MATHEMATICAL SCIENCES AND APPLICATIONS JOURNAL

Volume 5, Number 1 | ISSN: 2986-4180

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

Article

Aplikasi Teorema Fermat dalam Kriptografi

Hanifah Al Affiani, Muhammad Deni Johansyah, Sisilia Sylviani*

Departemen 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

Abstract

Cryptography is the science and technique of disguising messages in a unique form so that they can only be read and processed by the intended recipient. Many studies have been conducted to develop algorithms that can be used to encode information in a way that is difficult to crack and cannot be recognized by adversaries. One example of the most popular algorithms is the Rivest-Shamir-Adleman (RSA), which uses different key pairs for the encryption and decryption process of messages, usually known as the public key and private key. In public key-based encryption systems such as RSA, Fermat's theorem plays an important role because it enables modular exponential calculations on key pairs to be performed efficiently and provides a security basis for the RSA algorithm. Thus, this research aims to describe the application of Fermat's theorem in the RSA algorithm, where the encryption and decryption process involves modular exponentiation with public and private keys. As a result, using the properties of modular exponentiation in Fermat's theorem, this system ensures information remains secure from attacks by third parties without access to the private key, even if they succeed in intercepting encrypted messages. It can be concluded that Fermat's theorem plays a crucial role in establishing a solid mathematical foundation for creating secure and efficient cryptographic systems.

Keyword:
Fermat's theorem
Cryptography
RSA algortihm

1. Pendahuluan

Teori bilangan menjadi dasar dalam pengembangan kriptografi dalam hal menjaga keamanan dan kerahasiaannya. Salah satu contohnya adalah teorema Fermat yang menyatakan jika p adalah bilangan prima dan a adalah bilangan bulat yang tidak habis dibagi oleh p, maka $a^{p-1} \equiv 1 \ (mod\ p)$. Menurut teorema ini, prinsip ini memfasilitasi operasi eksponensial modular yang menjadi dasar algoritma seperti Rivest-Shamir-Adleman (RSA), El Gamal, dan Diffie-Hellman, memungkinkan proses enkripsi dan dekripsi dilakukan dengan efisien. (1-2].

Aplikasi teorema Fermat dalam kriptografi dapat memberi ketahanan terhadap serangan *brute force,* yaitu teknik yang digunakan dalam kriptografi dengan mencoba semua kemungkinan kombinasi secara sistematis untuk menemukan solusi atau meretas pesan dari pihak lawan. Selain itu, pada teorema Fermat, eksponensiasi modular memungkinkan pengolahan bilangan besar dalam modulo prima untuk memastikan hanya pihak dengan kunci privat yang dapat mendekripsi pesan. Di era digital saat ini, ancaman terhadap keamanan terus meningkat, termasuk dari potensi serangan komputer, sehingga menjadi motivasi pengembangan metode kriptografi baru. Meskipun demikian, eksponensiasi modular tetap menjadi bagian integral dari banyak algoritma dan menjaga dasar keamanan sistem seperti RSA [3-4].

RSA merupakan salah satu algoritma kriptografi yang menggunakan pasangan kunci berbeda umtuk proses enkripsi dan dekripsi pesan yang biasa dikenal kunci publik dan kunci privat. Keunggulan RSA didasarkan pada kesulitannya dalam memfaktorkan bilangan besar menjadi faktor-faktor prima. Selain RSA, konsep ini juga diterapkan dalam sistem lain seperti ElGamal, yang menggunakan eksponensiasi modular untuk menciptakan kunci aman. Keunggulan sistem ini terletak pada penggunaan bilangan prima besar, di mana sifat

https://doi.org/10.22437/msa.v5i1.38094

eksponensial modulonya memberikan prediktabilitas yang aman untuk operasi kriptografi. Dengan demikian, teorema Fermat terus menjadi fondasi matematis yang kokoh untuk menjaga kerahasiaan dan integritas data dalam berbagai aplikasi digital [5-6].

Berbagai penelitian telah dilakukan untuk memahami dan mengembangkan penerapan teorema Fermat dalam bidang kriptografi, khususnya pada algoritma seperti RSA. Zhang dan Wang (2012) mengkaji efisiensi algoritma RSA berbasis teorema Fermat dan menunjukkan bagaimana teorema ini mempercepat proses enkripsi dan dekripsi melalui eksponensiasi modular [1]. Galbraith (2013) serta Silverman dan Hoffstein (2014) juga membahas pentingnya teorema Fermat dalam membangun algoritma kunci publik seperti RSA, ElGamal, dan Diffie-Hellman, dengan menyoroti ketahanan terhadap serangan brute force dan keamanan yang dihasilkan dari manipulasi bilangan prima besar [2-3].

Meski demikian, penelitian-penelitian sebelumnya sebagian besar berfokus pada efisiensi algoritma dan penerapan dalam lingkungan komputasi klasik. Penelitian seperti Barak dan Boneh (2018) menunjukkan relevansi algoritma ini dalam keamanan modern, tetapi kurang membahas bagaimana elemen matematis fundamental seperti teorema Fermat dapat diimplementasikan secara langsung dalam enkripsi dan dekripsi pada sistem RSA [4]. Selain itu, Shi dan Yuan (2019) mengembangkan algoritma eksponensiasi modular yang lebih cepat menggunakan teorema Fermat, namun penelitian ini masih kurang dalam memberikan penjelasan praktis mengenai keterkaitan langsung antara mekanisme teorema Fermat dan aplikasi kunci publik dalam RSA [22].

