Analisis Pengaruh Nilai Kunci Privat terhadap Keamanan RSA dengan Teorema Wiener

Muchammmad Dimas Sakti Widyatmaja - 13521160
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13521160@std.stei.itb.ac.id

***Abstract*—Algoritma RSA (Rivest—Shamir—Adleman) merupakan algoritma kriptografi yang memiliki kunci enkripsi dan dekripsi berbeda (*asymmetric cryptography*). Salah satu kegunaan dari RSA yaitu untuk melakukan *digital signature*. Wiener's attack, dinamai berdasarkan ahli kriptologi Michael J. Wiener, merupakan salah satu jenis serangan terhadap algoritma RSA. Jenis serangan ini dapat digunakan untuk mencari private key jika private key yang digunakan dalam RSA cukup kecil. Serangan ini memanfaatkan pecahan berlanjut untuk mengekspos private key.**

***Keywords*—RSA, kriptografi, Wiener's attack, private key.**

I. PENDAHULUAN

Dalam era digital, pertukaran informasi berbasis internet sudah marak dilakukan oleh masyarakat dunia. Orang-orang yang tidak bertanggung jawab, bisa saja menginterupsi pertukaran pesan antar pengguna internet. Oleh karena itu, diperlukan suatu cara untuk mengetahui keaslian suatu pesan. Salah satu skema matematis untuk mengetahui keaslian suatu pesan adalah tanda tangan digital (*digital signature*). Tanda tangan digital terdiri dari deret fungsi hash yang dihasilkan dari proses algoritme fungsi hash tertentu yang kemudian disandikan (dienkripsi) dengan algoritme kriftografi kunci asimetris. Untuk memverifikasinya digunakan kunci publik dari algoritma tesebut [1].

RSA merupakan algoritme pertama yang cocok untuk digital signature seperti halnya enkripsi. Algortima RSA diciptakan pada tahun 1977 oleh tiga orang dari Massachusetts Institute of Technology, Ron Rivest, Adi Shamir dan Len Adleman. RSA banyak digunakan karena memiliki beberapa kelebihan, antara lain seperti, implementasi algoritma RSA relatif mudah, RSA aman dan terjamin untuk mentransmisikan data rahasia, pembagian kunci public yang mudah, dan RSA sangat sulit untuk dibobol karena memiliki algoritma yang kompleks. Akan tetapi, RSA juga memiliki beberapa kelemahan seperti, waktu transfer data yang lambat karena perhitungannya melibatkan angka-angka yang besar, pemrosesan tingkat tinggi diperlukan di bagian penerima untuk dekripsi, dan RSA juga tidak boleh dipakai untuk enkripsi data publik [2].

Selain beberapa kelemahan yang telah disebutkan di atas, RSA juga bukan merupakan algoritma yang tidak memiliki celah. Jika pengimplementasian algoritma RSA tidak memenuhi beberapa standar tertentu, RSA dapat dibobol. Salah satu celah keamanan dari RSA terdapat pada bagian kunci privat (*private key*). Apabila kunci privat yang digunakan untuk dekripsi terlalu kecil, orang-orang bisa melakukan serangan untuk mendapatkan kunci privat tersebut untuk mendekripsi *ciphertext* yang telah dienkripsi.

Untuk mengekspos algoritma RSA dengan kunci privat yang kecil, dapat digunakan Wiener's attack. Serangan ini menggunakan pecahan berlanjut untuk merusak system RSA. Serangan ini didasarkan pada Teorema Wiener, yang berlaku untuk nilai kunci privat yang kecil. Wiener telah membuktikan bahwa penyerang dapat secara efisien menemukan kunci privat, d, ketika $d < \frac{1}{3}N^{\frac{1}{4}}$ [3].

II. TEORI DASAR

A. Bilangan Bulat

Bilangan bulat adalah bilangan yang dapat dituliskan tanpa komponen desimal atau pecahan. Sebagai contoh, 21, 4, 0, -3, -67 dan -2048 merupakan bilangan bulat, sedangkan 9,75; dan 50,5 bukan. Himpunan bilangan bulat terdiri dari angka 0, semua bilangan bulat positif (juga disebut dengan bilangan asli), dan invers aditif-nya, semua bilangan bulat negatif [6].

