Matrix dan hasil perkalian (*dot product*) tersebut dimasukkan kedalam *block cipher* yang baru. Tahap terakhir yaitu *AddRoundKey* yang hampir sama dengan *AddRoundKey* pada tahap pertama, namun pada tahap ini dilakukan sebanyak jumlah *round* dan perhitungan dilakukan dengan meng-XORkan *state* sekarang dengan *round key*.

Tahap 3 : Pada tahap *Final round* ini akan dilakukan proses transformasi untuk putaran yang terakhir, proses yang dilakukan yaitu *SubBytes*, *Shift Rows* dan *AddRoundKey*.

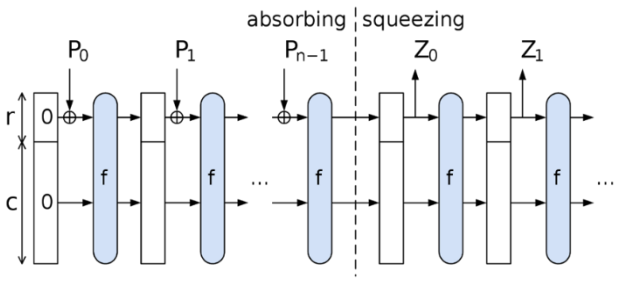
2.4 Algoritma Secure Hash Algorithm-3 (SHA3)

SHA dikembangkan oleh National Institute of Standards and Technology (NIST) dan dipublikasikan sebagai *Federal Information Processing Standards* (FIPS 180) pada tahun 1993.

Dibuatnya SHA-3 ini bukan untuk menggantikan SHA-2, tetapi karena kekhawatiran NIST terhadap keamanan fungsi *hash* MD5, SHA-0, SHA-1 yang telah berhasil ditembus. Karena celah keamanan itulah pada akhirnya NIST mencari algoritma fungsi hash alternatif yang jauh berbeda dengan algoritma sebelumnya dengan mengadakan kompetisi untuk mencari fungsi hash baru yang lebih baik. Pada bulan Agustus 2015, Fungsi *hash* Keccak terpilih oleh NIST sebagai algoritma fungsi *hash* baru yang paling aman dan efisien [11]. Algoritma Keccak merupakan fungsi hash yang dirancang oleh Guido Bertoni, Joan Daemen, Michaël Peeteres, dan Gilles Van Assche [3].

SHA-3 memiliki beragam ukuran *output* [7]. Jenis-jenis keluaran SHA-3 tersebut yaitu SHA-3 224, SHA-3 256, SHA-3 384, SHA-3 512, SHAKE128 dan SHAKE256.

SHA-3 menggunakan konstruksi spons (*sponge structure*) yang melakukan permutasi terhadap state dengan panjang yang tetap [2].



Gambar 2. Ilustrasi *Sponge Structure* pada SHA3

Gambar 2 menjelaskan fase-fase yang terdapat pada *sponge structure*. Fungsi *f* dianalogikan dalam sebuah spons terdapat 2 fase, yaitu fase "*absorbs*" yang merupakan *input* bit ke dalam *state*-nya, dan fase "*squeeze*" yang merupakan *output* bit dari *state*-nya.

2.5 Algoritma ElGamal

Algoritma El-Gamal termasuk dalam kelas algoritma kriptosistem asimetris dan didasarkan pada sistem enkripsi *Elliptic Curve* [12]. Hal itu juga sebanding dengan algoritma Diffie-Hellman dan banyak digunakan untuk enkripsi data serta tanda tangan digital. Keamanannya algoritma ini terletak pada penghitungan bidang hingga logaritma diskrit. Pada algoritma El-Gamal, output (*ciphertext*) yang dihasilkan pada tahap enkripsi dua kali lipat lebih panjang dibandingkan dengan plaintext-nya. Enkripsi menghasilkan bilangan N acak dari *ciphertext*.

El-Gamal banyak digunakan dalam protokol standard keamanan Internet seperti IPSEC, VPN, dan SSL untuk mengamankan transmisi data pada jaringan publik dan banyak digunakan pada *email* dan *web* [6].