Berdasarkan penelitian-penelitian sebelumnya, diperlukan eksplorasi lebih mendalam mengenai penerapan prinsip eksponensial modular dari teorema Fermat secara rinci dalam algoritma RSA. Penelitian ini akan membahas bagaimana prinsip tersebut digunakan dalam proses enkripsi dan dekripsi pada sistem kunci publik. Selain itu, penelitian ini juga menjelaskan bagaimana algoritma RSA memanfaatkan sifat-sifat matematis dari teorema Fermat untuk menghasilkan kunci enkripsi yang aman, serta memastikan bahwa hanya pihak yang memiliki kunci privat yang dapat mengakses informasi asli. Dengan memahami mekanisme ini, penelitian ini diharapkan dapat memberikan pemahaman yang lebih komprehensif mengenai peran teori bilangan dalam membangun sistem keamanan yang efektif dan efisien untuk melindungi data digital.

2. Metode Penelitian

2.1 Teorema Fermat

Teorema Fermat dikemukakan oleh Pierre de Fermat, seorang matematikawan Prancis terkenal dari abad ke-17.

Teorema 1. [11] Jika p adalah bilangan prima dan a adalah bilangan bulat yang tidak habis dibagi oleh p, dengan kata lain $\gcd(a,p) = 1$), maka:

$$a^{p-1} \equiv 1 \,(mod\ p) \qquad (1)$$

Bukti. Misalkan $\mathbb{Z}_p$ adalah himpunan bilangan bulat modulo p yang relatif prima dengan p. Misalkan $A = \{1, 2, \dots, p-1\}$, jika setiap elemen di himpunan A dikalikan dengan $a \bmod p$, dengan a adalah bilangan bulat positif yang relatif prima dengan p, maka didapat himpunan baru yang dapat dinyatakan dengan $B = \{1a \bmod p, 2a \bmod p, \dots, (p-1)a \bmod p\}$.

Selanjutnya, klaim A = B,

- $0 \notin B$, karena $p \nmid ja$, untuk suatu $j \in \mathbb{Z}$ maka B ⊂ A,
- Jika $i \in \mathbb{Z}$ dengan $i \neq j$, maka $ia \bmod p \neq ja \bmod p$

Karena $p \nmid (j-i)a$, dengan demikian$|B| = p - 1$, diperoleh $A = B$. Sehingga,

$$\Pi_{i\in A}\, i \equiv \Pi_{i\in B}\, i \,(mod\ p)$$
$$1.2 \dots .(p-1) \equiv a(2a) \dots \big((p-1)a\big) \ (mod\ p),$$
$$(p-1)! \equiv (p-1)!\ a^{p-1} (mod\ p),$$

dapat dinyatakan sebagai,

$$(a^{\,p-1} - 1)(p-1)! = kp, \text{ untuk suatu } k \in \mathbb{Z}.$$

Dengan kata lain,

$$p \mid (a^{\,p-1} - 1)(p-1)!$$

karena $\gcd(p, ((p-1)!) = 1$, artinya p tidak membagi $(p-1)!$ sehingga p haruslah membagi $(a^{\,p-1} - 1)$ atau $p \mid (a^{p-1} - 1)$ artinya $(a^{p-1} - 1) = lp$, untuk suatu $l \in \mathbb{Z}$. Sehingga diperoleh,

$$a^{p-1} = lp + 1 \Longrightarrow a^{p-1} \equiv 1 \,(mod\ p) \blacksquare$$

Jadi, terbukti bahwa $a^{p-1} \equiv 1 \,(mod\ p)$.

Teorema 2. [11] Misalkan $n = pq$, p dan q merupakan bilangan prima, tulis $m = (p-1)(q-1)$, dan misalkan $k > 2$ suatu bilangan bulat sedemikian sehingga $k \equiv 1(mod\ m)$, sehingga untuk setiap x dengan $gcd\ (x,n) = 1$, berlaku

$$x^k \equiv x(mod\ n) \qquad (2)$$

Bukti. Diketahui bahwa $k \equiv 1(mod\ m)$ yang dapat dinyatakan sebagai $k = 1 + lm$, untuk suatu $l \in \mathbb{Z}$. Dengan demikian, $x^k = x^{1+lm} = x.(x^m)^l$, dengan $m = (p-1)(q-1)$ dan $n = pq$, dengan p, q merupakan bilangan prima.

Akan dibuktikan bahwa $x^k \equiv x(mod\ n)$. Perhatikan jika $x^m \equiv 1(mod\ n)$, maka persamaan menjadi $x^k = x(x^m)^l = x(1)^l = x$, dengan kata lain $x^k \equiv x(mod\ n)$. Jadi cukup ditunjukkan bahwa $x^m \equiv 1(mod\ n)$ dimana $\gcd(x, n) = 1$ dimana $n = pq$. Kondisi ini menyiratkan bahwa p tidak membagi x. Oleh karena itu, dengan menggunakan Teorema Fermat,

$$x^{p-1} \equiv 1(mod\ p),$$

diperoleh

$$x^m = (x^{p-1})^{q-1} \equiv 1^{q-1} \equiv 1(mod\ p).$$

Demikian pula, $x^m \equiv 1(mod\ q)$ dan karena p and q relatif prima, maka $x^m \equiv 1(mod\ pq)$. ■

Telah dibuktikan bahwa $x^m \equiv 1(mod\ pq)$, maka bukti selesai.

