PROSIDING
Seminar Nasional Matematika dan Sains
Departemen Pendidikan Matematika dan Pendidikan Biologi
FKIP Universitas Wiralodra

11 Desember 2021

PENGAPLIKASIAN KRIPTOSISTEM RIVEST-SHAMIR-ADLEMAN (RSA) PADA TANDA TANGAN ELEKTRONIK

Intan Safira Eka Jatri[1)], Putranto Hadi Utomo[2)]
[1,2)]Universitas Sebelas Maret, Jl. Ir. Sutami No. 36, kota Surakarta.

Email: intansafira1805@student.uns.ac.id[1)], putranto@staff.uns.ac.id[2)]

Abstrak. Tanda tangan elektronik merupakan hal yang sangat penting untuk mengetahui integritas suatu dokumen. Tanda tangan elektronik dibentuk menggunakan mekanisme kunci kriptografi. Terdapat berbagai macam kriptosistem yang menyediakan algoritme tanda tangan elektronik. Pada penelitian ini akan digunakan kriptosistem Rivest-Shamir-Adleman (RSA). Kriptosistem RSA merupakan kriptosistem pertama yang menyediakan algoritme tanda tangan elektronik, selain itu kriptosistem RSA menggunakan metode yang cukup mudah pada proses pembangkitan kunci, proses enkripsi, dan proses dekripsi, namun tetap memiliki tingkat keamanan yang baik. Metode penelitian yang digunakan adalah metode kajian pustaka, dengan teori-teori yang tersedia dapat digunakan untuk mendesain algoritme tanda tangan elektronik dan baris perintah pada program yang akan dibangun. Penelitian ini menghasilkan algoritme tanda tangan elektronik dengan menggunakan kriptosistem RSA, contoh perhitungan dari algoritme tanda tangan elektronik yang telah dibuat, program yang dibangun dengan menggunakan bahasa pemrograman python, dan yang terakhir akan dilakukan uji coba program.

Kata Kunci : *Kriptosistem RSA, Tanda Tangan Elektronik, Python*

1. Pendahuluan

Tanda tangan merupakan suatu hal yang krusial, dikarenakan berguna sebagai identitas diri. Namun saat ini sedang marak pemalsuan tanda tangan, banyaknya kasus yang tercatat pada Direktori Putusan Mahkamah Agung RI adalah 2693 kasus. Pada tahun 2021 tercacat sebanyak 371 kasus pemalsuan tanda tangan dan jumlah kasus terus meningkat setiap tahunnya [1]. Banyaknya kasus pemalsuan tanda tangan mengakibatkan perlunya diadakan perubahan sistem tanda tangan. Tanda tangan manual ataupun tanda tangan manual yang digitalisasi perlu diubah ke dalam tanda tangan elektronik.

Tanda tangan elektronik merupakan tanda tangan yang terdiri atas informasi elektronik yang dilekatkan, terasosiasi atau terkait dengan informasi elektronik lainnya yang digunakan sebagai alat verifikasi dan autentikasi [2]. Tanda tangan elektronik membantu memenuhi tiga aspek keamanan informasi, yaitu autentikasi (keaslian) pengirim/penerima, integritas (keutuhan) data, dan mekanisme anti-sangkal (non-repudiasi) [3]. Cara kerja tanda tangan elektronik berbeda dengan tanda tangan digital pada umumnya seperti barcode, pena digital, dan tanda tangan yang dipindai, karena hal tersebut sangat mudah dipalsukan dan sulit diverifikasi keabsahannya. Seperti yang diketahui, karakteristik dari dokumen elektronik adalah mudah untuk diubah dan disalin, akibatnya dokumen mudah dipalsukan, maka dari itu dokumen elektronik yang dapat dipastikan keasliannya hanya dokumen yang memuat tanda tangan elektronik dengan menggunakan pasangan kunci kriptografi (kunci privat dan publik) [5].

