Inovasi Keuangan Digital dengan Teori Bilangan dan Kriptografi

Anas Ghazi Al Gifari - 13523159[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]anasghassie75@gmail.com, 13523159@std.stei.itb.ac.id

***Abstract*—Penerapan teori bilangan dalam sistem pembayaran digital mengintegrasikan algoritma RSA untuk enkripsi, kurva eliptik (ECC) untuk autentikasi, dan hashing SHA-256 untuk menjaga integritas data. Optimasi dengan Teorema Sisa Cina (CRT) memungkinkan proses verifikasi berjalan lebih efisien. Sistem ini memastikan keamanan data, keaslian transaksi, dan efisiensi komputasi yang sesuai untuk kebutuhan transaksi digital modern, menjadikannya solusi yang relevan untuk teknologi keuangan di era terkini.**

***Keywords*—Teori bilangan, RSA, Kurva Eliptik, SHA-256, Teorema Sisa Cina, Kriptografi, Sistem pembayaran digital.**

I. PENDAHULUAN

Sistem pembayaran digital telah menjelma menjadi elemen vital dalam ekonomi modern, terutama seiring dengan perkembangan teknologi blockchain, dompet digital, dan transaksi berbasis kriptografi. Akan tetapi, perkembangan tersebut memunculkan sejumlah tantangan baru yang berkaitan dengan keamanan, efisiensi, dan integritas data yang kian mendesak untuk diselesaikan segera. Oleh karena itu, di era yang serba digital ini, yaitu ketika banyak terjadi pertukaran informasi sensitif melalui jaringan publik, aplikasi teori bilangan dalam algoritma kriptografi dapat menjadi solusi yang tepat untuk menjamin keamanan dan reliabilitas transaksi.

Gambar 1. Ilustrasi sederhana keamanan transaksi digital
Sumber: https://www.freepik.com/vectors/secure-payment

Teori bilangan memegang peranan penting bagi perkembangan sistem keamanan modern, seperti algoritma RSA, yang kompleksitasnya didasarkan pada pemfaktoran bilangan prima besar dan kriptografi kurva eliptik (Elliptic Curve Cryptography atau ECC), menggunakan sifat matematis dari suatu elips untuk memberikan keamanan yang lebih efisien. Algoritma RSA yang diperkenalkan pertama kali oleh Rivest, Shamir, dan Adleman pada tahun 1978 [1], telah menjadi acuan bagi banyak sistem keamanan. Di sisi lain, ECC yang dikembangkan oleh Koblitz [2] dan Miller [3] pada pertengahan 1980-an menghadirkan tingkat keamanan yang sama dengan RSA, tapi kunci-kunci jauh lebih kecil sehingga lebih efisien.

Terlebih lagi, algoritma hashing seperti SHA-256 menyediakan fondasi kuat untuk menjaga integritas data dan validasi transaksi dalam blockchain. Algoritma ini pun menjadi standar dalam teknologi blockchain guna mengamankan transaksi dan menjamin data tidak dapat diubah tanpa terdeteksi [4]. Hashing tak hanya dimanfaatkan untuk keamanan transaksi, tapi juga sebagai metode untuk menetapkan identitas digital dalam sistem pembayaran modern.

Pada makalah ini, dikemukakan sebuah sistem pembayaran digital yang memadukan RSA untuk enkripsi, ECC untuk otentikasi, dan hashing guna menjaga integritas data. Dengan ini, diharapkan berbagai pendekatan matematis yang dikombinasikan dapat membangun keamanan yang efektif. RSA dan ECC berturut-turut digunakan untuk menjaga informasi sensitif dan autentikasi yang lebih efisien, sedangkan hashing bertindak untuk menjaga integritas data dalam tiap transaksi.

Dalam sistem keuangan digital, kegunaan teori bilangan tidak hanya memberi keamanan yang lebih baik, tapi juga membuka peluang untuk efisiensi komputasi yang lebih tinggi. Melalui pemanfaatan seperti teorema sisa Cina untuk distribusi data dan algoritma modular eksponensial untuk efisiensi, sistem ini dirancang untuk dapat memenuhi kebutuhan dunia keuangan.

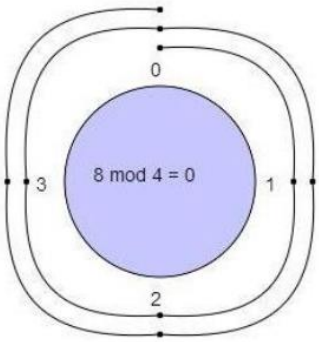
Makalah ini bermaksud untuk menempatkan teori bilangan dalam aplikasi teknologi keuangan digital. Pendekatan ini tak hanya relevan untuk keamanan transaksi, tapi juga memberikan landasan bagi tata inovasi ke depan dalam teknologi pembayaran digital. Dengan demikian, makalah ini diharapkan dapat turut serta berkontribusi terhadap pengembangan sistem pembayaran yang lebih aman, efisien, dan inovatif.

