

Aplikasi Enkripsi *File Text* Pada *Google Drive* Menggunakan Algoritma *Aes* dan *Rsa*

Text File Encryption Application on Google Drive Using Aes and Rsa Algorithms

Josua Purba¹, Ria Eka Sari²

^a Program Studi Informatika Universitas Potensi Utama, Jl. Kl. Yos Sudarso Km 6,5, Medan, 20241, Indonesia

^b Program Studi Informatika Universitas Potensi Utama, Jl. Kl. Yos Sudarso Km 6,5, Medan, 20241, Indonesia
josuapurba2001@gmail.com¹, ladiespure@gmail.com²

ABSTRAK

Penelitian ini berfokus pada pengembangan aplikasi enkripsi dan dekripsi file teks yang menggunakan kombinasi algoritma Advanced Encryption Standard (AES) dan Rivest-Shamir-Adleman (RSA), serta menyediakan fitur integrasi dengan Google Drive untuk keamanan data yang lebih optimal. Algoritma AES diterapkan sebagai metode enkripsi simetris untuk meningkatkan efisiensi dan kecepatan enkripsi, sementara RSA digunakan sebagai metode enkripsi asimetris yang menambah lapisan keamanan melalui kunci publik dan privat. Dengan menggunakan kombinasi dua algoritma ini, aplikasi ini dirancang untuk melindungi kerahasiaan file teks pengguna sebelum diunggah ke Google Drive. Aplikasi yang dikembangkan menyediakan antarmuka yang intuitif, termasuk fitur unggah otomatis yang memungkinkan pengguna untuk langsung mengunggah file hasil enkripsi ke Google Drive hanya dengan mengklik tombol yang disediakan. Integrasi ini bertujuan untuk meningkatkan pengalaman pengguna dalam mengamankan data, serta mencegah potensi akses tidak sah dari pihak lain yang mungkin memiliki akses ke tautan file di Google Drive tanpa izin pengguna. Dengan adanya sistem enkripsi ganda ini, penelitian ini diharapkan dapat memberikan kontribusi dalam meningkatkan perlindungan data di platform penyimpanan awan, serta mengurangi risiko kerentanan data yang diakses tanpa autentikasi tambahan. Penelitian ini juga memberikan gambaran solusi keamanan yang dapat diterapkan pada aplikasi penyimpanan cloud lainnya demi melindungi privasi pengguna.

Kata Kunci : Enkripsi, Dekripsi, AES, RSA, Cloud Storage, Keamanan Digital

ABSTRACT

This research focuses on the development of a text file encryption and decryption application that uses a combination of the Advanced Encryption Standard (AES) and Rivest-Shamir-Adleman (RSA) algorithms, and provides integration features with Google Drive for more optimal data security. The AES algorithm is applied as a symmetric encryption method to improve encryption efficiency and speed, while RSA is used as an asymmetric encryption method that adds a layer of security through public and private keys. By using a combination of these two algorithms, this application is designed to protect the privacy of users' text files before they are uploaded to Google Drive. The developed application provides an intuitive interface, including an auto-upload feature that allows users to directly upload encrypted files to Google Drive by simply clicking the button provided. This integration aims to improve the user experience in securing data, as well as preventing potential unauthorized access from other parties who may have access to linked files on Google Drive without user permission. With this dual encryption system, this research is expected to contribute to improving data protection on cloud storage platforms, as well as reducing the risk of data vulnerabilities accessed without additional authentication. This research also provides an overview of security solutions that can be applied to other cloud storage applications to protect user privacy.

Keywords : Encryption, Decryption, AES, RSA, Cloud Storage, Digital Security

Disubmit: 15 Agustus 2024

Info Artikel :
Direview: 19 September 2024

Diterima : 25 Oktober 2024

Copyright © 2024 – senadimu. All rights reserved.

1. PENDAHULUAN

Perkembangan teknologi informasi telah memperkenalkan berbagai kemudahan dalam penyimpanan dan berbagi data. Data yang merupakan sumber utama informasi saat ini dapat disimpan secara daring sehingga manusia dengan mudah dan kapan saja dapat mengaksesnya [1]. Khususnya melalui layanan cloud storage seperti Google Drive. Google Drive merupakan media penyimpanan online (Cloud Storage) milik Google yang meluncur pada 24 April 2012. Google Drive menjadi salah satu Cloud Storage yang menyediakan ruang penyimpanan hingga 15GB secara gratis, paling besar ketimbang layanan Cloud lainnya [2]. Cloud storage memungkinkan pengguna untuk menyimpan, mengakses, dan membagikan data secara mudah dari berbagai perangkat. Namun, dengan kemudahan akses ini, muncul juga tantangan serius terkait keamanan data, terutama pada data yang bersifat sensitif. Penyimpanan data di cloud memiliki potensi ancaman seperti akses tidak sah dan serangan siber yang dapat mengakibatkan kebocoran data pengguna [3]. Penyimpanan data di cloud menghadirkan tantangan besar terkait privasi dan integritas informasi, terutama mengingat adanya risiko akses tidak sah dan serangan siber.