Pada pengaplikasian teorema Fermat dalam kriptografi, untuk menemukan faktor persekutuan terbesar (FPB) atau *greatest common divisor* (gcd) antara dua bilangan, yang menjadi langkah dasar dalam berbagai operasi enkripsi dapat digunakan algoritma Euclid.

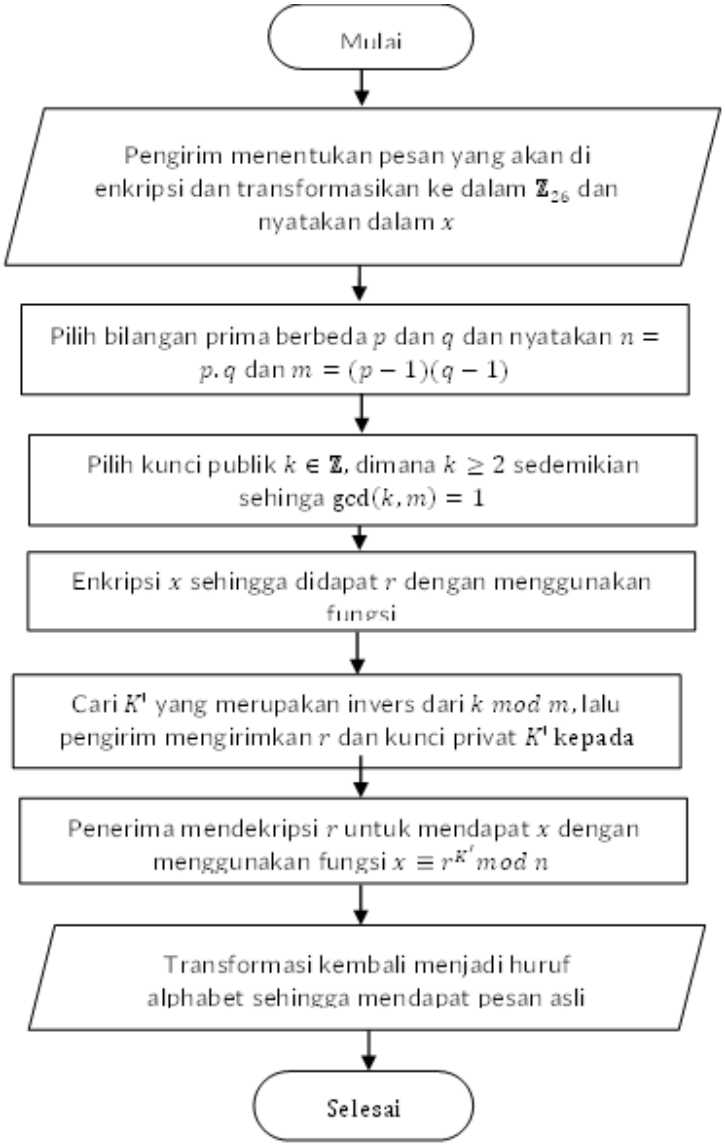
Definisi 3. [7] Misalkan m dan n bilangan bulat, $n > 0$. Jika m dibagi dengan n maka hasil pembagiannya adalah q *(quotient)* dan sisanya r *(remainder),* sedemikian sehingga

$$m = nq + r, 0 \leq r < n. \qquad (3)$$

Keterkaitan algoritma Euclid dalam teorema Fermat, yaitu algoritma ini dapat membantu memastikan bahwa nilai-nilai yang digunakan dalam operasi eksponensial modular adalah valid. Terutama dalam memastikan bahwa bilangan yang digunakan tidak habis dibagi oleh bilangan prima p. Dengan kata lain, algoritma Euclid membantu memastikan kondisi dasar yang diperlukan agar Teorema Fermat dapat diterapkan secara benar dalam perhitungan kriptografi.

2.2 Algoritma Kriptografi

Kriptografi adalah ilmu dan teknik untuk menyamarkan pesan agar hanyadapat dibaca oleh pihak yang berwenang membacanya. Penelitian mengenai algoritma kriptografi telah banyak dikembangkan. Salah satu contoh algoritma yang paling populer adalah algoritma Rivest-Shamir-Adleman (RSA).



Gambar 1. Diagram Alir Penelitian

Cara kerja dari algoritma RSA dapat diilustrasikan, sebagai berikut. Misalkan terdapat dua orang Budi dan Ani yang akan bertukar pesan, Budi harus mengirimkan kunci publik (k, n) kepada Ani, lalu Ani menggunakan kunci tersebut untuk mengenkripsi pesan asli (x). Kemudian mengirimkan *ciphertext* (r) dan kunci privat (K', n) kepada Budi yang selanjutnya akan digunakan untuk mendekripsi r dan membaca pesan aslinya. Pasangan kunci privat ini harus dirahasiakan dengan baik dan hanya pemilik kunci privat yang dapat mendekripsi pesan dan membaca pesan aslinya. Dengan demikian, keamanan algoritma RSA dikenal karena kesulitannya memfaktorkan bilangan yang besar menjadi faktor-faktor prima. Pemfaktoran ini dilakukan untuk memperoleh kunci privat yang akan digunakan untuk mendekripsi

pesan asli. Algoritma RSA didasarkan pada teorema Euler yang menyatakan bahwa ([2],[7]).

$$a^{\phi(n)} \equiv 1(mod\ n), \qquad (4)$$

dengan a harus relatif prima terhadap n dan $\phi(n) = n\left(1-\frac{1}{p_1}\right)\left(1-\frac{1}{p_2}\right)\ldots\left(1-\frac{1}{p_r}\right)$, dalam hal ini $p_1, p_2, \ldots, p_r$ adalah faktor prima dari n.

