Community Service Article (COMERS)
e-ISSN: 3063-9425

Volume : 1 Number : 2 Year : 2024
website:https://ejournal.univbhaktiasih.ac.id/index.php/comers

IMPLEMENTASI KRIPTOGRAFI TEKS MENGGUNAKAN RSA

AGUNG WIDYANTO

Program Studi Informatika, Fakultas Ilmu Kesehatan dan Sains, Universitas Bhakti Asih Tangerang
Jl. Raden Fatah No.62, Kota Tangerang, Banten, Indonesia
Email: agungwidyanto@gmail.com

Sari – Mempelajari teknologi keamanan data sangat relevan di era digital, salah satunya adalah dengan mempelajari kriptografi. Penelitian ini memiliki tujuan untuk menerapkan kriptografi enkripsi asimetris RSA pada teks sederhana dengan memanfaatkan paket python PyCryptodome. Pengujian script dengan menerapkan lingkungan pengembangan Python dengan fitur enkripsi pesan dan dekripsi pesan. Alih-alih menggunakan pustaka RSA yang telah diakui publik, menerapkan tanpa kehati-hatian dengan *self development* akan mengurangi efisiensi waktu, seperti yang dilakukan penelitian sebelumnya. Maka akan disandingkan konsumsi waktu keduanya. Terbukti penerapan kriptografi RSA dengan PyCryptodome lebih efisien dan cepat. Hasil dan manfaat penelitian ini adalah efisiensi waktu dan keamanan data yang lebih terjamin dan tersedianya prototipe *script* yang dapat diintegrasikan pada pengembangan aplikasi lebih lanjut dengan kemampuan mengenkripsi dan mendekripsi pesan menggunakan algoritma kriptografi RSA. Dengan *script* sederhana ini, diharapkan kita mengetahui cara mengimplementasikan kriptografi dengan menggunakan beberapa algoritma kriptografi.

Kata kunci: Kriptografi, RSA, Enkripsi, Dekripsi

Abstract *- Studying data security technology is very relevant in the digital era, one of which is studying cryptography. This research aims to apply RSA asymmetric encryption cryptography to simple text by utilizing the PyCryptodome Python package—script testing by implementing the Python development environment with message encryption and message decryption features. Instead of using a publicly recognized RSA library, implementing it without careful self-development will reduce time efficiency, as previous research has done. Then the time consumption of both will be compared. It has been proven RSA cryptography with PyCryptodome is faster and more efficient. The results and benefits of this research are more guaranteed time efficiency and data security, as well as the availability of script prototypes that can be integrated into further application development with the ability to encrypt and decrypt messages using the RSA cryptographic algorithm. With this simple script, we hope to know how to implement cryptography using several cryptography algorithms..*

Keywords*: Cryptographic, RSA, Encryption, Decryption.*

1. PENDAHULUAN

Mempelajari teknologi keamanan data memiliki banyak alasan yang sangat relevan di era digital ini. Ancaman keamanan data, seperti peretasan atau pelanggaran keamanan, dapat mengakibatkan pencurian identitas, penipuan, atau penggunaan informasi pribadi untuk kegiatan ilegal. Kegiatan bisnis menyimpan informasi yang sangat berharga seperti rahasia perusahaan, data pelanggan, dan strategi pemasaran. Keamanan data yang lemah dapat merugikan reputasi bisnis dan menyebabkan kerugian finansial yang signifikan. Seiring dengan berkembangnya teknologi, metode peretasan dan ancaman siber juga berkembang.

Salah satu cara untuk meningkatkan atau menjaga keamanan data dengan menerapkan kriptografi, yakni proses menyembunyikan informasi dengan cara mengkonversi data yang dapat dibaca, kedalam data yang tidak bisa dibaca menggunakan sejumlah kunci (*key*) atau algoritma enkripsi. Kriptografi dari kata *kryptos* yang artinya tersembunyi dan *graphia* yang artinya sesuatu yang tertulis sehingga kriptografi dapat juga disebut sebagai sesuatu yang tertulis secara rahasia atau tersembunyi. Dengan kata lain, kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan ketika pesan dikirim dari suatu tempat ke tempat yang lain.

Community Service Article (COMERS)
e-ISSN: 3063-9425

Volume : 1 Number : 2 Year : 2024
website:https://ejournal.univbhaktiasih.ac.id/index.php/comers

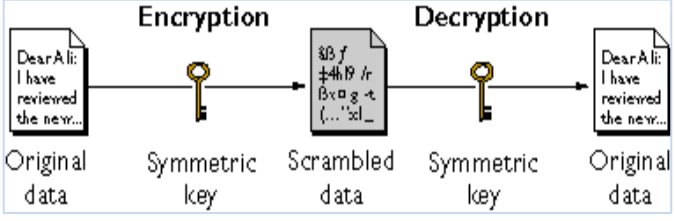
Definisi kriptografi (Munir, 2006) sebagai ilmu yang mempelajari teknik matematika yang berhubungan dengan keamanan informasi misalnya kerahasiaan, integritas data, otentikasi pengirim/penerima data, dan otentikasi data.