Pengamanan data di cloud dapat ditingkatkan melalui penerapan kriptografi. Kriptografi ialah bidang yang mempelajari metode matematika untuk melindungi data dan informasi, seperti validasi, otentikasi, dan integritas [4]. Enkripsi adalah proses konversi antara pesan teks asli (Plaintext) menjadi pesan terenkripsi (Ciphertext) melalui sistem Kriptografi. Kriptografi melindungi data dari akses yang tidak sah di dunia modern, memastikan bahwa data tetap rahasia, dapat dipercaya, dan asli [5]. Enkripsi data memungkinkan informasi sensitif disimpan secara aman dan hanya dapat diakses oleh pihak yang memiliki izin. Skripsi ini membahas pengembangan sistem enkripsi pada Google Drive dengan mengkombinasikan dua algoritma utama, yaitu Advanced Encryption Standard (AES) sebagai metode enkripsi simetris dan Rivest-Shamir-Adleman (RSA) sebagai metode enkripsi asimetris. Advanced Encryption Standard (AES) adalah algoritma kriptografi enkripsi yang standart untuk digunakan system keamanan pada semua data [6]. Algoritma RSA (Rivest Shamir Adleman) merupakan salah satu metode dalam cabang ilmu kriptografi, dimana RSA adalah jenis kriptografi asimetris yang menggunakan 2 kunci, yaitu kunci publik dan private [7]. Kedua metode ini secara bersama-sama meningkatkan keamanan data, karena AES menjadi pilihan utama untuk enkripsi data pada aplikasi yang memerlukan kecepatan dan efisiensi [8], sementara RSA menyediakan pengelolaan kunci enkripsi yang lebih aman melalui sistem kunci publik dan privat di mana kunci untuk melakukan enkripsi berbeda dengan kunci untuk melakukan dekripsi. Kunci untuk melakukan enkripsi disebut sebagai kunci publik, sedangkan kunci untuk melakukan dekripsi disebut sebagai kunci private [9]. Kombinasi AES dan RSA pada aplikasi penyimpanan data cloud dapat membantu meningkatkan keamanan data pengguna dari akses pihak yang tidak berwenang [10].

Beberapa penelitian sebelumnya telah mengeksplorasi penerapan enkripsi kombinasi AES dan RSA pada berbagai platform. Sebagai contoh, penelitian oleh Steven Richardo Siburian dan rekannya (2023) mengembangkan sistem kriptografi hybrid AES dan RSA untuk pengamanan pesan SMS. Namun, penelitian ini belum mencakup pengamanan data pada cloud storage [11]. Penelitian lain oleh Novanto Prio Utomo dan rekan (2023) juga menggunakan metode RSA untuk melindungi data dalam sistem email berbasis web, yang menunjukkan pentingnya RSA dalam menjaga integritas dan privasi data pada berbagai aplikasi daring [12]. Selain itu, Delisman Hulu dkk. (2020) mengimplementasikan algoritma AES untuk meningkatkan keamanan data medis, membuktikan bahwa AES sangat bermanfaat dalam enkripsi data sensitif lainnya, seperti hasil radiologi pasien [13]. Penelitian Fitri Nuraeni dan rekannya juga melakukan pengembangan tanda tangan digital dengan menggunakan kombinasi algoritma RSA dan AES dengan tujuan mendapatkan nilai kualitas enkripsi yang bagus dilihat dari segi waktu proses enkripsi-dekripsi, nilai entropi, nilai korelasi, dan nilai avalanche effect [14]. Penelitian Ahmad Pudoli dan rekannya juga melakukan pembuatan layanan messenger berbasis android dengan menggunakan fitur firebase, untuk memberikan solusi keamanan pesan pada messenger, jadi pesan yang dikirim pada layanan

tersebut dilakukan pengamanan, sehingga jika dilakukan sniffing oleh seseorang maka pesan tidak dapat dibaca. Teknik yang digunakan untuk melakukan pengamanan pesan Hybrid Cryptosystem dengan mengkombinasikan algoritma RSA untuk metode asimetris dan AES untuk metode simetris [15].

Dalam penelitian ini, sistem enkripsi dirancang khusus untuk Google Drive, dengan integrasi yang memungkinkan pengguna mengunggah file yang terenkripsi langsung ke platform tersebut, sehingga risiko akses tidak sah dapat ditekan. Dengan memperhatikan kebutuhan akan keamanan data dan minimnya aplikasi serupa, penelitian ini bertujuan untuk menghadirkan solusi enkripsi yang mudah diakses dan diintegrasikan dengan Google Drive. Diharapkan, hasil dari pengembangan ini dapat memberikan manfaat baik bagi pengguna individu maupun organisasi dalam menjaga keamanan data penting yang disimpan secara daring.

2. METODE

Di dalam penelitian ini penulis menggunakan 2 (dua) metode studi pengumpulan data yaitu:

1. Studi Pengumpulan Data

Pengumpulan data merupakan langkah awal yang penting untuk mendukung pengembangan sistem enkripsi ini. Data dikumpulkan melalui beberapa metode berikut:

a. Sampel

Peneliti mengumpulkan beberapa sampel data text sebagai data data yang akan di enkripsi untuk penelitian ini.

b. Analisis Sistem Google Drive

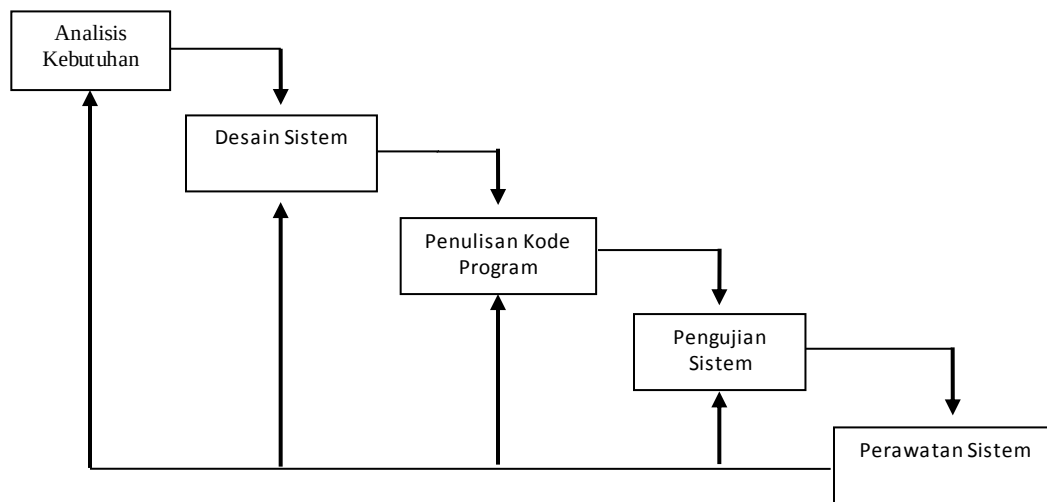
Penelitian ini juga melibatkan analisis terhadap sistem Google Drive, termasuk keamanan dan kebijakan akses yang diberlakukan oleh platform tersebut. Pemahaman ini penting untuk memastikan aplikasi yang dikembangkan dapat terintegrasikan dengan Google Drive dan mampu melindungi data yang diunggah ke platform tersebut.