B. Sifat Pembagian Bilangan Bulat

Misalkan a dan b bilangan bulat, $a \neq 0$. a habis membagi b (a *divides* b) jika terdapat bilangan bulat c, sedemikian sehingga b = ac. Notasi: $a \mid b$ jika $b = ac$, $c \in Z$ dan $a \neq 0$. Contoh 1: $4 \mid 12$ karena 12/4 = 3 (bilangan bulat) atau 12 = 4 x 3. Tetapi $4 \nmid 13$ karena 13/4 = 3.25 (bukan bilangan bulat). Berdasarkan Teorema Euclidean, 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 $m = nq + r$ dengan $0 \leq r < n$. Contohnya adalah sebagai berikut, 1987/97 = 20, sisa 47 $1987 = 20 \cdot 97 + 47$. –22/3 = –8, sisa 2 $-22 = (-8) \cdot 3 + 2$ [6].

C. Pembagi Bersama Terbesar (PBB)

Misalkan a dan b bilangan bulat tidak nol. Pembagi bersama terbesar (PBB – greatest common divisor atau gcd) dari a dan b adalah bilangan bulat terbesar d sedemikian hingga $d \mid a$ dan $d \mid b$ [6]. Dalam hal ini kita nyatakan bahwa PBB(a, b) = d, Dalam versi bahasa Inggris, dinotasikan sebagai gcd(a,b) = d atau GCD(a,b) = d. Ada beberapa penulisan notasi faktor persekutuan terbesar, yaitu g.c.d(a,b)} atau (a,b).

D. Relatif Prima

Dua buah bilangan bulat a dan b dikatakan relatif prima jika PBB(a, b) = 1. Contoh, 20 dan 3 relatif prima sebab PBB(20,3) = 1. Apabila dikaitkan dengan kombinasi linier, jika a dan b relatif prima, maka terdapat bilangan bulat m dan n sedemikian sehingga ma + nb = 1. Contoh, Bilangan 20 dan 3 adalah relatif prima karena PBB(20, 3) =1, atau dapat ditulis 2 . 20 + (–13) . 3 = 1 (m = 2, n = –13) Tetapi 20 dan 5 tidak relatif prima karena PBB(20, 5) = 5 ≠ 1 sehingga 20 dan 5 tidak dapat dinyatakan dalam m . 20 + n . 5 = 1 [6].

E. Aritmetika Modulo

Misalkan a dan m bilangan bulat (m > 0). Operasi a mod m (dibaca "a modulo m") memberikan sisa jika a dibagi dengan m.

Notasi: a mod m = r sedemikian sehingga a = mq + r, dengan 0 ≤ r < m. m disebut modulus atau modulo, dan hasil aritmetika modulo m terletak di dalam himpunan {0, 1, 2, …, m – 1} [6].

F. Kongruen dan Kekongurenan Linier

Definisi dari kekongruenan yaitu, misalkan a dan b bilangan bulat dan m adalah bilangan > 0, maka a ≡ b (mod m) jika dan hanya jika m | (a – b). Jika a tidak kongruen dengan b dalam modulus m, maka ditulis a ≡ b (mod m). Misalnya 38 mod 5 = 3 dan 13 mod 5 = 3, maka dikatakan 38 ≡ 13 (mod 5) (dibaca: 38 kongruen dengan 13 dalam modulus 5). a ≡ b (mod m) dalam bentuk "sama dengan" dapat dituliskan sebagai a = b + km (k adalah bilangan bulat). a mod m = r dapat juga ditulis a ≡ r (mod m) [6].

Misalkan m adalah bilangan bulat positif.

1)Jika a ≡ b (mod m) dan c adalah sembarang bilangan bulat maka

(i) (a + c) ≡ (b + c) (mod m),

(ii) ac ≡ bc (mod m),

(iii) a p ≡ b p (mod m), p bilangan bulat tak-negatif.

2) Jika a ≡ b (mod m) dan c ≡ d (mod m), maka

(i) (a + c) ≡ (b + d) (mod m),

(ii) ac ≡ bd (mod m).

