# Analisis Komparasi Penggunaan Algoritma RSA dan ElGamal pada Digital Signature

Tara Chandani Haryono 18221146
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
tara.haryono@gmail.com

***Abstract*—Keamanan dan privasi dalam era digital merupakan tantangan utama. Studi ini membandingkan algoritma RSA dan ElGamal untuk tanda tangan digital, dengan fokus pada efisiensi waktu. RSA dan ElGamal adalah algoritma kriptografi asimetris yang populer digunakan untuk memastikan integritas dan autentikasi data. RSA dikenal karena kesederhanaan dan efisiensinya, sementara ElGamal menawarkan keamanan tambahan melalui kompleksitas logaritma diskrit. Implementasi kedua algoritma menggunakan fungsi hash untuk menciptakan tanda tangan digital yang aman. Analisis dilakukan berdasarkan waktu komputasi. Hasil menunjukkan bahwa setiap algoritma memiliki keunggulan masing-masing. RSA lebih efisien dalam hal kecepatan komputasi dan integrasi sistem, menjadikannya pilihan yang lebih umum dalam aplikasi tanda tangan digital.**

***Keywords—RSA, Elgamal, Tanda Tangan Digital***

## I. PENDAHULUAN

Dengan pesatnya perkembangan teknologi informasi serta semakin banyaknya perangkat elektronik yang saling terhubung dan berkomunikasi satu sama lain, keamanan dan privasi menjadi tantangan penting saat ini. Misalnya, kurangnya otentikasi di antara entitas yang berkomunikasi, risiko berbagai serangan siber, akses tidak sah ke suatu layanan, dan kebocoran informasi rahasia. Sebelum memberikan akses ke suatu jaringan atau layanan, identitas pengakses perlu diverifikasi kemudian kebijakan kontrol akses diterapkan berdasarkan identitas pengguna tersebut. Tanda tangan digital berfungsi untuk mengotentikasi identitas pengguna, sementara otorisasi mengonfirmasi apakah pengguna memiliki hak yang sesuai untuk mengakses suatu sumber daya. [1]

Enkripsi diperlukan untuk memastikan keamanan informasi saat transmisi data dan komunikasi sebab kedua hal tersebut rentan terhadap serangan eksternal. Untuk meningkatkan keamanan, data diubah melalui enkripsi sebelum ditransmisikan dan kemudian dikembalikan ke bentuk aslinya melalui dekripsi saat sampai ke penerima.

Berbagai algoritma kriptografi telah dikembangkan untuk menyediakan transmisi data dan informasi yang aman. Berdasarkan kunci yang digunakan, algoritma-algoritma ini dikategorikan menjadi dua macam yakni algoritma kunci simetris dan kunci asimetris. Pada algoritma kunci simetris, kunci yang digunakan untuk proses enkripsi dan dekripsi sama. Pada algoritma kunci asimetris, kunci yang digunakan untuk melakukan enkripsi berbeda dengan kunci yang digunakan untuk dekripsi. Enkripsi dilakukan menggunakan kunci publik penerima sedangkan dekripsi dilakukan menggunakan kunci privat penerima. Algoritma RSA dan Elgamal merupakan algoritma kunci asimetris.

Tanda tangan digital mengonfirmasi keaslian pesan melalui proses kriptografi. Tanda tangan digital setara dengan tanda tangan tulisan tangan, yang menawarkan perlindungan terhadap peniruan identitas dalam jaringan komunikasi. Tanda tangan digital memberikan jaminan tambahan mengenai asal, integritas, dan status dokumen, tindakan, atau komunikasi elektronik, serta mengonfirmasi persetujuan penandatangan. Penandatanganan dokumen memanfaatkan kunci dan algoritma enkripsi. Algoritma enkripsi yang sering digunakan pada tanda tangan digital adalah RSA dan ElGamal.

Makalah ini bertujuan untuk membandingkan performa algoritma kriptografi RSA dan ElGamal yang digabung dengan fungsi hash SHA-256 untuk proses generasi dan verifikasi tanda tangan digital.

## II. DASAR TEORI

### *A. Algoritma RSA*

RSA (Rivest-Shamir-Adleman) adalah salah satu algoritma kriptografi asimetris yang paling terkenal dan banyak digunakan. Algoritma ini ditemukan pada tahun 1977 oleh Ron Rivest, Adi Shamir, dan Leonard Adleman. RSA digunakan untuk enkripsi dan dekripsi data serta pembuatan dan verifikasi tanda tangan digital. Keamanan algoritma ini didasarkan pada kesulitan faktorisasi bilangan besar menjadi dua bilangan prima.