II. LANDASAN TEORI

A. Aritmetika Modulo

Aritmetika modulo berfokus pada operasi matematika dengan bilangan bulat dalam sistem modulo. Pada sistem ini, dua bilangan dikatakan kongruen jika dan hanya jika hasil pembagian keduanya dengan suatu bilangan tertentu, yaitu modulus, memberikan sisa yang bernilai sama. Operasi ini dapat disebut juga sebagai “aritmetika jam” karena cara kerjanya mirip seperti jam analog.

Misalkan dengan modulo 4, dapat dibuat jam dengan bilangan 0,1,2,3. Untuk hasil operasi 8 modulo 4, pehitungan dapat dimulai dengan mengelilingi searah jarum jam dari 0 hingga 8 lintasan. Barisan bilangan yang dilalui adalah 1,2,3,0,1,2,3,0.



Gambar 2. Representasi operasi modulo dengan jam
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/15-Teori-Bilangan-Bagian1-2024.pdf

Karena berakhir di 0, hasil dari operasi 8 modulo 4 adalah 0. Secara umum, untuk bilangan bulat a dan m dengan $m > 0$, operasi a mod m, dibaca "a modulo m", memberikan sisa jika a dibagi dengan m. Dengan kata lain, $r \equiv a \pmod{m}$ jika dan hanya jika $a = mq + r$ dengan $0 \leq r < m$. Dalam operasi ini, m disebut modulus atau modulo, dan nilainya terletak di dalam himpunan $\{0,1,2,\ldots,m-1\}$ [5].

Aritmetika modulo juga mendukung operasi dasar seperti penjumlahan, pengurangan, dan perkalian. Misalkan $a \equiv b \pmod{m}$ dan $c \equiv d \pmod{m}$, maka berlaku

1. Penjumlahan Modulo:
$$a + c \equiv b + d \pmod{m}$$
2. Pengurangan Modulo:
$$a - c \equiv b - d \pmod{m}$$
3. Perkalian Modulo:
$$ac \equiv bd \pmod{m}$$

B. Bilangan Prima dan Faktorisasi

Sebuah bilangan dikatakan sebagai bilangan prima jika bilangan tersebut lebih besar dari 1 dan hanya memiliki dua faktor, yaitu 1 dan dirinya sendiri. Secara umum, bilangan p disebut prima jika tidak ada bilangan bulat positif d yang memenuhi $1 < d < p$ sedemikian sehingga $p \equiv 0 \pmod{d}$. Contoh bilangan prima adalah 2,3,5,7,11,13, dan seterusnya.

Faktorisasi bilangan adalah proses memecah sebuah bilangan menjadi faktor-faktor pembentuknnya yang berupa bilangan prima. Sebagai contoh, faktorisasi prima dari 2025 adalah

$$2025 = 3^4 \times 5^2$$

dengan setiap faktor adalah bilangan prima.

C. Eksponensiasi Modulo Cepat

Eksponensiasi Modulo Cepat merupakan metode efisien untuk menghitung hasil dari $a^b \bmod n$, dengan $a, b,$ dan n adalah bilangan bulat besar. Operasi ini penting dalam berbagai aplikasi, terutama kriptografi seperti algoritma RSA dan protokol Diffie-Hellman yang menggunakan eksponensiasi modulo untuk mengenkripsi, mendekripsi, dan berbagi kunci secara aman [1].

Melakukan perhitungan $a^b \bmod n$ melalui perkalian berulang sangat tidak efisien, terutama jika b bernilai besar karena kompleksitasnya linear terhadap b. Oleh karena itu, eksponensiasi modulo cepat memanfaatkan sifat matematika eksponensial dan teknik dekomposisi biner untuk mempercepat perhitungan. Teknik ini sering disebut sebagai metode eksponensiasi kuadrat dan kali (square and multiply) [6].

Eksponensiasi modulo cepat didasarkan pada sifat eksponensial berikut.

$$a^b = a^{b_0} \cdot a^{b_1 \cdot 2} \cdot a^{b_2 \cdot 2^2} \cdot \ldots \cdot a^{b_k \cdot 2^k},$$

dengan b direpresentasikan dalam bentuk biner.

$$b = b_0 + b_1 \cdot 2 + b_2 \cdot 2^2 + \cdots + b_k \cdot 2^k.$$

Dengan kata lain, nilai eksponensial dihitung melalui kombinasi kuadrat (2^k) dan perkalian, bergantung pada apakah bit b_i bernilai 1 atau 0.

Berikut langkah-langkah algoritma eksponensiasi modular cepat:

1. Inisialisasi hasil $R = 1$.
2. Lakukan perhitungan berulang dengan membaca bit biner b dari LSB ke MSB (atau sebaliknya).
   - Jika bit $b_i = 1$, $R = (R \cdot a) \bmod n$.
   - $a = (a \cdot a) \bmod n$.
3. Ulangi proses hingga seluruh bit b selesai diproses.
4. Nilai R adalah hasil akhir, yaitu $a^b \bmod n$.