(i) Algoritma untuk pembangkitan kunci

- Tentukan bilangan prima p.
- Tentukan bilangan acak g, dengan syarat (g < p).
- Tentukan bilangan acak x, dengan syarat (x < p-2).
- Kalkulasikan kunci publik y menggunakan perhitungan $y = g^x \pmod p$
- Bilangan (p, g, y) merupakan *public key* dan bilangan (x) adalah *private key*.

(ii) Algoritma untuk Enkripsi

Proses enkripsi menggunakan kunci publik (p, g, y) dan dilakukan dengan persamaan :

$$\mathbf{y^k m \bmod p} \quad (1)$$

Pada persamaan 1, *y* dipangkatkan dengan *k*, kemudian dikalikan dengan *m*, dimana *m* merupakan *message* yang akan dienkripsi. Hasil tersebut kemudian di-modulo dengan bilangan *p*.

(iii) Algoritma untuk Dekripsi

Proses dekripsi menggunakan kunci pribadi (p, x) dan dilakukan dengan rumus persamaan :

$$\mathbf{a^{p-1-x} \bmod p} \quad (2)$$

Pada persamaan 2, *a* dipangkatkan dengan (*p-1-x)*, dimana *a* merupakan *ciphertext* yang akan didekripsi. Hasil tersebut kemudian di-modulo dengan bilangan *p*.

3. METODE

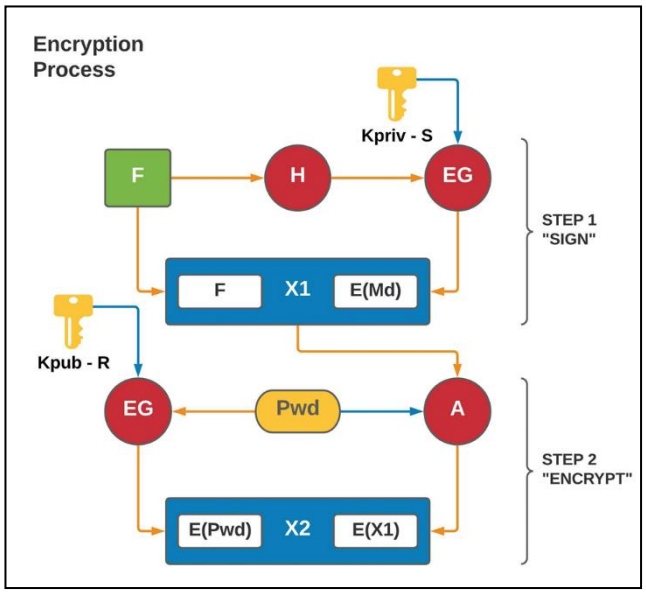
3.1 Desain Sistem

Penelitian ini menggunakan model skema *hybrid* yang menggabungkan algoritma AES, ElGamal, dan SHA3. Algoritma AES digunakan dalam proses enkripsi dan dekripsi data (*data encapsulation*), algoritma ElGamal digunakan dalam proses enkripsi dan dekripsi kunci (*key encapsulation*), sedangkan algoritma SHA3 digunakan untuk memverifikasi integritas data (data *integrity check*). Algoritma AES menggunakan AES 256-bit dengan panjang kunci 32 *bytes* dan menggunakan *round* sebanyak 14 putaran. Algoritma *hash* SHA3 menggunakan SHA3-512 yang mengeluarkan *output* berupa *message digest* sebesar 512-bit. Algoritma ElGamal memerlukan kunci yang berbeda untuk proses enkripsi dan dekripsinya. Bagian 3.2 dan Bab 3.3 menjelaskan rancangan proses untuk dekripsi dan enkripsi algoritma AES, ElGamal, dan SHA3. Adapun penjelasan parameter-parameter di tiap simbol yang terdapat pada skema yaitu sebagai berikut :

