Volume 4 Issue 1 (2024) Pages 97-107 P-ISSN:2798-2696 E-ISSN:2798-0057
JOSTECH Journal of Science and Technology
Website: https://ejournal.uinib.ac.id/jurnal/index.php/jostech Email: jostech@uinib.ac.id

Implementasi Kriptografi dengan Algoritma RSA (Rivest-Shamir-Adleman) dalam Penyandian Pesan Teks dan Dokumen

Septi Rahmita Sari[1], Resmawan[2], Nisky Imansyah Yahya[3]✉, Lailany Yahya[4]
Department of Mathematics, Faculty of Mathematics and Natural Sciences,
Universitas Negeri Gorontalo[1,2,3,4]

Received 26 Februari 2024, Accepted 31 Maret 2024, Published 31 Maret 2024

Abstrak

Kriptografi dapat digunakan dalam mencegah penyalahgunaan data, salah satu tindakan pencegahan dengan cara penyandian pesan. Kriptografi merupakan studi tentang penyandian pesan atau cara perlindungan data. Dalam penyandian isi pesan terdapat algoritma yang popular digunakan sampai saat ini yaitu algoritma RSA (Rivest-Shamir-Adleman). Algoritma RSA ialah metode yang mempunyai dua kunci yang berbeda untuk setiap proses enkripsi dan dekripsinya tetapi saling berkaitan sehingga lebih terjaga dalam proses keamanan data. Kemudian dalam proses pencarian kunci algoritma RSA memanfaatkan kaidah bilangan prima. Semakin besar bilangan prima yang dipakai sebagi kunci, semakin susah ditemukan angka besar sebagi faktornya. Dalam hal ini akan dibahas proses enkripsi pesan teks dan dokumen menggunakan algoritma RSA, selanjutnya akan dijelaskan proses dekripsi dan proses pembangkitan kunci. *Plaintext* diubah menjadi *ciphertext* yakni menggunakan kode ASCII yang panjangnya 256 serta menggunakan PKCS dalam melakukan proses *enkrispi* pada algoritma RSA. Kemudian pengimplementasian algoritma RSA pada pesan teks dan dokumen menggunakan bahasa pemrograman Python. Pada penelitian selanjutnya, disarankan untuk menggunakan algoritma atau Bahasa pemprograman lain dalam proses pengamanan pesan.

Kata Kunci: *algoritma RSA; pesan teks; enkripsi; dekripsi.*

Abstract

Cryptography can be used in prevent data, one of the preventive measures is by encoding messages. Cryptography is the study of encoding messages or ways of data protection. In encoding the content of messages, there is an algorithm conventionally used nowadays, the RSA (Rivest-Shamir-Adleman) algorithm. The RSA algorithm is a method that has two different keys for each encryption and decryption process but is still interrelated to maintain security in processing the data. In finding the key, the RSA algorithm utilizes the rule of prime number. The larger the prime number used as a key, the harder it is to find a large number as a factor. This research describes the process of encrypting text messages, the content of documents using the RSA algorithm, and the key generation process. Those processes are done by converting plaintext into *ciphertext* using ASCII code, which is 256 long, and using PKCS (Public Key Cryptography Standards) is the encryption process on the RSA algorithm. This study uses Pyhton programming language to implement the RSA algorithm on text messages and documents. As a recommendation to the subsequent studies, it is proper to use algorithms or other programming languages to secure messages.

Keywords: *RSA algorithm; text message; encryption; decryption.*

✉ septirahmita1234@gmail.com, nisky@ung.ac.id

PENDAHULUAN

Perkembangan teknologi semakin pesat, dapat dilihat dari proses komunikasi yang semakin mudah dengan adanya internet. Contohnya pada pengiriman sebuah pesan menggunakan email, layanan pesan singkat atau SMS, dan WhatsApp [1]. Selain memiliki keunggulan internet juga memiliki kelemahan, penggunaan internet yang begitu besar dapat memicu hal-hal negatif, seperti mendapatkan informasi hoax melalui aplikasi chatting (WhatsApp, Line, Telegram) dengan presentase mencapai 62.8% [2]. Maka dari itu, diperlukan tindakan pencegahan untuk mengamankan isi pesan dengan cara penyandian pesan sebelum dikirim ke pihak yang bersangkutan. Hal ini untuk melindungi kerahasiaan pesan dan mencegah pesan yang dikirim agar tidak mudah dimodifikasi untuk menjaga keutuhan pesan [1]. Upaya yang dapat dilakukan untuk melindungi pesan adalah penggunaan enkripsi dan dekripsi pada kriptografi [3].