Tanda tangan elektronik dibuat melalui mekanisme kriptografi kunci publik. Kriptografi kunci publik memanfaatkan dua buah kunci yang berbeda namun saling berkaitan secara matematika, yakni kunci publik dan kunci privat. Kunci publik dapat

diberikan atau diakses oleh siapa pun dan digunakan oleh pihak lain untuk melakukan verifikasi atas tanda tangan elektronik yang dibuat. Sedangkan kunci privat harus dijaga kerahasiaannya, atau dalam kata lain hanya pemilik kunci saja yang diperbolehkan mengetahui dan mengakses kunci tersebut. Kunci privat dapat digunakan untuk menandatangani dokumen elektronik. Pembuatan kunci publik dan kunci privat pada tanda tangan elektronik dapat menggunakan beberapa algoritme kriptosistem. Salah satu kriptosistem yang dapat disajikan ke dalam bentuk skema tanda tangan elektronik adalah kriptosistem Rivest-Shamir-Adleman (RSA).

Kriptosistem RSA merupakan algoritme pertama yang dianggap cocok untuk tanda tangan elektronik, dikarenakan banyaknya kriptosistem yang menyediakan algoritme tanda tangan elektronik namun algoritme RSA lebih mudah dipahami dan tetap dipercayai memiliki tingkat kemanan yang tinggi karena sulitnya memfaktorkan bilangan prima yang besar. Maka dari itu pada penelitian ini akan digunakan kriptosistem RSA sebagai dasar pemikiran algoritme tanda tangan elektronik. Selain itu akan digunakan SHA-256 sebagai bentuk biometrik dokumen. Hasil penelitian ini akan dibahas mengenai cara memperoleh skema tanda tangan elektronik dengan kriptosistem RSA, contoh perhitungan algoritme dan membangun program sederhana dengan bahasa pemrograman python.

2. Metode Penelitian

Metode penelitian yang digunakan dalam penelitian ini adalah kajian pustaka dengan mempelajari teori-teori yang berkaitan dengan penelitian sehingga dapat menunjang kelancaran penelitian. Berikut langkah-langkah yang dilakukan dalam penelitian.

1) Mempelajari serta memahami konsep kerja dari tanda tangan elektronik dan kriptosistem RSA.
2) Membuat diagram alir (*flowchart*) tanda tangan elektronik dengan algoritme RSA.
3) Membuat contoh perhitungan menggunakan algoritme yang telah dibuat.
4) Membangun program sederhana tanda tangan elektronik dari konsep yang telah dibuat.
5) Melakukan uji coba program dan menarik kesimpulan.

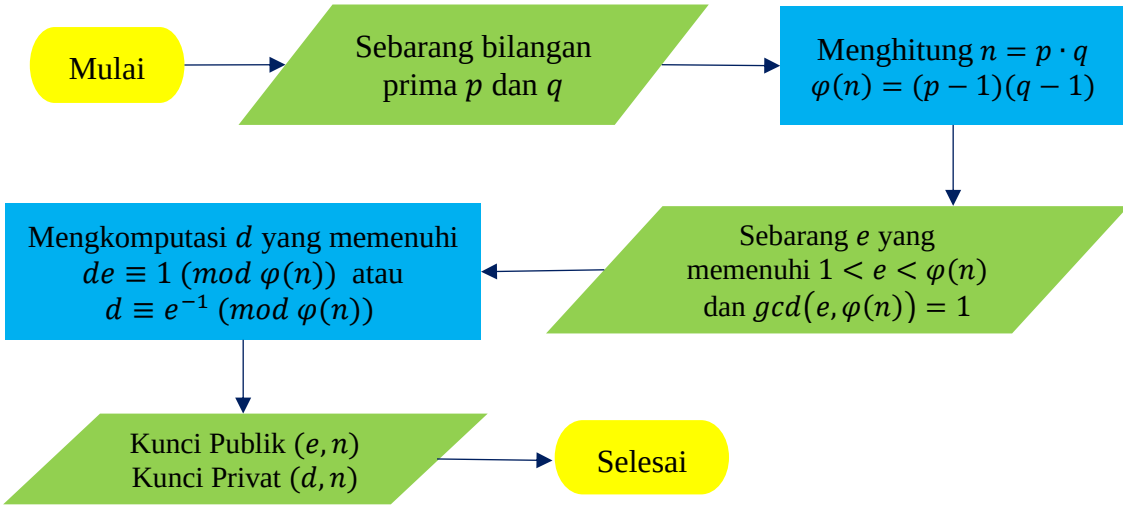
3. Hasil dan Pembahasan

3.1 Algoritme Tanda Tangan Elektronik dengan Kriptosistem RSA

Kriptosistem RSA merupakan kriptosistem kunci publik yang dibangun pada tahun 1978 oleh Ron Rivest, Adi Shamir, dan Len Adleman. RSA memanfaatkan kesulitan memfaktorkan dua bilangan prima yang besar sebagai keamanan kunci. Pada proses enkripsi dan dekripsi digunakan aritmatika modular sebagai pengubah pesan asli ke dalam *ciphertext* ataupun sebaliknya.