Kriptografi sebagai suatu disiplin yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, serta otentikasi (Munir, 2006). Kriptografi (Ariyus, 2006) juga digunakan untuk mengidentifikasi pengiriman pesan dan tanda tangan digital dan keaslian pesan dengan sidik jari digital. Dalam kriptografi, pesan atau informasi yang dapat di baca disebut sebagai *plaintext* atau *clear text*. Proses yang dilakukan untuk mengubah teks asli (*plaintext*) ke dalam teks rahasia (*chipertext*) disebut enkripsi. Pesan yang dibuat tidak terbaca disebut teks rahasia (*chipertext*). Proses kebalikan dari enkripsi disebut dekripsi. Dekripsi mengurai teks rahasia (*chipertext*) menjadi teks asli (*plaintext*). Proses enkripsi dan dekripsi membutuhkan penggunaan sejumlah informasi rahasia, yang sering disebut kunci (*key*).

Beberapa tujuan dari kriptografi ini yang menjadi aspek keamanan pada informasi yaitu :

1. Kerahasiaan (*confidentiality*) adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka/mengupas informasi yang telah disandi.
2. Untuk integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihakpihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubsitusian data lain kedalam data yang sebenarnya.
3. Autentikasi berhubungan dengan identifikasi/pengenalan, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui kanal harus diautentikasi keaslian, isi datanya, waktu pengiriman, dan lain-lain.
4. Keadaan untuk mencegah terjadinya penyangkalan terhadap pengiriman/terciptanya suatu informasi oleh yang mengirimkan/membuat.

Secara mendasar tipe kriptografi yakni simetrik dan asimetrik. *Symmetric encryption* menggunakan satu kunci (*key*) untuk mengenkripsi dan mendekripsi sebuah informasi yang dikirim atau diterima (Meko, 2018). Artinya, untuk mendekripsi informasi, seseorang harus memiliki kunci yang sama dengan yang digunakan untuk mengenkripsi informasi tersebut (lihat **Gambar 1**). Kuncinya, dalam praktiknya, sebagai rahasia antara dua pihak atau lebih yang dapat digunakan bersama. Persyaratan inilah, bahwa kedua belah pihak memiliki akses ke kunci rahasia ditengarai sebagai isu salah satu kelemahan utama enkripsi simetris, yakni pada kunci simetrisnya.



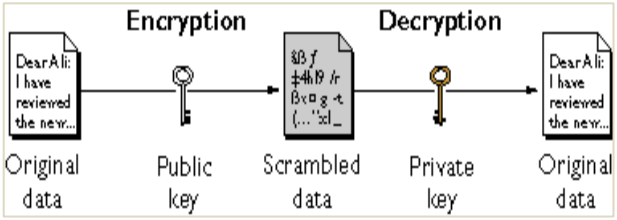
Gambar 1. Symmetric Encryption.

Community Service Article (COMERS)
e-ISSN: 3063-9425

Volume : 1 Number : 2 Year : 2024
website:https://ejournal.univbhaktiasih.ac.id/index.php/comers

Teknik kriptografi dengan enkripsi simetris diantaranya meliputi algoritma DES, AES, IDEA, Saphent, Blowfish, Skipjack, dan Twofish. Tiap algoritma simetris ini memiliki kelebihan dan kekurangan masing-masing dalam proses enkripsi dan dekripsi data dilihat dari segi kecepatan maupun keamanan data *ciphertext* yang dihasilkan. Dijelaskan (Meko, 2018) kecepatan enkripsi dan dekripsi data dengan menggunakan algoritma AES lebih baik dibanding algoritma ketiga algoritma lainnya yaitu DES, Blowfish dan IDEA dimana kecepatan tertinggi untuk proses enkripsinya adalah 48% dan kecepatan dekripsinya adalah 45% dan kecepatan terendah dimiliki oleh algoritma IDEA yaitu kecepatan enkripsi sebesar 5% dan dekripsi sebesar 2%.

Sedangkan *Asymmetric encryption* menggunakan kunci (key) yang berbeda - beda untuk mengenkripsi dan mendekripsi sebuah informasi yang dikirim atau diterima (lihat **Gambar 2**). Terdapat *public key* untuk melakukan enkripsi dan *private key* yang digunakan proses dekripsi.



Gambar 2. Asymmetric Encryption.

Algoritma RSA merupakan salah satu teknik kriptografi asimetris yaitu teknik modern riptografi yang memiliki dua kunci yaitu kunci publik yang digunakan untuk enkripsi dan satu teknik kunci privat untuk membuat deskripsi proses, terbukti populer dan masih teknologi ini masih sangat relevan diimplementasikan (Galla, 2016) (Sihotang *et al.*, 2020) (Indra & Cyra Nabila, 2023).

Dalam kebanyakan kasus, kriptografi memiliki dua fungsi yang saling terkait untuk digunakan, satu untuk enkripsi dan yang lainnya untuk dekripsi. Secara umum kriptografi modern, memiliki kemampuan untuk menjaga kerahasiaan informasi yang dienkripsi tidak didasarkan pada algoritma kriptografi yang dikenal luas sebelumnya (simetris), namun menggunakan kunci ganda yang harus digunakan bersama algoritma tersebut secara acak pada saat proses enkripsi atau dekripsi.