- F = *File*
- H = Fungsi *hash* SHA-3
- EG = Fungsi algoritma enkripsi ElGamal
- A = Fungsi algoritma enkripsi AES
- Pwd = *Password* untuk algoritma enkripsi AES
- Kpriv – S = Kunci pribadi milik pengirim (*sender*)
- Kpriv – R = Kunci pribadi milik penerima (*receiver*)
- Kpub – S = Kunci publik milik pengirim (*sender*)
- Kpub – R = Kunci publik milik penerima (*receiver*)
- X1 = *File Output* dari hasil proses "*Sign*"
- X2 = *File Output* dari hasil proses "*Encrypt*"
- E(…) = Sebuah *input* yang telah terenkripsi

3.2 Rancangan Proses Enkripsi

Rancangan proses enkripsi menggunakan kombinasi algoritma AES, ElGamal, dan SHA3 dapat dilihat pada Gambar 3.



Gambar 3. Rancangan Proses Enkripsi Algoritma AES, ElGamal, dan SHA3

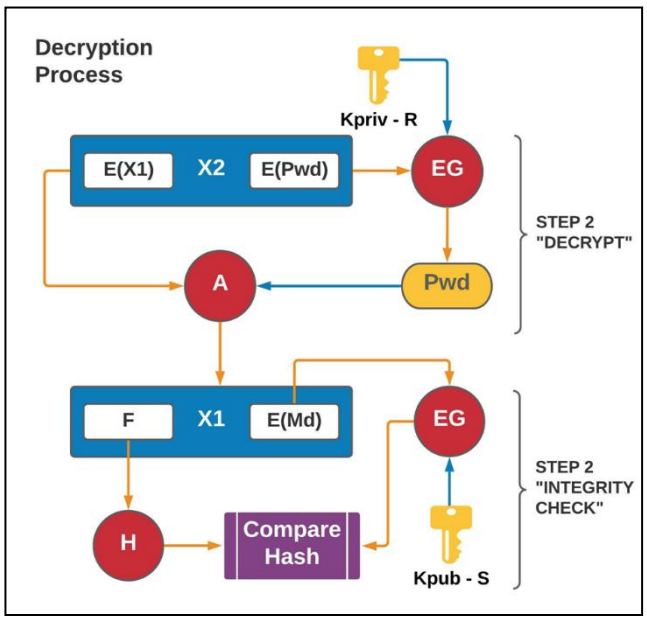
Pada Gambar 3 dijelaskan pula alur proses enkripsi pada program mulai dari file diinputkan hingga menghasilkan output berupa file yang telah terenkripsi. Proses enkripsi pada program terbagi menjadi dua tahap, yaitu tahap untuk pembuatan signature dan tahap peng-enkripsian file.

Pada tahap pembuatan signature, pertama file (F) akan di-hash menggunakan algoritma SHA3 (H) sehingga menghasilkan output berupa message digest sebesar 512-bit. Kemudian message digest tersebut dienkripsi menggunakan algoritma ElGamal (EG). Kunci yang digunakan untuk peng-enkripsian pada tahap signature ini menggunakan kunci pribadi dari pengirim (Kpriv – S). Hasil enkripsi dari algoritma ElGamal ini menghasilkan message digest yang terenkripsi (E(Md)). Selanjutnya message digest yang terenkripsi ini digabungkan dengan file input menjadi sebuah file baru (simbol "X1" pada skema).

Tahap berikutnya yaitu tahap peng-enkripsian file dimana (X1) yang merupakan output dari tahap pembuatan signature akan dilakukan enkripsi menggunakan algoritma simetris AES (A). Dalam proses meng-enkripsi file (X1) ini digunakan password (Pwd) sepanjang 256-bit yang diinputkan pengirim pada program untuk memperkuat keamanan. Hasil enkripsi dari algoritma AES ini menghasilkan (X1) yang terenkripsi, dinotasikan pada skema dengan simbol (E(X1)). Kemudian password input (Pwd) untuk algoritma AES tersebut juga dienkripsi lagi menggunakan algoritma ElGamal (EG). Berbeda dengan tahap signing, pada tahap ini kunci yang digunakan untuk algoritma ElGamal menggunakan kunci publik dari penerima file (Kpub - R). Hasil dari algoritma ElGamal ini menghasilkan password AES yang terenkripsi, dinotasikan pada skema dengan simbol (E(Pwd)). Kemudian kedua output tersebut (E(X1) dan E(Pwd)) digabungkan dengan fungsi byte padding sehingga menjadi sebuah file (X2) yang berekstensi ".mx_encrypted" yang menandakan bahwa file yang diinputkan telah berhasil dienkripsi dan siap untuk dikirimkan ke penerima.