Pada pengamanan pesan, kunci publik digunakan untuk mengenkripsi pesan sedangkan kunci privat digunakan untuk mendekripsi pesan. Berbanding terbalik dengan algoritme tanda tangan elektronik dimana kunci publik digunakan untuk memverifikasi tanda tangan dan kunci privat digunakan untuk menandatangani dokumen. Tanda tangan elektronik dengan kriptosistem RSA melalui 3 proses, yaitu (1) Proses pembangkitan kunci, (2) Proses penandatanganan, (3) Proses verifikasi tanda tangan. Setiap proses dijelaskan pada *flowchart* dibawah ini.

Proses Pembangkitan Kunci

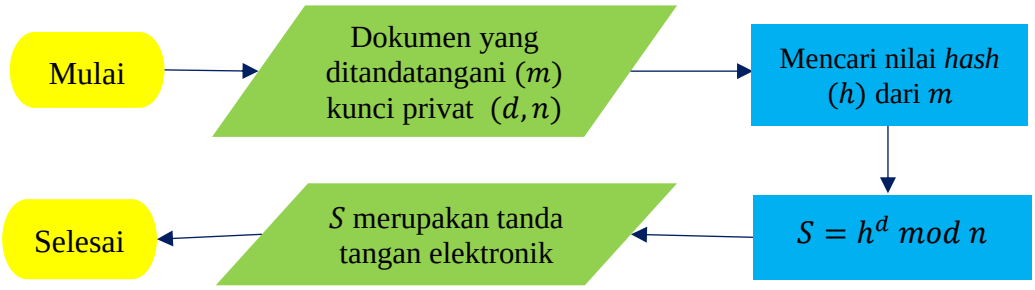


Gambar 1. *Flowchart* Proses Pembangkitan Kunci

Berikut algoritme dari proses pembangkitan kunci :

1. Memilih sebarang dua bilangan prima p dan q (minimal 200 digit).
2. Selanjutnya menghitung modulus $n = pq$.
3. Hitung fungsi Euler dari n yaitu $\varphi(n) = (p-1)(q-1)$.
4. Memilih sebarang e yang memenuhi $1 < e < \varphi(n)$ dan $gcd(e, \varphi(n)) = 1$.
5. Mengkomputasi d dengan menggunakan *extended Euclidean algorithm* yang memenuhi $de \equiv 1\ (mod\ \varphi(n))$ atau $d \equiv e^{-1}\ (mod\ \varphi(n))$.
6. Kemudian mempublikasikan kunci publik $K_E = (e, n)$ dan merahasiakan kunci privat $K_D = (d, n)$.

