# Aplikasi RSA, Caesar Cipher, dan Fungsi Hash dalam *Information Hiding*

Shabrina Maharani - 13522134[1]
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*[1]13522134@std.stei.itb.ac.id*

***Abstrak*—Kebocoran data merupakan salah satu tantangan dalam keamanan informasi, sering kali disebabkan oleh kesalahan manusia itu sendiri atau kesalahan sistem. Padahal, data dan informasi dengan sifat rahasia harus dilindungi dari pembacaan serta pengubahan yang dapat dilakukan oleh pihak-pihak yang tidak memiliki otoritas untuk mengetahui data tersebut. Untuk mengatasi masalah ini, teknik *information hiding* dapat menjadi solusi yang efektif. Dalam makalah ini, pembahasan akan berfokus pada salah satu teknik *information hiding*, yaitu kriptografi dengan pembuatan program *Cryptography Secret Generator* yaitu program berbasis CLI yang menerapkan kriptografi melalui kombinasi algoritma RSA yang diperkuat oleh Caesar Cipher, dan dibantu dengan fungsi hash untuk memperkuat keamanan pengguna. Kriptografi adalah ilmu sekaligus seni untuk menjaga keamanan sebuah pesan atau informasi [6]. Selain itu, kombinasi ini diharapkan akan semakin meningkatkan keamanan informasi yang akan dikirimkan.**

***Kata Kunci*—*Information Hiding*, Kriptografi, Algoritma RSA, Algoritma Caesar Cipher, Fungsi Hash**

## I. PENDAHULUAN

Kebocoran data merupakan ancaman serius dalam era informasi yang terus berkembang, di mana informasi dan data rahasia rentan terhadap pembacaan serta pengubahan oleh pihak yang tidak berhak. Baik disebabkan oleh kelalaian manusia, maupun kelemahan sistem, risiko kebocoran data membutuhkan pendekatan yang cermat dan efektif untuk melindungi keamanan informasi. Salah satu solusi yang telah dikenal dalam melawan tantangan ini adalah teknik *information hiding*, dengan fokus utama pada bidang kriptografi.

Kriptografi, sebagai ilmu dan seni untuk menjaga keamanan pesan atau informasi, telah menjadi fondasi penting dalam melindungi kerahasiaan data [6]. Dalam konteks ini, penulis berfokus untuk mendalami penerapan teknik *information hiding* dengan merancang *Cryptography Secret Generator*, sebuah program berbasis *Command Line Interface (CLI)* yang memanfaatkan kriptografi dengan menggabungkan algoritma RSA dan Caesar Cipher. Penggunaan kombinasi algoritma ini diperkuat lagi dengan fungsi hash yang menambahkan lapisan keamanan untuk melawan upaya pembacaan dan pengubahan data yang tidak sah.

Dalam proses pengembangan *Cryptography Secret Generator*, perhatian khusus diberikan pada penerapan algoritma RSA, yang dikenal dengan keunggulannya dalam keamanan kunci publik dan privat. Melalui kombinasi dengan Caesar Cipher, diharapkan dapat diciptakan suatu lapisan tambahan perlindungan, membuat data lebih sulit dipecahkan atau dimanipulasi oleh pihak yang tidak berhak. Penyempurnaan keamanan juga melibatkan penggunaan fungsi hash, sehingga integritas data dapat dipertahankan secara efektif.

Dengan mempertimbangkan kriptografi sebagai langkah kritis dalam menjaga keamanan informasi, *Cryptography Secret Generator* diharapkan dapat memberikan kontribusi positif dalam melindungi data dan informasi yang sensitif. Penerapan kombinasi algoritma RSA, Caesar Cipher, dan fungsi hash dalam program ini menjadi fokus utama pembahasan, dengan tujuan meningkatkan tingkat keamanan informasi yang dikirimkan dan disimpan.

## II. LANDASAN TEORI

### A. Bilangan Bulat

Teori bilangan merupakan cabang ilmu matematika murni yang mempelajari bilangan bulat (*integer*) atau fungsi yang berhubungan dengan bilangan bulat. Bilangan bulat merupakan bilangan yang tidak memiliki pecahan desimal dalam representasinya, contohnya 22, -9, 7, 8, dan sebagainya. Bilangan bulat umumnya dilambangkan dengan notasi $\mathbb{Z}$. Bilangan bulat terbagi menjadi bilangan bulat positif dan bilangan bulat negatif. Bilangan bulat positif adalah bilangan bulat yang dimulai dari 1 sampai seterusnya. Sedangkan, bilangan bulat negatif dimulai dari angka -1 dan seterusnya dikurangi 1. Selain itu, angka 0 juga termasuk dalam himpunan bilangan bulat.

### B. Sifat Pembagian Bilangan Bulat

Dalam konteks sifat pembagian pada bilangan bulat, apabila terdapat dua bilangan bulat, a dan b (dengan a tidak sama dengan 0), maka dapat dikatakan bahwa a habis membagi b jika terdapat suatu bilangan bulat c, sedemikian rupa sehingga hasil bagi b terhadap a adalah c.

$$a \mid b, \text{ jika } b = ac,\ c \in \mathbb{Z},\ a \neq 0$$

Hal ini dapat direpresentasikan dengan notasi di atas, a | b, yang berarti a dapat membagi b dengan hasil bilangan bulat c, di mana c termasuk dalam himpunan bilangan bulat, dan a tidak sama dengan 0.

Sebagai contoh, jika a adalah 4 dan b adalah 12, maka 4 dapat membagi 12 karena 12 dapat diwakili sebagai hasil kali antara 4

dan 3, di mana 3 adalah bilangan bulat. Notasi nya dapat direpresentasi menjadi seperti di bawah ini.

$$4 \mid 12,\ 12 = (3)(4),\ 3 \in \mathbb{Z},\ 4 \neq 0$$

Namun, sebaliknya, jika b adalah 13, maka 4 tidak dapat membagi 13 karena hasil bagi 13 terhadap 4 bukanlah bilangan bulat [4].

