MATHEMATICAL SCIENCES AND APPLICATIONS JOURNAL

Volume 5, Number 1 | ISSN: 2986-4180

https://online-journal.unja.ac.id/msa

Article

Kombinasi Algoritma Sandi Caesar dan Algoritma RSA untuk Pengamanan Pesan Teks

Alifa Raida Alamsyah, Edi Kurniadi, Anita Triska, dan Sisilia Sylviani*

Program Studi Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Padjadjaran, Sumedang, Jawa Barat 45363, Indonesia

*Corresponding author e-mail: sisilia.sylviani@unpad.ac.id

Article Info

Received October 2024
Accepted December 2024
Published December 2024

Keyword:
Number theory
Cryptography
Caesar cipher
RSA algorithm

Abstract

This article combines a simple cryptographic algorithm, Caesar Cipher, with a more complex algorithm, RSA, in order to increase the security of encrypted text messages. Text messages are first encrypted with the Caesar Cipher algorithm, which is then re-encrypted using the RSA algorithm. By utilizing number theory, specifically about integers and modulo arithmetic in the RSA algorithm, a public key and a secret key are obtained that will increase the security of the encryption process in this article. Due to the increased security of the text message, uninvolved parties cannot read the actual text message.

1. Pendahuluan

Beberapa penelitian terkait kriptografi telah dilakukan, khususnya dalam menggabungkan metode kriptografi yang sederhana, seperti Sandi Caesar dengan metode yang lebih kompleks seperti Algoritma RSA dengan tujuan untuk meningkatkan keamanan pesan yang ingin dienkripsi. Misalnya, penelitian yang dilakukan oleh [1] menggunakan Sandi Caesar untuk proses enkripsi pertama yang selanjutnya dienkripsi kembali menggunakan Algoritma RSA dalam mengenkripsi surat wasiat menggunakan bahasa pemrograman SQL. Ada pula penelitian yang dilakukan oleh [2] menggunakan Sandi Caesar untuk proses enkripsi pertama yang selanjutnya dienkripsi kembali menggunakan Algoritma RSA dalam mengenkripsi berkas dokumen dan pesan. Dalam penelitian tersebut, enkripsi berkas dan dokumen dienkripsi dan didekripsi menggunakan bahasa pemrograman Pascal. Oleh karena itu, proses enkripsi dan dekripsi dari algoritma Sandi Caesar dan RSA juga menarik untuk disimulasikan menggunakan bahasa pemrograman Python yang merupakan bahasa pemrograman yang mudah digunakan dan bersifat universal. Dengan demikian, dalam artikel ini dibahas penggunaaan algoritma Sandi Caesar dan algoritma RSA dengan cara manual serta disimulasikan menggunakan bahasa pemrograman Python.

Teori bilangan merupakan sebuah cabang ilmu matematika murni yang menyelidiki bilangan bulat serta fungsinya. Teori bilangan juga merupakan ilmu dasar dari setiap teori yang ada, karena pada setiap teori, minimal memiliki satu jenis bilangan [3]. Salah satu penerapan dari teori bilangan adalah kriptografi. Kriptografi merupakan ilmu untuk menjaga kerahasiaan dari suatu pesan dengan cara menyandikan pesan ke dalam sebuah bentuk yang tidak dapat dimengerti lagi maknanya [1, 4–7].

Terdapat beberapa istilah yang sering digunakan dalam kriptografi. *Plaintext* merupakan pesan asli yang ingin dikirim dan berisikan informasi untuk dibaca. *Ciphertext* merupakan data yang sudah dienkripsi dan tidak dapat dibaca secara langsung. Enkripsi adalah proses untuk menyembunyikan pesan. Proses enkripsi merubah pesan *plaintext* menjadi *ciphertext* menggunakan suatu kunci. Kunci merupakan suatu parameter yang digunakan dalam proses enkripsi dan dekripsi. Kunci yang digunakan dapat berbentuk apapun, baik abjad, bilangan, atau bit i [1, 8–10].

Dua jenis algoritma kriptografi adalah algoritma simetri dan algoritma asimetri. Algoritma simetri biasa disebut kriptografi kunci rahasia, karena kunci yang digunakan rahasia dan kunci tersebut digunakan untuk

https://doi.org/10.22437/msa.v5i1.38104

proses enkripsi dan dekripsi. Algoritma asimetri biasa disebut kriptografi kunci publik, karena kunci yang digunakan untuk proses enkripsi dan dekripsi berbeda, untuk melakukan proses enkripsi menggunakan kunci yang bersifat publik dan untuk melakukan proses dekripsi menggunakan kunci yang bersifat rahasia [4, 5, 7, 11, 12].