Popularitas algoritma RSA sebagai kriptografi asimetri, memiliki ciri yakni kunci yang digunakan untuk enkripsi berbeda dengan kunci yang digunakan untuk dekripsi. Dikembangkan pada tahun 1976 oleh tiga orang, yakni Ron Rivest, Adi Shamir, dan Leonard Adleman, Algoritma RSA dikenal sebagai kriptografi kunci public (*public-key cryptography*). RSA telah digunakan untuk transmisi data yang diklaim lebih aman dengan teknik mengenkripsi informasi yang dikirim dalam bentuk chiperteks yang tidak dapat dipahami sehingga hanya orang dengan kunci privatnya yang dapat memulihkan informasi data tersebut. Dengan memodifikasi Algoritma RSA, proses enkripsi dan dekripsi file berupa teks dapat dilakukan.

Community Service Article (COMERS)
e-ISSN: 3063-9425

Volume : 1 Number : 2 Year : 2024
website:https://ejournal.univbhaktiasih.ac.id/index.php/comers

Dalam proses kriptografi RSA mencakup *Key Generate*, *Encrypt* dan *Decypt*. (Galla, 2016) Keamanan RSA bergantung pada faktorisasi bilangan. Banyak algoritma efisien dikembangkan untuk meningkatkan konsep teori bilangan di RSA dan untuk mengatasi serangan. Pada kasus lainnya, dilakukan pengaturan kombinasi penerapan algoritma RSA dan algoritma *One Time Pad* (Indra & Cyra Nabila, 2023) yang diklaim membantu dalam pengamanan pesan teks yang lebih aman. Penggunaan perhitungan dengan struktur alfabet pada algoritma *One Time Pad* dan dipadukan dengan penggunaan pemfaktoran bilangan prima pada RSA dapat membuat sistem keamanan pesan teks menjadi lebih aman dan terjamin. Algoritma *One Time Pad* merupakan algoritma yang sederhana namun sangat aman karena kuncinya hanya digunakan satu kali, dan setelah itu diperbarui kembali. Oleh karena itu dibutuhkan kunci yang sama panjangnya dengan *plaintext*, sehingga semakin panjang *plaintext* maka semakin panjang pula kuncinya. Enkripsi menggunakan kriptografi sangat luas, dan meliputi data teks ((Adhar, 2019) (Fachri & Sembiring, 2020) (Satria *et al*., 2020), email, gambar (Rahmawati & Soekarta, 2018), video (Syahputri, 2019), file dan data sensitif lainnya.

Sebagai salah satu kajian tema teknologi keamanan data, diharapkan kita mengetahui cara mengimplementasikan kriptografi dengan menggunakan beberapa algoritma enkripsi. Oleh karenanya penulis melakukan eksperimen menerapkan kode kriptografi sederhana untuk teks. Penulis memiliki tujuan untuk melakukan pemutakhiran solusi permasalahan (Malik & Baharsyah -13521029, 2022) terhadap dugaan konsumsi waktu yang lebih banyak. Alih-alih menggunakan pustaka RSA yang telah diakui publik, menerapkan tanpa kehati-hatian dengan *self development* dirasa akan mengurangi efisiensi waktu. Selain dari sisi ukuran file hasil, efektifitas kriptografi dapat dibuktikan dari sisi konsumsi waktu yang lebih baik.

Manfaat yang didapat adalah efisiensi waktu dan keamanan data yang lebih terjamin. Penulis menyandingkan dari sisi efektifitas waktu penerapan yang telah dilakukan sebelumnya (Malik & Baharsyah -13521029, 2022) dengan pendekatan yang lebih praktis.

2. DATA DAN METODOLOGI

Kajian dilakukan dengan mengamati hasil eksperimen yang didapatkan dengan menerapkan kriptografi pada teks sederhana. Diberikan dalam bentuk file text sebanyak 3 buah file, dengan varian jumlah karakter yang berbeda. Pembuatan *script* di lingkungan pengembangan Python dengan memanfaatkan pustaka mandiri kriptografi berbasis Python, yakni PyCryptodome.

Diawal metoda disertakan *script* sederhana enkripsi simetris (DES, AES, dan Blowfish) untuk memperlihatkan kompleksitas pengamanan dengan menyandingkan kunci yang dihasilkan melalui enkripsi simeteris. Hal yang sama dilakukan pada enkripsi asimetris (RSA). Selanjutnya merujuk studi kasus *script* enkripsi asimetris menggunakan algoritma RSA (Malik & Baharsyah -13521029, 2022) dan script pendekatan yang diusulkan penulis.

Analisis data dilakukan dengan mengamati konsumsi waktu yang diperlukan pada saat menerapkan kriptografi pada suatu berkas. Algoritma RSA melakukan enkripsi dan dekripsi dengan menggunakan konsep pada bilangan prima berserta aritmetika dan modulo. Kunci enkripsi bersifat publik dan dekripsi bersifat *private* yang merupakan bilangan bulat.

Community Service Article (COMERS)
e-ISSN: 3063-9425

Volume : 1 Number : 2 Year : 2024
website:https://ejournal.univbhaktiasih.ac.id/index.php/comers