### C. Teorema Euclidean

Teorema Euclidean merupakan suatu prinsip mendasar dalam matematika yang terkait dengan pembagian bilangan bulat. Menurut teorema ini, untuk dua bilangan bulat (m) dan (n) dengan (n > 0), jika (m) dapat dibagi oleh (n), maka terdapat dua bilangan bulat unik, yakni (q) (quotient) dan (r) (remainder), sehingga persamaannya adalah sebagai berikut

$$m = nq + r, \text{ berlaku dengan batasan } (0 \leq r < n)$$

Dengan kata lain, teorema ini menyatakan bahwa setiap bilangan bulat (m) dapat diungkapkan sebagai hasil kali antara bilangan bulat (n) dan (q), ditambah dengan sisa pembagian (r) yang lebih kecil dari (n). Proses ini, yang dikenal sebagai algoritma pembagian atau algoritma Euclidean, memiliki aplikasi luas dalam matematika, terutama terkait pembagian bilangan bulat dan perhitungan sisa pembagian. Selain itu, Teorema Euclidean membentuk dasar untuk konsep Pembagi Bersama Terbesar (PBB) atau greatest common divisor (GCD), yang memiliki peran krusial dalam berbagai aspek matematika seperti teori bilangan dan kriptografi.

Pembagi Bersama Terbesar (PBB), juga dikenal sebagai *Greatest Common Divisor* (GCD), adalah konsep yang melibatkan dua bilangan bulat non-nol, a dan b. PBB(a, b) merupakan bilangan bulat terbesar d, sehingga d dapat membagi a dan b. Dalam istilah sekolah dasar, konsep ini sering disebut sebagai faktor persekutuan terbesar (FPB). Teorema keduanya menyatakan bahwa jika m dan n adalah bilangan bulat dengan syarat n > 0, dan m = nq + r, maka

$$\text{PBB}(m, n) = \text{PBB}(n, r)$$

Sebagai contoh, pada situasi di mana m = 60 dan n = 18, dengan persamaan 60 = 3 x 18 + 6, PBB(60, 18) setara dengan PBB(18, 6), yang keduanya bernilai 6[5].

### D. Aritmetika Modulo

Jika terdapat dua bilangan bulat, a dan m, di mana m adalah bilangan bulat positif. Operasi **a mod m**, yang dapat dibaca sebagai "a modulo m," menghasilkan sisa dari pembagian a dengan m. Notasi ini dapat diartikan sebagai hasil kali antara m dan suatu bilangan bulat q, ditambah dengan sisa pembagian r, dengan syarat bahwa $0 \leq r < m$.

Dalam konteks ini, m disebut sebagai modulus atau modulo, dan hasil aritmetika modulo m terbatas pada himpunan bilangan $\{0, 1, 2, \ldots, m - 1\}$. Sebagai contoh, beberapa contoh hasil dari operasi modulo dapat dijelaskan sebagai berikut:

(i) 22 mod 5 = 2,
(ii) 21 mod 3 = 0,
(iii) 2 mod 8 = 2,
(iv) 0 mod 234 = 0, dan
(v) –41 mod 8 = 7 (dengan penjelasan bahwa karena a negatif, hasil modulo dihitung sebagai m dikurangi sisa pembagian positif) [4].

### E. Kekongruenan

Misalnya, m adalah bilangan bulat positif. Sebuah bilangan bulat a dikatakan kongruen dengan bilangan bulat b modulo m jika, dan hanya jika, m membagi (a-b), dan hal ini dapat dinyatakan sebagai

$$a \equiv b \pmod{m}$$

Definisi kongruensi pertama kali diungkapkan oleh Carl Friedrich Gauss pada tahun 1790, di mana

$$x \equiv r \pmod{d}$$

berlaku jika dan hanya jika

$$x\text{-}r = kd, \text{ untuk suatu bilangan bulat } k=0,1,2,\ldots$$

Sifat distributif modulo menyatakan bahwa untuk bilangan bulat a, b, dan c, jika $a \equiv b \pmod{m}$, maka $a + c \equiv b + c \pmod{m}$ dan $ac \equiv bc \pmod{m}$. Ini berarti operasi penambahan dan perkalian dapat diterapkan pada kongruensi dengan cara yang konsisten dengan sifat-sifat distributif pada bilangan bulat.

Sifat keterhubungan modulo menyatakan bahwa jika

$$a \equiv b \pmod{m} \text{ dan } b \equiv c \pmod{m},$$
$$\text{maka } a \equiv c \pmod{m}$$

Dengan kata lain, kongruensi bersifat transitif, sehingga jika dua bilangan bulat kongruen terhadap modulus yang sama dan satu bilangan juga kongruen dengan bilangan lainnya, maka ketiganya kongruen satu sama lainnya[2].

### F. Information Hiding

*Information hiding* merupakan suatu bidang ilmu yang mempelajari metode untuk menyembunyikan pesan agar tidak dapat terdeteksi oleh orang lain, baik secara visual maupun auditif [4]. Tujuan utama dari *information hiding* adalah untuk menjaga kerahasiaan, integritas, dan autentikasi informasi.

Dasar dari konsep *information hiding* melibatkan beberapa prinsip utama. Pertama, *non-obscurity* menegaskan bahwa keamanan tidak hanya bergantung pada seberapa baik informasi disembunyikan, melainkan juga pada kekuatan algoritma dan metode pengamanan yang digunakan. Kedua, *robustness* adalah prinsip yang menekankan kemampuan sistem untuk tetap efektif meskipun dihadapkan pada tekanan atau serangan. Terakhir, *capacity* merujuk pada seberapa besar volume data yang dapat disembunyikan atau disisipkan tanpa mengurangi kualitas informasi yang tersembunyi.

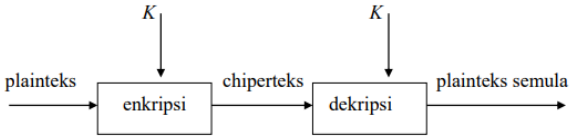
Manfaat dari *information hiding* mencakup aspek-aspek seperti kerahasiaan informasi, integritas data, autentikasi, dan perlindungan terhadap serangan seperti penyadapan atau manipulasi data. Organisasi-organisasi seperti International Association for Cryptologic Research (IACR), Steganography Analysis and Research Center (SARC), National Institute of Standards and Technology (NIST), dan International Organization for Standardization (ISO) memiliki peran penting dalam pengembangan dan penetapan standar terkait keamanan informasi, termasuk dalam bidang *information hiding*.