Sandi Caesar berasal dari nama seorang kaisar romawi, Julius Caesar. Sandi Caesar merupakan sebuah metode kriptografi yang sederhana, di mana prosesnya adalah menggeser karakter pada *plaintext* sebanyak tiga posisi setelahnya dalam urutan kode ASCII. Sandi Caesar termasuk ke dalam algoritma simetris, di mana kunci untuk proses enkripsi dan dekripsinya sama. Sehingga, semua karakter pada *plaintext* dan *ciphertext* digeser sebanyak posisi yang sama [13–17].

Algoritma RSA (Rivest Shamir Adleman) merupakan salah satu dari kriptografi sandi publik yang populer digunakan untuk merahasiakan suatu pesan. Algoritma RSA menggunakan dua kunci, yaitu kunci rahasia dan kunci publik. Konsep bilangan prima dan aritmetika modulo menjadi dasar dari proses enkripsi dan dekripsi pada algoritma RSA. Kunci yang digunakan pada algoritma RSA berbentuk bilangan bulat. Kunci yang digunakan untuk proses enkripsi adalah kunci publik, sedangkan kunci yang digunakan untuk proses dekripsi adalah kunci rahasia. Kunci rahasia diperoleh dengan cara memfaktorkan bilangan bulat menjadi faktor-faktor primanya [18–20].

Artikel ini mengombinasikan dua algoritma pada kriptografi, yaitu algoritma Sandi Caesar dan algoritma RSA, dengan tujuan untuk meningkatkan keamanan pesan teks yang akan dienkripsi agar pihak-pihak yang tidak berkepentingan tidak dapat membaca pesan teks yang diberikan. Dalam artikel ini, selain menggunakan cara manual, bahasa pemrograman Python juga akan digunakan dalam membantu perhitungan dalam proses enkripsi dan dekripsi sehingga perhitungan akan menjadi lebih mudah dan efisien.

2. Metode Penelitian

2.1. Bilangan Bulat

Bilangan $\dots, -3, -2, -1, 0, 1, 2, \dots$ dikatakan bilangan bulat; bilangan $0, 1, 2, 3, \dots$ dikatakan bilangan bulat non-negatif; dan bilangan $1, 2, 3, \dots$ dikatakan bilangan bulat positif.

Jika terdapat a dan b yang merupakan dua buah bilangan bulat dengan syarat $a \neq 0$. Dikatakan bahwa a habis membagi b (a divides b) jika terdapat bilangan bulat c sedemikian sehingga $b = ac$ [21].

Dinotasikan oleh $a|b$ jika $b = ac, c \in \mathbb{Z}$ dan $a \neq 0$. Pernyataan "a habis membagi b" dapat ditulis juga "b kelipatan a" [21].

Teorema 1. [21] Diberikan bilangan bulat a dan b, dengan $b \neq 0$, maka terdapat bilangan bulat q dan r sedemikian sehingga

$$a = qb + r \qquad 0 \leq r < |b|$$

Bilangan bulat q dan r masing-masing disebut hasil bagi dan sisa dari a oleh b.

Teorema 2. [21] Jika a, b, dan c semuanya bilangan bulat, c dikatakan sebagai pembagi bersama terbesar (gcd) dari a dan b, jika c adalah bilangan bulat terbesar sehingga c habis membagi a dan c juga habis membagi b atau dapat ditulis,

$$\gcd(a, b) = c$$

dengan syarat $c|a$ dan $c|b$.

2.2. Bilangan Prima

Bilangan bulat positif $a > 1$ dikatakan bilangan prima, jika pembaginya hanyalah 1 dan a. Bilangan bulat yang lebih besar dari 1 dan bukan bilangan prima dikatakan bilangan komposit [21].

2.3. Relatif Prima

Misalkan a dan b keduanya bilangan bulat. Bilangan a dan b dikatakan relatif prima jika dan hanya jika 1 adalah pembagi bersama terbesar dari a dan b [21], dinotasikan sebagai

$$\gcd(a, b) = 1$$

atau

$$x \cdot a + y \cdot b = 1$$

dengan x dan y keduanya bilangan bulat.