2. Studi Kepustakaan (*Library Research*)

Penulis melakukan studi pustaka untuk memperoleh pemahaman mendalam tentang keamanan data, algoritma enkripsi, dan teknologi cloud storage. Sumber bacaan mencakup buku panduan kriptografi modern, artikel ilmiah, dan jurnal terkait algoritma simetris dan asimetris atau jurnal yang membahas tentang konsep yang berhubungan dengan judul penelitian.

Dalam pelaksanaan penelitian ini aktivitas yang dilakukan didalamnya yaitu mengadakan studi terhadap aplikasi sistem enkripsi file text pada Google drive dengan menerapkan metode simetris dengan algoritma AES dan metode asimetris dengan algoritma RSA yang akan digunakan.

Metodologi penelitian merupakan tata cara dan langkah – langkah yang diperlukan untuk mencapai tujuan perancangan yang dilakukan. Langkah – langkahnya adalah :

Gambar 1. Perancangan *Waterfall*

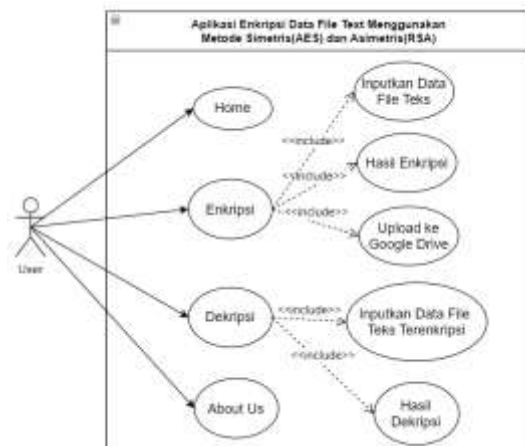
3. HASIL DAN PEMBAHASAN

3.1. Desain Sistem

Diagram alur dan use case dibuat untuk mendokumentasikan proses utama dalam aplikasi, termasuk enkripsi, dekripsi, dan integrasi Google Drive. Diagram tersebut membantu dalam memetakan alur kerja sistem agar lebih mudah diimplementasikan dan diuji pada tahap selanjutnya.

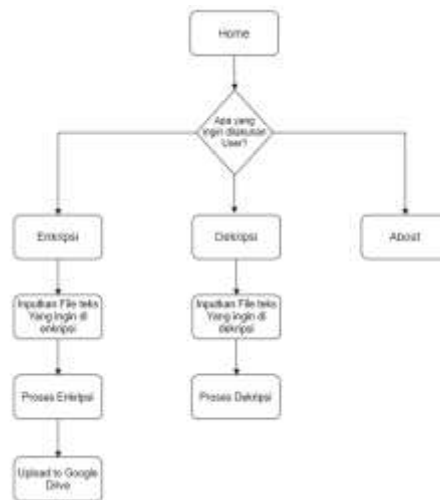
a. Use Case Diagram

Dalam penyusunan suatu program diperlukan suatu model data yang berbentuk diagram yang dapat menjelaskan suatu alur proses sistem yang akan dibangun. Use case diagram membantu dalam memahami keseluruhan sistem dengan mengidentifikasi tujuan utama yang ingin dicapai oleh pengguna dan cara interaksi dengan sistem tersebut. Pada gambar use case diagram dibawah menggambarkan bagaimana sistem enkripsi dan dekripsi data file text dibangun. Dapat dilihat pada Gambar 2.

Gambar 2. *Use Case Diagram* Enkripsi dan Dekripsi Data File Text

b. Flowchart Diagram

Flowchart Diagram adalah alat bantu visual yang digunakan untuk menggambarkan alur logika suatu sistem atau proses secara sistematis. Setiap langkah dalam proses direpresentasikan dengan simbol-simbol tertentu, dan hubungan antar langkah dihubungkan dengan garis atau panah. Pada *flowchart* diagram dibawah menggambarkan bagaimana alur logika sistem enkripsi dan dekripsi data *file text* yang di ingin dibangun. Dapat dilihat pada Gambar 3.