Kekongruenan linier (*linear congruence*) berbentuk: ax ≡ b (mod m) (m > 0, a dan b sembarang bilangan bulat, dan x adalah peubah bilangan bulat).

Pemecahan: ax = b + km → $x = \frac{b+km}{a}$

(Cobakan untuk k = 0, 1, 2, … dan k = –1, –2, … yang menghasilkan x sebagai bilangan bulat) [6].

G. Balikan Modulo

Balikan dari a (mod m) adalah bilangan bulat x sedemikian sehingga: xa ≡ 1 (mod m). Dalam notasi lainnya, a –1 (mod m) = x. Balikan modulo memiliki syarat yaitu jika a dan m relatif prima dan m > 1, maka balikan (invers) dari a (mod m) ada [6].

H. Bilangan Prima

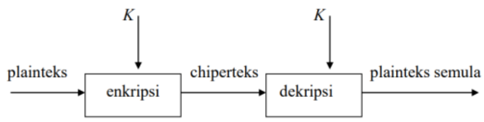
Bilangan prima yaitu bilangan bulat positif lebih dari 1 yang hanya mempunyai dua faktor pembagi, yaitu bilangan itu sendiri dan 1. Bilangan asli yang lebih dari 1 dan bukan bilangan prima disebut bilangan komposit. Karena bilangan prima harus lebih besar dari 1, maka barisan bilangan prima dimulai dari 2, yaitu 2, 3, 5, 7, 11, 13, …. Seluruh bilangan prima adalah bilangan ganjil, kecuali 2 yang merupakan bilangan genap. Setiap bilangan bulat positif yang lebih besar atau sama dengan 2 dapat dinyatakan sebagai perkalian satu atau lebih bilangan prima. Berdasarkan Teori Fermat, jika p adalah bilangan prima dan a adalah bilangan bulat yang tidak habis dibagi dengan p, yaitu PBB(a, p) = 1, maka: $a^{p-1} \equiv 1 \pmod p$ [7].

I. Kriptografi

Dari Bahasa Yunani yang artinya "secret writing". Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan dengan cara menyandikannya menjadi bentuk lain yang tidak bermakna. Tujuan: agar pesan yang bersifat rahasia tidak dapat dibaca oleh pihak yang tidak berhak.

Pesan: data atau informasi yang dapat dibaca dan dimengerti maknanya. Nama lain: plainteks (plaintext). Cipherteks (ciphertext): pesan yang telah disandikan sehingga tidak memiliki makna lagi.

Enkripsi (encryption): proses menyandikan plainteks menjadi cipherteks. Dekripsi (decryption): Proses mengembalikan cipherteks menjadi plainteksnya [8].



(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/TeoriBilangan-2020-Bagian3.pdf)

Terdapat berbagai jenis algoritma untuk kriptografi. Pada kriptografi modern, algoritmanya disusun berdasarkan pada teori matematis dan aplikasi komputer. Salah satu contoh dari algoritma kriptografi modern adalah RSA.

J. Algoritma RSA

RSA dibuat oleh tiga peneliti dari MIT (Massachussets Institute of Technology), yaitu Ronald Rivest, Adi Shamir, dan Leonard Adleman, pada tahun 1976. Keamanan RSA bergantung pada kesulitan praktis produk pemfaktoran dari dua bilangan prima besar, "*the factoring problem*" [4]. RSA merupakan algoritma yang relatif lambat. Oleh karena itu, RSA tidak umum digunakan untuk mengenkripsi data *user* secara langsung. RSA lebih sering digunakan untuk mengirimkan kunci bersama untuk kriptografi kunci simetris, yang kemudian digunakan untuk enkripsi-dekripsi massal.