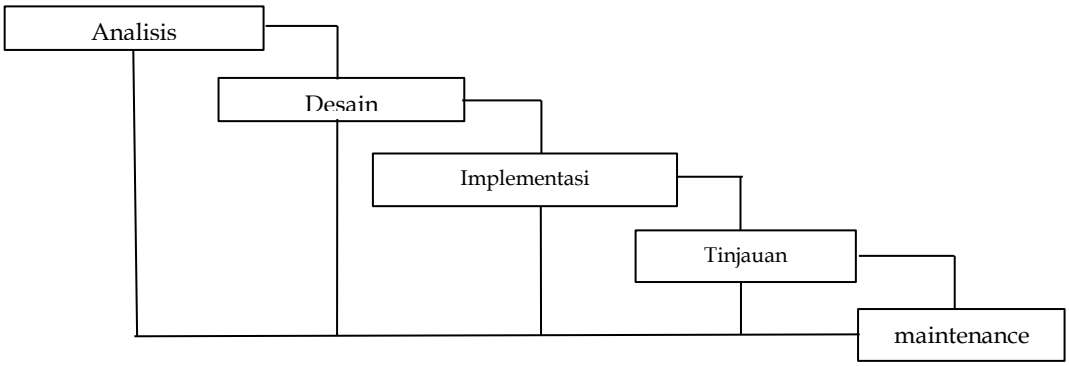
Kriptografi merupakan studi tentang penyandian pesan atau untuk melindungi data. Kriptografi menggunakan beberapa proses yang harus dilakukan sebelum mengirim pesan untuk memastikan keamanan data hanya pengirim dan penerima yang dikenal. Diantaranya ialah enkripsi dan dekripsi [4]. Langkah dasar dalam proses penyandian untuk melindungi pesan adalah proses enkripsi yaitu langkah dalam mengubah pesan menjadi kata sandi, dan dekripsi yaitu proses pengembalian pesan. Pesan yang dimaksud adalah informasi yang dapat dibaca dan dipahami maknanya disebut *plaintext*. Sedangkan pesan yang telah disandikan sehingga tidak memiliki makna lagi disebut *ciphertext*. Salah satu algoritma kriptografi yang digunakan dalam penyandian isi pesan atau penyandian informasi data, yaitu algoritma RSA. Algoritma RSA dinilai memiliki keamanan yang baik karena sulit untuk dipecahkan [5], dalam algoritma RSA terdapat operasi matematika yang terdiri dari mengalikan dua bilangan prima yang dipilih secara acak (p dan q). Hasil perkaliannya adalah n. Semakin besar n maka semakin tinggi tingkat keamanannya karena semakin sulit bagi penyerang untuk memfaktorkan nilai n [6]. Berdasarkan kunci yang digunakan Algoritma kriptografi dapat dibedakan menjadi dua, yaitu algoritma simetris dan algoritma asimetris [7].

RSA adalah proses penyandian kunci asimetrik [8], dan pertama kali ditemukan pada tahun 1976 oleh tiga peneliti MIT (*Massachusetts Institute of Technology*), yaitu Ron Rivest, Adi Shamir, dan Leonard Adleman. Proses perumusan algoritma RSA didasarkan pada Teorema Euler, setelah itu diperoleh kunci bersama dan kunci rahasia yang saling berkaitan. Saat proses penyandian pesan teks, ukuran kunci rahasia lebih besar dibandingkan dengan kunci umum, sehingga membutuhkan waktu yang lebih lama jika di hitung secara manual [9]. Hal ini digunakan untuk mencegah berbagai percobaan dalam memecahkan teks rahasia, semakin besar bilangan prima yang digunakan sebagai kunci, semakin susah ditemukan angka besar sebagai faktornya [1]. Hingga saat ini, banyak peneliti telah menggunakan algoritma RSA untuk mengamanan pesan. Diantaranya penelitian oleh [10] menganalisis kinerja kriptografi *hybrid* dari algoritma *Blowfish* dan algoritma RSA untuk menentukan kecepatan enkripsi dan dekripsi data menggunakan kedua algoritma tersebut. algoritma RSA banyak digunakan untuk mengamankan pesan karena memiliki pencarian kunci yang kompleks. Keamanan sandi RSA terletak pada tingginya kesulitan pemfaktoran angka yang besar, serta memakai kunci yang berbeda untuk setiap proses enkripsi dan dekripsinya sehingga kecil kemungkinan dapat dibaca oleh orang lain [11].