Adapun *plaintext* dalam penelitian ini dimodifikasi dengan mentrasformasi alphabet ke dalam $\mathbb{Z}_{26}$ dan algoritma yang digunakan disajikan dalam diagram alir pada Gambar 1.

3. Hasil dan Pembahasan

Pada pembahasan ini akan diperlihatkan aplikasi teorema fermat pada kriptografi, dimana algoritma yang digunakan adalah algoritma RSA dengan fungsi enkripsi:

$$x^k \equiv r(mod\ n), \qquad (5)$$

di mana,
x: pesan yang dienkripsi,
r: *ciphertext* yang akan dicari,
n: hasil kali dua bilangan prima pq,
k: eksponen enkripsi yang dipilih sedemikian sehingga $\gcd(k, m) = 1$, dengan $m = (p-1)(q-1)$.

Pada proses dekripsi diperlukan invers dari k, sehingga terlebih dahulu akan diselidiki eksistensi inverse dari k, sebut K'. Hal ini dapat ditunjukan sebagai berikut. Jika K'merupakan invers dari $k\ modulo\ m$, maka

$$K'k = 1 + km.$$

Sebelumnya diketahui bahwa $\gcd(k, m) = 1$, sehingga $1 = sk + tm$ atau dapat dinyatakan sebagai $1 \equiv sk\ mod\ m$. Hal ini mengakibatkan $K' = s$. Jadi, terbukti bahwa invers dari k selalu ada.
Dalam kasus dunia nyata, $\gcd(x, p) = 1$ dengan p merupakan bilangan prima yang sangat besar dan x bilangan bulat yang kecil. Berdasarkan Teorema Fermat,

$$x^{p-1} \equiv 1(mod\ p)$$
$$(x^{p-1})^{k(q-1)} \equiv 1^{k(q-1)}(mod\ p)$$
$$x.(x^{p-1})^{k(q-1)} \equiv x.1\ (mod\ p)$$

$$x^{(1+k)(p-1)(q-1)} \equiv x(mod\ p)$$

Dengan mengganti p menjadi q, diperoleh

$$x^{(1+k)(p-1)(q-1)} \equiv x(mod\ q).$$

Sementara itu pada $n = pq$, diperoleh

$$x^{(1+k)(p-1)(q-1)} \equiv x(mod\ pq) \equiv x(mod\ n).$$

Selanjutnya dari kekongruenan akan dicari fungsi dekripsinya,

$$r^{K'} \equiv x^{(1+k)(p-1)(q-1)} \qquad (6)$$
$$x^{(1+k)(p-1)(q-1)} \equiv x(mod\ n) \qquad (7)$$

Dari 6 dan 7 diperoleh $r^{K'} \equiv x(mod\ n)$. Dengan demikian, didapat fungsi dekripsi yaitu:

$$x \equiv r^{K'} mod\ n.$$

Setelah mengetahui fungsi enkripsi dan dekripsi, dalam proses pengkodean untuk menyamarkan pesan asli, tahap paling awal yang dilakukan adalah mentransformasikan dari alfabet ke bilangan yang termuat dalam $\mathbb{Z}_{26}$ (Tabel 1).

Contoh 1. Akan dienkripsi pesan "I" dengan $p = 11, q = 13$ dan $k = 7$. Kemudian, dilanjutkan dengan proses dekripsinya.

Penyelesaian. Pesan yang dienkripsi yaitu "I" berkorespondensi dengan $8 =$ "I". Pilih bilangan prima $p = 11, q = 13$, sehingga, $n = p.q = 11\,.13 = 143$, dan $m = (11-1)(13-1) = 10\,.12 = 120$. Kemudian pilih $k = 7 \ni \gcd(k, m) = \gcd(7, 120) = 1$. Enkripsi $x = 8$ untuk kedua blok dengan menggunakan fungsi $x^k \equiv r\ mod\ n, 0 \le r < n$, sedemikian sehingga $8^7 \equiv r\ mod\ 120, 0 \le r < 120$.

$$8^2 \equiv 64\ mod\ 120$$
$$8^3 \equiv 83\ mod\ 120$$
$$8^4 \equiv 92\ mod\ 120$$
$$\vdots$$
$$8^7 \equiv 57\ mod\ 120$$

sehingga, $r = 57$.

Tabel 1. Korespondensi Alfabet dan $\mathbb{Z}_{26}$

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	34	25

Kemudian cari K' invers dari $k = 7\ (mod\ 120)$ dengan menggunakan algoritma Euclid, diperoleh $7K' \equiv 1(mod\ 120)$ atau bisa dinyatakan dengan $7K' + 120y = 1$, sehingga

$$1 = 103\ (7) - 6(120),$$

dengan $K' \equiv 103$. Berikut dilampirkan program Phyton untuk mengenkripsi pesan x dan invers dari k. Program ini disertakan sebagai pelengkap dari penyelesaian sebelumnya di mana hasil yang didapat sesuai (Gambar 2).