Prosedur pembangkitan pasangan kunci di dalam RSA

1. Pilih dua bilangan prima, p dan q (rahasia)
2. Hitung n = pq. Besaran n tidak perlu dirahasiakan.
3. Hitung m = (p – 1)(q – 1). (rahasia)
4. Pilih sebuah bilangan bulat untuk kunci publik, e, yang relatif prima terhadap m, yaitu PBB(e, m) = 1.
5. Hitung kunci dekripsi, d, melalui kekongruenan ed ≡ 1 (mod m). [8]

K. Pecahan berlanjut

Pecahan berlanjut sederhana adalah ekspresi dari bentuk berikut x=a [5]:

$$x = a0 + \frac{1}{a1 + \left(\frac{1}{a2 + \left(\frac{1}{a3 + \cdots}\right)}\right)}$$

Dengan a*i* disebut hasil bagi. Notasi pecahan berlanjut dapat singkat dengan ekspansi pecahan berlanjutnya [5]:

$x=[a0,a1,a2,a3,\ldots,an]$

L. *Wiener's Attack*

Teorema Wiener menjelaskan bahwa. Misalkan N = pq dengan q < p < 2q. Misalkan d < $\frac{1}{3}N^{\frac{1}{4}}$. Diberikan <e,N> dengan ed = 1 mod m, d dapat dengan efisien didapatkan dengan mencari $\frac{k}{d}$ yang benar di anatara konvergensi-konvergensi dari $\frac{e}{N}$ [5].

Dalam memvalidasi kebenaran pada hasil serangan wiener, perlu dilakukan pemfaktoran N dengan m. Misalkan p dan q adalah bilangan prima yang hasil perkaliannya adalah N, maka kita memiliki:

m = N – (p + q) + 1

p + q = N – m + 1

Misal terdapat persamaan kuadrat (x-p)(x-q) = 0, yang akar-akarnya adalah p dan q, maka:

$$(x-p)(x-q)=0$$
$$x^2+x(p+q)-N=0$$
$$x^2+x(N-m+1)-N=0$$

Jika hasil m yang didapatkan benar, maka akar-akar dari persamaan tersebut akan menjadi bilangan bulat dan merupakan faktor dari N.

III. SKEMA SERANGAN WIENER

A. *Serangan Efektif*

Pada skema ini, penulis akan menguji keberhasilan serangan Wiener berdasarkan teorema Wiener yang menyatakan bahwa jika kunci privat yang digunakan cukup kecil, maka serangan Wiener dapat berhasil. Keseluruhan dari jalannya Serangan Wiener adalah sebagai berikut:

- Membuat pasangan kunci yang rentan (yang memiliki kunci privat pendek ($d<\frac{1}{3}N^{\frac{1}{4}}$)).
- Mencari konvergensi dari ekspansi pecahan berlanjut dari $\frac{e}{N}$.
- Melakukan iterasi dari konvergen $\frac{d_i}{k_i}$:
  - Menghitung m_i dengan k_i dan d_i.
  - Memastikan kebeneran melalui faktorisasi N dengan m_i. (Validasi kebenaran juga dapat dilakukan dengan melakukan tes enkripsi-dekripsi dengan pesan tertentu).

B. *Serangan Tidak Efektif*

Pada skema ini, penulis akan menguji serangan Wiener menggunakan kunci privat yang cukup besar, $d>\frac{1}{3}N^{\frac{1}{4}}$. Dengan nilai kunci privat yang cukup besar, harusnya hal ini dapat dengan efektif menggagalkan serangan Wiener.

IV. IMPLEMENTASI

Pada makalah ini, penulis menggunakan bahasa python untuk implementasi pembuatan pasangan kunci dan serangan Wiener. Penulis menggunakan bahasa python karena sudah tersedia banyak *library* yang dapat dimanfaatkan dalam rangka membuat pasangan kunci maupun melakukan *hashing* terhadap pesan. Penulis menggunakan kode yang diambil dari https://github.com/sagi/code_for_blog/tree/b0e4af242d11eedcf7cda3d58ec9df09e39d8388/2016/wieners-rsa-attack dengan sedikit perubahan untuk membuat pasangan kunci yang tidak rentan. Fungsi-fungsi yang digunakan dipisah menjadi tiga file, key.py, cf.py, dan wiener.py. key.py terdiri atas fungsi-fungsi yang diperlukan untuk membuat komponen-komponen RSA, yaitu fungsi get_prime, test_key, create_keypair_vuln, dan create_keypair_nonvuln. Program cf.py berisi fungsi get_cf_expansion dan get_convergence untuk menghitung konvergensi dan ekspansi konvergensi. Program wiener.py berisi program utama dan fungsi sha1 untuk *hashing*. Penjabaran fungsi-fungsi tersebut antara lain:

A. *get_prime*

Fungsi ini menghasilkan bilangan prima dengan ukuran *size* bit.

```
def get_prime(size):
    while True:
        r = urandom.getrandbits(size)
        if gmpy2.is_prime(r): # Miller-rabin
            return r
```

Gambar 4.1. Fungsi get_prime

B. *test_key*

Fungsi ini digunakan untuk mengecek kebenaran pasangan kunci yang digunakan.

```
def test_key(N, e, d):
    msg = (N - 123) >> 7
    c = pow(msg, e, N)
    return pow(c, d, N) ==  msg
```

Gambar 4.2. Fungsi test_key

C. *create_keypair_vuln*

```
def create_keypair_vuln(size):
    while True:
        p = get_prime(size // 2)
        q = get_prime(size // 2)
        if q < p < 2*q:
            break

    N = p * q
    m = (p - 1) * (q - 1)

    # Recall that: d < (N^(0.25))/3
    max_d = c_div(isqrt(isqrt(N)), 3)
    max_d_bits = max_d.bit_length() - 1

    while True:
        d = urandom.getrandbits(max_d_bits)
        try:
            e = int(gmpy2.invert(d, m))
        except ZeroDivisionError:
```

```
            continue
        if (e * d) % m == 1:
            print('N is', N)
            print('private key (d) used is', d)
            break
    assert test_key(N, e, d)

    return  N, e, d, p, q
```

Gambar 4.3. Fungsi create_keypair_vuln

Fungsi ini digunakan untuk membuat pasangan kunci dengan panjang kunci privat yang kecil untuk membuatnya rentan terhadap serangan Wiener. Fungsi ini pertama-tama membuat pasangan p dan q. Jika sudah mendapatkan p dan q, maka N dan m dapat dibuat. Setelah itu, kunci privat, d, dibuat dengan acak dengan nilai maksimal sebesar $\frac{1}{3}N^{\frac{1}{4}}$. Kunci public dicari dengan menggunakan invers modular dengan gmpy2.invert. Setelah semua elemen RSA didapat, N, e, dan d dites, apabila hasilnya sudah benar, maka N, e, d, p, dan q akan di-*return*. Apabila hasilnya tidak benar maka akan terjadi Assertion error.

D. create_keypair_nonvuln

```
def create_keypair_nonvuln(size):
    while True:
        p = get_prime(size // 2)
        q = get_prime(size // 2)
        if q < p < 2*q:
            break

    N = p * q
    m = (p - 1) * (q - 1)

    # Max d is N/3 now
    max_d = c_div(N, 3)
    max_d_bits = max_d.bit_length() - 1

    while True:
        d = urandom.getrandbits(max_d_bits)
        try:
            e = int(gmpy2.invert(d, m))
        except ZeroDivisionError:
            continue

        # d must be bigger than (N^(0.25))/3
        if (e * d) % m == 1 and c_div(isqrt(isqrt(N)), 3):
            print('N is', N)
            print('private key (d) used is', d)
            break
    assert test_key(N, e, d)

    return  N, e, d, p, q
```

Gambar 4.4. Fungsi create_keypair_nonvuln

Fungsi ini mirip dengan fungsi *create_keypair_vuln*. Beberapa elemen yang diubah antara lain nilai maksimal kunci privat, d, yang menjadi N/3 dan penempatan output nilai N dan d.

E. get_cf_expansion

```
def get_cf_expansion(n, d):
    e = []
    q = n // d
    r = n % d

    e.append(q)

    while r != 0:
        n, d = d, r
        q = n // d
        r = n % d
        e.append(q)

    return e
```

Gambar 4.5. Fungsi get_cf_expansion

Fungsi ini akan digunakan untuk mencari list yang memiliki elemen ekspansi pecahan berlanjut dari kunci publik dengan N.