Misalkan a dan b relatif prima, maka terdapat m dan n yang keduanya bilangan bulat sedemikian sehingga

$$ma + nb = 1.$$

2.4. Aritmetika Modulo

Misalkan a dan m keduanya bilangan bulat dan $m > 0$, operasi modulo $a \bmod m$ akan menghasilkan sisa (*remainder*) dari a dibagi oleh m [21], dinotasikan dengan $a \bmod m = r$ sedemikian sehingga

$$a = m \cdot q + r$$

dengan

$$0 \leq r < m.$$

2.5. Kongruen

Misalkan a dan b keduanya bilangan bulat dan $m > 0$ sehingga berlaku $a \bmod m = c$ dan $b \bmod m = c$, artinya a kongruen dengan b dalam modulo m, dinotasikan sebagai:

$$a \equiv b(mod\ m).$$

Kekongruenan $a \equiv b(mod\ m)$ juga dapat dinotasikan dalam bentuk

$$a = b + km$$

dengan k merupakan bilangan bulat [21].

2.6. Kekongruenan Lanjar

Kekongruenan lanjar dinotasikan

$$ax \equiv b(mod\ m)$$

Dengan a dan b keduanya bilangan bulat, $m > 0$, dan x merupakan peubah bilangan bulat, yang dicari menggunakan persamaan

$$ax = b + km.$$

Persamaan di atas dapat diubah menjadi

$$x = \frac{b + km}{a}$$

dengan k adalah sembarang bilangan bulat. Lakukan pengujian untuk k bilangan bulat sedemikian sehingga diperoleh x yang merupakan bilangan bulat [21].

2.7. Algoritma Sandi Caesar

Pada Sandi Caesar, setiap karakter pada *plaintext* akan digeser ke karakter ketiga berikutnya dari susunan ASCII. Pada Sandi Caesar, kuncinya adalah 3, yang merupakan banyaknya pergeseran pada karakter [22].

Algoritma Sandi Caesar untuk ASCII karakter adalah:

1. Proses enkripsi

$$c_i = E(p_i) = (p_i + 3)\ mod\ 256.$$

2. Proses dekripsi

$$p_i = D(c_i) = (c_i - k)\ mod\ 256.$$

2.8. Algoritma RSA

Proses pembentukan kunci dari algoritma sandi RSA adalah:

1. Pilih dua bilangan prima sembarang sebagai p dan q, di mana $p \neq q$.
2. Hitung nilai $n = p \cdot q$.
2 Hitung nilai $m = (p - 1)(q - 1)$.
3 Pilih kunci publik yang dinotasikan dengan e, di mana e relatif prima dengan m.
4 Peroleh kunci untuk proses dekripsi dengan menghitung kongruensi $e \cdot d \equiv 1\ (mod\ n)$. Ingat bahwa $e \cdot d \equiv 1\ (mod\ n)$ ekivalen dengan $e \cdot d = 1 + kn$, maka d dapat diperoleh dengan menghitung $d = \frac{1+kn}{e}$.

Algoritma tersebut menghasilkan:

1. Kunci proses enkripsi (e, n).
2. Kunci proses dekripsi (d, n).

Kunci proses enkripsi disebut dengan kunci publik, sedangkan kunci proses dekripsi disebut dengan kunci rahasia [18]. Algoritma RSA untuk ASCII karakter adalah:

1. Proses enkripsi

$$c_i = E(p_i) = p_i^e\ mod\ n.$$

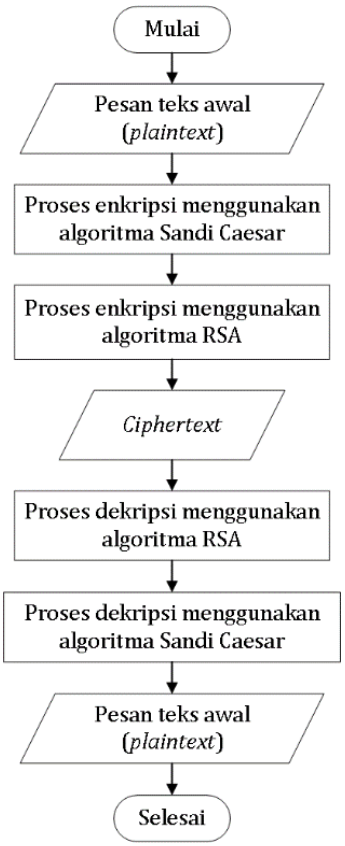
2. Proses dekripsi

$$p_i = D(c_i) = c_i^d\ mod\ n.$$

2.9. ASCII

American Standard Code for Information Interchange (ASCII) adalah kode standar yang digunakan untuk melakukan pertukaran informasi pada komputer. ASCII berjumlah sebanyak 256 karakter, dengan urutan karakter 0 sampai 127 adalah karakter yang digunakan untuk memanipulasi teks, sedangkan karakter 128 sampai dengan 255 digunakan untuk memanipulasi grafik [23]. ASCII akan digunakan sebagai acuan kode dalam proses enkripsi dan dekripsi yang digunakan pada artikel ini.