3.3 Rancangan Proses Dekripsi

Rancangan proses dekripsi menggunakan kombinasi algoritma AES, ElGamal, dan SHA3 dapat dilihat pada Gambar 4.



Gambar 4. Rancangan Proses Dekripsi Algoritma AES, ElGamal, dan SHA3

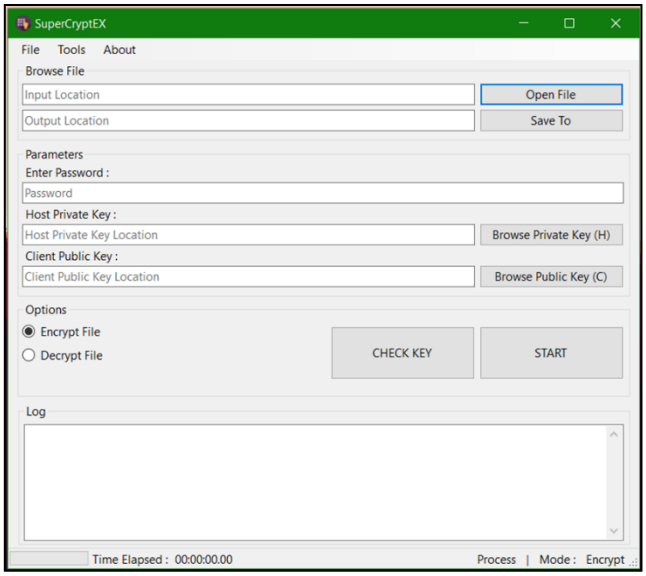
Pada Gambar 4 dijelaskan pula alur proses dekripsi pada program mulai dari file terenkripsi diinputkan hingga menghasilkan output berupa file asli yang telah didekripsi. Proses dekripsi pada program terbagi menjadi dua tahapan proses, tahap pertama merupakan proses dekripsi pada file dan tahap kedua merupakan proses untuk melakukan pengecekan integritas dari file yang telah didekripsi tersebut.

Tahap pertama dimulai dengan melakukan pemisahan byte (split byte) antara password AES yang terenkripsi (E(Pwd)) dan file yang terenkripsi (E(X1)). Password AES yang terenkripsi ini dilakukan proses dekripsi menggunakan algoritma ElGamal (EG). Kunci yang digunakan untuk proses dekripsi ini menggunakan kunci pribadi dari penerima (Kpriv - R). Setelah password AES berhasil didekripsi, selanjutnya password (Pwd) ini akan digunakan untuk mendekripsi file (E(X1)) dengan algoritma AES (A). Hasil dari proses dekripsi ini menghasilkan output berupa file dan sebuah message digest yang terenkripsi (E(Md)), dinotasikan pada skema dengan simbol "X1".

Tahap kedua merupakan tahap untuk melakukan pengecekan terhadap file integrity yang diterima. Tahap ini diawali dengan melakukan pemisahan byte (split byte) pada "X1" untuk memisahkan file (F) dengan message digest yang masih terenkripsi (E(Md)). Selanjutnya dilakukan proses dekripsi pada message digest yang terenkripsi ini menggunakan algoritma ElGamal (EG) sehingga dihasilkan output berupa sebuah message digest. Kunci yang digunakan pada proses tersebut menggunakan kunci publik dari pengirim file (Kpub – S). Proses terakhir diawali dengan melakukan hash pada file (F) dengan SHA-3 (H) dan membandingkannya dengan message digest yang didapat dari proses dekripsi ElGamal (EG). Jika message digest hasil dekripsi ElGamal dan message digest hasil dari hashing file sama, maka dapat dipastikan bahwa file yang diterima adalah file asli (authentic).