Gambar 3. *Flowchart* Diagram

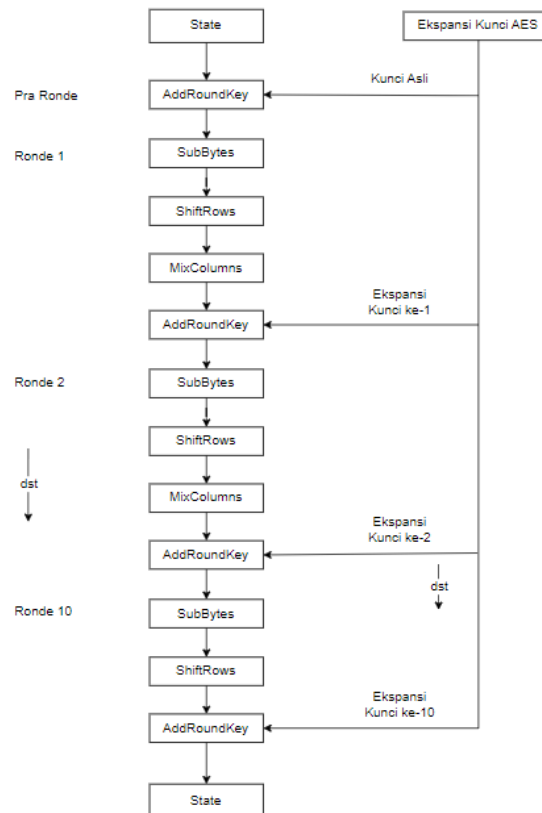
3.2. Pembahasan Metode

Penelitian ini menggunakan kombinasi algoritma Advanced Encryption Standard (AES) dan Rivest-Shamir-Adleman (RSA) untuk menjamin keamanan data di Google Drive. AES, sebagai enkripsi simetris, memberikan kecepatan dalam mengenkripsi data, sementara RSA, sebagai enkripsi asimetris, berperan dalam mengamankan kunci enkripsi AES. Kombinasi ini dirancang untuk mengoptimalkan efisiensi dan keamanan, memungkinkan data tetap terlindungi dari akses tidak sah.

1) Metode AES

Jenis kriptografi yang pertama yakni simetris key. Pada jenis kriptografi simetris key, pengirim dan penerima informasi hanya menggunakan satu kunci untuk melakukan enkripsi dan dekripsi informasi. AES (Advanced Encryption System) merupakan standar algoritma yang digunakan pada metode ini. Algoritma AES adalah blok ciphertext simetris yang dapat mengenkripsi (encipher) dan dekripsi (decipher) informasi.

Berikut ini merupakan flowchart dari metode RSA sebagai berikut :



Gambar 4. Flowchart Algoritma AES
(Sumber : Satya Wacana Christian University)

Contoh kita akan melakukan enkripsi menggunakan algoritma (AES)

Plaintext : *hello world 1234*

Kunci : *AES encryption!!*

a. Langkah pertama *ekspansi kunci*

AES-128 membutuhkan 11 *round key* untuk proses enkripsi. Mulai dengan *Key Round 0*, yaitu kunci asli kita. Kunci lainnya dihasilkan dari *RotWord*, *SubWord*, dan *Rcon*.

Untuk contoh ini kita hanya fokus *key round 0* atau kunci asli kita

Key Round 0 : 41 45 53 20 65 6e 63 72 79 70 74 69 6f 6e 21 21.

b. Langkah kedua kita melakukan *initial round (AddRoundKey)*

Kita melakukan operasi *XOR* antara plaintext dengan kunci round pertama (*Key Round 0*). *XOR* dilakukan *byte per byte*.

Plaintext (Hex): 68 65 6c 6c 6f 20 77 6f 72 6c 64 20 31 32 33 34

Key Round 0 (Hex): 41 45 53 20 65 6e 63 72 79 70 74 69 6f 6e 21 21

Kita melakukan *XOR* setiap *byte* dari plaintext dengan *key round* :

Maka hasil *XOR* adalah :

29 20 3f 4c 0a 4e 14 1d 0b 1c 10 49 5e 5c 12 15

c. Langkah ketiga *SubBytes*

Langkah selanjutnya melakukan *SubBytes*. Setiap *byte* dari hasil *XOR* akan diganti menggunakan tabel *S-Box*. *S-Box* adalah tabel substitusi tetap yang digunakan untuk menukar setiap *byte* dengan *byte* lain berdasarkan nilai tertentu.

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

Gambar 5. Tabel AES S-Box
(Sumber : Wikipedia)

Setelah *byte* dari hasil *XOR* sudah diganti menggunakan tabel *S-Box* maka diperoleh hasil *SubByte*.

Hasil setelah *SubByte* adalah :

a5 b7 75 29 67 2f fa a4 2b 9c ca 3b 58 4a c9 59

d. Langkah ke empat *ShiftRows*

Setelah *SubBytes*, kita melakukan operasi *ShiftRows*. Pada *ShiftRows*, setiap baris dalam blok matriks 4x4 akan digeser ke kiri dengan jumlah tertentu:

- Baris pertama tidak di-*shift*.
- Baris kedua digeser 1 *byte* ke kiri.
- Baris ketiga digeser 2 *byte* ke kiri.
- Baris keempat digeser 3 *byte* ke kiri.

Kita susun hasil dari *SubBytes* ke dalam matriks 4x4:

a5	b7	75	29
67	2f	fa	a4
2b	9c	ca	3b
58	4a	c9	59

a5	b7	75	29
----	----	----	----

2f	fa	a4	67
ca	3b	2b	9c
59	58	4a	c9

Setelah di *ShiftRows*,

maka hasilnya adalah:

| a5 | b7 | 75 | 29 | | 2f | fa | a4 | 67 | | ca | 3b | 2b | 9c | | 59 | 58 | 4a | c9 |

e. Langkah kelima *MixColumns*

Selanjutnya melakukan operasi *mixcolumns*, *mixcolumns* adalah langkah transformasi matriks yang dilakukan pada setiap kolom dari blok 4x4, di mana kita mengalikan setiap kolom dengan matriks konstan.

Matriks konstan yang digunakan di AES adalah :

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$$

Proses *mixcolumns* dengan kolom pertama dari *shiftrows*

Kolom pertama dari hasil *shiftrows* = [a5, 2f, ca, 59]

Kita akan kali kan kolom hasil *shiftrows* dengan matriks konstan

Langkah 1: Baris Pertama (02, 03, 01, 01)

Kita hitung baris pertama :

$$02 \times a5 \oplus 03 \times 2f \oplus 01 \times ca \oplus 01 \times 59$$

1. $02 \times a5$

- a5 dalam heksadesimal = 10100101 dalam biner.
- Perkalian dengan 02 berarti shift left satu bit: 10100101 $\rightarrow$ 01001010 = 4a (heksadesimal).

Jadi, $02 \times a5 = 4a$.