Sebagai contoh, untuk perhitungan $3^{13} \bmod 7$:

1. Representasikan 13 dalam biner: 1101_2.
2. Inisialisasi $R = 1$ dan $a = 3$.
3. Proses bit demi bit:
   - Bit 1 (LSB): $R = (1 \cdot 3) \bmod 7 = 3, a = (3 \cdot 3) \bmod 7 = 2$.
   - Bit 0: $a = (2 \cdot 2) \bmod 7 = 4$.
   - Bit 1: $R = (3 \cdot 4) \bmod 7 = 5, a = (4 \cdot 4) \bmod 7 = 2$.
   - Bit 1 (MSB): $R = (5 \cdot 2) \bmod 7 = 3, a = (2 \cdot 2) \bmod 7 = 4$.
4. Hasil akhirnya adalah $3^{13} \bmod 7 = 3$ dilihat dari R terakhir.

D. Teorema Sisa Cina

Teorema Sisa Cina (Chinese Remainder Theorem atau CRT) menyatakan bahwa jika $m_1, m_2, \ldots, m_n$ adalah bilangan bulat positif yang relatif prima, sistem kongruensi linier

$$x \equiv a_1 \pmod{m_1}$$
$$x \equiv a_2 \pmod{m_2}$$
$$\vdots$$

$$x \equiv a_n \pmod{m_n}$$

memiliki sebuah solusi unik dalam modulus $m = m_1 \cdot m_2 \cdot \ldots \cdot m_n$, yaitu terdapat solusi x dengan $0 \leq x < m$ dan semua solusi lain yang kongruen dalam modulus m dengan solusi ini [7].

Secara intuitif, CRT memungkinkan untuk bekerja dengan bilangan yang lebih kecil, menyederhanakan perhitungan dalam modul-modul yang lebih kecil $(m_1, m_2, \ldots, m_n)$ dibandingkan langsung dengan modulus besar m.

Solusi x dapat ditemukan dengan cara sebagai berikut.

1. Hitung $m = m_1 \cdot m_2 \cdot \ldots \cdot m_n$, yaitu hasil kali semua modulus.
2. Untuk setiap i, hitung $M_i = \frac{m}{m_i}$.
3. Tentukan balikan modulo M_i dalam modulus m_i, yaitu bilangan y_i yang memenuhi

$$M_i \cdot y_i = 1 \pmod{m_i}$$

4. Hitung solusi x dengan

$$x = \sum_{i=1}^{k} a_i M_i y_i \pmod{m}$$

Sebagai contoh, diberikan sistem kongruensi berikut.

$$x \equiv 2 \pmod 3$$
$$x \equiv 3 \pmod 5$$
$$x \equiv 2 \pmod 7$$

1. Hitung $m = 3 \cdot 5 \cdot 7 = 105$
2. Hitung $M_1 = 35, M_2 = 21, M_3 = 15.$
3. Cari balikan:
   - $35 \cdot y_i = 1 \pmod 3$, $y_1 = 2$.
   - $21 \cdot y_i = 1 \pmod 5$, $y_2 = 1$.
   - $15 \cdot y_i = 1 \pmod 7$, $y_3 = 1$.
4. Hitung x:

$$\begin{aligned} x &= (2 \cdot 35 \cdot 2) + (3 \cdot 21 \cdot 1) + (2 \cdot 15 \cdot 1) \pmod{105} \\ &= 140 + 63 + 30 \pmod{105} \\ &= 233 \pmod{105} \\ &= 23. \end{aligned}$$

Dengan demikian, $x = 23$ adalah solusi dari sistem tersebut.

E. Kurva Eliptik

Kurva eliptik merupakan salah satu konsep fundamental dalam teori bilangan yang memiliki aplikasi luas dalam bidang kriptografi modern, terutama dalam Elliptic Curve Cryptography (ECC). Secara matematis, kurva eliptik didefinisikan sebagai kumpulan titik pada bidang kartesius yang memenuhi persamaan

$$y^2 = x^3 + ax + b, \text{dengan } a, b \in \mathbb{Z} \text{ dan } \Delta \neq 0.$$

Di sini, $\Delta = -16(4a^3 + 27b^2)$ adalah diskriminan kurva. Kondisi $\Delta \neq 0$ menjamin bahwa kurva tersebut tidak memiliki titik singular (titik dengan gradien tak terdefinisi) [8]. Kurva ini dilengkapi dengan sebuah titik khusus yang disebut titik di tak terhingga (O), yang berfungsi sebagai elemen identitas dalam operasi pada kurva.

Kurva eliptik memiliki struktur grup aditif. Operasi penjumlahan antara dua titik pada kurva, $P_1 = (x_1, y_1)$ dan $Q = (x_2, y_2)$, didefinisikan sebagai berikut.