Proses Penandatanganan Dokumen

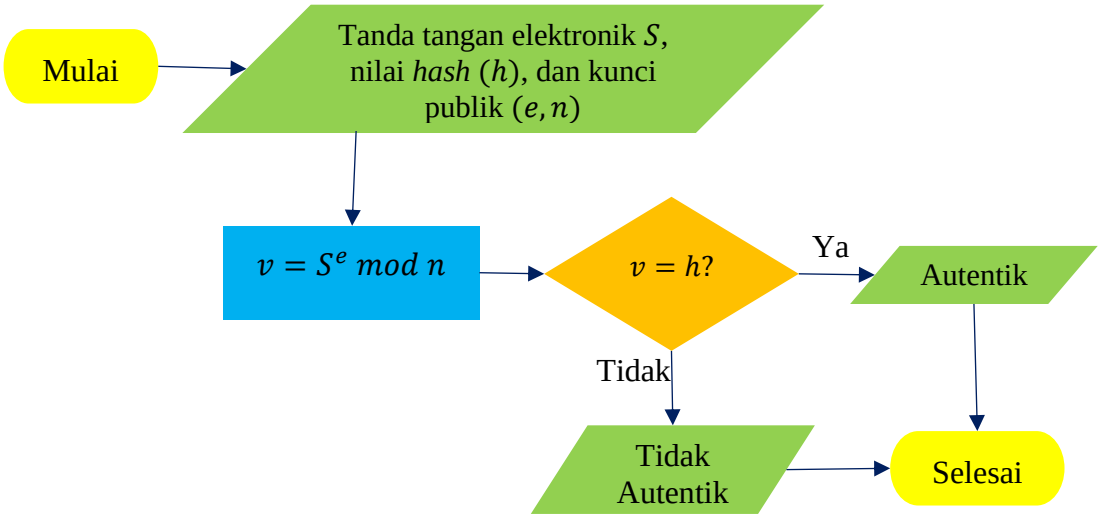


Gambar 2. *Flowchart* Proses Penandatanganan Dokumen

Berikut algoritme dari proses penandatanganan :

1. Masukkan dokumen yang akan ditandatangani (m) dan mencari kunci privat (d, n).
2. Mencari nilai *hash* (h) dari pesan yang akan ditandatangani (m).
3. Nilai *hash* (h) dienkripsi menggunakan kunci privat (d, n) dengan $S = h^d\ mod\ n$, dimana S merupakan representasi tanda tangan elektronik.

Proses Verifikasi Dokumen



Gambar 3. *Flowchart* Proses Verifikasi Dokumen

Berikut algoritme dari proses verifikasi :

1. Mencari tanda tangan elektronik S, nilai *hash* (h) pesan, dan kunci publik (e, n).
2. Tanda tangan elektronik S didekripsi menggunakan kunci publik dengan $v = S^e \ mod \ n$
3. Bandingkan hasil v dengan h, jika $v = h$ maka dokumen masih autentik/asli, namun jika $v \neq h$ maka dokumen tidak autentik/telah dipalsukan.

3.2 Contoh Perhitungan Tanda Tangan Elektronik dengan Kriptosistem RSA

Berikut akan diberikan contoh perhitungan dari algoritme tanda tangan elektronik. Pada penelitian ini perhitungan dilakukan dengan bantuan komputer dan digunakan bilangan yang cukup besar dikarenakan kunci yang dihasilkan harus lebih panjang dari pesan yang dikirimkan dimana fungsi *hash* yang digunakan adalah SHA-256 dan nilai *hash* menghasilkan >80 digit angka, maka dari itu panjang kunci harus >80 digit juga.

Dimisalkan suatu ketika Rizki ingin menandatangani sebuah dokumen yang berisikan "Hai Akbar, sampai bertemu besok malam jam 7.30" sebelum menandatangani pesan Rizki melakukan proses pembangkitan kunci dengan :

1. Dipilih $p = 130562573974831339072104487701522849564649723874197050331724679704031120218119785519253657 7$ dan $q = 1984130341616133283866919456086480177359555921641313094460706853499054396146892019111342021.$
2. $n = p \cdot q = 130562573974831339072104487701522849564649723874197050331724679704031120218119785519253657 7 \cdot 1984130341616133283866919456086480177359555921641313094460706853499054396146892019111342021$
   $n = 259053164502963777663851728700146366166576477196697281942897405433471925286172451897847508554571263573890723748897826778897465158971258209697805107162741065013800092488132959960211 7$
3. $\varphi(n) = (130562573974831339072104487701522849564649723874197050331724679704031120218119785519253657 7 - 1)(1984130341616133283866919456086480177359555921641313094460706853$

$$34990543961468920191113420 21 - 1)$$
$$= (13056257397483133907210448770152284956464972387419705$$
$$03317246797040311202181197855192536576)(19841303416161$$
$$33283866919456086480177359555921641313094460706853499 0$$
$$54396146892019111342020)$$
$$= 25905316450296377766385172870014636616657647719669728 1$$
$$94289740543347192528617245189784750822559565543744605 6$$
$$29010139346872659785836594217136944532936737601107724 0$$
$$2596791455295723520.$$