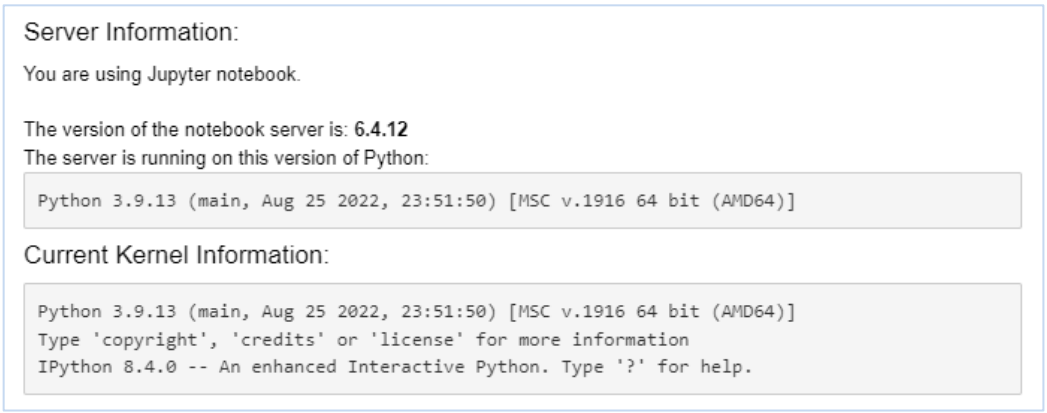
Mendapatkan kunci dekripsi dapat dilakukan dengan metode yang umum yakni faktorisasi dengan menggunakan pohon faktor. Kelemahannya adalah semakin besar bilangan yang difaktorkan tentu membutuhkan konsumsi waktu yang lebih banyak, karena kompleksitas tingkat pemfaktoran semakin tinggi.

Besaran-besaran yang digunakan pada algoritma RSA:

p dan q bilangan prima (rahasia)
r = p* q (tidak rahasia)
Ø(r) = (p – 1)(q – 1) (rahasia)
PK (kunci enkripsi) (tidak rahasia)
SK (kunci dekripsi) (rahasia)
X (plainteks) (rahasia)
Y (cipherteks) (tidak rahasia)

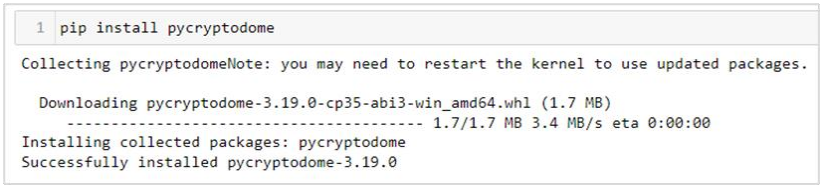
3. HASIL DAN PEMBAHASAN

Untuk pembelajaran *script*, penulis menyiapkan lingkungan pengembangan dengan perangkat lunak yang biasa dipergunakan seperti yang diperlihatkan pada **Gambar 3**.



Gambar 3. Informasi Python Environment.

Digunakan pustaka pada lingkungan Python yakni PyCryptodome, pustaka mandiri Python untuk kriptografi tingkat-rendah yang memiliki banyak fungsi algoritma enkripsi. Cara memasang pustaka PyCryptodome diperlihatkan pada **Gambar 4**.



Gambar 4. Instalasi pustaka pycryptodome.

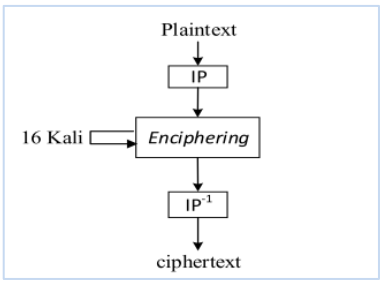
Pengenalan kriptografi pada teks penulis awali dengan penjelasan yang paling mendasar yakni enkrip dan dekrip. Dan menerapkan kode untuk menyembunyikan informasi penting

Community Service Article (COMERS)
e-ISSN: 3063-9425

Volume : 1 Number : 2 Year : 2024
website:https://ejournal.univbhaktiasih.ac.id/index.php/comers

dalam bentuk yang disamarkan, dan menormalkannya kembali sehingga informasi dapat diterima utuh.

1. DES (Data Encryption Standard) adalah standar enkripsi data menggunakan algoritma symmetric encryption. Pada algoritma ini, kunci (key) hanya memiliki ukuran blok data tetap sebesar 8 byte. Tetapi, *Secret Key* yang digunakan untuk enkripsi dan dekripsi ada 64 byte. 56 byte digenerate secara random, dan 8 byte digunakan untuk error checking (lihat **Gambar 5**).



Gambar 5. Skema Global DES.

Runtutan kode yang ditulis pada **Gambar 6** merupakan cara menerapkan algortima DES dengan menggunakan paket PyCryptodome.

```
# encryption
from Crypto.Cipher import DES
from Crypto.Util.Padding import pad

data = "Data ini akan dienkripsi"

# 8 byte block
key = b'1n1kunC1'

# Menetapkan panjang data yang akan dienkripsi
BLOCK_SIZE = 32

des = DES.new(key, DES.MODE_ECB)
padded_text = pad(data.encode(), BLOCK_SIZE)
encrypted_text = des.encrypt(padded_text)

# decryption
key = b'1n1kunC1' # 8 byte block
des = DES.new(key, DES.MODE_ECB)
decrypted_text = des.decrypt(encrypted_text)

print("Text : ", data)
print("Encrypted text :", encrypted_text)
print("Decrypted text", decrypted_text.decode())
```

```
Text :  Data ini akan dienkripsi
Encrypted text : b'p8\xf1\\f\xd9\xd1\n*\x8d\taa\xb0\xa4\xec\x1a\x19\xb4sB\x84E\xcd\xad\xc8\xb4\xfd\x89\xc3k!'
Decrypted text Data ini akan dienkrip
```

Gambar 6. DES untuk kriptografi teks sederhana.