F. get_convergence

```
def get_convergents(e):
    n = [] # Nominators
    d = [] # Denominators

    for i in range(len(e)):
        if i == 0:
            ni = e[i]
            di = 1
        elif i == 1:
            ni = e[i]*e[i-1] + 1
            di = e[i]
        else: # i > 1
            ni = e[i]*n[i-1] + n[i-2]
            di = e[i]*d[i-1] + d[i-2]

        n.append(ni)
        d.append(di)
        yield (ni, di)
```

Gambar 4.6. Fungsi get_convergents

Fungsi ini akan digunakan untuk mencari konvergensi-konvergensi dari ekspansi pecahan berlanjut dari kunci publik dengan N.

G. sha1

```
def sha1(n):
    h = hashlib.sha1()
    h.update(str(n).encode('utf-8'))
    return h.hexdigest()
```

Gambar 4.7. Fungsi sha1

Fungsi ini digunakan untuk melakukan *hashing* dengan tipe SHA1 kepada n. Semua elemen RSA nantinya akan di-*hash* dengan fungsi ini.

H. Program utama

```python
if __name__ == '__main__':
    vuln = str(input('Would you like to make the private key short? (y/n) '))
    if vuln == 'y' :
        N, e, d, p, q = vk.create_keypair_vuln(1024)
    elif vuln == 'n' :
        N, e, d, p, q = vk.create_keypair_nonvuln(1024)
    else :
        print('invalid input. exiting program')
    print('[+] Generated an RSA keypair with a short private exponent.')
    print('[+] For brevity, keypair components are crypto. hashed:')
    print('[+] ++ SHA1(e):    ', sha1(e))
    print('[+] -- SHA1(d):    ', sha1(d))
    print('[+] ++ SHA1(N):    ', sha1(N))
    print('[+] -- SHA1(p):    ', sha1(p))
    print('[+] -- SHA1(q):    ', sha1(q))
    print('[+] -- SHA1(m):    ', sha1((p - 1)*(q - 1)))
    print('[+] -----------------')

    cf_expansion = cf.get_cf_expansion(e, N)
    convergents = cf.get_convergents(cf_expansion)
    print('[+] Found the continued fractions expansion convergents of e/N.')

    print('[+] Iterating over convergents; '
            'Testing correctness through factorization.')
    print('[+] ...')
    for pk, pd in convergents: # pk - possible k, pd - possible d
        if pk == 0:
            continue;

        possible_m = (e*pd - 1)//pk

        p = Symbol('p', integer=True)
        roots = solve(p**2 + (possible_m - N - 1)*p + N, p)

        if len(roots) == 2:
            pp, pq = roots # pp - possible p, pq - possible q
            if pp*pq == N:
                print('[+] Factored N! :) derived keypair components:')
                print('[+] ++ SHA1(e):    ', sha1(e))
                print('[+] ++ SHA1(d):    ', sha1(pd))
                print('[+] ++ SHA1(N):    ', sha1(N))
                print('[+] ++ SHA1(p):    ', sha1(pp))
                print('[+] ++ SHA1(q):    ', sha1(pq))
                print('[+] ++ SHA1(m):    ', sha1(possible_m))
                sys.exit(0)

    print('[-] Wiener\'s Attack failed; Could not factor N')
    sys.exit(1)
```

Gambar 4.8. Program utama

Bagian ini akan menjalankan serangan Wiener. Program pertama-tama akan menanyai pengguna Apakah akan menggunakan kunci privat yang pendek agar rentan terhadap serangan Wiener atau tidak. Jika pengguna pengguna memilih menggunakan kunci privat yang rentan maka program akan menjalankan fungsi create_keypair_vuln. Sebaliknya, program akan menjalankan fungsi create_key_pair_nonvuln.

Setelah mendapatkan komponen-komponen dari RSA, program akan melakukan hashing terhadap komponen-komponen tersebut. Di sini, semua komponen akan dicetak ke layer, namun, dalam program ini, pengguna akan dianggap hanya memiliki e (kunci publik) dan N. Setelah itu, akan dicari ekspansi pecahan berlanjut dari e dan N serta konvergensinya. Setelah list dari konvergensinya ditemukan, program akan melakukan iterasi terhadapnya dan melakukan validasi dengan menggunakan persamaan kuadrat. Jika akar-akarnya berhasil ditemukan, maka Wiener's attack berhasil dan program telah menemukan N yang dapat membuat pengguna dapat menemukan komponen-komponen lainnya. Jika tidak ada akar-akarnya, maka Wiener's attack gagal dan pengguna tidak dapt mendapatkan N.