Selanjutnya, dengan mengirimkan $r = 57$ dan $K' = 103$, penerima dapat melakukan proses dekripsi dengan menggunakan fungsi $x \equiv r^{K'} mod\ n$. Berdasarkan sifat eksponen dalam operasi modulo, $K' = 103$, perhatikan bahwa

$$103 \equiv 1100111,$$

dapat diubah menjadi $103 \equiv 2^0 + 2^1 + 2^2 + 2^5 + 2^6$. Perhatikan untuk $r = 57$, penerima dapat menghitung pemangkatan berulang dari $57\ mod\ 143$. Yaitu,

$$57^1 \equiv 57\ mod\ 143$$
$$57^{2^1} \equiv 103\ mod\ 143$$
$$57^{2^2} \equiv 27\ mod\ 143$$
$$57^{2^5} \equiv 92\ mod\ 143$$
$$57^{2^6} \equiv 27\ mod\ 143$$

Sehingga, $x \equiv 57^{103} mod\ 143 \equiv 57^{1+2^1+2^2+2^5+2^6} \equiv 57 \times 103 \times\ 27 \times\ 92 \times\ 27 \equiv 8$. Jadi, diperoleh $x = 8$ yang berkorespondensi dengan "I".

```
# Mendefinisikan ulang parameter dan fungsi RSA setelah reset
p = 11
q = 13
n = p * q
k = 7
phi_n = (p - 1) * (q - 1)

# Algoritma Euclid yang diperluas untuk menemukan invers modular dari k (kunci privat d)
def egcd(a, b):
    if a == 0:
        return b, 0, 1
    gcd, e1, y1 = egcd(b % a, a)
    e = y1 - (b // a) * e1
    y = e1
    return gcd, e, y

# Fungsi untuk menemukan invers modular
def modinv(a, x):
    gcd, e, y = egcd(a, x)
    if gcd != 1:
        raise Exception('Tidak ada invers modular')
    else:
        return e % x
```

```
# Menemukan kunci privat d
d = modinv(k, phi_n)

# Pesan yang akan dienkripsi (x = 0)
x = input("Masukkan x: ")

# Mengonversi x menjadi bilangan bulat sebelum menggunakannya dalam pow()
x = int(x) # Baris ini mengonversi input string menjadi bilangan bulat.

# Enkripsi RSA: r = x^k mod n
r = pow(x, k, n)

# Dekripsi RSA: x_decrypted = r^d mod n
m_decrypted = pow(r, d, n)

r, m_decrypted
print("Hasil enkripsi:", r)
print("Invers dari k:", d)
```

```
Masukkan x: 8
Hasil enkripsi: 57
Invers dari k: 103
```

Gambar 2. Program untuk mengenkripsi masalah 1

Contoh 2. Akan dienkripsi pesan "CAN" dengan $p = 41, q = 43$ dan $k = 23$. Kemudian, dilanjutkan dengan proses dekripsinya.

Penyelesaian. Pesan yang dienkripsi yaitu "CAN" berkorespondensi dengan $2 = \text{C}", 0 = "A", 13 = "N"$. Kelompokan menjadi 3 blok, sehingga didapat "2 0 13". Pilih bilangan prima $p = 41, q = 43$, sehingga

$n = p.q = 41 . 43 = 1763$, dan $m = (41 - 1)(43 - 1) = 40 . 42 = 1680$. Kemudian pilih $k = 23 \ni \gcd(k, m) = \gcd(23, 1680) = 1$. Enkripsi x untuk ketiga blok dengan fungsi $x^k \equiv r \; mod \; n, 0 \leq r < n$.

- Untuk blok 1: $x_1 = 2$, sedemikian sehingga $2^{23} \equiv r_1 \; mod \; 1763, 0 \leq r_1 < 1763$.

$$2^2 \equiv 4 \; mod \; 1763$$
$$2^3 \equiv 8 \; mod \; 1763$$
$$2^4 \equiv 16 \; mod \; 1763$$
$$\vdots$$
$$2^{23} \equiv 254 \; mod \; 1763$$

diperoleh, $r_1 = 254$.

- Untuk blok 2: $x_2 = 0$, sedemikian sehingga $0^{23} \equiv r_2 \; mod \; 1763, 0 \leq r_2 < 1763$.

$$0^{23} \equiv 0 \; mod \; 1763$$

diperoleh, $r_2 = 0$.

- Untuk blok 3: $x_3 = 13$, sedemikian sehingga $13^{23} \equiv r_3 \; mod \; 1763, 0 \leq r_3 < 1763$.