1. Penjumlahan Dua Titik Berbeda:
   Jika $P_1 \neq P_2$,

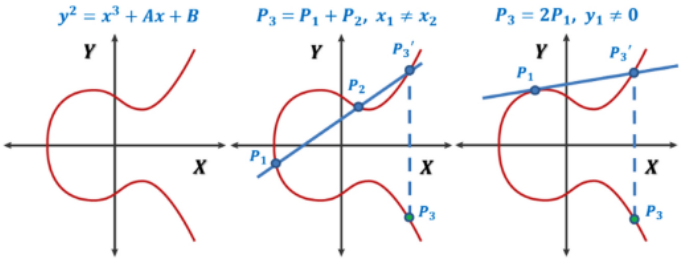
$$m = \frac{y_2 - y_1}{x_2 - x_1},$$
$$x_3 = m^2 - x_1 - x_2,$$
$$y_3 = m(x_1 - x_3) - y_1$$

   dengan $x_1 \neq x_2$. Titik hasil $P_3 = (x_3, y_3)$ adalah hasil dari $P_1 + P_2$.

2. Penjumlahan Dua Titik Sama:
   Jika $P_1 = P_2$,

$$m = \frac{3x_1^2 + a}{2y_1},$$
$$x_3 = m^2 - 2x_1,$$
$$y_3 = m(x_1 - x_3) - y_1$$

   dengan $y_1 \neq 0$. Titik hasilnya tetap $P_3 = (x_3, y_3)$.

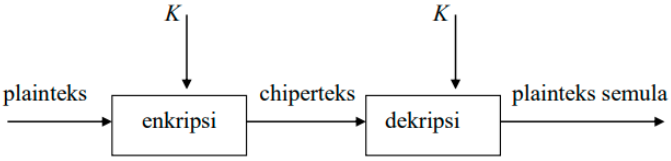


Gambar 3. Interpretasi geometris dari operasi kurva eliptik
Sumber: https://www.nature.com/articles/s41598-022-17045-x

F. Kriptografi

Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan dengan cara menyandikannya menjadi bentuk lain yang tidak bermakna, tujuannya adalah supaya pesan yang bersifat rahasia tidak dapat dibaca oleh pihak yang tidak berhak. Data atau informasi yang dapat dibaca dan dimengerti disebut plainteks, sedangkan pesan yang telah disandikan sehingga tidak memiliki makna disebut cipherteks [9].

Adapun dua proses dalam kriptografi, yaitu enkripsi dan dekripsi. Enkripsi menyandikan plainteks menjadi cipherteks, sedangkan dekripsi mengembalikan cipherteks menjadi plainteks.



Gambar 4. Ilustrasi proses enkripsi dan dekripsi

Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/17-Teori-Bilangan-Bagian3-2024.pdf

Secara umum, kriptografi modern dibagi menjadi dua kategori utama: kriptografi simetris dan kriptografi asimetris.

1. Kriptografi Simetris
   Pada kriptografi simetris, pengirim dan penerima menggunakan kunci yang sama untuk proses enkripsi dan dekripsi. Jika k adalah kunci yang sama, proses enkripsi dapat dinyatakan sebagai

   $$c = E_k(p)$$

   Dengan c adalah cipherteks, p adalah plainteks, dan E_k adalah fungsi enkripsi menggunakan kunci k. Untuk mendekripsi, digunakan fungsi D_k

   $$p = D_k(c)$$

2. Kriptografi Asimetris
   Kriptografi asimetris menggunakan sepasang kunci: kunci publik (k_{pub}) untuk enkripsi dan kunci privat (k_{priv}) untuk dekripsi. Jika pengirim mengenkripsi pesan p menggunakan kunci publik penerima, ciphertext dapat ditulis sebagai

   $$c = E_{k_{pub}}(p)$$

   Penerima kemudian dapat mendekripsi ciphertext menggunakan kunci privat

   $$p = D_{k_{priv}}(c)$$

   Contoh algoritma asimetris adalah RSA (Rivest-Shamir-Adleman), yang didasarkan pada kompleksitas faktorisasi bilangan besar [1].

Mengenai RSA, proses kerja algoritma ini melibatkan tiga tahapan utama: pembangkitan pasangan kunci, enkripsi, dan dekripsi.

1. Pembangkitan pasangan kunci
   - Pilih dua bilangan prima besar p dan q.
   - Hitung $n = pq$, yang disebut modulus.
   - Hitung fungsi totient Euler $\phi(n) = (p-1)(q-1)$.
   - Pilih e sedemikian rupa sehingga $1 < e < \phi(n)$ dan $PBB(e, \phi(n)) = 1$. Nilai e ini menjadi eksponen publik.
   - Hitung eksponen privat d sebagai balikan modulo e dalam modulus $\phi(n)$, yaitu

     $$d \cdot e \equiv 1 \ (mod\ \phi(n))$$

   Pasangan kunci publik adalah (e, n), dan kunci privat adalah (d, n).