4. HASIL PENGUJIAN

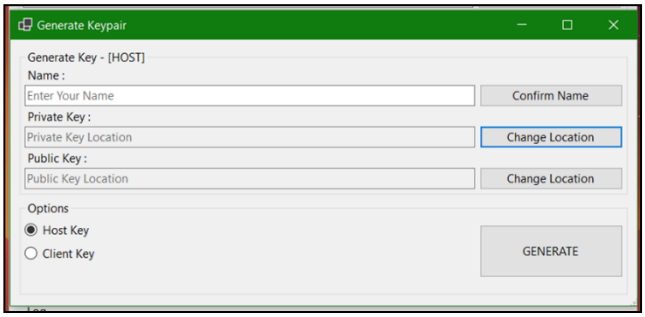
Program yang dibangun untuk mendukung penelitian ini menggunakan bahasa C#. Pengujian dilakukan dengan tujuan untuk mendapatkan data hasil, baik itu dari proses enkripsi maupun dekripsinya serta untuk menghitung kecepatan rata-rata program dalam proses enkripsi dan dekripsi. Adapun tampilan halaman utama program yang dibuat ditunjukkan pada Gambar 5.



Gambar 5. Tampilan *Main Form* pada Program

4.1 Pengujian Program

Sebelum dilakukan proses enkripsi dan dekripsi, program membutuhkan dua buah pasang kunci yang masing-masing merupakan sebuah kunci *host* dan sebuah kunci *client*. Pada program telah disediakan fitur untuk men-*generate* kunci yang diperlukan program.

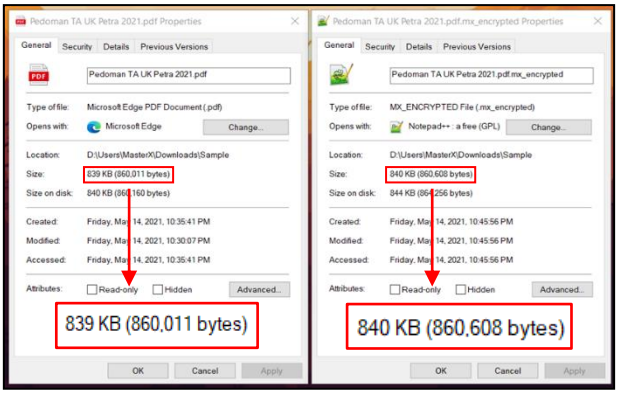


Gambar 6. Tampilan *Form Generate Key* pada Program

Gambar 6 merupakan tampilan form pada program untuk membangkitkan sepasang kunci (*generate key*) untuk algoritma ElGamal. Pada penelitian ini, digunakan kunci *host* dengan nama Bob selaku pengirim *file* dan kunci *client* dengan nama Alice selaku penerima *file*.

Pada pengujian tahap ini, dilakukan proses enkripsi dan dekripsi terhadap sebuah *file* dengan nama "Pedoman TA UK Petra 2021.pdf" yang berukuran 860,011 *bytes* (840 *kilobytes*).

Untuk proses enkripsinya digunakan password sepanjang 32 karakter dengan kalimat yaitu "saya-sangat-senang-bermain-hujan". Kunci pribadi pengirim (*host private key*) yang digunakan yaitu kunci dengan nama *file* "Bob_HostKey.mx_hpriv". Sedangkan kunci publik penerima (*client public key*) yang digunakan yaitu kunci dengan nama *file* "Alice_ClientKey.mx_cpub".