2. $03 \times 2f$

- 2f dalam heksadesimal = 00101111 dalam biner.
- Perkalian dengan 03 berarti :
 - $02 \times 2f$ (shift left): 01011110 = 5e.
 - XOR dengan 2f (karena $03 = 02 + 01$) : 5e XOR 2f = 71.

Jadi, $03 \times 2f = 71$.

3. $01 \times ca$

- $01 \times ca$ berarti hasilnya adalah nilai itu sendiri, yaitu ca .

4. 01×59

- 01×59 berarti hasilnya adalah nilai itu sendiri, yaitu 59.

XOR Semua Hasil:

Sekarang kita gabungkan semua hasil dengan operasi *XOR* : $4a \oplus 71 \oplus ca \oplus 59$

Lalu lakukan operasi *XOR* satu per satu dalam bentuk biner:

- $4a = 01001010$
- $71 = 01110001$
- $ca = 11001010$
- $59 = 01011001$
- 1. $4a \text{ XOR } 71 : 01001010 \text{ XOR } 01110001 = 00111011 \text{ (3b)}$
- 2. $3b \text{ XOR } ca : 00111011 \text{ XOR } 11001010 = 11110001 \text{ (f1)}$
- 3. $f1 \text{ XOR } 59 : 11110001 \text{ XOR } 01011001 = 10101000 \text{ (a8)}$

Jadi, hasil untuk baris pertama adalah a8.

Begitu juga cara untuk langkah baris ke 2, 3 dan ke 4.

Setelah melakukan operasi untuk setiap baris di kolom pertama, maka hasil akhirnya adalah : [a8, fc, f5, b8]

Hasil akhir *mixcolumns* :

Setelah melakukan *MixColumns* untuk keempat kolom, hasil akhirnya adalah sebagai berikut :

| a8 | 25 | 6d | 6a | | fc | 89 | a2 | 7e | | f5 | 4b | fc | d0 | | b8 | 8b | c9 | 70 |

f. Langkah ke enam *AddRoundKey*

Pada tahap *AddRoundKey*, blok hasil *MixColumns* akan di-*XOR* dengan Round Key yang dihasilkan dari proses Key Expansion.

Sebelum melanjutkan kita harus mendapatkan Round Key 1 dari *Key Round 0*, Untuk mendapatkan Round Key 1 dari *Key Round 0*, kita perlu melakukan Key Expansion dalam algoritma *AES*. Proses Key Expansion ini menggunakan tiga operasi utama: *RotWord*, *SubWord*, dan *Rcon*. Mari kita lakukan langkah-langkah tersebut secara terperinci untuk menghasilkan Round Key 1 dari *Key Round 0*.

Key Round 0 dalam heksadesimal = 41 45 53 20 65 6e 63 72 79 70 74 69 6f 6e 21 21

Pisahkan *Key Round 0* menjadi 4 kata (masing-masing 4 *byte*). Setiap kata (*word*) terdiri dari 32 bit atau 4 *byte*:

$w[0] = 41 \ 45 \ 53 \ 20$

$w[1] = 65 \ 6e \ 63 \ 72$

$w[2] = 79 \ 70 \ 74 \ 69$

$w[3] = 6f \ 6e \ 21 \ 21$

Untuk menghasilkan $w[4]$ hingga $w[7]$ (yang membentuk *Round Key 1*), kita akan menggunakan operasi *RotWord*, *SubWord*, dan *XOR* dengan *Rcon*.

1. Langkah 1: Menghitung $w[4]$

a. *RotWord*($w[3]$)

RotWord menggeser satu *byte* dari kata terakhir ke kiri. Jadi :

$$w[3] = 6f\ 6e\ 21\ 21 \rightarrow \text{RotWord}(w[3]) = 6e\ 21\ 21\ 6f$$

b. *SubWord*(*RotWord*(*w*[3]))

Gunakan *S-Box* (tabel *substitusi*) untuk mengganti setiap *byte* dari hasil *RotWord*. Dengan menggunakan *S-Box*, kita dapatkan:

$$\text{SubWord}(6e\ 21\ 21\ 6f) = 9f\ c4\ c4\ bb$$

c. *XOR* dengan *Rcon*[1]

Rcon[1] untuk ronde pertama adalah 01 00 00 00. Kita lakukan *XOR* antara hasil *SubWord* dan *Rcon* :

$$9f\ c4\ c4\ bb\ XOR\ 01\ 00\ 00\ 00 = 9e\ c4\ c4\ bb$$

d. *XOR* dengan *w*[0] untuk mendapatkan *w*[4]

Sekarang, kita hitung *w*[4] dengan *XOR* antara hasil sebelumnya (9e c4 c4 bb) dengan *w*[0] (41 45 53 20) :

$$w[4] = 41\ 45\ 53\ 20\ XOR\ 9e\ c4\ c4\ bb = df\ 81\ 97\ 9b$$

2. Langkah 2: Menghitung *w*[5]

Untuk mendapatkan *w*[5], kita lakukan *XOR* antara *w*[4] dan *w*[1] :

$$w[5] = w[4]\ XOR\ w[1] = df\ 81\ 97\ 9b\ XOR\ 65\ 6e\ 63\ 72 = ba\ ef\ f4\ e9$$

Begitu juga untuk *w*[6] dan *w*[7]

Setelah proses selesai maka kita memperoleh *Round Key* 1 adalah gabungan dari *w*[4], *w*[5], *w*[6], dan *w*[7] :

Round Key 1:

| df | 81 | 97 | 9b | | ba | ef | f4 | e9 | | c3 | 9f | 80 | 80 | | ac | f1 | a1 | a1 |

Kita lanjutkan ke tahap *AddRoundKey*

Pada tahap ini, kita akan melakukan operasi *XOR* antara hasil *MixColumns* dari ronde sebelumnya dengan *Round Key* 1.