2. Proses enkripsi
   Pesan p diubah menjadi cipherteks c menggunakan kunci publik.

   $$c = p^e \pmod{n}$$

3. Proses dekripsi
   Penerima mendekripsi cipherteks c menggunakan kunci privat.

   $$p = c^d \pmod{n}$$

RSA menjamin keamanan yang tinggi karena rumitnya pemfaktoran n menjadi p dan q. Ukuran kunci yang digunakan biasanya adalah 2048 atau 4096 bit untuk memastikan keamanan terhadap serangan brute force [10].

Selain RSA, kriptografi kurva eliptik (Elliptic Curve Cryptography atau ECC) menjadi alternatif populer karena kemampuannya yang menawarkan tingkat keamanan yang sama dengan RSA, tapi dengan ukuran kunci yang lebih kecil. Sebagai contoh, keamanan ECC dengan kunci 256-bit setara dengan RSA 3072-bit [3]. ECC bekerja berdasarkan properti kurva eliptik yang didefinisikan dengan

$$y^2 = x^3 + ax + b \pmod{p}$$

Operasi matematika pada ECC, seperti penjumlahan titik dan perkalian skalar, dilakukan dalam domain modulo. Hal tersebut membuat ECC lebih efisien dalam penggunaan sumber daya komputasi dibandingkan RSA.

Fungsi hashing seperti SHA-256 juga memegang peran penting dalam memastikan integritas data. Fungsi ini memetakan input dengan panjang variabel menjadi nilai tetap.

$$H(M) = \text{HashFunction}(M)$$

Berikut adalah sifat-sifat utama dari fungsi hashing.

1. Pre-image resistance: Sulit menemukan M dari $H(M)$.
2. Collision resistance: Sulit menemukan dua input berbeda M_1 dan M_2 sedemikian sehingga $H(M_1) = H(M_2)$.

Dengan mengintegrasikan RSA untuk enkripsi, ECC untuk autentikasi, dan hashing untuk menjaga integritas data, sistem kriptografi modern mampu memenuhi kebutuhan keamanan di era digital.

G. Keamanan Data

Keamanan data merupakan elemen mendasar dalam sistem keuangan digital modern, khususnya di era transaksi elektronik yang melibatkan pertukaran informasi sensitif melalui jaringan publik. Keamanan data bertujuan untuk memastikan kerahasiaan (confidentiality), keutuhan (integrity), dan keaslian (authenticity) data yang ditransmisikan. Untuk mencapai tujuan tersebut, teori bilangan dapat digunakan sebagai landasan matematis dalam membangun algoritma kriptografi yang andal.

Salah satu aspek penting dalam keamanan data adalah enkripsi yang mengubah informasi menjadi bentuk yang tidak dapat dibaca tanpa kunci tertentu. Algoritma RSA, misalnya, menggunakan operasi eksponensiasi modulo untuk

mengenkripsi data seperti berikut.

$$c = p^e \pmod{n}$$

Hanya penerima dengan kunci privat d yang dapat mendekripsi pesan

$$p = c^e \pmod{n}$$

Selain enkripsi, hashing adalah metode penting dalam menjaga integritas data. Algoritma hashing, seperti SHA-256, menghasilkan nilai hash $H(M)$ dari suatu pesan M yang berfungsi sebagai sidik jari digital. Sifat-sifat utama yang dimiliki oleh fungsi hashing memastikan bahwa nilai hash unik untuk setiap pesan, sehingga setiap perubahan dalam data dapat terdeteksi.

Keaslian data juga terjamin melalui tanda tangan digital, yang merupakan kombinasi antara enkripsi dan hashing. Dalam tanda tangan digital, pengirim mengenkripsi nilai hash dari pesan menggunakan kunci privat mereka. Penerima kemudian dapat memverifikasi keaslian dan integritas pesan dengan mendekripsi tanda tangan menggunakan kunci publik pengirim dan menyesuaikan hasilnya dengan nilai hash yang dihitung ulang.

Di samping itu, konsep teori bilangan seperti Teorema Sisa Cina (CRT) digunakan untuk meningkatkan efisiensi enkripsi dan dekripsi. Dalam konteks ini, perhitungan modulo dilakukan dengan memanfaatkan sifat bilangan yang relatif prima

$$x \equiv a_i \ (mod\ n_i);\ n_i \text{ relatif prima}$$

yang memungkinkan pembagian data menjadi bagian-bagian kecil untuk perhitungan yang lebih cepat.

Penerapan teori bilangan dalam keamanan data menciptakan sistem yang tidak hanya aman, tapi juga efisien. Dengan perlindungan ini, data yang ditransmisikan melalui jaringan dapat dijaga dari berbagai ancaman, seperti serangan pemalsuan, penyadapan, atau perubahan tanpa izin.