$$13^2 \equiv 169 \; mod \; 1763$$
$$13^3 \equiv 434 \; mod \; 1763$$
$$13^4 \equiv 353 \; mod \; 1763$$
$$\vdots$$
$$13^{23} \equiv 427 \; mod \; 1763$$

diperoleh, $r_3 = 427$.

```python
# Mendefinisikan ulang parameter dan fungsi RSA setelah reset
p = 41
q = 43
n = p * q
k = 23
phi_n = (p - 1) * (q - 1)

# Algoritma Euclid yang diperluas untuk menemukan invers modular dari k (kunci privat d)
def egcd(a, b):
    if a == 0:
        return b, 0, 1
    gcd, e1, y1 = egcd(b % a, a)
    e = y1 - (b // a) * e1
    y = e1
    return gcd, e, y

# Fungsi untuk menemukan invers modular
def modinv(a, x):
    gcd, e, y = egcd(a, x)
    if gcd != 1:
        raise Exception('Tidak ada invers modular')
    else:
        return e % x

# Menemukan kunci privat d
d = modinv(k, phi_n)

# Daftar nilai x yang akan diuji
x_values = [2, 0, 13]

# Loop untuk menguji enkripsi dan dekripsi untuk setiap nilai x
for x in x_values:
    # Enkripsi RSA: r = x^k mod n
    r = pow(x, k, n)

    # Dekripsi RSA: x_decrypted = r^d mod n
    m_decrypted = pow(r, d, n)
    print(f"Untuk x = {x}:")
    print(f"Hasil enkripsi: {r}")
print("Invers dari k:", d)
```

```
Untuk x = 2:
Hasil enkripsi: 254
Untuk x = 0:
Hasil enkripsi: 0
Untuk x = 13:
Hasil enkripsi: 427
Invers dari k: 1607
```

Gambar 3. Program untuk mengenkripsi masalah 2

Jadi, dengan menggabungkan ketiga blok diperoleh hasil enkripsi menjadi "254 0 427".

Kemudian cari K' invers dari $k = 23\ (mod\ 1680)$ dengan menggunakan algoritma Euclid, diperoleh $23K' \equiv 1(mod\ 1680)$ atau bisa dinyatakan dengan $23K' + 1680y = 1$, sehingga

$$1 = -73\ (23) + 1(1680),$$

dengan $K' = -73\ mod\ 1680 = 1607$.

Gambar 3 merupakan program Phyton untuk mengenkripsi pesan x dan invers dari k. Program ini disertakan sebagai pelengkap dari penyelesaian sebelumnya di mana hasil yang didapat sesuai.

Selanjutnya, dengan mengirimkan $r = 254\ 0\ 427$ dan $K' = 1607$, penerima dapat melakukan proses dekripsi dengan menggunakan fungsi $x \equiv r^{K'} mod\ n$. Berdasarkan sifat eksponen dalam operasi modulo, $K' = 1607$, perhatikan bahwa

$$1607 \equiv 11001000111,$$

dapat diubah menjadi, $1607 = 2^0 + 2^1 + 2^2 + 2^6 + 2^9 + 2^{10}$.

- Untuk blok 1, $r_1 = 254$ dan penerima dapat menghitung pemangkatan berulang dari $254\ mod\ 1763$.

$$\begin{aligned} 254^1 &\equiv 254\ mod\ 1763 \\ 254^{2^1} &\equiv 1048\ mod\ 1763 \\ 254^{2^2} &\equiv 1718\ mod\ 1763 \\ 254^{2^6} &\equiv 1595\ mod\ 1763 \\ 254^{2^9} &\equiv 305\ mod\ 1763 \\ 254^{2^{10}} &\equiv 1349\ mod\ 1763 \end{aligned}$$

diperoleh,

$$\begin{aligned} x_1 \equiv 254^{1607} &\equiv 2081^{1+2^1+2^2+2^6+2^9+2^{10}} \\ &\equiv 254 \times 1048 \times 1718 \times 1595 \times 305 \times 1349 \\ &\equiv 2. \end{aligned}$$

- Untuk blok 2, $r_2 = 0$, sedemikian sehingga didapat $x_2 \equiv 0^{1607} \equiv 0$
- Untuk blok 3, $r_3 = 427$ dan penerima dapat menghitung pemangkatan berulang dari $427\ mod\ 1763$.

$$\begin{aligned} 427^1 &\equiv 427\ mod\ 1763 \\ 427^{2^1} &\equiv 740\ mod\ 1763 \\ 427^{2^2} &\equiv 1070\ mod\ 1763 \\ 427^{2^6} &\equiv 857\ mod\ 1763 \\ 427^{2^9} &\equiv 756\ mod\ 1763 \\ 427^{2^{10}} &\equiv 324\ mod\ 1763 \end{aligned}$$

diperoleh,

$$\begin{aligned} x_3 \equiv 254^{1607} &\equiv 2081^{1+2^1+2^2+2^6+2^9+2^{10}} \\ &\equiv 427 \times 740 \times 1070 \times 857 \times 756 \times 324 \\ &\equiv 13. \end{aligned}$$

Jadi, dengan menggabungkan ketiga blok didapat $x =$ "2 0 13" yang berkorespondensi dengan "CAN".

Contoh 3. Akan dienkripsi pesan "$STOP$" dengan $p = 43, q = 59$ dan $k = 13$. Kemudian, dilanjutkan dengan proses dekripsinya.

Penyelesaian. Pesan yang dienkripsi yaitu "$STOP$" berkorespondensi dengan $18 =$ "S", $19 =$ "T", $14 =$ "O", $15 =$ "P". Kelompokkan menjadi 2 blok, sehingga didapat "1819 1415". Pilih bilangan prima $p = 43, q = 59$, sehingga $n = p.q = 43\,.59 = 2537$, dan $m = (43-1)(59-1) = 42\,.58 = 2436$. Kemudian pilih $k = 13 \ni \gcd(k, m) = \gcd(13, 2436) = 1$. Enkripsi x untuk kedua blok dengan menggunakan fungsi $x^k \equiv r\ mod\ n, 0 \leq r < n$.