4. $e = 1 < e < \varphi(n) = 14532740503202224263804467343324300205302452$
$$00384088165675670014295974713687095564620450$$
$$53165644815330781285466074012099699341913420$$
$$14837209094417439053942197594973335436952175$$
$$44857.$$

5. $d \equiv e^{-1} \left(mod\ \varphi(n)\right) = 11239771655727392311259782582020917987448$
$$23017020046680142929000211661159130739395$$
$$96166874153384758463071433823967014984264$$
$$59710503116477425381594679325596521124804$$
$$24623377001991593.$$

6. Kunci publik $K_E = (e,\ n)$
$$= (11239771655727392311259782582020917987448230 1$$
$$70200466801429290002116611591307393959616687 41$$
$$53384758463071433823967014984264597105031164 77$$
$$42538159467932559652112480424623377001991593, 2$$
$$59053164502963777663851728700146366166576477 19$$
$$66972819428974054334719252861724518978475085 54$$
$$57126357389072374889782677889746515897125820 96$$
$$97805107162741065013800092488132959960211 7)$$

Kunci privat $K_D = (d,\ n)$
$$= (11239771655727392311259782582020917987448230 1$$
$$70200466801429290002116611591307393959616687 41$$
$$53384758463071433823967014984264597105031164 77$$
$$42538159467932559652112480424623377001991593, 2$$
$$59053164502963777663851728700146366166576477 19$$
$$66972819428974054334719252861724518978475085 54$$
$$57126357389072374889782677889746515897125820 96$$
$$97805107162741065013800092488132959960211 7).$$

Setelah mendapatkan kunci privat dan kunci publik, selanjutnya dilakukan proses penandatangan dokumen.

1. m : "Hai Akbar, sampai bertemu besok malam jam 7.30" dan kunci privat (d, n).
2. Nilai *hash* (h) dari m atau $h(m) = 23caa79b7f6518197341a89dc8d1a201682$
$$45be91b1781090a424b62bed23e66.$$ Kemudian mengubah nilai $h(m)$ kedalam bentuk *integer* menjadi $h(m_i) = 46246909572983688134135101761177911610$
$$35780232408033885662393967920980412 96.$$
3. $S = h(m_i)^d\ mod\ n = (46246909572983688134413510176117791161035$
$$78023240880338562393960980412963)^{1123977\ldots991593}$$
$$mod\ 25905316450296377766385172870014636616$$
$$65764771966972819428974054334719252861724 5$$

$$
\begin{aligned}
&189784750855457126357389072374889782677889\\
&746515897125820969780510716274106501380092\\
&488132959960211\\
S \;=\; &130354267257059527784634156032787352693177\\
&815632803648554940347195951151348600663908\\
&112828804883748063175763558688572421155955\\
&790995898275097531208188327027555944464145\\
&0812199631514
\end{aligned}
$$

Setelah melakukan penandatanganan, selanjutnya akan dilakukan proses verifikasi dokumen yang ditandatangani dengan

$$
\begin{aligned}
v = S^e \bmod n = \;&(130354267257059527784634156032787352693177815632\\
&036485549403471959511513486006639081128288048837418\\
&063175163558688572421155955790995898275097531208118\\
&8832702755594446414508121996315 14)^{1123971\ldots329599602117}\\
&\bmod 259053164502963777663851728700146366166576477119\\
&669728194289740543347192528617245189784750855457126\\
&357389072374889782677889746515897125820969780510716\\
&27410650138000924881329599602 11\\
v \;=\; &462469095729836881344135101761177911610357802324080\\
&338856623939679209804129 6 = h(m_i)
\end{aligned}
$$

Karena $v = h(m)$, maka pesan masih autentik atau asli. Dimisalkan dokumen telah diubah menjadi “Hai Akbar, sampai bertemu besok malam jam 7.00” kemudian dokumen yang telah diubah menghasilkan S sebagai berikut.

$$
\begin{aligned}
S = \;&236830541656387638165012948686638289393053801318109256082273\\
&810990352558128000901499227540500793355991055179570704231317\\
&102755435529716033059613857370142195302891179290150534835167
\end{aligned}
$$