Blok Hasil *MixColumns*:

a8	25	6d	6a
fc	89	a2	7e
f5	4b	fc	d0
b8	8b	c9	70

Round Key 1 :

df	81	97	9b
ba	ef	f4	e9
c3	9f	80	80
Ac	f1	a1	a1

Sekarang kita akan melakukan *XOR byte per byte* antara hasil *MixColumns* dan *Round Key 1*.

Proses *XOR Byte per Byte* :

1. Baris 1:
 - a8 *XOR* df = 77
 - 25 *XOR* 81 = a4
 - 6d *XOR* 97 = fa
 - 6a *XOR* 9b = f1

Hasil Baris 1: 77 a4 fa f1

Begitu juga untuk baris ke 2 sampai baris ke 4

Setelah proses selesai maka hasil akhirnya adalah sebagai berikut:

| 77 | a4 | fa | f1 | | 46 | 66 | 56 | 97 | | 36 | d4 | 7c | 50 | | 14 | 7a | 68 | d1 |

g. Ronde Berikutnya

Langkah-langkah ini diulang untuk ronde-ronde berikutnya (hingga 10 ronde untuk AES-128), dan pada ronde terakhir *MixColumns* tidak digunakan.

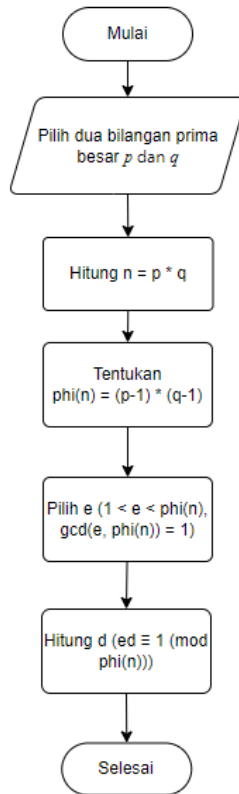
h. Ciphertext Akhir

Setelah menyelesaikan semua ronde, hasil akhirnya adalah ciphertext, Ini adalah blok terakhir yang dihasilkan setelah ronde terakhir atau ciphertextnya

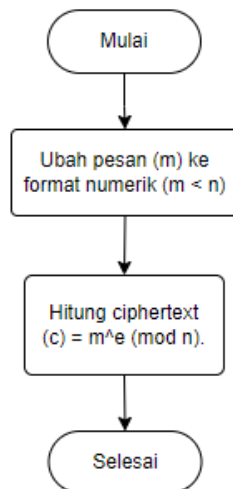
| 6b | f0 | 32 | d0 | | f7 | 16 | d9 | 69 | | 4d | 64 | e7 | 6d | | 20 | f6 | 6d | f8 |

2) Metode RSA

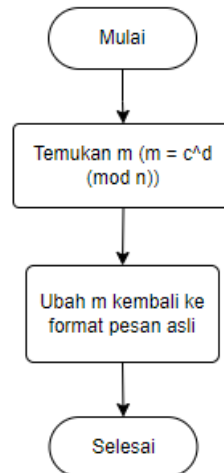
Asimetris atau public key cryptography adalah jenis kriptografi yang memanfaatkan dua kunci yang saling berkaitan sebagai public key dan private key. Algoritma RSA adalah algoritma untuk enkripsi dan dekripsi pesan. RSA menggunakan dua kunci, yaitu kunci public untuk enkripsi dan kunci private untuk dekripsi.



Gambar 6. Flowchart Generate Kunci Public dan Private
(Sumber : Alessius)



Gambar 7. Flowchart Enkripsi Algoritma RSA
(Sumber : Alessius)



Gambar 8. Flowchart Dekripsi Algoritma RSA
(Sumber : Alessius)

Contoh kita akan melakukan enkripsi dan dekripsi menggunakan algoritma (RSA) pertama kita harus melakukan ekspansi kunci atau generate key public dan private.

Tahapan Ekspansi Kunci :

1. Memilih dua buah bilangan prima berukuran besar p dan q . Kedua bilangan ini tidak boleh sama. Untuk memperoleh tingkat keamanan yang tinggi pilih p dan q yang berukuran hingga misalnya 1024 bit.
2. Menghitung n
Hitung n dengan mengalihkan p dan q
 $n = p \times q$
3. Langkah ke tiga menghitung m
Menghitung fungsi euler atau m dengan rumus :
 $\text{Phin}(n) = (p - 1) \times (q - 1)$
4. Memilih e yang relatif prima terhadap $\text{phin}(n)$. Untuk memilih e dalam algoritma RSA, ada beberapa syarat yang harus dipenuhi :
 1. e harus lebih besar dari 1 dan lebih kecil dari m yaitu $1 < e < \text{phin}(n)$.
 2. e harus relatif prima terhadap m , artinya $\text{gcd}(e, \text{phin}(n)) = 1$, di mana gcd adalah greatest common divisor atau faktor persekutuan terbesar (FPB). Ini berarti bahwa e dan m tidak memiliki faktor yang sama selain 1.

Tapi pada contoh perhitungan nanti kita menggunakan algoritma Euclidean karena metode ini efisien untuk menghitung gcd tanpa faktorisasi. Algoritma ini bisa diterapkan untuk menghitung gcd antara dua bilangan a dan b (dengan $a > b$) sebagai berikut :

1. Bagi a dengan b , dapatkan sisanya r .
2. Jika $r = 0$, maka $\text{gcd}(a, b) = b$
3. Jika $r \neq 0$, gantikan a dengan b , dan b dengan r , lalu ulangi langkah ini.

Sebagai contoh untuk $\text{gcd}(60, 24)$

Bagi 60 dengan 24 :

$60 / 24 = 2$ (sisa = 12)

Karna sisa r tidak sama dengan 0, ulangi algoritma dengan $a = 24$ dan $b = 12$. Bagi 24 dengan 12 :
 $24 / 12 = 2$ (sisa 0) Karena sisanya 0, maka $\text{gcd}(60, 24) = 12$.