Berdasarkan paparan diatas, penelitian ini menggunakan algoritma RSA yang memiliki dua kunci dalam proses penyandian pesan, kemudian mengimplementasikannya pada GUI Python untuk proses pembentukan kunci, enkripsi dan dekripsi sebuah pesan teks serta isi dokumen agar pesan yang akan dikirim secara konvensional melalui sistem keamanan ganda sehingga semakin aman dari orang yang berusaha mencuri pesan, selain itu diharapkan juga dalam penggunaan algoritma RSA memiliki keefektifan dan efisien dalam hal waktu.

METODOLOGI

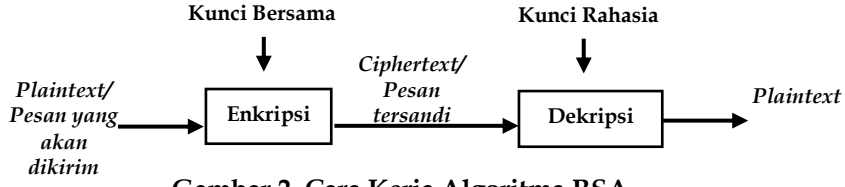
Penelitian ini merupakan penelitian terapan atau penelitian kualitatif dan data menggunakan data simulasi. Setelah itu diimplemetasikan ke dalam program aplikasi menggunakan GUI *python*.



Gambar 1. Diagram Alir Pengembangan Program Aplikasi berbasis GUI Python dengan Metode Waterfall

Gambar 1 menjelaskan tahapan dalam pengimplementasian ke dalam program aplikasi menggunakan metode waterfall [12], metode ini merupakan metode pengembangan perangkat lunak sekuensial, yang terdiri dari lima fase yang saling berhubugan dan berpengaruh. *Output* dari sebuah fase dalam metode Waterfall berfungsi sebagai *input* bagi fase berikutnya, hal ini terlihat dari adanya hubungan dan dampak antara fase ini. Akibatnya, kekurangan pada hasil pelaksanaan fase sebelumnya merupakan awal dari kekurangan pada tahap berikutnya. Sebelum memasuki fase implementasi penulisan kode program, sangat penting untuk bekerja sama untuk mengevaluasi dan mendesain sistem sebaik mungkin. [13]. dan metode algoritma yang digunakan ialah algoritma RSA.

Implementasi algoritma RSA pada proses enkripsi dan dekripsi suatu pesan atau dokumen dapat dilihat pada gambar 2 [14].



Gambar 2. Cara Kerja Algoritma RSA

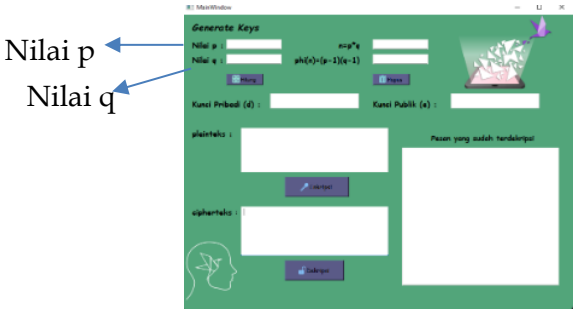
HASIL DAN PEMBAHASAN

Perancangan Sistem Berdasarkan Metode Waterfall

1. Analisa Kebutuhan (Requirement)

   Analisis masalah yang relevan adalah keamanan isi pesan rahasia pada aplikasi messaging masih belum maksimal, contohnya SMS (tidak memiliki fitur *enkripisi end-to-end)* dan aplikasi bajakan seperti WhatsApp MOD. Sehingga dibutuhkan sistem keamanan ganda.

2. Desain

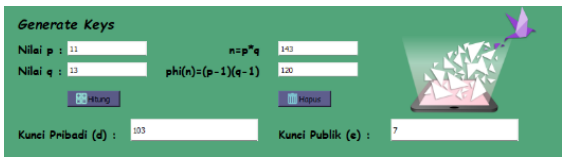


Gambar 3. GUI Enkripsi dan Dekripsi Pesan

Berdasarkan Gambar 3 menjelaskan tentang GUI enkripsi dan dekripsi pesan, dimana diatas terdapat 3 kotak input yaitu nilai p, nilai q dan plaintext, sedangkan untuk kotak outputnya terdiri dari kotak hasil dari perhitungan n, phi(n), kunci pribadi, kunci umum dan *ciphertext* dan juga pesan yang sudah terenkripsi. Kemudian terdapat juga tombol hitung untuk proses pembentukan kunci, tombol hapus untuk menghapus, tombol enkripsi untuk proses enkripsi dari plaintext menjadi *ciphertext*, dan tombol dekripsi untuk proses pengembalian pesan yakni dari *ciphertext* menjadi *plaintext*.