Selanjutnya tanda tangan dari dokumen yang diubah akan diverifikasi dengan menggunakan kunci publik

$$
\begin{aligned}
v = S^e \bmod n = \;&(236830541656387638165012948686638289393053801318109256082273\\
&810990352558128000901499227540500793355991055179570704231317\\
&102755435529716033059613857370142195302891179290150534835167)^{11239716557\ldots329599602117}\\
&\bmod 259053164502963777663851728700146366166576477119\\
&669728194289740543347192528617245189784750855457126\\
&357389072374889782677889746515897125820969780510716\\
&27410650138000924881329599602 11\\
v \;=\; &462469095729836881344135101761177911610357802324080\\
&338856623939679209804126 57 \neq 46246909572983688134\\
&135101761177911610357802324080338856623939679209804\\
&1296 = h(m_i)
\end{aligned}
$$

Karena $v \neq h(m)$, maka pesan sudah tidak autentik atau telah dipalsukan. Dikarenakan fungsi *hash* selalu menghasilkan nilai yang berbeda setiap 1 bit perubahan pesan.

1.3 Program Sederhana Tanda Tangan Elektronik dengan Kriptosistem RSA

Program sederhana yang dibuat menggunakan bahasa pemrograman python dan ditulis dengan *jupyter notebook*. Algoritme tanda tangan yang telah dibuat akan diubah ke dalam bahasa pemrograman agar dapat digunakan secara luas. Berikut dijelaskan secara singkat baris perintah yang digunakan pada pembuatan program sederhana dari algoritme tanda tangan elektronik dengan algoritme RSA.

Berdasarkan diagram alir (*flowchart*) dari proses pembangkitan kunci digunakan sebarang dua buah bilangan prima p dan q dengan minimal 200 digit, untuk menghasilkan bilangan prima yang sangat besar tidaklah mudah, maka dari itu digunakan baris perintah berikut untuk mendapatkan bilangan prima yang besar secara acak dan efisien.

```
def qBitRandom(n):
    return random.randrange(2**(n-1)+1, 2**n - 1)

def getLowLevelPrime(n):
    while True:
        pc = nBitRandom(n)
        for divisor in first_primes_list:
            if pc % divisor == 0 and divisor**2 <= pc:
                break
        else: return pc

def isMillerRabinPassed(mrc):
    maxDivisionsByTwo = 0
    ec = mrc-1
    while ec % 2 == 0:
        ec >>= 1
        maxDivisionsByTwo += 1
    assert(2**maxDivisionsByTwo * ec == mrc-1)

if __name__ == '__main__':
    while True:
        e = 300
        prime_candidate = getLowLevelPrime(e)
        if not isMillerRabinPassed(prime_candidate):
            continue
        else:
            q = prime_candidate
            print("q =", prime_candidate)
            break
```

Gambar 4. Baris Perintah Menentukan Nilai p dan q

Setelah bilangan prima p dan q didapatkan akan dilanjutkan proses pembangkitan kunci ketahap selanjutnya yaitu menghitung *modulus* n, menghitung fungsi *phi Euler*, mencari bilangan e, dan menghitung nilai d atau *inverse* dari e *modulo* $\varphi(n)$. Berikut baris perintah yang digunakan tertera pada Gambar 5.

```
n = p*q
phi = (p-1) * (q-1)

def coprime(a, b):
    while b != 0:
        a, b = b, a % b
    return a

e = random.randrange(1, phi)
g = coprime(e, phi)
while g != 1:
        e = random.randrange(1, phi)
        g = coprime(e, phi)

def extended_gcd(aa, bb):
    lastremainder, remainder = abs(aa), abs(bb)
    x, lastx, y, lasty = 0, 1, 1, 0
    while remainder:
        lastremainder, (quotient, remainder) = remainder, divmod(lastremainder, remainder)
        x, lastx = lastx - quotient*x, x
        y, lasty = lasty - quotient*y, y
    return lastremainder, lastx * (-1 if aa < 0 else 1), lasty * (-1 if bb < 0 else 1)

def modinv(a, m):
    g, x, y = extended_gcd(a, m)
    if g != 1:
        raise Exception('Modular inverse does not exist')
    return x % m

d = modinv(e, phi)

print ('private key =', (e, n))
print ('public key =', (d, n))
```