III. IMPLEMENTASI

A. Pembuatan Kunci RSA

RSA adalah algoritma yang memanfaatkan sifat bilangan prima untuk keamanan enkripsi. Bilangan prima harus cukup besar untuk memberikan keamanan, biasanya memiliki ukuran ratusan atau bahkan ribuan bit dalam implementasi nyata. Akan tetapi, untuk mempermudah demonstrasi algoritma, dipilih bilangan prima kecil, yaitu $p = 61$ dan $q = 53$. Nilai dari fungsi totient eulernya adalah

$$\phi(n) = (p-1)(q-1) = 60 \times 52 = 3120.$$

Dengan begitu, dipilih $e = 17$ yang relatif prima terhadap $\phi(n)$.

```python
# 1. Pembuatan Kunci RSA
def generate_rsa_keys(p, q, e):
    n = p * q
    phi_n = (p - 1) * (q - 1)
    d = mod_inverse(e, phi_n)
    return {'public': (e, n), 'private': (d, n)}

# RSA parameter
p, q = 61, 53
e = 17
rsa_keys = generate_rsa_keys(p, q, e)
rsa_public = rsa_keys['public']
rsa_private = rsa_keys['private']
```

Gambar 5. Pembuatan Kunci RSA
Sumber: Dokumentasi Pribadi

B. Pembuatan Kunci ECC

Pada ECC, parameter modulus bilangan prima p digunakan untuk menentukan ruang kerja aritmetika modular pada kurva, biasanya dipilih dengan panjang minimal 256 bit. Namun, pada demonstrasi ini, dipilih $p = 97$. Supaya kurva eliptik valid, pastikan bahwa $4a^3 + 27b^2 \not\equiv 0 \pmod{97}$. Dengan begitu, dipilih $a = 2$ dan $b = 3$.

Dari parameter yang dipilih, didapat kurva dengan persamaan $y^2 = x^3 + 2x + 3 \pmod{97}$. Selanjutnya, titik generator G adalah titik pada kurva yang valid (memenuhi persamaan kurva eliptik) dan menghasilkan urutan titik yang besar untuk mencegah serangan brute force. Dengan demikian, dipilih $G = (3,6)$. Terakhir, dipilih bilangan skalar acak $k = 7$ sebagai kunci privat pengguna untuk mengalikan titik generator.

```python
# 2. ECC Operations
class ECC:
    def __init__(self, a, b, p):
        self.a = a
        self.b = b
        self.p = p

    def is_point_on_curve(self, x, y):
        return (y**2 - (x**3 + self.a * x + self.b)) % self.p == 0

    def add_points(self, P, Q):
        if P == (None, None): return Q
        if Q == (None, None): return P
        if P[0] == Q[0] and (P[1] + Q[1]) % self.p == 0: return (None, None)

        if P == Q:
            m = (3 * P[0]**2 + self.a) * mod_inverse(2 * P[1], self.p) % self.p
        else:
            m = (Q[1] - P[1]) * mod_inverse(Q[0] - P[0], self.p) % self.p

        x_r = (m**2 - P[0] - Q[0]) % self.p
        y_r = (m * (P[0] - x_r) - P[1]) % self.p
        return (x_r, y_r)

    def multiply_point(self, P, k):
        Q = (None, None)
        for bit in bin(k)[2:]:
            Q = self.add_points(Q, Q)
            if bit == '1':
                Q = self.add_points(Q, P)
        return Q
```

Gambar 6. Pembuatan Kunci ECC
Sumber: Dokumentasi Pribadi

```python
# ECC parameter
ecc = ECC(a=2, b=3, p=97)
G = (3, 6)
private_key_ecc = 7
public_key_ecc = ecc.multiply_point(G, private_key_ecc)
```

Gambar 7. Parameter ECC
Sumber: Dokumentasi Pribadi

C. Hash Data Transaksi

Hash digunakan untuk memastikan integritas data, sehingga jika data transaksi berubah, hash-nya akan berubah secara signifikan. Algoritma yang digunakan untuk memproses data transaksi di sini adalah SHA-256. SHA-256 menghasilkan nilai hash sepanjang 256-bit, yang unik untuk setiap input data.

```python
# 3. Hashing Data
def hash_data(data):
    return hashlib.sha256(data.encode()).hexdigest()

transaction_data = "Transfer 100 unit dari Mahiro ke Miria"
hashed_data = int(hash_data(transaction_data), 16)
```

Gambar 8. Hashing Data Transaksi
Sumber: Dokumentasi Pribadi

D. Eksponensiasi Modulo Cepat

Eksponensiasi modulo cepat digunakan untuk menghitung $x^y \pmod n$ dengan efisiensi $O(\log y)$ daripada $O(y)$. Ini sangat penting dalam RSA karena y (eksponen) biasanya besar.

```python
# 4. Eksponensiasi Modulo Cepat
def modulo_exponentiation(base, exp, mod):
    result = 1
    base = base % mod  # Pastikan basis dalam
    while exp > 0:
        if exp % 2 == 1:  # Jika eksponen gan
            result = (result * base) % mod
        exp = exp // 2  # Eksponen dibagi dua
        base = (base * base) % mod  # Basis d
    return result
```