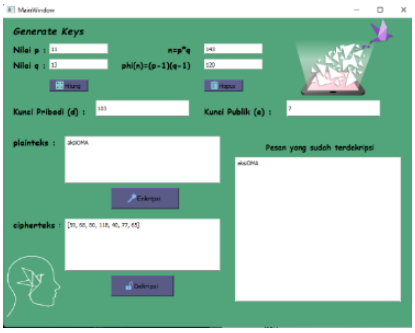
3. Implementasi Algoritma RSA

   Proses selanjutnya adalah Implementasi algoritma kriptografi RSA dalam penyandian pesan berbasis GUI Python terhadap pesan teks menggunakan GUI PyQt5. Proses yang dilakukan sebelum menyandikan pesan ialah membangkitkan kunci umum dan kunci rahasia menggunakan nilai p dan q berupa bilangan prima. Kemudian tekan tombol hitung untuk melakukan pembentukkan kunci. Sehingga di hasilkan nilai n dan nilai phi(n), sehingga diperoleh kunci umum dan kunci rahasia seperti pada gambar 4.



Gambar 4. Pembentukan Kunci Umum dan Kunci Rahasia

Setelah terbentuk kunci umum dan kunci rahasia, pengirim atau penerima pesan melakukan proses enkripsi. Dengan memasukkan pesan teks yang akan disandikan ke dalam textbox, kemudian tekan tombol enkripsi dan dekripsi untuk mendapatkan *ciphertext* dan pesan aslinya seperti pada Gambar 5.



Gambar 5. Enkripsi dan Dekripsi Pesan Teks

4. Tinjauan

   Pada fase ini, sistem akan di uji secara komputasi dan secara perhitungan biasa.

5. Pemeliharaan (Maintenance)

   Pada tahapan ini, perangkat dapat dijalankan sesuai perintah. Untuk pemeliharaan selanjutnya dapat dilakukan setelah ada kesalahan yang terjadi atau diperuntukkan dalam pengembangan aplikasi yang dapat berdiri sendiri nantinya.

Penyandian Pesan

Dalam merumuskan Algoritma RSA yang hasilnya merupakan rumus enkripsi dan dekripsi yang saling berkaitan itu berdasarkan pada penggunaan Teorema Euler.

Teorema 1 [8] *Jika m adalah bilangan bulat positif dan a adalah bilangan bulat dengan gcd (a,m) = 1, maka* $a^{\phi(m)} \equiv 1\ (mod\ m)$.

Sehingga menghasilkan enkripsi dan dekripsi sebagai berikut

$$Enkripsi : m^e mod\ n \qquad (1)$$

$$Dekripsi : c^d mod\ n \qquad (2)$$

Teorema Fermat berkaitan dengan pemanfaatan bilangan prima dalam proses pembangkit pasangan kunci.

Teorema 2 [15] *Jika p adalah bilangan prima dan a adalah bilangan bulat dimana p tidak membagi a, maka* $e^{(p-1)} \equiv 1\ (mod\ p)$.

Membangkitkan kunci dengan menggunakan persamaan

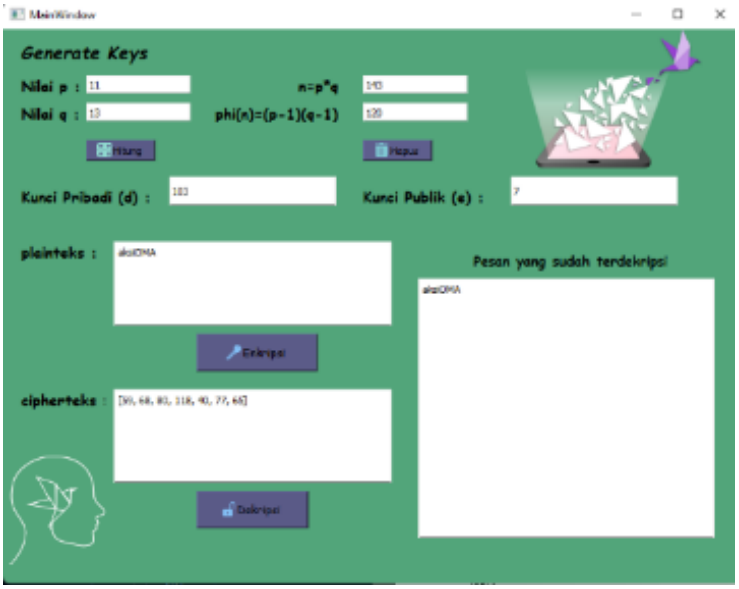
$$ed \equiv 1\ mod\ \phi(m)\ atau\ d \equiv e^{-1}\ mod\ \phi(m) \qquad (3)$$