Gambar 5. Baris Perintah Pembangkitan Kunci

Setelah proses pembangkitan kunci selesai dan menghasilkan kunci publik dan kunci privat, selanjutnya diberikan baris perintah untuk memasukkan dokumen yang akan

ditandatangani, pada penelitian ini hanya akan digunakan pesan singkat untuk ditandatangani, baris perintah yang digunakan untuk memasukkan pesan, tertera pada Gambar 6.

```
msg = b'A message for signing'
```

Gambar 6. Baris Perintah Masukkan Dokumen yang Akan Ditandatangani

Setelah pesan dimasukkan akan dicari nilai *hash* dari pesan. Dikarenakan nilai *hash* menghasilkan sebuah data *string* maka dari itu nilai *hash* perlu diubah ke dalam *integer* agar dapat dikomputasi, baris perintah yang digunakan dapat dilihat pada Gambar 7.

```
hash = sha256(msg).hexdigest()
print("hash =", hash)
hash_int = int.from_bytes(sha256(msg).digest(), byteorder='little')
print('hash integer =', hash_int)
```

Gambar 7. Baris Perintah Mencari Nilai *Hash* dari Dokumen dan Mengubah Nilai *Hash* ke dalam *Integer*

Setelah nilai *hash* berubah menjadi integer, kemudian proses penandatanganan dapat dikomputasi, baris perintah yang digunakan tertera pada Gambar 8.

```
signature = pow(hash_int, d, n)
print('signature', signature)
```

Gambar 8. Baris Perintah Penandatanganan Dokumen

Proses penandatanganan selesai, kemudian dilanjutkan proses verifikasi dokumen untuk memastikan apakah dokumen masih autentik atau sudah diubah/dipalsukan, untuk memverifikasi dokumen digunakan baris perintah yang dapat dilihat pada Gambar 9.

```
hashFromSignature = pow(signature, e, n)
print("Signature valid:", hash_int == hashFromSignature)
```

Gambar 9. Baris Perintah Verifikasi Dokumen

Uji Coba Program

Setelah program selesai dibuat akan dilakukan uji coba, pertama akan menentukan nilai p dan q dengan menggunakan baris perintah yang tertera pada Gambar 4. Berikut keluaran yang dihasilkan dapat dilihat pada Gambar 10.

```
p = 12869852281383745611606842932195628475334014051909492791916923710915404476754274795455433 67
q = 19472824976575387105374215163163319392377170787119977565188753150637911255942272622184091 37
```

Gambar 10. Nilai p dan q

Dari nilai p dan q yang telah ditentukan, tahap selanjutnya menghitung *modulus n*, *phi Euler*, dan proses lainnya, sehingga didapatkan kunci publik dan kunci privat. Gambar 11 merupakan keluaran dari kunci publik dan kunci privat berdasarkan baris perintah pada Gambar 5.

```
private key = (532673419537120139736390704949702285906927347644020947537655921461455528351691027887360947466865067573275002204850012463384601730565830025237663912528956245832444982557620029087395, 250612380949765128436907259195452861129624270721280763094196503628825907288806284321190966848418665701121298603667958862970997946865518055279709709842093814451088618829843298254427919)
public key = (229297685204903107090942450974398412589621555026962681177549231201194372634522596104805974539988805990172197175481953582065550600424690015827857645318501757645564358157326074356810 7, 250612380949765128436907259195452861129624270721280763094196503628825907288806284321190966848418665701121298603667958862970997946865518055279709709842093814451088618829843298254427919)
```

Gambar 11. Kunci Publik dan Kunci Privat

Setelah kunci publik dan kunci privat didapatkan, langkah selanjutnya memasukkan pesan yang akan ditandatangani, dimisalkan pada Gambar 12 merupakan pesan yang akan ditandatangani.

```
msg = b'Temui saya di kantor sekarang'
```

Gambar 12. Pesan yang Akan Ditandatangani

Kemudian pesan yang telah dimasukkan akan dihitung nilai *hash* nya dan keluaran nilai *hash* akan diubah ke dalam data *integer*. Berikut keluaran dari nilai *hash* dan nilai *hash* yang telah diubah.

```
hash = 82ffde89e52d1c05fad77fb4960ec3cb1f81079bf9a278dd5cda33f07335ea94
hash integer = 673561127217280951135558149773346254469611383335721295225020543347275825806 10
```