Tantangan dan perkembangan terkini dalam *information hiding* melibatkan aspek seperti penanganan *big data*, pemanfaatan teknologi *machine learning* dalam analisis steganografi, keamanan mobile, dan dampak *quantum computing* pada kriptografi. Pemahaman yang mendalam terhadap konsep-konsep ini menjadi sangat penting dalam perancangan sistem *information hiding* yang efektif dan aman di tengah dinamika perkembangan teknologi informasi.

## *G. Kriptografi*

Kriptografi merupakan bidang ilmu komputasi yang mempelajari teknik-teknik dalam bidang matematika yang berkaitan dengan berbagai aspek keamanan dalam penyampaian informasi seperti informasi yang bersifat rahasia, integritas dari sebuah data, dan otentikasi (Menez, 1996)[6]. Kriptografi merupakan bidang yang sangat digunakan dalam menjaga keamanan informasi. Berdasarkan sejarahnya, kriptografi digunakan untuk kegiatan mata-mata saat pemerintahan, militer, dan perang di zaman dahulu. Istilah kriptografi berasal dari bahasa Yunani yaitu *cryptos* yang artinya rahasia dan *graphein* yang artinya menulis. Dari kedua pengertian tersebut kriptografi juga dapat disimpulkan sebagai bidang ilmu sekaligus sebuah seni untuk menjaga privasi atau keamanan pesan (informasi).

Dalam bidang kriptografi, pesan yang dijaga kerahasiaannya disebut sebagai plainteks, sementara pesan hasil penyandian disebut sebagai cipherteks. Untuk mengembalikan pesan yang telah disandikan ke bentuk aslinya, seseorang perlu memahami metode penyandian dan memiliki kunci penyandian yang sesuai.




*Gambar 2.G.1 Alur dalam Kriptografi*
*Sumber : Teori Bilangan (itb.ac.id)*

Hal ini memastikan bahwa hanya pihak yang berhak yang dapat melakukan pengembalian ke bentuk awal pesan tersebut. Proses mengubah plainteks menjadi cipherteks dikenal sebagai enkripsi, sementara proses mengembalikan cipherteks menjadi plainteks disebut dekripsi. Terdapat tiga jenis algoritma kriptografi, diantaranya algoritma kriptografi simetri, algoritma kriptografi nir-simetri, dan fungsi hash.

Algoritma kriptografi simetri melibatkan penggunaan kunci yang sama untuk proses enkripsi dan dekripsi. Kunci ini harus dijaga dengan ketat sebagai informasi rahasia. Jenis kriptografi ini telah dikenal sejak ribuan tahun yang lalu, dan prinsipnya tetap relevan hingga tahun 1976.

Algoritma kriptografi nir-simetri, atau sering disebut kriptografi kunci-publik, melibatkan penggunaan dua kunci yang berbeda untuk enkripsi dan dekripsi. Kunci enkripsi dapat diketahui publik (*public key*), sementara kunci dekripsi harus dijaga sebagai informasi rahasia (*private key*). Model kriptografi ini pertama kali muncul pada tahun 1976 dan membawa paradigma baru dalam keamanan informasi.

Fungsi Hash, sebagai jenis kriptografi lainnya, berfokus pada mengkompresi pesan dengan ukuran sembarang menjadi *message-digest* dengan ukuran tetap. Proses ini bersifat *irreversible*, artinya tidak mungkin mengembalikan pesan ke bentuk aslinya. Fungsi Hash menjadi penting dalam mengamankan integritas data dan memastikan bahwa pesan tidak dapat dimodifikasi tanpa deteksi.

## *H. Fungsi Hash*

Hash dalam komputasi mengacu pada fungsi hash atau fungsi penghas, yakni metode matematis yang mengubah data, seperti teks atau file, menjadi nilai hash yang unik. Nilai hash tersebut berupa serangkaian angka atau huruf acak yang mewakili data yang telah di-hash. Fungsi hash ini juga merupakan salah satu jenis dari algoritma kriptografi.

Fungsi hash memiliki beberapa sifat kunci, termasuk keunikan, deterministik, irreversibilitas, dan panjang nilai tetap. Keunikan memastikan bahwa setiap data yang berbeda menghasilkan nilai hash yang berbeda, meskipun kemungkinan tabrakan tetap ada. Deterministik menunjukkan bahwa fungsi hash akan selalu menghasilkan nilai yang sama untuk input yang sama. Irreversibilitas menandakan bahwa mengembalikan nilai hash menjadi data asli sulit atau bahkan tidak mungkin. Panjang nilai hash tetap mempermudah pemrosesan data.

Salah satu fungsi hash yang digunakan dalam dunia komputasi adalah fungsi hash modulo sebagai berikut

$$h(k) = k \bmod m$$

Fungsi tersebut dapat diterapkan dalam pembuatan program, khususnya dalam konteks keamanan data. Dengan menerapkan fungsi hash, program memastikan keunikan dan mengontrol akses hanya untuk pihak yang berhak.

## *I. Algoritma Caesar Cipher*

Algoritma Caesar Cipher, yang termasuk dalam kategori cipher klasik dengan teknik substitusi, adalah metode pengamanan informasi yang melibatkan penggantian huruf plainteks dengan huruf cipherteks. Teknik substitusi ini dilakukan dengan cara menggeser setiap huruf dalam plainteks sebanyak n posisi secara wrapping, yang berarti jika mencapai akhir alfabet, penggeseran akan kembali ke awal. Algoritma ini dikenal telah digunakan pada zaman Raja Julius Caesar, yang memberikan nama pada metode ini[7].

Dalam konteks enkripsi dan dekripsi pada Caesar Cipher, terdapat prosedur khusus. Pada proses enkripsi, setiap huruf dalam plainteks digeser ke posisi yang ditentukan secara wrapping, menghasilkan cipherteks. Sedangkan pada proses dekripsi, penggeseran dilakukan ke arah sebaliknya untuk mengembalikan cipherteks ke bentuk aslinya, yaitu plainteks. Walaupun Caesar Cipher termasuk dalam kategori cipher klasik, keamanannya relatif rendah karena metodenya yang sederhana. Oleh karena itu, dalam penggunaannya, lebih disarankan untuk dikombinasikan dengan metode lain agar lebih aman.

Untuk melakukan enkripsi dan dekripsi pada Caesar Cipher dapat dilakukan dengan menggunakan prosedur sebagai berikut[7].