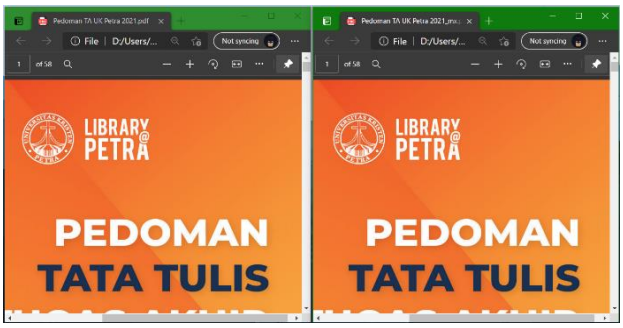


Gambar 7. Perbandingan Ukuran *File* Asli (kiri) dengan *File* Hasil Enkripsi (kanan)

Pada Gambar 7 dapat dilihat bahwa *file* hasil enkripsi dengan *file* aslinya terdapat perbedaan ukuran. dikarenakan penambahan beberapa atribut dan variabel untuk proses dekripsi pada program. *File* hasil enkripsi mengalami peningkatan sebesar 597 *bytes*, yaitu dari 860,011 *bytes* menjadi 860,608 *bytes*.

Untuk proses dekripsinya digunakan kunci publik pengirim (*host public key*) yaitu kunci dengan nama *file* "Bob_HostKey.mx_hpub". Sedangkan kunci pribadi penerima (*client private key*) yang digunakan yaitu kunci dengan nama *file* "Alice_ClientKey.mx_cpriv".

Setelah proses dekripsi *file* berhasil dilakukan, selanjutnya dilakukan pengamatan terhadap *file* hasil dekripsi ("Pedoman TA UK Petra 2021_mx.pdf") dengan *file* yang asli ("Pedoman TA UK Petra 2021.pdf"). Pengamatan dilakukan terhadap dua aspek, yaitu perbandingan ukuran *file* dan perbandingan isi *file*.



Gambar 8. Perbandingan *File* Asli (kiri) dengan *File* Hasil Dekripsi (kanan)

Pada Gambar 8 terlihat bahwa isi dari *file* hasil dekripsi sama/identik dengan *file* aslinya.

Pengujian pada program juga dilakukan terhadap sepuluh buah file dengan jenis (ekstensi) yang berbeda-beda tiap file-nya. Pasangan kunci yang dipakai pada pengujian tahap ini yaitu menggunakan kunci pengirim (*host key*) Bob dan kunci penerima (*client key*) Alice. Rata-rata ukuran file yang diuji kurang lebih adalah sekitar 5 megabytes. Kesepuluh *file* hasil dekripsi tersebut kemudian

dibandingkan dengan tiap-tiap *file* aslinya seperti pada pengujian tahap pertama. Sepuluh file yang diuji tersebut dijabarkan pada Tabel 1.

Tabel 1. Sepuluh Sampel Pengujian *File*

Nama File	Tipe File	Ukuran File (*byte*)
big_buck_bunny_720p_5mb.mkv	MKV	5,247,430
big_buck_bunny_720p_5mb.mp4	MP4	5,253,880
file_example_MP3_5MB.mp3	MP3	5,289,384
SampleDOCFile_5000kb.doc	DOC	5,275,136
SampleXLSFile.xls	XLS	5,275,648
kriptografi week 1.pptx	PPTX	5,249,987
Quiz Parampaa.exe	EXE	4,761,230
SampleJPGImage_5mb.jpg	JPG	5,266,467
sample-large-zip-file.zip	ZIP	5,216,156
SampleTextFile.txt	TXT	5,289,384

Dari hasil pengujian terhadap sepuluh file pada Tabel 1 tersebut, seluruh file dari hasil dekripsi dapat dibuka kembali dan isinya identik dengan file aslinya sehingga dapat disimpulkan bahwa program mampu menjalankan proses enkripsi dan dekripsi terhadap berbagai jenis file yang berbeda-beda.