2. AES adalah lanjutan dari algoritma enkripsi standar DES dengan pendekatannya *blokchipertext* simetrik. Istilah yang dikenal sebagai enkripsi disebut ciphertext, dan dekripsi sebagai plaintext. Algoritma AES menggunakan kunci kriptografi 28, 192, dan 256 bit untuk mengenkrip dan dekripsi data pada blok tetap 128 bits (Anoop, 2007), yang nantinya akan menjadi tiga kunci yang masing - masing memiliki panjang 128, 192 dan 256 byte. AES dikenal cukup cepat dan aman. Untuk menerapkan AES dapat digunakan script pada **Gambar 7.**

Community Service Article (COMERS)
e-ISSN: 3063-9425

Volume : 1 Number : 2 Year : 2024
website:https://ejournal.univbhaktiasih.ac.id/index.php/comers

```python
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

data = 'Data ini akan di enkripsi!'
key = b'1n1kuNc1k174h3h3' # 16 block size

# encryption
cipher = AES.new(key, AES.MODE_EAX)
nonce = cipher.nonce
ciphertext, tag = cipher.encrypt_and_digest(data.encode())
print("Text : ", data)
print("#nonce : ", nonce)
print("#tag : ", tag)
print("#enkripsi : ", ciphertext)

# decryption
key = b'1n1kuNc1k174h3h3' # 16 block size

cipher = AES.new(key, AES.MODE_EAX, nonce)
data = cipher.decrypt(ciphertext)
try:
    cipher.verify(tag)
    print("Decrypted text :",data.decode())
except ValueError:
    print("Kunci salah atau data korup!")
```

```
Text :  Data ini akan di enkripsi!
#nonce :  b'y\x86#}\x93\x0e\xb9\xdbv\xd2\x14\x95\x03#V5'
#tag :  b'\xc5\xd0\xccL\xc8\x07\xab\x1e4\xc6\x1d{\xb2\xb5!\xb8'
#enkripsi :  b"\xe2\xb5EB\xcd.r\xc3\x1fa\xcb\xdb\x07\xab\xdf+^\xae'\x19\x8f\x0e\x8c\xd6\xcf\xab"
Decrypted text : Data ini akan di enkripsi!
```

Gambar 7. AES untuk kriptografi teks sederhana.

3. Bruce Schneier memperkenalkan algoritma simetris dalam blok, yang disebut sebagai Blowfish. Memiliki blok data tetap sebesar 8 byte dan memiliki panjang kunci (key) yang bervariasi, mulai dari 32 sampai 448 byte. Blowfish dipercaya sangat cepat dan aman. Namun, kunci yang dimiliki oleh algoritma ini cukup besar untuk menahan serangan secara *brute force*. Pada distribusi Cipher.4 Blowfish melakukanenkripsi yang hampir mirip DES, dengan panjang kunci yang bervariasi 32bit hingga 448bit (Utami & Tambunan, 2010). Istilah *Key-Expansion* pada Blowfish digunakan untuk melakukan *update* kunci yang dipisahkan pada tipe data array (subkey) dengan ukuran 4168 byte. Sedangkan pada proses enkripsi dilakukan iterasi sebuah fungsi yang dikenal sebagai *Feistel Network*. Iterasi berlangsung hingga 16 kali perulangan. Yang menjadi unik adalah pada setiap perulangan (putaran) permutasi kunci dan data dependent terjadi substitusi dengan seluruh operasinya adalah penambahan + XOR (32-bit). Untuk menerapkan Blowfish dapat digunakan script pada **Gambar 8.**

```python
from Crypto.Cipher import Blowfish
from struct import pack

key = b'KeyIniSangatlahPanjang'
data = b'Mari kita enkripsi!'

bs = Blowfish.block_size

cipher = Blowfish.new(key, Blowfish.MODE_CBC)

plen = bs - len(data) % bs

padding = [plen]*plen
padding = pack('b'*plen, *padding)
msg = cipher.iv + cipher.encrypt(data + padding)

print(msg)
```

```
b'\x0b\x80}]\xc2\x84\xd0\xf9\x98\x16\xed\x1bW\xf7\xdb\xaeq\x1c\xcd\x0b\x9bp3\t1\xa7#N8\xb1~\xea'
```

Gambar 8. Blowfish untuk enkripsi teks sederhana.

Community Service Article (COMERS)
e-ISSN: 3063-9425

Volume : 1 Number : 2 Year : 2024
website:https://ejournal.univbhaktiasih.ac.id/index.php/comers

4. RSA (Rivest–Shamir–Adleman) adalah algoritma untuk generate public key paling populer. Sering digunakan dalam mengamankan data konfidensial (enkripsi) dan autentikasi (digital signature). RSA juga menggunakan kedua private dan public key saat proses enkripsi dan dekripsi.

```python
from Crypto.PublicKey import RSA

key = RSA.generate(2048)
private_key = key.export_key()
file_out = open("private.pem", "wb")
file_out.write(private_key)
file_out.close()

public_key = key.publickey().export_key()
file_out = open("receiver.pem", "wb")
file_out.write(public_key)
file_out.close()
```