Enkripsi: $c = E(p) = (p + k) \bmod 26$
Dekripsi: $p = D(c) = (c - k) \bmod 26$
$k$ = kunci rahasia

*Gambar 2.I.1 Alur dalam Kriptografi*
*Sumber : Teori Bilangan (itb.ac.id)*

### *J. Algoritma RSA*

Rivest-Shamir-Adleman atau umumnya disingkat menjadi RSA adalah salah satu penerapan kriptografi dengan jenis algoritma kriptografi nir-simetri, atau sering disebut kriptografi kunci-publik yang melibatkan penggunaan dua kunci yang berbeda untuk enkripsi dan dekripsi. RSA merupakan algoritma kunci-publik yang paling popular dalam dunia keamanan informasi. Algoritma ini dibuat oleh kelompok peneliti yang berasal dari *Massachussets Institute of Technology* pada 1976, yaitu Ronald Rivest, Adi Shamir, dan Leonard Adleman. Algoritma RSA memiliki tingkat keamanan yang sangat baik karena memiliki kesulitan untuk memfaktorkan bilangan bulat yang besar menjadi factor-faktor prima [9] .

Dalam algoritma RSA itu sendiri, setiap pengguna yang akan menggunakan metode ini akan memiliki sepasang kunci, yaitu kunci publik (e) yang bersifat tidak rahasia atau dapat diketahui oleh publik digunakan untuk melakukan enkripsi pesan dan kunci privat (d) yang bersifat rahasia (hanya diketahui oleh pemilik kunci digunakan untuk melakukan dekripsi pesan. Untuk mendapatkan pasangan kunci di dalam RSA pengguna dapat melakukan prosedur dalam beberapa langkah. Pertama, pengguna dapat memilih dua bilangan prima, p dan q yang sifatnya rahasia. Kemudian, p dan q akan dihitung menjadi

$$n = pq$$

Besaran n tidak bersifat rahasia dan dapat diketahui oleh publik. Setelah itu, akan dihitung variabel m yang sifatnya rahasia.

$$m = (p - 1)(q - 1)$$

Kemudian, pengguna akan diminta untuk memilih sebuah bilangan bulat untuk kunci publik, e, yang relatif prima terhadap m, yaitu

$$PBB(e, m) = 1$$

Terakhir, kunci dekripsi, d, akan dihitung melalui kekongruenan

$$ed \equiv 1 \pmod{m}$$

Enkripsi dan dekripsi pada RSA dilakukan dengan menggunakan prosedur sebagai berikut

**Enkripsi:** $p^e \equiv c \pmod{n}$ atau dapat ditulis: $c = p^e \bmod n$

**Dekripsi:** $c^d \equiv p \pmod{n}$ atau dapat ditulis: $p = c^d \bmod n$

*Gambar 2.J.1 Prosedur Enkripsi-Dekripsi Algoritma RSA*
*Sumber : Teori Bilangan (itb.ac.id)*

## III. Analisis Permasalahan Pembuatan Program

Dalam pengembangan program *Cryptography Secret Generator* ini, terdapat beberapa langkah analisis permasalahan yang penulis tempuh. Pertama-tama, penulis melakukan analisis permasalahan terkait fungsi hash yang digunakan untuk mengamankan penggunaan password dan username. Selanjutnya, akan dibahas analisis permasalahan pada tahap enkripsi, dimana penggunaan Caesar Cipher yang diperkuat dengan RSA menjadi bagian dari keamanan program. Terakhir, tahapan analisis permasalahan untuk proses dekripsi, menggambarkan langkah-langkah yang diperlukan untuk mengembalikan informasi yang telah dienkripsi. Dengan pemahaman mendalam terhadap permasalahan ini, program dapat dirancang dengan lebih efektif dan dapat diandalkan.

### *A. Fungsi Hash*

Fungsi hash digunakan untuk menentukan kevalidan password yang dimasukkan oleh pengguna. Pada tahap ini, perlu dipertimbangkan kekuatan dan keamanan dari fungsi hash yang digunakan. Pertanyaan-pertanyaan mendasar meliputi bagaimana username diubah menjadi nilai hash dengan fungsi hash tertentu, serta bagaimana proses verifikasi dilakukan dengan membandingkan nilai hash hasil enkripsi password dengan nilai hash yang disimpan.

Misalnya, jika pengguna memasukkan username dengan input *maha*, program seharusnya akan melakukan pengubahan terhadap karakter m, a, dan h ke representasi bilangan ASCII. Dalam hal ini, m bernilai 109, a bernilai 97, dan h bernilai 104. Kemudian, program akan menjumlahkan bilangan ASCII tersebut menjadi 419. Melalui fungsi hash modulo, angka 419 akan di modulo dengan m dan itulah yang seharusnya menjadi password dari pengguna dengan username *maha*.

Program harus mampu menangani permasalahan apabila password yang dimasukkan tidak sesuai dengan hasil hash dari username yang telah diubah menjadi ASCII tersebut.

### *B. Enkripsi*

Enkripsi pada program ini melibatkan penggunaan Caesar Cipher yang diperkuat oleh RSA. Tujuan dari proses enkripsi yaitu untuk meningkatkan keamanan informasi yang disimpan atau dikirimkan. Program harus dibuat untuk menjawab pertanyaan bagaimana teks dapat diubah menjadi bentuk terenkripsi dengan Caesar Cipher, serta bagaimana RSA memberikan lapisan keamanan tambahan. Pemahaman mendalam tentang bagaimana kedua metode ini bekerja bersama dibutuhkan untuk menciptakan sistem yang andal, aman serta mampu menghasilkan pesan yang sudah dienkripsi dengan baik.

### *C. Dekripsi*

Dekripsi merupakan langkah krusial dalam mengembalikan informasi terenkripsi ke bentuk semula. Analisis permasalahan di tahap dekripsi mencakup pemahaman tentang bagaimana nilai-nilai ASCII hasil dekripsi RSA diubah kembali menjadi teks dengan dekripsi RSA terlebih dahulu kemudian dilanjutkan dengan dekripsi melalui prosedur dekripsi Caesar Cipher. Program harus dibuat untuk menggabungkan kedua metode ini sehingga informasi dapat dipulihkan tanpa kehilangan integritas.

Dengan memahami secara mendalam permasalahan pada setiap tahap, program *Cryptography Secret Generator* ini dapat dikembangkan dengan lebih efisien dan menghasilkan sistem keamanan yang handal.