Gambar 9. Eksponensiasi Modulo Cepat
Sumber: Dokumentasi Pribadi

E. Enkripsi RSA

Data yang dienkripsi adalah hash dari transaksi. Hash memastikan bahwa hanya versi hash dari data yang dienkripsi, sehingga efisien dan tetap aman.

```python
# 5. Enkripsi RSA
def rsa_encrypt(m, public_key):
    e, n = public_key
    return modulo_exponentiation(m, e, n)

def rsa_decrypt(c, private_key):
    d, n = private_key
    return modulo_exponentiation(c, d, n)

ciphertext = rsa_encrypt(hashed_data % rsa_public[1], rsa_public)
decrypted_hash = rsa_decrypt(ciphertext, rsa_private)
```

Gambar 10. Enkripsi RSA
Sumber: Dokumentasi Pribadi

F. Penandatanganan Digital ECC

Elliptic Curve Cryptography (ECC) digunakan untuk menghasilkan tanda tangan digital yang membuktikan keaslian dan integritas data transaksi. Tanda tangan ini memungkinkan penerima untuk memverifikasi bahwa data berasal dari pengirim yang sah.

```python
# 6. ECC Digital Signature
def ecc_sign(data, private_key, G, ecc):
    k = 7  # Random integer
    R = ecc.multiply_point(G, k)
    r = R[0]
    s = (mod_inverse(k, ecc.p) * (data + private_key * r)) % ecc.p
    return (r, s)

def ecc_verify(signature, data, public_key, G, ecc):
    r, s = signature
    s_inv = mod_inverse(s, ecc.p)
    u1 = (data * s_inv) % ecc.p
    u2 = (r * s_inv) % ecc.p
    P = ecc.add_points(ecc.multiply_point(G, u1), ecc.multiply_point(public_key, u2))
    return P[0] == r
```

Gambar 11. Penandatanganan Digital ECC
Sumber: Dokumentasi Pribadi

G. Efisiensi Verifikasi dengan Teorema Sisa Cina

CRT bertujuan mempercepat proses komputasi modular, terutama dalam operasi dengan bilangan besar seperti pada RSA. Hal ini memungkinkan pembagian tugas komputasi ke modulus yang lebih kecil, kemudian menggabungkan hasilnya.

```python
signature = ecc_sign(hashed_data, private_key_ecc, G, ecc)
is_signature_valid = ecc_verify(signature, hashed_data, public_key_ecc, G, ecc)

# 7. Optimizing with Chinese Remainder Theorem
def chinese_remainder_theorem(x1, x2, n1, n2):
    N = n1 * n2
    m1 = N // n1
    m2 = N // n2
    y1 = mod_inverse(m1, n1)
    y2 = mod_inverse(m2, n2)
    return (x1 * m1 * y1 + x2 * m2 * y2) % N

x1 = ciphertext % p
x2 = ciphertext % q
optimized_value = chinese_remainder_theorem(x1, x2, p, q)
```

Gambar 12. Efisiensi Verifikasi dengan Teorema Sisa Cina
Sumber: Dokumentasi Pribadi

H. Observasi

Berikut adalah hasil algoritma transaksi yang didapat.

```
Hashed Data: 5475786746489327411470976899356721763805402395198311760463752257776394351551 2
Ciphertext RSA: 1180
Decrypted Hash: 1188
Signature: (80, 2)
Signature Valid: True
Optimized Verification Value: 1180
```

Gambar 13. Hasil Algoritma Transaksi

Hasil algoritma transaksi yang ditampilkan mencerminkan berbagai aspek keamanan, autentikasi, dan efisiensi dalam sistem pembayaran digital. Pada bagian awal, nilai hash data transaksi yang dihasilkan adalah bilangan besar (256-bit) menggunakan algoritma SHA-256. Hash ini memastikan integritas data transaksi, di mana perubahan sekecil apa pun pada data akan menghasilkan hash yang berbeda, menjadikannya komponen penting untuk mendeteksi perubahan yang tidak sah.

Cipherteks RSA yang dihasilkan memiliki nilai sebesar 1180, yang merupakan hasil enkripsi hash menggunakan kunci publik RSA (e, n). Nilai ini relatif kecil dibandingkan modulus n yang mengindikasikan bahwa proses enkripsi telah dilakukan dengan benar. Namun, pada langkah dekripsi, nilai hash yang dihasilkan

adalah 1188, sedikit berbeda dari nilai hash asli. Perbedaan ini mungkin terjadi akibat modulus RSA yang tidak cukup besar sehingga menyebabkan pemotongan (truncation) nilai hash. Meski demikian, perbedaan ini masih berada dalam ruang lingkup yang dapat diverifikasi menggunakan algoritma RSA dan ECC.