5. Mencari d dengan rumus : $e \times d \bmod m = 1$, disini kita menebak nilai d yang jika dikalikan dengan e dan di mod dengan $\text{phin}(n)$ maka hasilnya adalah 1.

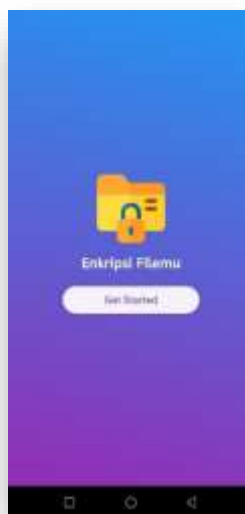
6. Kemudian di dapatlah sepasang kunci : kunci public = (e,n) dan kunci private = (d,n) .

3.3. Hasil Aplikasi

Berikut ini akan dijelaskan tentang tampilan hasil dari aplikasi sistem enkripsi *File Text* pada *Google Drive* menggunakan kombinasi algoritma *AES* dan *RSA* berbasis *Android* dapat dilihat sebagai berikut:

1. Tampilan *Splash Screen*

Tampilan *Splash Screen* merupakan tampilan yang pertama kali muncul ketika menjalankan aplikasi, Tampilan *Splash Screen* dapat dilihat pada gambar 9. sebagai berikut :



Gambar 9. Tampilan *Splash Screen*

2. Tampilan *Home*

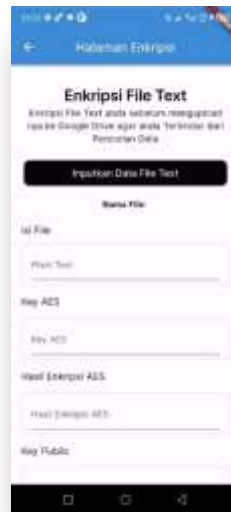
Tampilan *Home* merupakan tampilan yang akan muncul di halaman utama yang menampilkan 3 button atau menu, yaitu menu enkripsi, menu dekripsi, menu about ketika membuka aplikasi, Tampilan *Home* dapat dilihat pada gambar 10. sebagai berikut :



Gambar 10. Tampilan *Home*

3. Tampilan Menu Enkripsi

Tampilan Menu Enkripsi merupakan tampilan halaman enkripsi yang dimana di Menu Enkripsi ini kita dapat melakukan enkripsi data *File Text* kita yang ingin di enkripsi, Tampilan Menu Enkripsi dapat dilihat pada gambar 11. sebagai berikut :



Gambar 11. Tampilan Menu Enkripsi

4. Tampilan Menu Dekripsi

Tampilan Menu Dekripsi merupakan tampilan halaman dekripsi yang di mana di Menu Dekripsi ini kita dapat melakukan dekripsi data *File Text* kita yang ingin di dekripsi, Tampilan Menu Dekripsi dapat dilihat pada gambar 12. sebagai berikut :



Gambar 12. Tampilan Menu Dekripsi

5. Tampilan Menu *About*

Tampilan Menu *About* merupakan tampilan halaman *About* yang dimana di Menu *About* ini berisikan fungsi, tujuan, tentang aplikasi ini. Tampilan Menu *About* dapat dilihat pada gambar 13. sebagai berikut :



Gambar 13. Tampilan Menu About

4. KESIMPULAN

Berdasarkan rumusan masalah dan tujuan penelitian dan berdasarkan penelitian yang telah dilakukan selama membuat aplikasi ini, maka dapat ditarik beberapa kesimpulan sebagai berikut :

1. Algoritma enkripsi AES dan RSA dapat diimplementasikan secara efektif dalam aplikasi sistem enkripsi *File Text*. Penggunaan kombinasi kedua algoritma ini terbukti meningkatkan keamanan data yang disimpan di *Google Drive*. Algoritma AES digunakan untuk kecepatan proses enkripsi, sementara algoritma RSA memberikan keamanan tambahan melalui manajemen kunci yang lebih aman.
2. Aplikasi yang dikembangkan berhasil diintegrasikan dengan *Google Drive*. Proses integrasi ini memungkinkan pengguna untuk mengunggah *File Text* terenkripsi secara langsung ke *Google Drive*.
3. Dengan penerapan algoritma enkripsi AES dan RSA, keamanan data text yang disimpan di *Google Drive* dapat dipastikan aman. Sistem enkripsi yang digunakan mampu melindungi data dari akses tidak sah, sehingga memberikan solusi yang efisien dan mudah digunakan dalam menjaga kerahasiaan informasi data *File Text* di *Google Drive*.

Saran untuk penelitian lebih lanjut dalam penelitian “Aplikasi sistem enkripsi *File Text* pada *Google Drive* menggunakan kombinasi algoritma AES dan RSA”, maka dapat diberikan saran sebagai berikut :

1. Saat ini aplikasi hanya bisa mengenkripsi data *File Text* saja untuk penelitian selanjutnya diharapkan aplikasi dapat dikembangkan juga untuk dapat mengenkripsi lebih banyak lagi jenis data seperti data *file pdf*, *word*, *image*, dll.
2. Aplikasi ini dibuat berbasis *Android* untuk penelitian selanjutnya juga diharapkan aplikasi dapat dibuat berbasis website dan dekstop sehingga memudahkan pengguna dapat mengakses aplikasi darimana saja.
3. Aplikasi ini dibuat untuk mengenkripsi *File Text* yang di khususkan pada *Google Drive* untuk penelitian selanjutnya diharapkan aplikasi juga dapat dibuat untuk mengenkripsi data pada jenis *cloud storage* lainnya seperti *One Drive*, *Dropbox*, *iCloud*, *Amozon S3*, *NextCloud*, dll.