## IV. Penyelesaian Masalah

Pada pembahasan makalah ini, penulis telah berhasil menanggapi permasalahan yang dianalisis dengan mengimplementasikan solusi melaui pembuatan program dalam bahasa pemrograman Python. Bahasa pemrograman Python dipilih oleh penulis karena lebih memudahkan dalam pembuatan proses melakukan implementasi program.

### *A. Fungsi Hash (hash.py)*

Penulis berhasil menyelesaikan masalah terkait dengan fungsi hash yang akan menjadi pembanding password yang dimasukkan oleh pengguna. Solusi tersebut dibuat dengan cara mengimplementasikan kelas Hash dengan metode hash_func untuk menghasilkan nilai hash dari sebuah key (username dalam bentuk penjumlahan ASCII dari tiap karakter). Pertama-tama, objek Hash dibuat dengan parameter m yang mewakili ukuran tabel hash atau range nilai hash. Selanjutnya, metode hash_func digunakan untuk menghasilkan nilai hash dari key yang diberikan.

```python
src > hash.py
class Hash:
    def __init__(self, m):
        self.m = m

    def hash_func(self, key):
        # Menghitung jumlah dari representasi ASCII setiap karakter dalam key
        key_sum = sum(ord(char) for char in key)
        # Menggunakan fungsi hash h(K) = K mod m
        hashed_value = key_sum % self.m
        return hashed_value
```

*Gambar 4.A.1 Implementasi Program Fungsi Hash*
*Sumber : Dokumen Milik Pribadi*

Dalam metode hash_func, pertama-tama, dilakukan penghitungan jumlah dari representasi ASCII setiap karakter dalam key dengan menggunakan fungsi *sum(ord(char) for char in key)*. Hal ini dilakukan untuk mendapatkan nilai numerik yang unik untuk setiap key. Selanjutnya, nilai numerik tersebut dimodulo dengan m (h(K) = K mod m) untuk mendapatkan nilai hash akhir. Hasil nilai hash ini kemudian dapat digunakan sebagai indeks dalam tabel hash untuk penyimpanan atau pencarian.

### *B. Algoritma RSA (rsa.py)*

Implementasi algoritma RSA dalam penyelesaian masalah enkripsi dan dekripsi juga dibuat untuk menyelesaikan permasalahan. Implementasi tersebut, disebut sebagai rsa.py, mengikuti prosedur yang sesuai dengan landasan teori RSA. Rivest-Shamir-Adleman (RSA) adalah salah satu algoritma kriptografi kunci-publik yang digunakan untuk melibatkan dua kunci berbeda dalam proses enkripsi dan dekripsi.

```python
rsa.py
class RSA:
    def __init__(self, p, q, public_key):
        self.n = p * q
        self.phi = (p - 1) * (q - 1)
        self.public_key = public_key
        self.private_key = self.get_private_key()

    def get_private_key(self):
        d = pow(self.public_key, -1, self.phi)
        return d

    def enkripsi(self, plaintext):
        return pow(plaintext, self.public_key, self.n)

    def dekripsi(self, ciphertext):
        return pow(ciphertext, self.private_key, self.n)
```

*Gambar 4.B.1 Implementasi Program Algoritma RSA*
*Sumber : Dokumen Milik Pribadi*

Dalam implementasi rsa.py, terdapat kelas RSA yang mengikuti langkah-langkah pembentukan kunci dan proses enkripsi serta dekripsi pada algoritma RSA. Pertama, pada metode konstruktor (init), program menginisialisasi nilai n sebagai hasil perkalian dua bilangan prima rahasia, yaitu p dan q. Selanjutnya, menghitung nilai phi (ϕ) menggunakan rumus (p - 1)(q - 1) yang juga merupakan nilai rahasia. Penggunaan kunci publik (e) sebagai parameter konstruktor memungkinkan inisialisasi nilai kunci publik secara langsung. Selain itu, metode konstruktor juga menghitung kunci privat (d) dengan memangkatkan kunci publik ke invers modulo phi menggunakan fungsi pow.

Metode get_private_key() bertanggung jawab untuk mengembalikan nilai kunci privat (d) yang telah dihitung sebelumnya. Ini dilakukan dengan menggunakan fungsi pow untuk menghitung kebalikan modulo dari kunci publik terhadap phi.

Proses enkripsi diimplementasikan dalam fungsi enkripsi(plaintext). Pada langkah ini, pesan teks biasa diangkat ke pangkat kunci publik (e) dan diambil modulo n. Hasilnya adalah teks terenkripsi yang dapat dikirimkan dengan aman melalui saluran komunikasi.

Proses dekripsi diimplementasikan dalam fungsi dekripsi (ciphertext). Pada langkah ini, teks terenkripsi diangkat ke pangkat kunci privat (d) dan diambil modulo n. Hasilnya adalah pesan teks biasa yang dapat dibaca oleh penerima yang memiliki kunci privat yang sesuai.

Dengan demikian, implementasi rsa.py secara komprehensif menggambarkan alur pembentukan kunci dan proses enkripsi serta dekripsi dalam algoritma RSA. Setiap langkahnya sesuai dengan prinsip-prinsip landasan teori RSA, memastikan keamanan informasi melalui kriptografi kunci-publik.

### *C. Algoritma Caesar Cipher (caesar_cipher.py)*

Implementasi algoritma Caesar Cipher dalam penyelesaian masalah enkripsi dan dekripsi juga dibuat untuk menyelesaikan permasalahan. Implementasi tersebut berada dalam caesar_cipher.py, dibuat mengikuti prosedur yang sesuai dengan landasan teori Caesar Cipher.

```python
caesar_cipher.py
class CaesarCipher:
    def __init__(self, shift):
        self.shift = shift

    def enkripsi(self, plaintext):
        result = ""
        for char in plaintext:
            if char.isalpha():
                ascii_offset = ord('a') if char.islower() else ord('A')
                result += chr((ord(char) - ascii_offset + self.shift) % 26 + ascii_offset)
            else:
                result += char
        return result

    def dekripsi(self, ciphertext):
        result = ""
        for char in ciphertext:
            if char.isalpha():
                ascii_offset = ord('a') if char.islower() else ord('A')
                result += chr((ord(char) - ascii_offset + 26 + self.shift) % 26 + ascii_offset)
            else:
                result += char
        return result
```

*Gambar 4.C.1 Implementasi Program Algoritma Caesar Cipher*
*Sumber : Dokumen Milik Pribadi*

Dalam fungsi enkripsi, setiap karakter dalam plainteks diperiksa. Jika karakter tersebut merupakan huruf, maka dilakukan penggeseran sesuai dengan aturan yang telah dijelaskan dalam teori. Penggeseran ini melibatkan perhitungan ASCII *offset* tergantung pada apakah karakter tersebut huruf kecil atau huruf besar. Hasil penggeseran kemudian dihitung

secara modulo 26 untuk memastikan *wrapping*, dan karakter baru ditambahkan ke hasil akhir. Jika karakter bukan huruf, karakter tersebut langsung ditambahkan ke hasil enkripsi. Proses ini dilakukan untuk setiap karakter dalam plainteks.