Dimana :

$\phi(m)$: hasil perhitungan dari (p-1)(q-1). Dapat dikatakan bahwa ϕ (m) adalah kardinalitas dari himpunan residu sederhana modulo m dan ϕ (m) sering disebut indikator m. Hubungannya dengan pesan teks yakni digunakan dalam proses pencarian kunci pribadi untuk deskripsi.

e : Kunci publik (kunci enkripsi)
d : Kunci rahasia (kunci dekripsi)
m : *plaintext* bersifat rahasia
c : *ciphertext* bersifat tidak rahasia
n : hasil perkalian dari dua buah bilangan prima

Pengujian akan dilakukan pada kata aksiOMA, hasil pengujian adalah sebagai berikut:



Gambar 6. Hasil Enkripsi dan Dekripsi Pesan Teks

Berdasarkan dari aplikasi Python di atas, diperoleh *ciphertext* sebesar (59,68,80,118,40,77,65) dan dalam proses pengembalian pesan ke semula menjadi aksiOMA. Pada pengujian secara perhitungan langsung tanpa menggunakan aplikasi, misalkan teks yang akan dienkripsi adalah P = aksiOMA, dalam decimal ASCII nya menjadi menjadi a =97; k=107; s=115; i=105; O=79; M=77; A=65. Masing-masing nilai akan dikalikan dengan 256, karena Panjang kunci yang digunakan ialah 256.

Kemudian *plaintext* tersebut dipecah menjadi 1 blok yang terdiri dari satu huruf. Jika kalimat yang dienkripsi merupakan kalimat yang cukup panjang maka dalam satu blok bisa lebih dari satu huruf.

Dalam algoritma RSA ada standar proses enkripsi yang disebut PKCS (*Public Key Cryptography Standard*), pada umumnya digunakan untuk mengaitentikasi pesan biner terenskripsi, dan didekripsi oleh penerima pesan [16]. Jika hanya menggunakan kode ASCII dalam mengkonversi teks, program akan mudah ditebak. Berikut konversi yang menggunakan PKCS dan kode ASCII:

Tabel 1. Konversi dari ASCII

Teks (Karakter)	PKCS	Hasil
A	97×256^0	97
K	107×256^0	107
S	115×256^0	115
I	105×256^0	105
O	79×256^0	79
M	77×256^0	77
A	65×256^0	65

Ciphertext dalam bentuk desimal ini akan dienkripsi menggunakan kunci pribadi. Kunci pribadi diperoleh dengan cara sebagai berikut:

1. Memilih nilai p = 11 dan q = 13
2. Mengitung nilai n
   n = pq = 11x13=143
3. Menghitung nilai ϕ(n) dengan persamaan
   ϕ(n)=(p-1)(q-1) = (11-1)(13-1)= 120
4. Dipilih e = 7, dimana gcd(e,m(n)) = 1. Lalu menghitung nilai d menggunakan algoritma *euclid*:
   $d \equiv e^{-1} \ (mod \phi(n))$
   $d \equiv 7^{-1} \ (mod \ 120)$
   $d \equiv -17 \ mod \ 120$
   $d \equiv 103$
   Diperoleh nilai d = 103 berdasarkan perhitungan diatas. Maka diperoleh kunci pribadi untuk enkripsi yaitu d= 103
   Dengan,
   e, n : pasangan kunci umum
   d, n : pasangan kunci rahasia
   Kemudian plainteks dienkripsikan dengan

$$C = m^e \ mod \ n$$

C1= $97^7 mod \ 143 \equiv 59$
C2= $107^7 mod \ 143 \equiv 68$
C3= $115^7 mod \ 143 \equiv 80$
C4= $105^7 mod \ 143 \equiv 118$
C5= $79^7 mod \ 143 \equiv 40$
C6= $77^7 mod \ 143 \equiv 77$
C7= $65^7 mod \ 143 \equiv 65$