- Untuk blok 1: $x_1 = 1819$, sedemikian sehingga $1819^{13} \equiv r_1\ mod\ 2537, 0 \leq r < 2537$.

$$\begin{aligned} 1819^2 &\equiv 513\ mod\ 2537 \\ 1819^3 &\equiv 2068\ mod\ 2537 \\ 1819^4 &\equiv 1858\ mod\ 2537 \\ &\vdots \\ 1819^{13} &\equiv 2081\ mod\ 2537 \end{aligned}$$

diperoleh, $r_1 = 2081$.

- Untuk blok 2: $x_2 = 1415$, sedemikian sehingga $1415^{13} \equiv r_1\ mod\ 2537, 0 \leq r < 2537$.

$$\begin{aligned} 1415^2 &\equiv 532\ mod\ 2537 \\ 1415^3 &\equiv 1828\ mod\ 2537 \\ 1415^4 &\equiv 1417\ mod\ 2537 \\ &\vdots \\ 1415^{13} &\equiv 2182\ mod\ 2537 \end{aligned}$$

diperoleh, $r_2 = 2182$.

Jadi, pesan asli "$STOP$" dienkripsi menjadi "2081 2182". Berikut dilampirkan program Phyton untuk mengenkripsi pesan x dan invers dari k. Program ini disertakan sebagai pelengkap dari penyelesaian sebelumnya di mana hasil yang didapat sesuai Kemudian cari K' invers dari $k = 13\ (mod\ 2436)$. Dengan menggunakan algoritma Euclid, diperoleh $13K' \equiv 1(mod\ 2436)$ atau bisa dinyatakan dengan $13K' + 2436y = 1$, sehingga

$$1 = 937\ (13) - 5(2436)$$

dengan $K' = 937$.

```
# Mendefinisikan ulang parameter dan fungsi RSA setelah reset
p = 43
q = 59
n = p * q
k = 13
phi_n = (p - 1) * (q - 1)

# Algoritma Euclid yang diperluas untuk menemukan invers modular dari k (kunci privat d)
def egcd(a, b):
    if a == 0:
        return b, 0, 1
    gcd, e1, y1 = egcd(b % a, a)
    e = y1 - (b // a) * e1
    y = e1
    return gcd, e, y

# Fungsi untuk menemukan invers modular
def modinv(a, x):
    gcd, e, y = egcd(a, x)
    if gcd != 1:
        raise Exception('Tidak ada invers modular')
    else:
        return e % x

# Menemukan kunci privat d
d = modinv(k, phi_n)
```

```
# Daftar nilai x yang akan diuji
x_values = [1819, 1415]

# Loop untuk menguji enkripsi dan dekripsi untuk setiap nilai x
for x in x_values:
    # Enkripsi RSA: r = x^k mod n
    r = pow(x, k, n)

    # Dekripsi RSA: x_decrypted = r^d mod n
    m_decrypted = pow(r, d, n)
    print(f"Untuk x = {x}:")
    print(f"Hasil enkripsi: {r}")
print("Invers dari k:", d)
```

```
Untuk x = 1819:
Hasil enkripsi: 2081
Untuk x = 1415:
Hasil enkripsi: 2182
Invers dari k: 937
```

Gambar 4. Program untuk mengenkripsi masalah 3

Selanjutnya, dengan mengirimkan $r = 2081\ 2182$ dan $K' = 937$, penerima dapat melakukan proses dekripsi dengan menggunakan fungsi $x \equiv r^{K'} mod\ n$. Berdasarkan sifat eksponen dalam operasi modulo, $K' = 937$, perhatikan bahwa

$$937 \equiv 1110101001,$$

dapat diubah menjadi, $937 = 2^0 + 2^3 + 2^5 + 2^7 + 2^8 + 2^9$.

- Untuk blok 1, $r_1 = 2081$ dan penerima dapat menghitung pemangkatan berulang dari $2081\ mod\ 2537$.

$$2081^1 \equiv 2081\ mod\ 2537$$
$$2081^{2^3} \equiv 1644\ mod\ 2537$$
$$2081^{2^5} \equiv 497\ mod\ 2537$$
$$2081^{2^7} \equiv 1579\ mod\ 2537$$
$$2081^{2^8} \equiv 1907\ mod\ 2537$$
$$2081^{2^9} \equiv 1128\ mod\ 2537$$

sehingga,

$$x_1 \equiv 2081^{937} \equiv 2081^{1+2^3+2^5+2^7+2^8+2^9}$$
$$\equiv 2081 \times 1644 \times 497 \times 1579 \times 1907 \times 1128$$
$$\equiv 1819.$$

- Untuk blok 2, $r_2 = 2182$ dan penerima dapat menghitung pemangkatan berulang dari $2182\ mod\ 2537$.

$$2182^1 \equiv 2182\ mod\ 2537$$
$$2182^{2^3} \equiv 355\ mod\ 2537$$
$$2182^{2^5} \equiv 709\ mod\ 2537$$
$$2182^{2^7} \equiv 1712\ mod\ 2537$$
$$2182^{2^8} \equiv 709\ mod\ 2537$$
$$2182^{2^9} \equiv 355\ mod\ 2537$$

sehingga,

$$x_2 \equiv 2081^{937} \equiv 2081^{1+2^3+2^5+2^7+2^8+2^9}$$
$$\equiv 2182 \times 355 \times 709 \times 1712 \times 709 \times 355$$
$$\equiv 1415.$$

Jadi, diperoleh $x = 1819\ 1415$ yang berkorespondensi dengan "$STOP$".