4.2 Pengujian Kecepatan Rata-Rata Program

Pengujian kecepatan pada program dilakukan dengan tujuan untuk mengetahui rata-rata waktu yang diperlukan dalam melakukan proses enkripsi dan dekripsi. Pengujian kecepatan program dilakukan dengan menggunakan sebuah laptop yang memiliki spesifikasi sebagai berikut :

Processor : Intel(R) Core(TM) i7-7700HQ (8 CPUs) @ 2,80GHz
Memory : 16,384MB RAM
OS : Microsoft Windows 10 Home 64-bit

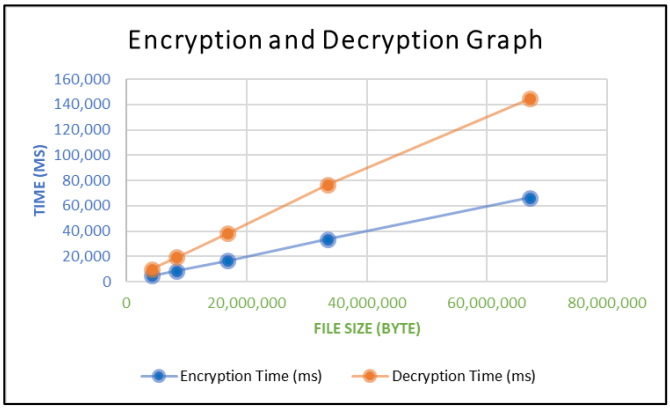
Pada pengujian tahap ini, dilakukan proses enkripsi dan dekripsi terhadap lima buah *file* teks yang masing-masing berukuran 4 MB, 8 MB, 16 MB, 32 MB, dan 64 MB. Pengujian pada tahap ini dilakukan dengan tujuan untuk mengetahui rata-rata kecepatan program dalam melakukan enkripsi dan dekripsi pada *file*.

Tabel 2. Perbandingan Waktu Enkripsi dan Dekripsi Terhadap Lima *File* dengan Ukuran yang Berbeda-Beda

File Name	File Size (*byte*)	Encryption Time (*ms*)	Decryption Time (*ms*)
Dummy_4MB.txt	4,194,304	4,671	10,152
Dummy_8MB.txt	8,388,608	8,613	19,250
Dummy_16MB.txt	16,777,216	16,711	38,169
Dummy_32MB.txt	33,554,432	33,641	76,629
Dummy_64MB.txt	67,108,864	66,462	144,496
TOTAL	**130,023,424**	**130,098**	**288,696**

Pada Tabel 2 dapat dilihat bahwa dari kelima *file* teks yang diuji, didapatkan total ukuran *file* yaitu sebesar 130,023,424 *bytes* dengan total waktu enkripsi selama 130,088 milidetik (sekitar 2 menit) dan total waktu dekripsi selama 288,696 milidetik (sekitar 4 menit). Rata-rata kecepatan enkripsi dan dekripsi dapat dihitung dengan cara membagi total ukuran *file* dengan total waktu yang diperlukan pada setiap prosesnya, sehingga dapat dituliskan sebagai berikut :

a. Rata-rata kecepatan enkripsi = 130,023,424 / 130,098 = **999.43 *bytes/ms***
b. Rata-rata kecepatan dekripsi = 130,023,424 / 288696 = **450.38 *bytes/ms***



Gambar 9. Grafik Perbandingan Kecepatan Enkripsi dan Dekripsi pada Program

Gambar 9 merupakan sebuah grafik yang menunjukkan pengaruh ukuran sebuah *file* terhadap kecepatan baik untuk proses enkripsi maupun dekripsinya.

Pada grafik tersebut dapat disimpulkan bahwa semakin besar ukuran dari sebuah *file*, maka perbedaan waktu enkripsi terhadap waktu dekripsinya juga akan semakin signifikan. Rata-rata kecepatan dekripsi pada program adalah 2x kecepatan enkripsinya.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil implementasi dan pengujian yang telah dilakukan, maka dapat diambil beberapa kesimpulan, yaitu :