Diperlukan proses untuk melakukan enkripsi dan dekripsi dengan kombinasi Algoritma Sandi Caesar dan RSA pada pesan teks. Berikut diagram alir dan rincian dari proses tersebut (Gambar 1).



Gambar 1. Diagram alir penelitian

3. Hasil dan Pembahasan

Bagian ini akan berisikan penjelasan mengenai proses enkripsi dan dekripsi menggunakan kombinasi algoritma Sandi Caesar dan algoritma RSA. Misalkan, diberikan *plaintext* "Kriptografi 2024!" yang mengandung huruf kapital, huruf kecil, spasi, angka, dan simbol tanda seru.

3.1. Enkripsi

Pada bagian ini, akan dilakukan proses enkripsi dua tahap pada *plaintext* "Kriptografi 2024!". Pertama-tama, *plaintext* akan dienkripsi menggunakan algoritma Sandi Caesar yang menggunakan kunci $k = 3$ dengan modulo 256. Konversi karakter pada *plaintext* ke kode ASCII lalu lakukan proses enkripsi.

- K = 75, maka $E(75) = (75 + 3)\ mod\ 256 = 78 =$ N
- r = 114, maka $E(114) = (114 + 3)\ mod\ 256 = 117 =$ u
- i = 105, maka $E(105) = (105 + 3)\ mod\ 256 = 108 =$ l
- p = 112, maka $E(112) = (112 + 3)\ mod\ 256 = 115 =$ s
- t = 116, maka $E(116) = (116 + 3)\ mod\ 256 = 119 =$ w
- o = 111, maka $E(111) = (111 + 3)\ mod\ 256 = 114 =$ r
- g = 103, maka $E(103) = (103 + 3)\ mod\ 256 = 106 =$ j
- r = 114, maka $E(114) = (114 + 3)\ mod\ 256 = 117 =$ u
- a = 97, maka $E(97) = (97 + 3)\ mod\ 256 = 100 =$ d
- f = 102, maka $E(102) = (102 + 3)\ mod\ 256 = 105 = i$
- i = 105, maka $E(105) = (105 + 3)\ mod\ 256 = 108 =$ l
- Spasi = 32, maka $E(32) = (32 + 3)\ mod\ 256 = 35 =$ #
- 2 = 50, maka $E(50) = (50 + 3)\ mod\ 256 = 53 =$ 5
- 0 = 48, maka $E(48) = (48 + 3)\ mod\ 256 = 51 =$ 3
- 2 = 50, maka $E(50) = (50 + 3)\ mod\ 256 = 53 =$ 5
- 4 = 52, maka $E(52) = (52 + 3)\ mod\ 256 = 55 =$ 7
- ! = 33, maka $E(33) = (33 + 3)\ mod\ 256 = 36 =$ $

Dari proses enkripsi menggunakan algoritma Sandi Cipher, diperoleh *ciphertext*-nya adalah "Nulswrjudil#5357$". Selanjutnya, akan dilakukan enkripsi menggunakan algoritma RSA. *Plaintext* yang digunakan pada proses enkripsi ini adalah *ciphertext* yang dihasilkan dari proses enkripsi menggunakan Sandi Caesar, yaitu "Nulswrjudil#5357$".

Pilih $p = 29, q = 5$ di mana $p \neq q$. Hitung $n = 29 \cdot 5 = 145$ dan $m = (29 - 1)(5 - 1) = 112$. Pilih $e = 89$, di mana $\gcd(89, 112) = 1$. Hitung nilai d yang memenuhi $89 \cdot d \equiv 1\ (mod\ 145)$, diperoleh $d = 73$. Akan dilakukan proses enkripsi menggunakan algoritma RSA dengan kunci publik yang diperoleh yaitu $(103, 145)$.