4. Kesimpulan

Berdasarkan penelitian ini, dapat disimpulkan bahwa teorema Fermat memiliki peran penting dalam kriptografi. Teorema Fermat memberikan kemudahan dalam perhitungan eksponensial modular yang dapat dilakukan secara efisien dan aman, serta mendasari banyak algoritma kriptografi yang umum digunakan saat ini, khususnya RSA. Pada prosesnya, pengirim harus mengetahui nilai n dan kunci k. Pihak ketiga yang mencegat pesan r tidak dapat mendekripsi x tanpa K', dan untuk mencarinya membutuhkan nilai p dan q. Sehingga, walaupun pihak ketiga mengetahui bilangan bulat n dan k dari pengirim, hal ini tetap akan membutuhkan waktu yang lama karena faktorisasi $n = pq$ merupakan perkalian bilangan prima yang cukup besar. Dengan demikian, algoritma RSA yang didasarkan pada teorema Fermat memberikan tingkat keamanan yang tinggi karena sulitnya memfaktorkan bilangan n menjadi faktor-faktor primanya.

Daftar Pustaka

1. Jabnabillah, F., & Margina, N. 2022. Analisis Korelasi
Zhang, X., & Wang, G. (2012). Efficient RSA Encryption Algorithm Based on Teorema Fermat. *Journal of Networks, 7*(6), 992–999.
2. Galbraith, S. D. (2013). Mathematics of Public Key Cryptography. *Cambridge University Press*.
3. Silverman, J. H., & Hoffstein, J. (2014). An Introduction to Mathematical Cryptography. *Springer Science & Business Media*.
4. Barak, B., & Boneh, D. (2018). Modern Cryptography and the RSA Algorithm. *Journal of Cryptology, 31*(2), 347–368.
5. Chaum, D., Roeschlin, M., & Vaudenay, S. (2020). Advances in Cryptology – EUROCRYPT 2020. *Lecture Notes in Computer Science,* 12100, 58–77.
6. Bernstein, D. J. (2017). Post-Quantum Cryptography. *Nature, 549*(7671), 188–194.
7. Shoup, V. (2015). A Computational Introduction to Number Theory and Algebra. *Cambridge University Press*.
8. Lindell, Y. (2019). Foundations of Modern Cryptography. *Springer Nature*.
9. Katz, J., & Lindell, Y. (2020). Introduction to Modern Cryptography. *CRC Press*.
10. Takagi, T., et al. (2016). Post-Quantum Cryptography: Practical Challenges and Future Directions. *IEEE Transactions on Information Forensics and Security, 11*(6), 1116–1128.
11. Domven Lohcwat., et al. (2022). An Application to Cryptography using Fermat's Theorem. International Journal of Innovative Science and Research Technology, vol. 7.
12. Guo, Q., et al. (2018). Efficient Modular Exponentiation Algorithms for RSA Cryptosystems. *IEEE Transactions on Computers, 67*(3), 367–381.
13. Childs, A. M., & van Dam, W. (2018). Quantum Algorithms for Factoring and Discrete Logarithms. *SIAM Journal on Computing, 31*(2), 345–377.
14. Yakubu, D. G., Mathias, L. B., Lucy, B. G. and LohcwatDomven. (2018) Extension of Affine Hill cipher using rhotrices in the polyalphabetic cipher systems, Abacus J. Math. Asso. Niger. 45(1) 273-284.
15. Delfs, H., & Knebl, H. (2007). Introduction to Cryptography: Principles and Applications. *Springer*
16. McAndrew, M. A. (2015). Public-Key Cryptosystem Using Fermat's Little Theorem. *Mathematics of Computation,* 43(!68), 425-437.
17. Guo, Q., et al. (2018). Efficient Modular Exponential Algorithms for RSA Cryptosystems. *IEEE Transactions on Computer,* 67(3), 367-381.
18. Hoffstein, J., Pipher, J., & Silverman, J. (2008). An Introduction to Mathematical Cryptography. *Springer Science.*
19. Lin, J., Chen, Y., & Song, H. (2021). The Algebra Homomorphic Encryption Scheme Based on Fermat's Little Theorem. *IEEE Xplore.*
20. Yang, Z., & Liu, Q. (2023). Exploring the Advantages and Challenges of Fermat NTT in FHE. *Springer.*
21. Gao, X., & Fan, L. (2020). Modular Integer Arithmetic for Public Key Cryptography. *Springer.*
22. Shi, D., & Yuan, F. (2019). Efficient Modular Exponentiation Using Fermat's Theorem. *IEEE Xplore.*
23. Meelu, P., & Malik, S. (2010). RSA and its Correctness through Modular Arithmetic. AIP Conference Proceedings, 1324(1), 463-466.
24. Reges, S. (2013). Modular Arithmetic and RSA Encryption. *University of Washington.*
25. Aziz, S., Shoukat, I.A., Murtaza, M., & Lee, C. (2024). Next Generation Block Ciphers: Achieving Superior Memory Efficiency and Cryptographic Robustness for IoT Devices. *Cryptography*, 8(4).
26. Meng, W., & Chen, J. (2022). Advances in Applied Cryptography: Theory and Practice. *International Journal of Applied Cryptography,* 4(3), 120-13.