RSA bekerja dengan dua kunci yang berbeda namun terkait, yaitu kunci publik dan kunci pribadi. Kunci publik digunakan untuk enkripsi data dan dapat dibagikan secara luas, sedangkan kunci pribadi digunakan untuk dekripsi data dan harus dijaga kerahasiaannya.

Keamanan RSA didasarkan pada kesulitan faktorisasi bilangan besar n yang merupakan hasil perkalian dua bilangan prima besar p dan q. Saat ini, tidak ada algoritma yang efisien untuk faktorisasi bilangan besar dalam waktu yang wajar, sehingga membuat RSA sangat aman jika kunci yang digunakan cukup besar (misalnya, 2048 bit atau lebih).

*1) Proses Pembangkitan Kunci*

Pembangkitan kunci dalam RSA melibatkan beberapa langkah berikut:

*a) Pilih dua bilangan prima besar secara acak, p dan q*

*b) Hitung nilai modulus n dengan rumus p * q*

*c) Hitung nilai totient Euler, $\phi(n)$ dengan rumus*

$$\phi(n) = (p-1) \times (q-1)$$

*d) Pilih eksponen publik e yang merupakan bilangan bulat positif kecil dan koprima dengan $\phi(n)$ (umumnya, nilai eee yang sering digunakan adalah 65537 karena efisien dalam komputasi).*

*e) Hitung eksponen pribadi d sebagai invers modulo dari e terhadap $\phi(n)$, yang memenuhi*

$$e \times d \equiv 1 \pmod{\phi(n)}$$

Hasil dari pembangkitan kunci adalah kunci publik yaitu (e, n) dan kunci privat yaitu (d, n).

*2) Proses Enkripsi*

Untuk mengenkripsi pesan M, lakukan langkah berikut

*a) Konversikan pesan M menjadi bilangan bulat m sedemikian sehingga $0 \leq m < n$*

*b) Hitung ciphertext C dengan rumus*

$$C = m^e \pmod{n}$$

*3) Proses Dekripsi*

Untuk mendekripsi ciphertext C, lakukan langkah berikut

*a) Hitung kembali pesan m dengan rumus*

$$m = C^d \pmod{n}$$

*b) Konversikan bilangan bulat m kembali ke pesan asli M.*

*4) Pembuatan Tanda Tangan Digital*

Pengirim mengenkripsi hash dari pesan dengan kunci pribadi untuk menghasilkan tanda tangan digital.

$$S = H(M)^d \pmod{n}$$

*5) Verifikasi Tanda Tangan Digital*

Penerima menggunakan kunci publik pengirim untuk mendekripsi tanda tangan dan memverifikasi bahwa hasilnya cocok dengan hash pesan asli.

$$H(M) = S^e \pmod{n}$$

*B. Algoritma Elgamal*

ElGamal adalah algoritma kriptografi asimetris yang ditemukan oleh Taher ElGamal pada tahun 1985. Algoritma ini digunakan untuk enkripsi data dan pembuatan tanda tangan digital. Keamanan ElGamal didasarkan pada kesulitan masalah Logaritma Diskret dalam grup bilangan bulat modulo bilangan prima besar.

ElGamal bekerja dengan dua kunci yang berbeda namun terkait, yaitu kunci publik dan kunci privat. Kunci publik digunakan untuk enkripsi data dan dapat dibagikan secara luas, sedangkan kunci privat digunakan untuk dekripsi data dan harus dijaga kerahasiaannya.

Keamanan ElGamal didasarkan pada kesulitan masalah Logaritma Diskrit. Secara khusus, diberikan g, $g^x$, dan $g^k$, sulit untuk menemukan k. Hal ini memastikan bahwa tanpa mengetahui kunci privat x, sangat sulit untuk mendekripsi pesan atau memalsukan tanda tangan.

*1) Proses Pembangkitan Kunci*

Pembangkitan kunci dalam ElGamal melibatkan beberapa langkah berikut

*a) Pilih bilangan prima besar p.*

*b) Pilih bilangan g yang merupakan generator dari grup bilangan bulat modulo p.*

*c) Pilih bilangan acak x sedemikian sehingga $1 \leq x \leq p-21$*

*d) Hitung y dengan rumus*

$$y = g^x \pmod{p}$$

Kunci publik adalah pasangan (p, g, y) dan kunci privat adalah x.