- Program dapat melakukan proses enkripsi dan dekripsi pada *file* dengan baik, sehingga dapat disimpulkan bahwa algoritma simetris AES, algoritma kunci publik ElGamal, dan fungsi *hash* SHA3 dapat digunakan secara bersamaan untuk menciptakan sebuah sistem enkripsi yang aman.
- Proses enkripsi dengan sistem ini dapat dilakukan secara berulang kali (*Cipher Block Chaining / CBC*) dengan menggunakan *password* dan kunci yang berbeda sehingga dapat memberikan proteksi berlapis pada *file*.
- Program dapat melakukan enkripsi dan dekripsi terhadap jenis *file* yang berbeda-beda sehingga dapat disimpulkan bahwa program dapat digunakan untuk meng-enkripsi berbagai jenis *file*.
- *File* yang terenkripsi hanya dapat didekripsi dengan menggunakan kunci publik dari pengirim dan kunci pribadi dari penerima, sehingga program dapat memastikan bahwa pengirim dan penerima *file* saja yang berwenang atas *file* tersebut.

- Rata-rata kecepatan proses enkripsi pada program adalah **999.43 bytes/ms**, sedangkan rata-rata kecepatan proses dekripsi pada program adalah **450.38 bytes/ms**.

5.2 Saran

Sejalan dengan perkembangan jaman, sudah pasti teknologi juga akan terus berkembang, oleh karena itu untuk penelitian sistem lebih lanjut di masa mendatang, terdapat beberapa saran yang dianjurkan, diantaranya yaitu program dapat dikembangkan agar bisa dijalankan pada perangkat *smartphone*.

Kemudian dalam melakukan proses algoritma, program dapat dikembangkan agar bisa berjalan pada *multiple thread* untuk meningkatkan efisiensi program.

Dan yang terakhir, tampilan pada program dapat dibuat lebih menarik agar tampilan program tidak terkesan tua/klasik.

6. DAFTAR PUSTAKA

[1] Antoni. 2018. *Kejahatan Dunia Maya (Cyber Crime) Dalam Simak Online. Nurani: Jurnal Kajian Syari'ah dan Masyarakat, 17*(2), 261-274. doi:10.19109/nurani.v17i2.1192.

[2] Azdy, R. A. 2016. *Tanda tangan Digital Menggunakan Algoritme Keccak dan RSA. JNTETI, 5*(3).

[3] Bentivenga, C., Christie, F., & Kitson, M. (2010). *Keccak Final Paper*. Retrieved from https://www.cs.rit.edu/~ark/winter2012/482/team/u5/report.pdf.

[4] Gunawan, I. 2018. *Kombinasi Algoritma Caesar Cipher dan Algoritma RSA utuk Pengamanan File Dokumen dan Pesan Teks. Jurnal Nasional Informatika dan Teknologi Jaringan*, 124-129.

[5] Hofheinz, D., & Kiltz, E. 2019. *Secure Hybrid Encryption from Weakened Key Encapsulation*. pp. 553-571.

[6] Kakish, M. J. 2012. *Authenticated and secure el-gamal cryptosystem over Elliptic curves*. *International Journal of Recent Research and Applied Studies, 10*(2).

[7] Kurniawan, F., Kusyanti, A., & Nurwarsito, H. 2017. *Analisis dan Implementasi Algoritma SHA-1 dan SHA-3 pada Sistem Autentikasi Garuda Training Cost*. *Pengemb. Teknol. Inf. dan Ilmu Komput., 1*(9), 803-812.

[8] Menezes, A. J., Oorschot, P. C., & Vanstone, S. A. 2001. *Handbook of Applied Cryptography*. CRC Press.

[9] Paar, C., & Pelzl, J. 2010. *Understanding Cryptography*. doi:10.1007/978-3-642-04101-3.

[10] Rachmawati, D., Sharif, A., Jaysilen, & Budiman, M. A. 2018. *Hybrid Cryptosystem Using Tiny Encryption Algorithm and LUC Algorithm*. *IOP Conference Series: Materials Science and Engineering, 300*, 12-42. doi:10.1088/1757-899x/300/1/012042.

[11] Romine, C. H. 2015. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. Retrieved from http://csrc.nist.gov/publications/drafts/fips-202/fips_202_draft.pdf.

[12] Wu, Z., Su, D., & Ding, G. 2014. *ElGamal Algorithm for Encryption of Data Transmission. 2014 International Conference on Mechatronics and Control (ICMC)*.