Gambar 13. Nilai *Hash* dari Pesan

Selanjutnya tanda tangan dapat dikomputasi dengan menggunakan baris perintah yang tertera pada Gambar 8. Berikut keluaran dari tanda tangan yang telah dikomputasi.

```
signature =  4972935107402409839329809554377233151566588216921845202619822365701260079216728672277185337422543550606678183176336243730836204767007053405316174563940456588427913168866600527045 47
```

Gambar 14. Tanda Tangan Pesan

Setelah tanda tangan didapatkan, dilakukan verifikasi terhadap dokumen yang telah ditandatangani, jika dokumen masih asli maka akan menghasilkan keluaran yang tertera pada Gambar 15.

```
Signature valid: True
```

Gambar 15. Verifikasi Pesan Asli

Dimisalkan sebuah dokumen telah diubah isinya menjadi seperti yang tertera pada Gambar 16.

```
message = b'temui saya di kantor sekarang'
```

Gambar 16. Pesan yang Diubah

Kemudian dokumen tersebut diverifikasi keasliannya dengan mendekripsi tanda tangan dan membandingkan nilai *hash* dari dokumen yang telah dipalsukan. Pada Gambar 17 dapat dilihat keluaran yang dihasilkan dari verifikasi dokumen yang telah dipalsukan.

```
hash integer = 673561127217280951135558149773346254469611383335721295225020543347275825806 1
Signature valid: False
```

Gambar 17. Verifikasi Pesan yang Diubah

4. Kesimpulan

Kriptosistem RSA memanfaatkan dua bilangan prima yang besar untuk mendapatkan kunci (kunci publik dan kunci privat), pada proses penandatangan digunakan kunci privat dan proses verifikasi digunakan kunci publik. Algoritme tanda tangan RSA memiliki langkah yang cukup sederhana namun tingkat keamanan dapat dijamin dengan sulitnya memfaktorkan dua bilangan prima yang besar. Program yang dibuat dapat digunakan dan dijalankan dengan baik sesuai algoritme yang telah dibuat. Program yang dibuat memperlihatkan bahwa perubahan 1 bit pada dokumen akan berpengaruh pada tanda tangan yang dihasilkan.

Saran untuk penelitian yang lebih luas dapat digunakan digit bilangan yang lebih besar agar keamanan dokumen terjaga. Program yang dibangun dapat untuk menyisipkan dokumen asli yang berformat .doc/.pdf sehingga dapat digunakan contoh nyata jika terjadi pemalsuan dokumen.

5. Ucapan Terima Kasih

Terima kasih kepada seluruh pihak yang telah membantu kelancaran penelitian ini sehingga penelitian dapat berjalan lancar dan terselesaikan dengan baik.

6. Daftar Pustaka

[1] Agung, M. (2020). *Pemalsuan Tanda Tangan*. Retrieved from Direktori Putusan Mahkamah Agung Republik Indonesia: https://putusan3.mahkamahagung.go.id/search.html?q=%22Pemalsuan+tanda+tangan%22&page=4

[2] BSrE, B. (2021). *Cara Kerja Tanda Tangan Elektronik*. Retrieved from Balai Sertifikasi Elektronik: https://bsre.bssn.go.id/2021216-berita-cara-kerja-tanda-tangan-elektronik

[3] BSrE, B. (2021). *Jenis Tanda Tangan Elektronik*. Retrieved from Balai Sertifikasi Elektronik: https://bsre.bssn.go.id/202113-artikel-jenis-tanda-tangan-elektronik

[4] Kurniawat, C. M. (2020). *TANDA TANGAN ELEKTRONIK (DIGITAL SIGNATURE).* Aceh: Media Aceh.

[5] Naji Maruf Ilyas, K. (2018). TANDA TANGAN DIGITAL DENGAN SISTEM KRIPTOGRAFI ALGORITMA RIVEST, SHAMIR, ADLEMAN (RSA). *Jurnal Matematika-S1*, 7(4), 25-33.

[6] Kromodimoeljo, S. (2010). *Teori dan Aplikasi Kriptografi.* Jakarta: SPK IT Consulting.