*2) Proses Enkripsi*

Untuk mengenkripsi pesan M, lakukan langkah berikut

*a) Konversikan pesan M menjadi bilangan bulat m sedemikian sehingga $0 \leq m < p$.*

*b) Pilih bilangan acak k sedemikian sehingga $1 \leq k \leq p-2$.*

*c) Hitung nilai $c_1$ dan $c_2$ dengan rumus*

$$c_1 = g^k \pmod{p}$$
$$c_2 = m \cdot y^k \pmod{p}$$

Ciphertext yang dihasilkan adalah pasangan $(c_1, c_2)$.

*3) Proses Dekripsi*

Untuk mendekripsi ciphertext ($c_1$, $c_2$) lakukan langkah berikut

*a) Hitung kembali pesan m dengan rumus*

$$m = c_2 \cdot (c_1^x)^{-1} \pmod{p}$$

Di mana $(c_1^x)^{-1}$ adalah invers modulo p dari $(c_1^x)$.

*b) Konversikan bilangan bulat mmm kembali ke pesan asli M*

*4) Pembuatan Tanda Tangan Digital*

Pengirim menghitung tanda tangan digital (r, s) untuk pesan M

$$r = g^k \pmod{p}$$

$$s = (H(M) - xr) \cdot k^{-1} \pmod{(p-1)}$$

Di mana H(M) adalah hash dari pesan M, dan $k^{-1}$ adalah invers dari k modulo p−1.

*5) Verifikasi Tanda Tangan Digital*

Penerima memverifikasi tanda tangan digital (r, s) dengan memeriksa apakah

$$g^{H(M)} \equiv y^r \cdot r^s \pmod{p}$$

Jika persamaan ini benar, maka tanda tangan valid.

### C. *Digital Signature*

Tanda tangan digital adalah mekanisme kriptografi yang digunakan untuk memverifikasi keaslian dan integritas pesan, perangkat lunak, atau dokumen digital. Tanda tangan digital memberikan jaminan bahwa pesan atau dokumen tersebut benar-benar berasal dari pengirim yang diaku dan tidak diubah selama transmisi.

Tanda tangan digital menggunakan algoritma kriptografi asimetris, yang melibatkan pasangan kunci publik dan kunci privat. Berikut adalah langkah-langkah umum dalam pembuatan dan verifikasi tanda tangan digital.

*1) Hashing*

Pengirim menggunakan fungsi hash kriptografi untuk menghasilkan nilai hash dari pesan atau dokumen yang akan ditandatangani. Fungsi hash mengubah input menjadi output dengan panjang tetap yang unik untuk setiap input yang berbeda.

$$h = H(M)$$

Di mana H adalah fungsi hash dan M adalah pesan.

*2) Enkripsi*

Pengirim mengenkripsi nilai hash menggunakan kunci pribadi untuk menghasilkan tanda tangan digital.

$$S = E_{kprivat}(h)$$

Di mana E adalah algoritma enkripsi dan $k_{privat}$ adalah kunci pribadi pengirim.

*3) Pengiriman*

Pengirim mengirim pesan asli bersama dengan tanda tangan digital S kepada penerima.

Sedangkan untuk proses verifikasi tanda tangan digital dapat dilakukan dengan cara menghitung nilai hash dari pesan yang diterima menggunakan fungsi hash yang sama. Lalu, penerima mendekripsi tanda tangan digital menggunakan kunci publik pengirim untuk mendapatkan nilai hash asli. Penerima membandingkan nilai hash yang dihitung dari pesan diterima (h′) dengan nilai hash yang didekripsi dari tanda tangan (h). Jika keduanya sama, tanda tangan valid; jika tidak, tanda tangan tidak valid atau pesan telah diubah.

### D. *SHA-256*

SHA-256 (Secure Hash Algorithm 256-bit) adalah fungsi hash kriptografis yang merupakan bagian dari keluarga SHA-2, yang dikembangkan oleh National Security Agency (NSA) dan diterbitkan oleh National Institute of Standards and Technology (NIST). SHA-256 menghasilkan output hash sepanjang 256bit (32 byte) dari input dengan panjang berapa pun. Fungsi hash ini digunakan secara luas dalam berbagai aplikasi keamanan digital, termasuk enkripsi, tanda tangan digital, dan autentikasi data.