Gambar 9. Membuat *Public Key* dan *Private Key* pada RSA

Kode pada **Gambar 9**, menghasilkan *public key* yang disimpan di receiver.pem dan *private key* disimpan di private.pem. Setiap kali kode tersebut dieksekusi maka pasangan *public key* dan *private key* yang dihasilkan berbeda pula. File tersebut dipergunakan untuk mendekripkan pesan yang dienkripsi. Kode pada **Gambar 10**, memiliki fungsi untuk mengenkripsi data untuk penerima yang memiliki public key RSA-nya. Kunci publik RSA disimpan dalam file bernama receiver.pem.

```python
from Crypto.PublicKey import RSA
from Crypto.Random import get_random_bytes
from Crypto.Cipher import AES, PKCS1_OAEP

data = "I met aliens in UFO. Here is the map.".encode("utf-8")
file_out = open("encrypted_data.bin", "wb")

recipient_key = RSA.import_key(open("receiver.pem").read())
session_key = get_random_bytes(16)

# Encrypt the session key with the public RSA key
cipher_rsa = PKCS1_OAEP.new(recipient_key)
enc_session_key = cipher_rsa.encrypt(session_key)

# Encrypt the data with the AES session key
cipher_aes = AES.new(session_key, AES.MODE_EAX)
ciphertext, tag = cipher_aes.encrypt_and_digest(data)
[ file_out.write(x) for x in (enc_session_key, cipher_aes.nonce, tag, ciphertext) ]
file_out.close()
```

Gambar 10. Membuat *public key* dan *private key* pada RSA.

Agar dapat mengenkripsi data dalam jumlah berapa pun, digunakan skema enkripsi hibrid. Yakni menggunakan RSA dengan PKCS#1 OAEP untuk enkripsi asimetris kunci sesi AES. Kunci sesi kemudian dapat digunakan untuk mengenkripsi semua data sebenarnya. Seperti pada contoh pertama, mode EAX memungkinkan deteksi modifikasi data yang tidak sah. Kemudian *private key* RSA yang diterima digunakan untuk mendekripsi (lihat **Gambar 11**).

Community Service Article (COMERS)
e-ISSN: 3063-9425

Volume : 1 Number : 2 Year : 2024
website:https://ejournal.univbhaktiasih.ac.id/index.php/comers

```python
from Crypto.PublicKey import RSA
from Crypto.Cipher import AES, PKCS1_OAEP

file_in = open("encrypted_data.bin", "rb")

private_key = RSA.import_key(open("private.pem").read())

enc_session_key, nonce, tag, ciphertext = \
   [ file_in.read(x) for x in (private_key.size_in_bytes(), 16, 16, -1) ]
file_in.close()

# Decrypt the session key with the private RSA key
cipher_rsa = PKCS1_OAEP.new(private_key)
session_key = cipher_rsa.decrypt(enc_session_key)

# Decrypt the data with the AES session key
cipher_aes = AES.new(session_key, AES.MODE_EAX, nonce)
data = cipher_aes.decrypt_and_verify(ciphertext, tag)
print(data.decode("utf-8"))
```

```
I met aliens in UFO. Here is the map.
```

Gambar 11. Dekripsi dengan private key RSA

RSA memiliki keunggulan dari sisi kompleksitasnya, karena RSA bisa diibaratkan seperti koper yang akan dikirim, *private key* adalah kopernya dan *public key* adalah isi dari koper tersebut. Ketika koper sudah sampai, kunci yang digunakan untuk membuka dan menutup tetap sama. RSA menjadi pilihan diantara metode kriptografi yang diimplementasikan oleh penulis pada studi kasus penelitian sebelumnya (Malik & Baharsyah -13521029, 2022).

Tabel 1. Perbandingan konsumsi waktu.

Kriptografi	Cycle Time (Encrypt-Decrypt)
Asimetris – RSA	0 days 00:00:02.261766
Simetris – AES	0 days 00:00:00.003770
Simetris – DES	0 days 00:00:00.002990
Simetris – Blowfish	0 days 00:00:00.000996

Tabel 1. Memperlihatkan bahwa enkripsi asimetris RSA memiliki konsumsi waktu yang lebih tinggi. Namun hal tersebut sesuai dengan tingkat keamanan data yang jauh lebih baik (Sihotang *et al*., 2020) yakni tahapan generate *public key* dan *private key*, dan ini digenerate secara random dan unik, yang berarti tidak sama bagi setiap pemilik kunci private.

Selanjutnya, dengan implementasi RSA pada penelitian (Malik & Baharsyah -13521029, 2022), didapatkan konsumsi waktu “Total_Time used “ : 0:00:26.488808, diperlihatkan pada Gambar 12. Sementara itu script yang diusulkan penulis menggunakn PyCryptodome mendapatkan “Total_Time used “ : 0:00:01.157900 diperlihatkan pada **Gambar 13.**

Community Service Article (COMERS)
e-ISSN: 3063-9425

Volume : 1 Number : 2 Year : 2024
website:https://ejournal.univbhaktiasih.ac.id/index.php/comers

```
================================================================================================================
Simple Cryptographic Text using RSA base manual logic-formulate. By Agung Widyanto Universitas Bhakti Asih Tangerang
================================================================================================================

Generate PublicKey dan PrivateKey
Hasil nilai p dan q (bilangan prima):
p =  577
q =  599
Public Key:  (255575, 345623)
Private Key:  (287591, 345623)
====================================

#Teks yang akan dienkripsi#
SlavÃ¢â‚¬â„¢sya, Otechestvo nashe svobodnoye,

#Teks hasil enkripsi#
51316 121152 280355 319458 167691 319987 116234 117655 90550 116234 119718 319987 197799 133225 280355 57374 155008 22722 58025
19340 68383 27045 19340 197799 58025 319458 191074 155008 150048 280355 197799 27045 19340 155008 197799 319458 191074 3139 191
074 3226 150048 191074 133225 19340 57374

#Teks hasil deskripsi#
SlavÃ¢â‚¬â„¢sya, Otechestvo nashe svobodnoye,

Encrypt_Time used :  0:00:05.256987
Decrypt_Time used :  0:00:26.116804
Total_Time used   :  0:00:26.488808
```