- N = 78, maka $E(78) = 78^{89}\ mod\ 145 = 53$
- u = 117, maka $E(117) = 117^{89}\ mod\ 145 = 117$
- l = 108, maka $E(108) = 108^{89}\ mod\ 145 = 118$
- $s = 115$, maka $E(115) = 115^{89}\ mod\ 145 = 115$
- w = 119, maka $E(119) = 119^{89}\ mod\ 145 = 69$
- r = 114, maka $E(114) = 114^{89}\ mod\ 145 = 84$
- j = 106, maka $E(106) = 106^{89}\ mod\ 145 = 21$
- $u = 117$, maka $E(117) = 117^{89}\ mod\ 145 = 117$
- d = 100, maka $E(100) = 100\ mod\ 145 = 35$
- i = 105, maka $E(105) = 105^{89}\ mod\ 145 = 15$
- l = 108, maka $E(108) = 108^{89}\ mod\ 145 = 118$
- # = 35, maka $E(35) = 35^{89}\ mod\ 145 = 120$
- 5 = 53, maka $E(53) = 53^{89}\ mod\ 145 = 123$
- 3 = 51, maka $E(51) = 51^{89}\ mod\ 145 = 71$
- 5 = 53, maka $E(53) = 53^{89}\ mod\ 145 = 123$
- 7 = 55, maka $E(55) = 55^{89}\ mod\ 145 = 105$

- $\$ = 36$, maka $E(36) = 36^{89}\ mod\ 145 = 16$

Dari proses enkripsi menggunakan algoritma RSA, diperoleh *ciphertext*-nya adalah $(53, 117, 118, 115, 69, 84, 21, 117, 35, 15, 118, 120, 123, 71, 123, 105, 16)$. Jadi, pesan awal yang bertuliskan "Kriptografi 2024!", setelah dienkripsi berubah menjadi $(53, 117, 118, 115, 69, 84, 21, 117, 35, 15, 118, 120, 123, 71, 123, 105, 16)$.

Proses enkripsi juga telah disimulasikan menggunakan bahasa pemrograman Python, yang terlampir pada Gambar 2 dan 3.

```
[ ]  def enkripsi_Caesar():
       # Input plainteks
       plainteks = input("Masukkan plainteks : ")

       # String kosong untuk menyimpan cipherteks
       cipherteks = ""

       # Loop untuk setiap karakter(pi) dalam plainteks
       for char in plainteks:
           # Tentukan pi sebagai indeks karakter dalam ASCII
           pi = ord(char)
           # Melakukan enkripsi
           enkripsi_karakter = ((pi + 3) % 256)
           # Tambahkan karakter yang telah dienkripsi ke cipherteks
           cipherteks += chr(enkripsi_karakter)

       # Print cipherteks
       print("Cipherteks:", cipherteks)
       return cipherteks

[ ]  cipherteks_caesar = enkripsi_Caesar()

     Masukkan plainteks : Kriptografi 2024!
     Cipherteks: Nulswrjudil#5357$
```

Gambar 2. Proses enkripsi menggunakan algoritma Sandi Caesar

```
[ ]  # Fungsi enkripsi RSA
     def enkripsi_RSA(plainteks,public_key):
       print("Plainteks:",plainteks)
       e, n = public_key
       cipherteks = [(ord(char)**e) % n for char in plainteks]
       print("Kunci Publik:",public_key)
       print("Cipherteks:",cipherteks)
       return cipherteks

[ ]  cipherteks_RSA = enkripsi_RSA(cipherteks_caesar, public_key)

     Plainteks: Nulswrjudil#5357$
     Kunci Publik: (89, 145)
     Cipherteks: [53, 117, 118, 115, 69, 84, 21, 117, 35, 15, 118, 120, 123, 71, 123, 105, 16]
```

Gambar 3. Proses enkripsi menggunakan algoritma RSA

3.2. Dekripsi

Pada Pada bagian ini, akan dilakukan proses dekripsi dua tahap pada *ciphertext* $(53, 117, 118, 115, 69, 84, 21, 117, 35, 15, 118, 120, 123, 71, 123, 105, 16)$ agar kembali menjadi pesan awal yaitu "Kriptografi 2024!". Pertama-tama, *ciphertext* akan didekripsi menggunakan algoritma RSA dengan kunci rahasia yang diperoleh yaitu $(73, 145)$.