Fungsi hash kriptografis adalah algoritma yang mengubah input data (pesan) menjadi output tetap yang disebut hash atau digest. Fungsi hash yang baik memiliki karakteristik deterministic, cepat, resisten terhadap *collision*, dan memiliki *avalanche effect* yakni perubahan kecil pada input harus menghasilkan perubahan besar yang tidak dapat diprediksi pada hash yang dihasilkan.

Langkah-langkah pada SHA-256 meliputi hal berikut:

*1) Padding*

Pesan asli di-padding sehingga panjangnya menjadi kelipatan 512 bit. Padding dimulai dengan bit '1' diikuti oleh sejumlah bit '0', diakhiri dengan panjang asli pesan dalam bit.

*2) Pemecahan Blok*

Pesan yang telah di-padding dipecah menjadi blok-blok 512-bit.

*3) Inisialisasi Nilai Hash*

Delapan nilai hash awal (H0 hingga H7) diinisialisasi. Nilai-nilai ini adalah konstanta yang telah ditentukan sebelumnya.

*4) Pemrosesan Blok*

Setiap blok 512-bit diproses melalui serangkaian transformasi menggunakan fungsi-fungsi bitwise, konstanta, dan variabel kerja.

*5) Iterasi Kompresi*

Setiap blok diolah melalui 64 iterasi menggunakan fungsi kompresi yang melibatkan variabel kerja dan konstanta khusus (K).

*6) Kombinasi Nilai Hash*

Nilai-nilai hash dari setiap iterasi kemudian digabungkan dengan nilai hash awal untuk menghasilkan hash akhir setelah semua blok diproses.

SHA-256 dianggap aman dan tahan terhadap berbagai jenis serangan kriptografi, termasuk serangan tabrakan (*collision*), dengan ketahanan hingga $2^{128}$ operasi untuk menemukan tabrakan. Hal ini membuat SHA-256 sangat cocok untuk aplikasi yang membutuhkan tingkat keamanan tinggi.

## III. RANCANGAN

Berikut merupakan rancangan implementasi kode program untuk melakukan analisis performa algoritma RSA dan Elgamal pada tanda tangan digital

### A. *Library*

Library yang digunakan antara lain modul time, cryptography, dan Crypto.

### B. *Modul*

*1) Waktu Enkripsi dan Dekripsi RSA*

```
def rsa_encryption_decryption(file_path):
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048,
    )
    public_key = private_key.public_key()
```

```
    with open(file_path, 'rb') as file:
        data = file.read()

    start_time = time.time()
    encrypted = public_key.encrypt(
        data,
        padding.OAEP(

mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    encryption_time = time.time() -
start_time

    start_time = time.time()
    decrypted = private_key.decrypt(
        encrypted,
        padding.OAEP(

mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    decryption_time = time.time() -
start_time

    return encryption_time, decryption_time
```

*2) Waktu Sign dan Verify Tanda Tangan Digital menggunakan RSA*

```
def rsa_sign_verify(file_path):
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=2048,
    )
    public_key = private_key.public_key()

    with open(file_path, 'rb') as file:
        data = file.read()

    hasher = hashes.Hash(hashes.SHA256())
    hasher.update(data)
    digest = hasher.finalize()

    start_time = time.time()
    signature = private_key.sign(
        digest,
        padding.PSS(

mgf=padding.MGF1(hashes.SHA256()),

salt_length=padding.PSS.MAX_LENGTH
        ),
        Prehashed(hashes.SHA256())
    )
    sign_time = time.time() - start_time

    start_time = time.time()
    public_key.verify(
        signature,
        digest,
        padding.PSS(

mgf=padding.MGF1(hashes.SHA256()),

salt_length=padding.PSS.MAX_LENGTH
        ),
        Prehashed(hashes.SHA256())
    )
    verify_time = time.time() - start_time

    return sign_time, verify_time
```

*3) Waktu Enkripsi dan Dekripsi Elgamal*

```
def
elgamal_encryption_decryption(file_path):
    key = ElGamal.generate(2048,
get_random_bytes)
    public_key = key.publickey()

    with open(file_path, 'rb') as file:
        data = file.read()

    h = SHA256.new(data)

    start_time = time.time()
    k = get_random_bytes(32)
    ciphertext =
public_key.encrypt(h.digest(), k)
```

```
    encryption_time = time.time() -
start_time

    start_time = time.time()
    decrypted = key.decrypt(ciphertext)
    decryption_time = time.time() -
start_time

    return encryption_time, decryption_time
```