Gambar 12. Informasi hasil menggunakan *script* pada penelitian sebelumnya.

```
================================================================================================================
Simple Cryptographic Text using RSA base PyCrytodome. By Agung Widyanto Universitas Bhakti Asih Tangerang
================================================================================================================

#Generate PublicKey dan PrivateKey#
Proses Generate Keys Selesai


#Teks to deliver#
SlavÃ¢â‚¬â„¢sya, Otechestvo nashe svobodnoye,

#Encrypt Text#
b'\xcb\xaa\xe1\x02\x957noN6\\k!LD\x15%\xe1Y\x86<-\x9d\\\xc6\x81\xae\xe3\xdc\xe1\xbe)#Z>.\x05\xf5\x0c\x99\xc6\xc1b`:!\xcar\xe0=
\xfc&\xc9y\xf4'

#Decrypt Text#
SlavÃ¢â‚¬â„¢sya, Otechestvo nashe svobodnoye,

Encrypt_Time used :  0:00:01.151917
Decrypt_Time used :  0:00:00.005983
Total_Time used   :  0:00:01.157900
```

Gambar 13. Informasi hasil menggunakan *script* dengan PyCrytodome.

Pengujian dilanjutkan dengan menggunakan 4 file yang berisi pesan teks dengan kombinasi pengaturan dan jumlah karakter. Hasilnya diperlihatkan pada **Tabel 2**.

Tabel 2. Perbandingan konsumsi waktu penelitian sebelumnya dan saat Ini.

Kriptografi	**File**	***∑ Character***	***Time use***		
			Encrypt	***Decrypt***	**Total**
Penelitian Sebelumnya (Malik & Baharsyah - 13521029, 2022)	File -1	153	0:00:02.558661	0:00:53.110399	0:00:53.188279
	File -2	313	0:00:02.637694	0:01:37.822257	0:01:38.009155
	File -3	2801	0:00:00.418253	0:00:01.337662	0:00:01.343858
	File -4	19621	0:00:03.289583	0:00:04.107728	0:00:04.120109
Penulis (menerapkan PyCryptodome)	File -1	153	0:00:00.484330	0:00:00.006527	0:00:00.490857
	File -2	313	0:00:00.299440	0:00:00.007014	0:00:00.306454
	File -3	2801	0:00:00.806378	0:00:00.005410	0:00:00.811788
	File -4	19621	0:00:01.512883	0:00:00.007053	0:00:01.520530

Tabel 2. Memperlihatkan konsumsi waktu yang lebih efisien dengan menerapkan

Community Service Article (COMERS)
e-ISSN: 3063-9425

Volume : 1 Number : 2 Year : 2024
website:https://ejournal.univbhaktiasih.ac.id/index.php/comers

PyCryptodome. Jika konsumsi waktunya dipisahkan masing-masing segmen, tahap *encrypt* dan *decrypt* tetap lebih unggul dengan menerapkan PyCryptodome, yakni pada File-1 mendahului 50 detik lebih cepat, pada File-2 selisih waktu 38 detik lebih cepat, pada File-3 dengan selisih waktu 1 detik lebih cepat, dan pada File-4 lebih cepat 3 detik. Hal ini membuktikan, implementasi kernel PyCryptodome untuk melakukan kriptografi RSA memberikan pengaruh perbaikan proses waktu yang lebih cepat dibandingkan dengan penelitian sebelumnya.

Mengapa lebih lambat pada penelitian sebelumnya, hal ini dipengaruhi oleh beberapa faktor dalam alur proses RSA yakni pada proses pembuatan kunci, enkripsi, dan deskripsi. Pada tahapan pembuatan kunci, telah dijelaskan diawal bahwa penggunaan faktorisasi akan menjadi lebih komplek dan membutuhkan waktu lebih lama jika bilangan bulatnya semakin besar. Kemudian enkripsi dan dekripsi pesan juga akan terkena imbasnya pada sisi konsumsi waktu, karena melibatkan pengkodean untuk menghitung chipertext secara matematis dengan *public key* dan proses ini berjalan serial. Ilustrasi sederhananya dapat digambarkan ketika Bejo ingin mengirimkan pesan "Z" kepada Untung, maka setelah dilakukan pembuatan kunci, proses selanjutnya adalah mengirimkan *public key* (n,e) nya untuk Untung, dan menyimpan secara rahasia private key-nya. Pesan “Z” diubah menjadi ascii-code dan selanjutnya ciphertext "c" (nilai yang telah terenkripsi) dihitung dengan menggunakan public key yang dikirimkan oleh Untung kepada Bejo. Dan sekaligus Bejo merespon dengan mengirimkan nilai “c” kepada Untung (proses decrypt) dengan menggunakan private-key miliknya. Pada proses ini (*decrypt*), nilai “Z” diganti dengan nilai c *ciphertext* (enkripsi) dan nilai e public key diganti nilai d dari private key. Hanya nilai n dari *public key* akan selalu identik nilai n dari private key -nya.