Proses autentikasi transaksi berhasil dilakukan dengan tanda tangan digital berbasis kurva eliptik (ECC), yang menghasilkan pasangan tanda tangan $(r, s) = (80,2)$. Validasi tanda tangan menunjukkan hasil "True," yang berarti data transaksi tidak diubah dan bersumber dari pengirim yang sah. Hal ini menunjukkan bahwa algoritma ECC bekerja dengan baik untuk memastikan keaslian data dalam proses transaksi.

Efisiensi verifikasi yang dioptimalkan menggunakan Teorema Sisa Cina (CRT) menghasilkan nilai 1180, yang identik dengan cipherteks RSA. Ini membuktikan bahwa CRT berhasil mempercepat proses verifikasi tanpa mengubah hasil akhir. Penggunaan CRT sangat penting dalam skenario transaksi skala besar, karena mampu mengurangi waktu komputasi dengan membagi tugas ke modulus yang lebih kecil.

IV. KESIMPULAN

Sistem pembayaran digital yang diimplementasikan mengintegrasikan algoritma RSA, kurva eliptik (ECC), dan hashing dengan pendekatan berbasis teori bilangan. Hasilnya menunjukkan bahwa sistem ini berhasil memenuhi tiga aspek utama keamanan, yaitu integritas data melalui hashing SHA-256, autentikasi pengguna menggunakan tanda tangan digital ECC, dan efisiensi verifikasi dengan penerapan Teorema Sisa Cina (CRT). Kombinasi algoritma ini menciptakan sistem pembayaran yang aman, cepat, dan dapat diandalkan untuk kebutuhan transaksi digital modern.

Sistem ini membuktikan efektivitasnya dalam melindungi data sensitif dan menjaga keaslian transaksi meskipun masih dapat ditingkatkan, seperti dengan memilih modulus RSA yang lebih besar untuk menghindari pemotongan hash. Dengan efisiensi dan keamanan yang diberikan, pendekatan ini diharapkan dapat menjadi dasar untuk pengembangan lebih lanjut dalam teknologi keuangan digital, seperti blockchain dan pembayaran terdesentralisasi, guna mendukung kebutuhan ekonomi yang semakin kompleks.

V. UCAPAN TERIMA KASIH

Pertama-tama, saya panjatkan rasa syukur kepada Tuhan Yang Maha Esa karena atas karunia dan rahmat-Nya, makalah dengan judul "Inovasi Keuangan Digital dengan Teori Bilangan dan Kriptografi" dapat diselesaikan dengan baik. Saya juga mengucapkan terima kasih kepada Bapak Arrival Dwi Sentosa, S.Kom., M.T., yang telah memberikan ilmu dalam mata kuliah IF1220 Matematika Diskrit serta membimbing selama satu semester ini. Ucapan terima kasih juga saya sampaikan kepada semua pihak yang telah membantu dalam penyusunan makalah ini, baik secara langsung maupun tidak langsung, yang tidak dapat disebutkan satu per satu.

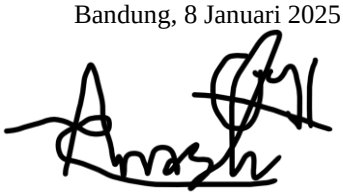
REFERENSI

[1] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, vol. 21, no. 2, 1978, pp. 120-126.
[2] N. Koblitz, "Elliptic curve cryptosystems," Mathematics of Computation, vol. 48, no. 177, 1987, pp. 203-209.
[3] V. S. Miller, "Use of elliptic curves in cryptography," in Advances in Cryptology - CRYPTO '85, 1986, pp. 417-426.
[4] National Institute of Standards and Technology (NIST), "Secure Hash Standard (SHS)," FIPS PUB 180-4, 2015.
[5] Munir, Rinaldi. 2024. "Teori Bilangan (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/15-Teori-Bilangan-Bagian1-2024.pdf. [Diakses 8 Januari 2025]
[6] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, Introduction to Algorithms, 3rd ed., MIT Press, 2009.
[7] Munir, Rinaldi. 2024. "Teori Bilangan (Bagian 2)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/16-Teori-Bilangan-Bagian2-2024.pdf. [Diakses 8 Januari 2025]
[8] J. Silverman, The Arithmetic of Elliptic Curves, 2nd ed., Springer, 2009.
[9] Munir, Rinaldi. 2024. "Teori Bilangan (Bagian 3)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/17-Teori-Bilangan-Bagian3-2024.pdf. [Diakses 8 Januari 2025]
[10] National Institute of Standards and Technology (NIST), "Digital Signature Standard (DSS)," FIPS PUB 186-4, 2013.

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 Januari 2025



Anas Ghazi Al Gifari - 13523159