Sehingga dihasilkan chiperteks C = 59 68 80 118 40 77 65

Proses dekripsi dilakukan menggunakan kunci rahasia d = 103
Jadi :

$$p = m^d \ mod \ n$$

1= $59^{103} mod \ 143 \equiv 97$
2= $68^{103} mod \ 143 \equiv 107$
3= $80^{103} mod \ 143 \equiv 115$
4= $118^{103} mod \ 143 \equiv 105$
5= $40^{103} mod \ 143 \equiv 79$
6= $77^{103} mod \ 143 \equiv 77$
7= $65^{103} mod \ 143 \equiv 65$

Akhirnya kita peroleh kembali plainteks semula
P = 97 107 115 105 79 77 65
P = aksiOMA

Jadi, diperoleh pehitungan di Python yang dapat di lihat pada gambar 6, dan perhitungan biasa menghasilkan hasil yang sama atau pesan dapat kembali di dekripsi setelah proses enkripsi.

Setiap pesan yang akan disandikan memiliki waktu yang relatif singkat, dan waktu tercepat berdasarkan panjangnya *plaintext* yang akan di enkripsi dan dekripsi. Dapat dilihat pada Tabel 2 di bawah ini dengan pengukuran waktu perdetik.

Tabel 2. Hasil Penyandian Pesan dan Waktu Enkripsi dan Dekripsi Tiap Pesan Teks

No.	Pesan dan Kunci	Cipher text	Plain text	Waktu
1.	Ayu (3, 107)	109 110 145	65 121 117	0,01002669334411 6211
2.	aksiOMA (7, 103)	97 107 115 105 79 77 65	59 68 80 118 40 77 65	0,05023789405822 754
3.	V14n4 (3, 235)	290 349 239 36 239	86 49 52 110 52	0,06394386291503 906
4.	Wildiyanti Adam (5, 107)	83, 101, 112, 116, 105, 32, 82, 97, 104, 109, 105, 116, 97, 32, 83, 97, 114, 105	329, 545, 801, 231, 564, 706, 656, 792, 808, 290, 564, 231, 792, 706, 329, 792, 91, 564	0,06406283378601 074

Jadi, berdasarkan Tabel 2 diatas terlihat bahwa jika semakin panjang atau semakin kompleks *plaintext* yang akan disandikan maka waktu yang dibutuhkan juga akan semakin sedikit lebih lama.

Penyandian Dokumen

Proses penyandian isi dokumen memiliki proses yang sama dengan penyandian isi pesan. Dengan proses pertama ialah pembentukan kunci, kemudian input file yang akan di enkripsi menggunakan kunci yang telah dibuat sebelumnya, setelah itu, output berupa file yang isinya telah dienkripsi menggunakan kunci. *Syntax* yang digunakan dapat dilihat pada Gambar 7 berikut :

```
def key_create(self):
    key = Fernet.generate_key()
    return key

def key_write(self, key, kunci):
    with open('kunci.txt', 'wb') as mykey:
        mykey.write(key)

def key_load(self, kunci):
    with open('kunci.txt', 'rb') as mykey:
        key = mykey.read()
    return key


def file_encrypt(self, key, enk, encrypted_file):

    f = Fernet(key)

    with open('input_file.pdf', 'rb') as file:
        original = file.read()

    encrypted = f.encrypt(original)

    with open ('output_ciphertext.pdf', 'wb') as file:
        file.write(encrypted)

def file_decrypt(self, key, encrypted_file, decrypted_file):

    f = Fernet(key)

    with open('output_ciphertext.pdf', 'rb') as file:
        encrypted = file.read()

    decrypted = f.decrypt(encrypted)

    with open('output_plaintext.pdf', 'wb') as file:
        file.write(decrypted)
```

Gambar 7. *Syntax* Proses Penyandian Dokumen

Syntax diatas memperlihatkan bahwa kunci.txt merupakan file yang memuat kunci yang dibuat sebelum enkripsi dan dekripsi pesan, untuk *key write* digunakan untuk membaca kunci yang telah diinput kedalam *syntax* dan *key load* akan membentuk kunci sesuai dengan kunci yang telah dimasukkan. Kemudian file dokumen yang dapat disandikan diantaranya dengan ekstensi .txt, .pdf, dan .csv. File tersebut akan diinput pada *open input* file, selanjutnya dengan menggunakan *key write* dokumen akan membentuk *output ciphertext* dan dengan menggunakan *key load file* dokumen akan membentuk *output plaintext*. Selanjutnya hasil enkripsi dan dekripsi tersebut akan disimpan ke dalam nama file yang telah dituliskan pada *output ciphertext* dan *output plaintext.*

1) Penyandian Isi Dokumen dengan format txt.