- $c_1 = 53$, maka $D(53) = 53^{73}\ mod\ 145 = 78 = \text{N}$
- $c_2 = 117$, maka $D(117) = 117^{73}\ mod\ 145 = 117 = \text{u}$
- $c_1 = 118$, maka $D(118) = 118^{73}\ mod\ 145 = 108 = \text{l}$
- $c_2 = 115$, maka $D(115) = 115^{73}\ mod\ 145 = 115 = \text{s}$
- $c_1 = 69$, maka $D(69) = 69^{73}\ mod\ 145 = 119 = \text{w}$
- $c_2 = 84$, maka $D(84) = 84^{73}\ mod\ 145 = 114 = \text{r}$
- $c_1 = 21$, maka $D(21) = 21^{73}\ mod\ 145 = 106 = \text{j}$
- $c_2 = 117$, maka $D(117) = 117^{73}\ mod\ 145 = 117 = \text{u}$
- $c_1 = 35$, maka $D(35) = 35^{73}\ mod\ 145 = 100 = \text{d}$
- $c_2 = 15$, maka $D(15) = 15^{73}\ mod\ 145 = 105 = \text{i}$
- $c_1 = 118$, maka $D(118) = 118^{73}\ mod\ 145 = 108 = \text{l}$
- $c_2 = 120$, maka $D(120) = 120^{73}\ mod\ 145 = 35 = \#$
- $c_1 = 123$, maka $D(123) = 123^{73}\ mod\ 145 = 53 = 5$
- $c_2 = 71$, maka $D(71) = 71^{73}\ mod\ 145 = 51 = 3$
- $c_1 = 123$, maka $D(123) = 123^{73}\ mod\ 145 = 53 = 5$
- $c_2 = 105$, maka $D(105) = 105^{73}\ mod\ 145 = 55 = 7$
- $c_1 = 16$, maka $D(16) = 16^{73}\ mod\ 145 = 36 = \$$

Dari proses dekripsi menggunakan algoritma RSA, diperoleh *plaintext*-nya adalah "Nulswrjudil#5357$". Selanjutnya, akan dilakukan dekripsi menggunakan algoritma Sandi Caesar. *Ciphertext* yang digunakan pada proses dekripsi ini adalah *plaintext* yang dihasilkan dari proses dekripsi menggunakan algoritma RSA, yaitu "Nulswrjudil#5357$".

- $\text{N} = 78$, maka $D(78) = (78 - 3)\ mod\ 256 = 75 = \text{K}$
- $\text{u} = 117$, maka $D(117) = (117 - 3)\ mod\ 256 = 114 = \text{r}$
- $\text{l} = 108$, maka $D(108) = (108 - 3)\ mod\ 256 = 105 = \text{i}$
- $\text{s} = 115$, maka $D(115) = (115 - 3)\ mod\ 256 = 112 = \text{p}$
- $\text{w} = 119$, maka $D(119) = (119 - 3)\ mod\ 256 = 116 = \text{t}$
- $\text{r} = 114$, maka $D(114) = (114 - 3)\ mod\ 256 = 111 = \text{o}$
- $\text{j} = 106$, maka $D(106) = (106 - 3)\ mod\ 256 = 103 = \text{g}$
- $\text{u} = 117$, maka $D(117) = (117 - 3)\ mod\ 256 = 114 = \text{r}$

- d = 100, maka $D(100) = (100 - 3)\ mod\ 256 = 97$ = a
- i = 105, maka $D(105) = (105 - 3)\ mod\ 256 = 102$ = f
- l = 108, maka $D(108) = (108 - 3)\ mod\ 256 = 105$ = i
- # = 35, maka $D(35) = (35 - 3)\ mod\ 256 = 32$ = spasi
- 5 = 53, maka $D(53) = (53 - 3)\ mod\ 256 = 50$ = 2
- 3 = 51, maka $D(51) = (51 - 3)\ mod\ 256 = 48$ = 0
- 5 = 53, maka $D(53) = (53 - 3)\ mod\ 256 = 50$ = 2
- 7 = 55, maka $D(55) = (55 - 3)\ mod\ 256 = 52$ = 4
- $ = 36, maka $D(36) = (36 - 3)\ mod\ 256 = 33$ = !

Dari proses dekripsi menggunakan algoritma Sandi Caesar, diperoleh *plaintext*-nya adalah "Kriptografi 2024!" sama seperti *plaintext* yang diberikan. Jadi, dapat dilihat dari proses enkripsi dan dekripsi menggunakaan kombinasi algoritma Sandi Caesar dan RSA, pesan awal dapat dienkripsi menjadi sebuah kode dan dapat didekripsi kembali menjadi pesan awal yang diberikan sebelumnya. Dapat disimpulkan bahwa proses enkripsi dan dekripsi dilakukan dengan benar.