Fungsi dekripsi mengikuti prinsip yang sama, tetapi dengan penggeseran ke arah sebaliknya. Penggeseran dilakukan dengan menambahkan 26 ke shift untuk memastikan perhitungan modulo memberikan hasil yang benar. Seperti pada proses enkripsi, setiap karakter diperiksa, diubah, dan ditambahkan ke hasil dekripsi.

### *D. Program Utama (main.py)*

```
# Langkah 1: Meminta pengguna memasukkan username
username = input("Masukkan username (tanpa spasi): ")

# Langkah 2: Mengubah username menjadi ASCII dan menghitung sum
username_sum = sum(ord(char) for char in username)
# Memastikan bahwa username tidak mengandung spasi
if ' ' in username:
    raise ValueError("Username tidak boleh mengandung spasi.")

# Membuat objek Hash dengan m = 100 untuk hash username_sum
hash_obj = Hash(m=100)
hashed_username_sum = hash_obj.hash_func(str(username_sum))

# Memeriksa kecocokan hashed_username_sum dengan hash dari password
while True:
    input_password = input("Masukkan password: ")
    if input_password == str(hashed_username_sum):
        print("Password benar.")
        break
    else:
        print("Password salah. Silakan coba lagi.")
        print("Clue : ")
        print("1. Ubah karakter username menjadi representasi angka ASCII")
        print("2. Jumlahkan seluruh hasil representasi ASCII dari username")
        print("3. Password yang seharusnya adalah jumlah representasi ASCII di modulo dengan 100")

# Langkah 4: Meminta pengguna untuk memilih enkripsi atau dekripsi
choice = input("Pilih 'e' untuk enkripsi atau 'd' untuk dekripsi: ")
```

*Gambar 4.D.1 Implementasi Program Utama Bagian 1*
*Sumber : Dokumen Milik Pribadi*

Pada program utama main.py, ketiga implementasi kriptografi dari bagian A, B, dan C dikombinasikan untuk membentuk suatu *Cryptography Secret Generator*. Program utama ini memainkan peran penting dalam keamanan dan privasi pengguna. Mulai dari baris ke-40, program meminta pengguna untuk memasukkan username, yang kemudian diubah menjadi representasi ASCII dan dihitung jumlahnya. Proses ini dilakukan untuk menciptakan hash dari jumlah ASCII username, dan pengguna diminta untuk memasukkan password yang sesuai. Setelah otentikasi berhasil, pengguna dapat memilih untuk melakukan enkripsi atau dekripsi. Pilihan ini mengarahkan program ke langkah selanjutnya, di mana teks yang akan dienkripsi atau didekripsi dimasukkan.

```
# Langkah 5: Melakukan enkripsi atau dekripsi sesuai dengan pilihan pengguna
if choice == 'e':
    # Meminta pengguna memasukkan teks yang akan dienkripsi
    plaintext = input("Masukkan teks yang akan dienkripsi: ")

    # Enkripsi dengan Caesar Cipher
    caesar_cipher = CaesarCipher(shift=3)
    encrypted_caesar = caesar_cipher.enkripsi(plaintext)

    # Mengubah setiap karakter pada teks hasil enkripsi dari Caesar Cipher menjadi ASCII
    ascii_values = [ord(char) for char in encrypted_caesar]

    # Enkripsi dengan RSA
    rsa = RSA(p=61, q=53, public_key=17)
    encrypted_rsa = [rsa.enkripsi(value) for value in ascii_values]

    print("Hasil enkripsi: ", encrypted_rsa)
    print("Kunci publik: ", rsa.public_key)
    print("------------------------------------------------------------")
```

*Gambar 4.D.2 Proses Enkripsi pada Program Utama*
*Sumber : Dokumen Milik Pribadi*

Pada langkah enkripsi, teks pertama kali dienkripsi menggunakan Caesar Cipher dengan pergeseran sebanyak 3 posisi ke kanan. Hasil enkripsi tersebut kemudian diubah menjadi representasi ASCII, dan teks kembali dienkripsi menggunakan RSA. Hasil akhir dari proses ini adalah teks terenkripsi beserta kunci publik yang digunakan.

```
elif choice == 'd':
    # Meminta pengguna memasukkan teks yang akan didekripsi
    ciphertext = input("Masukkan teks yang akan didekripsi: ")

    # Meminta pengguna memasukkan public key dan private key
    public_key = int(input("Masukkan public key: "))

    # Dekripsi dengan RSA
    rsa = RSA(p=61, q=53, public_key=public_key)
    #rsa.private_key = get_private_key(rsa)
    while True:
        private_key = int(input("Masukkan private key: "))
        if private_key == rsa.private_key:
            break
        else:
            print("Kunci privat salah. Silakan coba lagi.")

    # Membuat objek CaesarCipher dengan shift ke kiri 3 huruf
    caesar_cipher = CaesarCipher(shift=3)

    # Dekripsi RSA untuk mendapatkan list nilai ASCII
    decrypted_rsa = [rsa.dekripsi(value) for value in eval(ciphertext)]

    # Mengubah nilai ASCII ke teks dengan Caesar Cipher
    caesar_text = "".join(chr(value) for value in decrypted_rsa)


    # Dekripsi dengan Caesar Cipher
    decrypted_caesar = caesar_cipher.dekripsi(caesar_text)

    #Mengeluarkan output berupa hasil dekripsi
    print("************************************************************")
    print("           Hasil dekripsi:", decrypted_caesar)
    print("************************************************************")


    print("------------------------------------------------------------")

else:
    print("Pilihan tidak valid.")
    print("------------------------------------------------------------")
```

*Gambar 4.D.3 Proses Dekripsi pada Program Utama*
*Sumber : Dokumen Milik Pribadi*

Sementara itu, pada langkah dekripsi, program meminta pengguna untuk memasukkan teks yang akan didekripsi. Pengguna juga diminta untuk memasukkan kunci publik dan kunci privat yang digunakan dalam algoritma RSA. Proses dekripsi dimulai dengan dekripsi RSA untuk mendapatkan list nilai ASCII, yang selanjutnya diubah kembali menjadi cipherteks. Kemudian, cipherteks akan dilanjutkan dengan dekripsi menggunakan algoritma Caesar Cipher yaitu melakukan pergeseran ke kanan 3 posisi secara *wrapping*.