Berikut file dengan format .txt yang akan dienkripsi dapat dilihat pada Gambar 8, dimana file tersebut dapat dibaca secara langsung atau dapat dimengerti ketika orang awam membacanya.

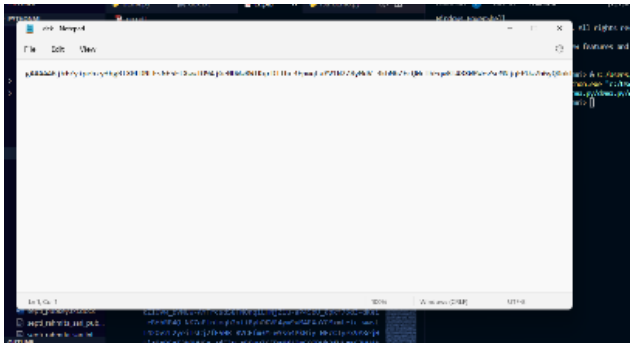
Gambar 8. File yang akan di Enkripsi

Gambar 8 menunjukkan dokumen akan dienkripsi, kemudian dengan menginput kunci dan file dokumen .txt seperti pada syntax Gambar 7 diatas. Setelah itu Gambar 9 menunjukkan dua proses yakni mengenkripsi file dek.txt dan dekripsi enk.txt yang kemudian hasil dari proses enkripsi dan dekripsi berupa *ciphertext* tersimpan dalam nama file hasil.txt dapat dilihat sebagai berikut:

```
mykey=encryptor.key_create()

encryptor.key_write(mykey, 'mike.txt')

loaded_key=encryptor.key_load('mike.txt')

encryptor.file_encrypt(loaded_key, 'dek.txt', 'enk.txt')

encryptor.file_decrypt(loaded_key, 'enk.txt', 'hasil.txt')
```

Gambar 9. Proses Enkripsi dan Dekripsi

Kemudian pada Gambar 10, file tersebut sudah menjadi *ciphertext* yang proses enkripsinya dapat dilihat pada Gambar 7 dan Gambar 9. Dari hasil enkripsi terlihat bahwa kalimat yang di input mengasilkan kalimat yang akan sulit dimengerti ketika orang awam membacanya. Karena isi dari file tersebut berupa tulisan acak dengan kombinasi huruf, angka dan simbol-simbol.



Gambar 10. File dengan isi berupa *Ciphertext*

Setelah proses enkripsi dilakukan dan menghasilkan *ciphertext*, proses selanjutnya adalah mengembalikan pesan menjadi pesan semula (*plaintext*) dengan cara mendekripsi file tersebut menggunakan kunci pribadi. Kemudian hasil dari pengembalian pesan dapat dilihat pada Gambar 11.

Gambar 11. Hasil Dekripsi File

SIMPULAN

Hasil dari penelitian ini ialah berdasarkan implementasi algoritma RSA (Rivest-Shamir-Adleman) yang dibuat pada GUI Python dalam penyandian pesan teks dapat dijalankan. Hasil pengujian ini diperoleh dari pembuktian antara perhitungan manual dan perhitungan di program menghasilkan hasil yang sama dengan menggunakan kunci umum dan kunci rahasia yang sama. Pada penelitian selanjutnya, disarankan untuk menggunakan algoritma atau Bahasa pemrograman lain dalam proses pengamanan pesan dan dokumen.

DAFTAR PUSTAKA

[1] A. Ginting, R. R. Isnanto, dan I. P. Windasari, "Implementasi Algoritma Kriptografi RSA untuk Enkripsi dan Dekripsi Email," *J. Teknol. dan Sist. Komput.*, vol. 3, no. 2, hal. 253, 2015, doi: 10.14710/jtsiskom.3.2.2015.253-258.

[2] M. W. Indriyanto, D. Hariyadi, dan M. Habibi, "Investigasi Dan Analisis Forensik Digital Pada Percakapan Grup Whatsapp Menggunakan Nist Sp 800-86 Dan Support Vector Machine Digital Forensics Investigation and Analysis on Whatsapp Group Chats Using Nist Sp 800-86 and Support Vector Machine," *Cyber Secur. dan Forensik Digit.*, vol. 3, no. 2, hal. 34–38, 2020.

[3] A. A. Rakhman dan A. W. Kurniawan, "Implementasi Algoritma Kriptografi Rivest Shamir Adleman (Rsa) Dan Vigenere Cipher Pada Gambar Bitmap 8 Bit," *Techno.COM*,

vol. 14, no. 2, hal. 122–134, ISSN:2356-2579, 2015.