Proses dekripsi juga telah disimulasikan menggunakan bahasa pemrograman Python, yang terlampir pada Gambar 4 dan 5.

4. Kesimpulan

Algoritma Sandi Caesar dan algoritma RSA dapat dikombinasikan untuk mengamankan sebuah pesan teks sehingga kerahasiaannya lebih terjaga. Hasil yang diperoleh dari proses enkripsi kedua algoritma juga cukup sulit untuk dipecahkan oleh pihak-pihak yang tidak terkait/tidak memiliki kepentingan. Dalam melakukan proses enkripsi dan dekripsi telah menggunakan bantuan bahasa pemrograman Python, sehingga pihak-pihak yang berkepentingan dapat melakukan proses enkripsi dan dekripsi tanpa harus menghitung secara manual. Pemrograman yang dibuat untuk artikel ini masih bisa dikembangkan secara lebih lanjut agar bisa melakukan proses enkripsi dan dekripsi pesan yang lebih panjang ataupun pesan berbentuk dokumen agar lebih efisien.

```
[ ]  # Fungsi dekripsi RSA
     def dekripsi_RSA(cipherteks, private_key):
       print("Cipherteks:",cipherteks)
       d, n = private_key
       print(d)
       plainteks = ''.join([chr((num ** d) % n) for num in cipherteks])
       print("Kunci Rahasia:",private_key)
       print("Plainteks:",plainteks)
       return plainteks

[ ]  plainteks_RSA = dekripsi_RSA(cipherteks_RSA, private_key)

     Cipherteks: [53, 117, 118, 115, 69, 84, 21, 117, 35, 15, 118, 120, 123, 71, 123, 105, 16]
     73
     Kunci Rahasia: (73, 145)
     Plainteks: Nulswrjudil#5357$
```

Gambar 4. Proses dekripsi menggunakan algoritma RSA

```
[ ]  def deskripsi_Caesar(cipherteks):
       print("Cipherteks:",cipherteks)
       # String kosong untuk menyimpan cipherteks
       plainteks = ""

       # Loop untuk setiap karakter(ci) dalam cipherteks
       for char in cipherteks:
           # Tentukan ci sebagai indeks karakter dalam ASCII
           ci = ord(char)
           deskripsi_karakter = (ci - 3) % 256
           # Tambahkan karakter yang telah dienkripsi ke plainteks
           plainteks += chr(deskripsi_karakter)

       # Print plainteks
       print("Plainteks:", plainteks)
       return plainteks

[ ]  plainteks_caesar = deskripsi_Caesar(plainteks_RSA)

     Cipherteks: Nulswrjudil#5357$
     Plainteks: Kriptografi 2024!
```