Hasil akhir dari proses enkripsi atau dekripsi akan ditampilkan sebagai output, memberikan pengguna akses ke informasi yang aman dan terenkripsi. Program utama ini menciptakan pengalaman pengguna yang interaktif sambil menyediakan lapisan keamanan ganda melalui Caesar Cipher dan RSA. Dengan kombinasi ini, program *Cryptography Secret Generator* memberikan keamanan tambahan terhadap serangan dan upaya pembongkaran teks terenkripsi.

## V. UJI COBA PROGRAM

Dalam bagian V ini, akan ditunjukkan hasil simulasi uji coba program untuk membuktikan keefektifan dan kehandalan *Cryptography Secret Generator* yang telah diimplementasikan. Uji coba ini dirancang untuk mengevaluasi fungsionalitas program dalam mengamankan informasi pengguna serta memverifikasi bahwa langkah-langkah enkripsi dan dekripsi dapat dijalankan dengan sukses. Melalui simulasi ini, akan dipastikan kembali bahwa program dapat menghasilkan teks terenkripsi yang aman dan mampu mengembalikan teks asli

melalui proses dekripsi yang benar. Berikut adalah hasil uji coba program pada berbagai skenario untuk meneguhkan integritas dan kredibilitas *Cryptography Secret Generator* yang telah penulis kembangkan.

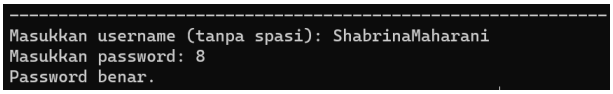
1. Antarmuka Pengguna


*Gambar 5.1 Tampilan Antarmuka Pengguna*
*Sumber : Dokumen Milik Pribadi*

Antarmuka pengguna yang disediakan dalam program *Cryptography Secret Generator* bukan hanya bertujuan untuk memperindah program, tetapi juga untuk memberikan pengalaman yang lebih intuitif dan interaktif kepada pengguna. Dengan menggunakan antarmuka pengguna yang ramah, pengguna dapat dengan mudah memasukkan informasi yang dibutuhkan, seperti username, password, dan pilihan untuk enkripsi atau dekripsi. Desain antarmuka yang baik tidak hanya meningkatkan estetika program, tetapi juga memastikan pengguna dapat menggunakan fungsionalitas program dengan nyaman dan tanpa kesulitan, menjadikan *Cryptography Secret Generator* lebih *user-friendly*.

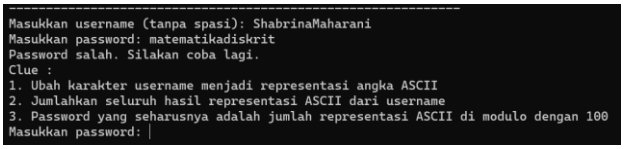
2. Proses input username dan password

Terdapat dua kondisi yang mungkin terjadi dalam proses ini. Pertama, jika password yang dimasukkan sesuai dengan hasil dari fungsi hash yang dilakukan pada jumlah ASCII dari username, program akan mengeluarkan tampilan sebagai berikut.




*Gambar 5.2 Simulasi Input Password Benar*
*Sumber : Dokumen Milik Pribadi*

Kedua, jika password yang dimasukkan tidak sesuai dengan hasil dari fungsi hash yang dilakukan pada jumlah ASCII dari username, program akan mengeluarkan tampilan sebagai berikut dan akan meminta masukan password kembali.



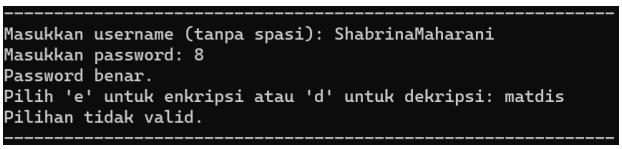

*Gambar 5.3 Simulasi Input Password Salah*
*Sumber : Dokumen Milik Pribadi*

3. Proses pemilihan antara melakukan enkripsi atau dekripsi

Program akan meminta input huruf e atau d kepada pengguna.

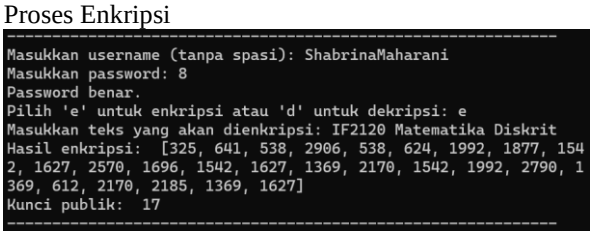

*Gambar 5.4 Pemilihan Enkripsi-Dekripsi*
*Sumber : Dokumen Milik Pribadi*

Apabila input yang dimasukkan selain huruf e dan d maka program akan mengeluarkan tampilan sebagai berikut.




*Gambar 5.5 Pemilihan Enkripsi-Dekripsi dengan Input selain e dan d*
*Sumber : Dokumen Milik Pribadi*

4. Proses Enkripsi

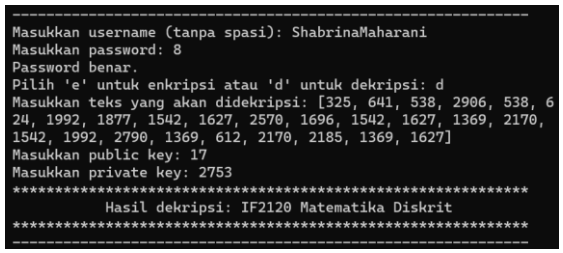



*Gambar 5.6 Proses Enkripsi*
*Sumber : Dokumen Milik Pribadi*

Proses enkripsi akan mengeluarkan list yang berupa hasil enkripsi dari plainteks yang diubah dengan Caesar Cipher, kemudian diperkuat lagi dengan algoritma RSA. Program juga akan mengeluarkan kunci publik yang dapat diketahui oleh siapapun itu.