UCAPAN TERIMA KASIH

Dengan penuh rasa syukur, saya ingin mengucapkan terima kasih yang sebesar-besarnya kepada Ibu Ria Eka Sari, M.Kom atas bimbingan, dukungan, serta arahan yang sangat berharga selama proses penelitian dan pengembangan aplikasi enkripsi *File Text* pada *Google Drive* menggunakan algoritma AES dan RSA ini. Selalu memberikan masukan yang konstruktif, semangat, dan panduan yang tak ternilai, sehingga penelitian ini dapat berjalan dengan lancar.

Semoga segala ilmu dan pengalaman yang diberikan menjadi bekal yang berguna bagi saya di masa mendatang. Sekali lagi, saya ucapkan terima kasih atas segala dukungan dan bimbingan yang tak ternilai.

REFERENSI

- [1] Z. Masyhur, A. Rizaldy, and P. Kartini, "Studi Literatur Keamanan dan Privasi Data Sistem Cloud Computing Pada Platform Google Drive," *J. Software, Hardw. Inf. Technol.*, vol. 1, no. 2, pp. 31–38, 2021, doi: 10.24252/shift.v1i2.15.
- [2] R. Reza, F. Fitriani, L. Lifanda, L. W. Mardan, and L. Hasimu, "Kesadaran Pengguna Terhadap Keamanan Penyimpanan Data Dalam Google Drive Studi Kasus Pada Dinas Komunikasi Kabupaten Wakatobi," *J. Multidisipliner Kapalamada*, vol. 1, no. 01, pp. 39–46, 2022, doi: 10.62668/kapalamada.v1i01.56.
- [3] H. Rahmawati, S., & Susanto, "Tantangan keamanan pada penyimpanan data di cloud," *Keamanan Inf. Indones.*, vol. 5, no. 1, p. 20, 2020.
- [4] D. I. Mulyana, A. P. Heryani, and V. Khoirunnisa, "Implementasi Metode Rivest Shamir Adleman untuk Enkripsi dan Dekripsi Text," vol. 03, no. 01, pp. 32–39, 2022.
- [5] I. Dan, A. Kombinasi, and A. E. S. Dan, "IMPLEMENTATION AND ANALYSIS OF THE COMBINATION OF RSA, AES AND STEGANOGRAPHY IN THE ENCRYPTION OF URBAN," vol. 3, no. September, pp. 20–29, 2024.
- [6] C. Irawan and E. H. Rachmawanto, "Keamanan Data Menggunakan Gabungan Kriptografi AES dan RSA," *Proceeding SENDIU*, vol. 1, no. 2, pp. 978–979, 2021.
- [7] D. Alfiani Fauzan and A. Fathurrozi, "Perancangan Aplikasi Pengamanan Data Menggunakan Algoritma RSA (Rivest Shamir Adleman) dan AES (Advanced Encryption Standard) Berbasis Web," *J. Inf. Inf. Secur.*, vol. 4, no. 1, pp. 2722–4058, 2023, [Online]. Available: <http://ejurnal.ubharajaya.ac.id/index.php/jiforty>
- [8] M. Suryanegara, "Implementasi algoritma AES untuk enkripsi data pada aplikasi penyimpanan daring," *J. Sist. Inf. Indones.*, vol. 12, no. 2, p. 45, 2020.
- [9] A. Tampubolon, "Implementasi Kombinasi Algoritma RSA dan Algoritma DES Pada Aplikasi Pengaman Pesan Teks," *J. SAINTIKOM (Jurnal Sains Manaj. Inform. dan Komputer)*, vol. 20, no. 1, p. 38, 2021, doi: 10.53513/jis.v20i1.2598.
- [10] F. Prasetya, D., & Rahman, "Kombinasi algoritma AES dan RSA untuk keamanan data cloud storage," *J. Keamanan Siber Indones.*, vol. 10, no. 3, p. 33, 2021.
- [11] S. R. Siburian, R. Alek, S. Sinaga, and F. Yudistira, "Kriptosistem Hybrid Menggunakan Kombinasi AES Dan RSA Untuk Enkripsi Teks Pesan," *J. JOCOTIS - J. Sci. Inform. Robot.*, vol. 1, no. 1, pp. 22–31, 2023, [Online]. Available: <https://jurnal.itcc.web.id/index.php/jct/>
- [12] N. P. Utomo, N. Fahriani, and M. Amirul, "Implementasi Kriptografi Dengan Metode RSA Untuk

- Keamanan Data Pada Email Berbasis PHP,” ... *Semin. Nas. Teknol. ...*, vol. 2, no. 1, pp. 97–105, 2023, [Online]. Available: <https://journal.unilak.ac.id/index.php/Semaster/article/view/18461>
- [13] D. Hulu, B. Nadeak, and S. Aripin, “Implementasi Algoritma AES (Advanced Encryption Standard) Untuk Keamanan File Hasil Radiologi di RSUD Imelda Medan,” *KOMIK (Konferensi ...)*, vol. 4, pp. 78–86, 2020, doi: 10.30865/komik.v4i1.2590.
- [14] F. Nuraeni, Y. H. Agustin, D. Kurniadi, and I. D. Ariyanti, “Implementasi Skema QR-Code dan Digital Signature menggunakan Kombinasi Algoritma RSA dan AES untuk Pengamanan Data Sertifikat Elektronik,” pp. 43–52, 2020.
- [15] A. Pudoli and D. Kusumaningsih, “Penggunaan Hybrid Cryptosystem Untuk Enkripsi Dan Dekripsi Pesan Messenger Menggunakan Algoritma Rivest Shamir Adleman (Rsa) Dan Advanced Encryption Standard (Aes) Dengan Firebase Pada Android,” vol. 9, no. 3, pp. 125–131, 2017.