Gambar 5. Proses dekripsi menggunakan algoritma Sandi Caesar

Daftar Pustaka

1. Silalahi, R., Parlina, I., Sumarno, S., Gunawan, I., & Saputra, W. 2021. Implementasi Algoritma Caesar Cipher dan Algoritma RSA untuk Keamanan Data Surat Wasiat pada Kantor Notaris/PPAT Robert Tampubolon, S.H. *Jurnal Sosial Teknologi*, *1*(4), 282–293. https://doi.org/10.59188/jurnalsostech.v1i4.64.
2. Gunawan, I. 2018. Kombinasi Algoritma Caesar Cipher dan Algoritma RSA untuk pengamanan File Dokumen dan Pesan Teks. *InfoTekJar (Jurnal Nasional Informatika dan Teknologi Jaringan)*, *2*(2), 124–129. https://doi.org/10.30743/infotekjar.v2i2.266.
3. Wardani, R. D. 2019. The Application of Number Theory to Determine Congruence in Traffic Lights. *BAREKENG: Jurnal Ilmu Matematika dan Terapan*, *13*(1), 047–052. https://doi.org/10.30598/barekengvol13iss1pp047-052ar697.
4. Haryadi, E., & Ladjamuddin, S. M. 2017. Teknik Keamanan Pesan Menggunakan Kriptografi Dengan Algoritma Vernam Chiper. *incomtech*, *6*(1).
5. Ridho, M. F. 2017. Perancangan Aplikasi Keamanan Data Dengan Algoritma Serpent. *Pelita Informatika: Informasi dan Informatika*, *6*(1), 116–120.
6. Yazirwan, Y. 2021. Perancangan Aplikasi Pengamanan File Menggunakan Algoritma Paillier Berbasis Android. *Jurnal Sistem Komputer dan Informatika (JSON)*, *2*(2), 137–140.
7. Irawan, M. D. 2017. Implementasi Kriptografi Vigenere Cipher Dengan Php. *(JurTI) Jurnal Teknologi Informasi*, *1*(1), 11–21.
8. Azhari, M., Mulyana, D. I., Perwitosari, F. J., & Ali, F. 2022. Implementasi Pengamanan Data pada Dokumen Menggunakan Algoritma Kriptografi Advanced Encryption Standard (AES). *Jurnal Pendidikan Sains dan Komputer*, *2*(01), 163–171.
9. Hidayat, M., Tahir, M., Sukriyadi, A., & Sulton, A. 2023. Penerapan Kriptografi Caesar Chiper dalam Pengamanan Data. *Jurnal Ilmiah Multidisiplin*, *2*(03), 35–41.
10. Nurcahya, S. D. 2022. Implementasi Aplikasi Kriptografi Metode Kode Geser Berbasis Java. *Jurnal Nasional Komputasi dan Teknologi Informasi*, *5*(4).
11. Agustina, A. N. 2017. Pengamanan Dokumen Menggunakan Metode RSA (Rivest Shamir Adleman) Berbasis Web.
12. Azis, N. 2018. Perancangan aplikasi enkripsi dekripsi menggunakan metode caesar chiper dan operasi xor. *IKRA-ITH Informatika: Jurnal Komputer dan Informatika*, *2*(1), 72–80.
13. Pratiwi, I. D., & Efendi, N. P. (n.d.). Jurnal Pengamanan Aplikasi Pesan dengan Algoritma Caesar Chiper dan Affine Chiper.
14. Azmi, M., & Zulkarnaen, Z. 2021. Implementasi Kombinasi Caesar Cipher dan Hill Cipher Menggunakan Modifikasi Sandi Morse Untuk Pengamanan Pesan Berbasis Teks. *JTIM: Jurnal Teknologi Informasi Dan Multimedia*, *3*(1), 8–13.
15. Siburian, A., & Harianja, A. P. 2017. Perancangan Aplikasi Pengamanan Basis Data Menggunakan Algoritma Caesar Cipher. *Jurnal Teknik Informatika UNIKA Santo Thomas*, 1–6.
16. Nasution, A. B. 2019. Implementasi Pengamanan Data Dengan Menggunakan Algoritma Caesar Cipher Dan Transposisi Cipher. *(JurTI) Jurnal Teknologi Informasi*, *3*(1), 1–6.
17. Priyono, P. 2016. Penerapan Algoritma Caesar Cipher dan Algoritma Vigenere Cipher Dalam Pengamanan Pesan Teks. *JURIKOM (Jurnal Riset Komputer)*, *3*(5).
18. Rizki, M., & Ariyani, P. F. 2021. Penerapan Kriptografi Dengan Menggunakan Algoritma Rsa Untuk Pengamanan Data Berbasis Desktop Pada Pt Trias Mitra Jaya Manunggal. *SKANIKA: Sistem Komputer dan Teknik Informatika*, *4*(2), 77–82.
19. Tampubolon, A. 2021. Implementasi Kombinasi Algoritma RSA dan Algoritma DES Pada Aplikasi Pengaman Pesan Teks. *Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika Dan Komputer)*, *20*(1), 38–43.
20. Rahayu, A., Ardana, A. P., Pramudhita, C., Syafitri, D., & Sirega, R. Z. 2024. Perbandingan Algoritma RSA dengan Algoritma Blowfish Pada Perancangan Aplikasi Keamanan Data. *Jurnal Ilmu Komputer dan Sistem Informasi (JIKOMSI)*, *7*(1), 203–207.
21. Burton, D. 2010. *Ebook: Elementary number theory*. McGraw Hill.
22. Harris, F., & Sari, R. E. 2024. Meningkatkan Keamanan Source Code Web Melalui Teknik Enkripsi dan Dekripsi Dengan Metode Reverse Cipher dan Caesar Cipher. *Jurnal Info Digit (JID)*, *2*(1), 405–417.
23. Winarno, N. P. S., & Cahyanto, T. A. 2021. Penggunaan Karakter Kontrol ASCII Untuk Integrasi Data Pada Hasil Enkripsi Algoritma Caesar Cipher. *INFORMAL: Informatics Journal*, *6*(3), 197–204.