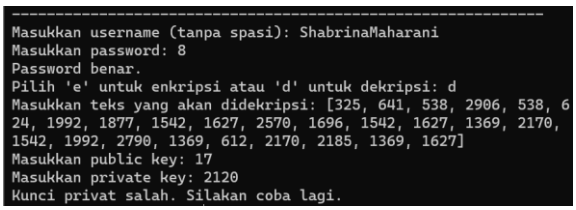
5. Proses Dekripsi

Terdapat dua kondisi yang mungkin terjadi dalam proses ini. Pertama, jika kunci publik dan kunci privat yang dimasukkan sesuai, maka program akan mengeluarkan tampilan sebagai berikut.




*Gambar 5.7 Proses Dekripsi dengan Kunci Publik-Privat sesuai*
*Sumber : Dokumen Milik Pribadi*

Kedua, apabila kunci publik dan kunci privat yang dimasukkan tidak sesuai, maka program akan mengeluarkan tampilan sebagai berikut.




*Gambar 5.8 Proses Dekripsi dengan Kunci Publik-Privat tidak sesuai*
*Sumber : Dokumen Milik Pribadi*

Pengembangan program buatan penulis dapat lebih lanjut dilihat dengan mengakses *repository* github di bawah ini :
https://github.com/Maharanish/Cryptography-Secret-Generator.git

## VI. KESIMPULAN

Melalui penerapan kombinasi algoritma RSA, Caesar Cipher, dan fungsi hash dalam *Cryptography Secret Generator*, keamanan informasi dapat ditingkatkan secara signifikan. Penggunaan fungsi hash untuk autentikasi pengguna, bersamaan dengan enkripsi Caesar Cipher yang sederhana dan enkripsi RSA yang kompleks, menciptakan lapisan keamanan ganda. Dengan demikian, pesan yang dihasilkan tidak hanya terenkripsi secara menyeluruh, tetapi juga sulit diakses oleh pihak yang tidak berwenang. Dalam konteks *information hiding*, program ini menjadikan informasi lebih terlindungi, memastikan bahwa hanya orang-orang yang memiliki kunci dan otoritas yang sesuai yang dapat mengakses isi pesan asli, meningkatkan keamanan dan kerahasiaan data.

## VII. UCAPAN TERIMA KASIH

Segenap rasa syukur penulis panjatkan kepada Allah SWT, sumber segala kebijaksanaan, atas rahmat dan petunjuk-Nya yang melimpah, memandu penulis dalam menyelesaikan makalah ini. Terima kasih tak terhingga penulis sampaikan untuk Dr. Ir. Rinaldi Munir, M.T. dan Monterico Adrian, S.T., M.T., selaku dosen pengampu mata kuliah IF2120 Matematika Diskrit kelas K03, yang dengan penuh dedikasi memberikan bimbingan, ilmu, dan inspirasi yang menjadi landasan utama penulisan. Penghargaan setinggi-tingginya juga diberikan kepada orang tua dan teman-teman penulis yang senantiasa memberikan dukungan moral, motivasi, dan semangat. Terima kasih juga kepada semua pihak yang berkontribusi dengan adanya referensi dan bahan pendukung, memperkaya pemahaman penulis terhadap materi. Semua bantuan, dukungan, dan kontribusi ini menjadi pondasi penting dalam kelancaran penyusunan makalah ini. Semoga hasil karya ini dapat memberikan manfaat dan kontribusi positif bagi pembaca.

## REFERENSI

[1] Syahriszani, Rezi. 2023. Pengertian Hash. Ilmuelektro.id. https://ilmuelektro.id/hash-adalah/ (diakses pada 8 Desember 2023, pukul 19.19 WIB).
[2] Sukardi. 2022. Materi, Soal, dan Pembahasan – Kongruensi Modulo. Mathcyber1997.https://mathcyber1997.com/materi-soal-dan-pembahasan-kongruensi-modulo/ (diakses pada 8 Desember 2023, pukul 18.51 WIB).
[3] Sereliciouz, Natasa, Bilangan Bulat – Pengertian, Jenis, Operasi Hitung. 2021. Quipper. https://www.quipper.com/id/blog/mapel/matematika/bilangan-bulat/#1_Bilangan_bulat_positif (diakses pada 8 Desember 2023, pukul 18.17 WIB).
[4] Johnston, Phillip. 2020. Information Hiding. Embedded Artistry LLC. https://embeddedartistry.com/fieldmanual-terms/information-hiding/ (diakses pada 8 Desember 2023, pukul 19.27 WIB).
[5] Munir, Rinaldi. (2023). Teori Bilangan (Bagian 1). https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/14-Teori-Bilangan-Bagian1-2023.pdf (diakses pada 8 Desember 2023).
[6] Munir, Rinaldi. (2023). Teori Bilangan (Bagian 2). https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/14-Teori-Bilangan-Bagian3-2023.pdf (diakses pada 8 Desember 2023).
[7] Munir, Rinaldi. (2022). Pengantar Kriptografi. https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/01-Pengantar-Kriptografi-(2023).pdf (diakses pada 8 Desember 2023).
[8] Munir, Rinaldi. (2022). Ragam Cipher Klasik (Bagian 1). https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/02-Ragam-Cipher-Klasik-Bagian1-(2023).pdf (diakses pada 8 Desember 2023).
[9] Munir, Rinaldi. (2020). Steganografi (Bagian 1). https://informatika.stei.itb.ac.id~rinaldi.munir/Kriptografi/2020-2021/Steganografi-Bagian1-2020.pdf (diakses pada 8 Desember 2023).
[10] Munir, Rinaldi. (2023). Algoritma RSA. https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/19-Algoritma-RSA-2023.pdf (diakses pada 8 Desember 2023).
[11] Rosen, Kenneth H. 2012. DISCRETE MATHEMATICS AND ITS APPLICATIONS, SEVENTH EDITION. The McGraw-Hill Companies. United States. Bab 4, halaman 237-306.

## PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Desember 2023

Shabrina Maharani 13522134