[4] Y. Reswan dan D. A. Prabowo, "Perancangan Aplikasi Pengamanan Data Text Menggunakan Kombinasi Algoritma Hill Cipher Dan Algoritma RSA," *JSI J. Sist. Inf.*, vol. 10, no. 2, hal. 1535–1545, 2018, doi: 10.36706/jsi.v10i2.8057.

[5] R. Sahara, H. Prastiawan, dan A. Rohman, "Barat 11650 3 Sekolah Tinggi Ilmu Komputer Cipta Karya Informatika 3 Jl," *Univ. Mercu Buana 12 Jl. Raya Meruya Selatan*, vol. 5, no. 9, hal. 118–122, 2017.

[6] D. Wulansari, Alamsyah, F. A. Setyawan, dan H. Susanto, "Mengukur Kecepatan Enkripsi dan Dekripsi Algoritma RSA pada Pengembangan Sistem Informasi Text Security," *Semin. Nas. Ilmu Komput. (SNIK 2016)*, no. Snik, hal. 85–91, 2016.

[7] T. Rahajoeningroem dan M. Aria, "Studi dan Implementasi Algoritma RSA untuk pengamanan Data transkip Mahasiswa," *Maj. Ilm. Unikom*, vol. 8, no. 1, hal. 77–90, 2011, [Daring]. Tersedia pada: http://jurnal.unikom.ac.id/_s/data/jurnal/v08-n01/volume-81-artikel-9.pdf/pdf/volume-81-artikel-9.pdf.

[8] A. N. Agustina, Aryanti, dan Nasron, "Pengamanan Dokumen Menggunakan Kombinasi Metode Rsa (Rivest Shamir Adleman) Berbasis Web," *Pros. Semin. Nas. Multi Disiplin Ilmu Call Pap. UNISBANK*, hal. 14–19, 2017, [Daring]. Tersedia pada: https://www.neliti.com/id/publications/174360/pengamanan-dokumen-menggunakan-metode-rsa-rivest-shamir-adlemanberbasis-web.

[9] N. T. E. Hermawan, E. Winarko, dan A. Ashari, "Multi prime numbers principle to expand implementation of CRT method on RSA algorithm," *AIP Conf. Proc.*, vol. 2331, no. April, 2021, doi: 10.1063/5.0041856.

[10] S. Suhandinata, R. A. Rizal, D. O. Wijaya, P. Warren, dan Srinjiwi, "Analisis Performa Kriptografi Hybrid Algoritma RSA," *Jurteksi*, vol. VI, no. 1, hal. 1–10, 2019.

[11] I. Halik dan Y. Prayudi, "Studi dan Analisis Algoritma Rivest Code 6 (RC6) dalam Enkripsi/Dekripsi Data," *Snati*, vol. 6, no. D, 2005, [Daring]. Tersedia pada: http://journal.uii.ac.id/index.php/Snati/article/view/1402.

[12] A. A. Wahid, "Analisis Metode Waterfall Untuk Pengembangan Sistem Informasi," *J. Ilmu-ilmu Inform. dan Manaj. STMIK*, no. November, hal. 1–5, 2020, [Daring]. Tersedia pada: https://www.researchgate.net/profile/Aceng_Wahid/publication/346397070_Analisis_Metode_Waterfall_Untuk_Pengembangan_Sistem_Informasi/links/5fbfa91092851c933f5d76b6/Analisis-Metode-Waterfall-Untuk-Pengembangan-Sistem-Informasi.pdf.

[13] M. Natsir, "Pengembangan Prototype Sistem Kriptografi Untuk Enkripsi Dan Dekripsi Data Office Menggunakan Metode Blowfish Dengan Bahasa Pemrograman Java," *Jurnal*, vol. 6, no. 2, hal. 2089–5615, 2016.

[14] R. Munir, *Kriptografi*, Edisi Kedu. Bandung: Informatika, 2019.

[15] D. B. Ginting, "Peranan Aritmetika Modulo dan Bilangan Prima pada Algoritma Kriptografi RSA (Rivest-Shamir-Adleman)," *Media Inform.*, vol. 9, no. 2, 2010.

[16] A. M. S. Pangan, I. L. Lacuesta, R. C. Mabborang, dan F. P. Ferrer, "Authenticating Data Transfer Using RSA-Generated QR Codes," *Eur. J. Inf. Technol. Comput. Sci.*, vol. 2, no. 4, hal. 18–30, 2022, doi: 10.24018/compute.2022.2.4.73.