*4) Waktu Sign dan Verify Tanda Tangan Digital Menggunakan Elgamal*

```
def elgamal_sign_verify(file_path):
    key = ElGamal.generate(2048,
get_random_bytes)

    with open(file_path, 'rb') as file:
        data = file.read()

    h = SHA256.new(data)

    start_time = time.time()
    signer = DSS.new(key, 'fips-186-3')
    signature = signer.sign(h)
    sign_time = time.time() - start_time

    start_time = time.time()
    verifier = DSS.new(key.publickey(),
'fips-186-3')
    verifier.verify(h, signature)
    verify_time = time.time() - start_time

    return sign_time, verify_time
```

## IV. ANALISIS HASIL PENGUJIAN

Kunci yang digunakan untuk RSA maupun Elgamal berukuran 2048bit. Waktu enkripsi, dekripsi, pembuatan tanda tangan, dan verifikasi tanda tangan dinyatakan dalam millisekon

*A. Waktu Enkripsi*

| Ukuran File (KB) | RSA | Elgamal |
|---|---|---|
| 7 | 57 | 2105 |
| 17 | 135 | 5109 |
| 34 | 576 | 8320 |
| 64 | 1630 | 17360 |
| 91 | 1876 | 17990 |

Elgamal membutuhkan waktu yang lebih banyak untuk proses enkripsi

*B. Waktu Dekripsi*

| Ukuran File (KB) | RSA | Elgamal |
|---|---|---|
| 7 | 2340 | 423 |
| 17 | 5960 | 1150 |
| 34 | 16300 | 3075 |
| 64 | 31565 | 6272 |
| 91 | 30946 | 5463 |

Elgamal melakukan dekripsi lebih cepat disbanding RSA

*C. Waktu Pembuatan Tanda Tangan*

| Ukuran File (KB) | RSA + SHA-256 | Elgamal + SHA256 |
|---|---|---|
| 7 | 478 | 138 |
| 17 | 466 | 135 |
| 34 | 478 | 142 |
| 64 | 491 | 139 |
| 91 | 490 | 138 |

Elgamal memiliki performa yang lebih baik pada pembuatan tanda tangan

*D. Waktu Verifikasi Tanda Tangan*

| Ukuran File (KB) | RSA + SHA-256 | Elgamal + SHA-256 |
|---|---|---|
| 7 | 14 | 164 |
| 17 | 14 | 175 |
| 34 | 14 | 169 |
| 64 | 15 | 188 |
| 91 | 21 | 178 |

RSA memerlukan waktu yang lebih sedikit untuk memverifikasi tanda tangan digital

## V. KESIMPULAN

Dari berbagai hasil pengujian yang sudah dilakukan, didapat bahwa algoritma RSA memiliki performa yang lebih baik disbanding algoritma Elgamal pada proses enkripsi dan verifikasi tanda tangan. Sedangkan, algoritma Elgamal memiliki performa yang lebih baik dalam melakukan dekripsi dan pembuatan tanda tangan. Maka, dapat disimpulkan bahwa setiap algoritma memiliki keunggulan masing-masing.

Namun, pada kenyataannya algoritma RSA jauh lebih popular digunakan untuk tanda tangan digital karena beberapa alasan utama. Pertama, RSA memiliki keunggulan dalam hal efisiensi komputasi. Algoritma RSA relatif lebih sederhana dibandingkan ElGamal, yang memerlukan lebih banyak operasi eksponensial dan modular. Kedua, RSA memiliki dukungan yang luas dan telah menjadi standar dalam berbagai protokol keamanan seperti SSL/TLS, yang digunakan untuk mengamankan komunikasi di internet. Hal ini membuat RSA lebih mudah diintegrasikan ke dalam sistem yang ada dan

lebih dipercaya oleh komunitas keamanan siber. Ketiga, RSA menawarkan fleksibilitas dalam ukuran kunci dan dapat dengan mudah disesuaikan untuk meningkatkan keamanan sesuai kebutuhan. Di sisi lain, tanda tangan digital ElGamal lebih kompleks dan memerlukan pengelolaan kunci yang lebih rumit, yang dapat meningkatkan risiko kesalahan dalam implementasi. Faktor-faktor ini menjadikan RSA pilihan yang lebih populer dan praktis untuk penggunaan tanda tangan digital dalam berbagai aplikasi keamanan.

## REFERENCES


[1] Sejfuli-Ramadani, N. The Role and the Impact of Digital Certificate and Digital Signature in Improving Security During Data Transmission. Eur. J. Sustain. Dev. Res. 2017, 2, 116–120.


## 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, 12 Juni 2024

Tara Chandani Haryono 18221146