Dengan perhitungan tersebut, sudah dapat mengimplementasikan *Private* dan *Public Key* sebagai sarana untuk melakukan enkripsi dengan menggunakan alghoritma RSA, dimana Budi melakukan enkripsi data M=77 dengan *public key* dan mendapatkan nilai c=3123, kemudian mengirimkannya kepada Andi untuk di dekripsi dengan menggunakan *private key* dan mendapatkan data yang sama dengan yang dimaksudkan oleh Budi, yaitu M2=77. Proses perhitungan, untuk menghasilkan *public key* dan *private key* menjadi *bottle-neck* setiap kali program dijalankan. Script yang digunakan belum dilakukan optimasi, contohnya belum melibatkan kernel publik untuk mendapatkan bilangan prima dan faktorial yang terbaik dengan memperhitungkan jumlah karakter pesan.

4. KESIMPULAN

Berdasarkan implementasi dan pengujian pada variasi pesan maka dapat disimpulkan bahwa konsumsi waktu yang dihasilkan adalah lebih baik dengan menerapkan PyCryptodome seperti yang ditampilkan pada hasil pengujian di tabel konsumsi waktu tiap segmen pada tahap *encrypt* dan *decrypt* di dominasi waktu yang lebih cepat dengan PyCryptodome.

Penulis merekomendasikan eksplorasi pada penelitian selanjutnya untuk mengejawentahkan purwarupa pada *script* yang telah dihasilkan pada lintas sistem operasi menggunakan PyCrytodome.

Community Service Article (COMERS)
e-ISSN: 3063-9425

Volume : 1 Number : 2 Year : 2024
website:https://ejournal.univbhaktiasih.ac.id/index.php/comers

PUSTAKA

Adhar, D. (2019). Implementasi Algoritma DES (Data Encryption Standard) Pada Enkripsi Dan Deskripsi SMS Berbasis Android. *JTIK (Jurnal Teknik Informatika Kaputama)*, *3*(2), 53–60.

Anoop, M. S. (2007). Public Key Cryptography. *Retrieved October*, *6*, 2010.

Ariyus, D. (2006). Kriptografi keamanan data dan komunikasi. *Yogyakarta: Graha Ilmu*.

Fachri, B., & Sembiring, R. M. (2020). Pengamanan Data Teks Menggunakan Algoritma DES Berbasis Android. *JURNAL MEDIA INFORMATIKA BUDIDARMA*, *4*(1), 110–116.

Galla, L. K. , K. V. S. , & N. N. (2016, D. I. of RSA. I. 2016 I. C. on C. I. C. and C. T. (ICCICCT) (pp. 81-87). IEEE. (2016). 2016 International Conference on Control, Instrumentation, Communication and Computational Technologies : ICCICCT-2016 : 16 & 17 December 2016. *2016 International Conference on Control, Instrumentation, Communication and Computational Technologies : ICCICCT-2016 : 16 & 17 December 2016*.

Indra, Z., & Cyra Nabila, R. (2023). Implementation of the RSA Algorithm and the One Time Pad Algorithm for Text Message Security. *Formosa Journal of Science and Technology (FJST)*, *2*(1), 379. https://doi.org/10.55927/fjst.v2i1.2999

Malik, M., & Baharsyah -13521029, I. (2022). *Makalah IF2120 Matematika Diskrit-Sem. I Tahun*.

Meko, D. A. (2018). Perbandingan Algoritma DES, AES, IDEA Dan Blowfish dalam Enkripsi dan Dekripsi Data. *Jurnal Teknologi Terpadu*, *4*(1).

Munir, R. (2006). Kriptografi. bandung. *Informatika*, *1*(7).

Rahmawati, M. S., & Soekarta, R. (2018). Teori Grup Pada Algoritma DES Dan Transformasi Wavelet Diskrit Dalam Program Aplikasi Keamanan Citra Digital. *Insect (Informatics and Security): Jurnal Teknik Informatika*, *4*(1), 1–6.

Satria, M., Rosnelly, R., & Nurhayati, N. (2020). Perancangan Aplikasi Keamanan Data Dokumen Word dengan Menggunakan Algoritma Triple DES. *Jurnal Mahasiswa Fakultas Teknik Dan Ilmu Komputer*, *1*(1), 463–475.

Sihotang, H. T., Efendi, S., Zamzami, E. M., & Mawengkang, H. (2020). Design and Implementation of Rivest Shamir Adleman's (RSA) Cryptography Algorithm in Text File Data Security. *Journal of Physics: Conference Series*, *1641*(1). https://doi.org/10.1088/1742-6596/1641/1/012042

Syahputri, N. (2019). RANCANG BANGUN APLIKASI KRIPTOGRAFI PENGAMANAN TRANSMISI DATA MULTIMEDIA MENGGUNAKAN ALGORITMA DATA ENCRYPTION STANDARD (DES). *Majalah Ilmiah METHODA*, *9*(2), 57–63.

Utami, E., & Tambunan, S. E. A. (2010). Penerapan Algoritma Blowfish Untuk Membuat Sebuah Model Kriptosistem Algoritma Dan Menganalisis Kinerja Algoritma Blowfish Dengan Simulasi Data Terbatas. *Data Manajemen Dan Teknologi Informasi (DASI)*, *11*(2), 33.