V. UJI COBA

Berikut merupakan hasil percobaan terhadap Wiener's attack dengan menggunakan kunci privat yang pendek dan panjang. Untuk kunci privat yang pendek, penulis membatasi nilai kunci privat sesuai dengan batas efektif serangan Wiener yang telah ditemukan di teorema Wiener yaitu $\frac{1}{3}N^{\frac{1}{4}}$. Sedangkan untuk kunci privat yang panjang, penulis menentukan batas atasnya sebesar $\frac{1}{3}N$ dan batas bawahnya sebesar $\frac{1}{3}N^{\frac{1}{4}}$, jadi nilainya berada di atas ambang efektif serangan Wiener.

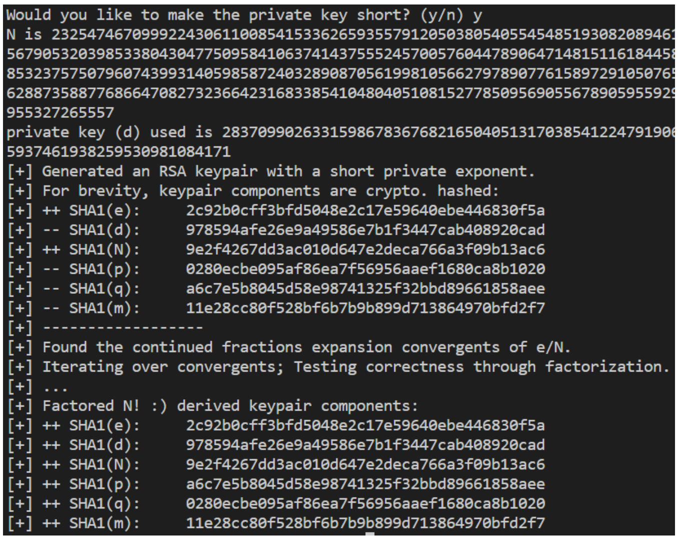
Berikut merupakan beberapa hasil dengan kunci privat yang pendek.

```
Would you like to make the private key short? (y/n) y
N is 13005831959301599134038653655413716329233458122820929763595460187899355
23619215893243451805787859283951262236066286837421652573443016985919280296
45275154327679827032851351764199731313453208401676506504290287185884697487
10985590545681207389792537739527462927875438389533912800917527998490858208
6761291349809187
private key (d) used is 66010191007644322060263751103406553800605158558738
69119396020023397351021353
[+] Generated an RSA keypair with a short private exponent.
[+] For brevity, keypair components are crypto. hashed:
[+] ++ SHA1(e):     7e1335c91a7e50c06105e32e8ebb0ccc30b4d9ed
[+] -- SHA1(d):     53adde144f7baacb3e5547b00d13868ce8b0dc21
[+] ++ SHA1(N):     941b35b8370e7d9d113882a6069c524bd3f513c6
[+] -- SHA1(p):     c7ccc15f8d773bac293020d8180e71d399e34590
[+] -- SHA1(q):     4e7dfa89043c87185a118195f6959a822a5185a3
[+] -- SHA1(m):     d30552b7be6ce8cfb78278c19158ae800f2fce75
[+] -----------------
[+] Found the continued fractions expansion convergents of e/N.
[+] Iterating over convergents; Testing correctness through factorization.
[+] ...
[+] Factored N! :) derived keypair components:
[+] ++ SHA1(e):     7e1335c91a7e50c06105e32e8ebb0ccc30b4d9ed
[+] ++ SHA1(d):     53adde144f7baacb3e5547b00d13868ce8b0dc21
[+] ++ SHA1(N):     941b35b8370e7d9d113882a6069c524bd3f513c6
[+] ++ SHA1(p):     4e7dfa89043c87185a118195f6959a822a5185a3
[+] ++ SHA1(q):     c7ccc15f8d773bac293020d8180e71d399e34590
[+] ++ SHA1(m):     d30552b7be6ce8cfb78278c19158ae800f2fce75
```

Gambar 5.1. Hasil uji coba kunci privat pendek 1

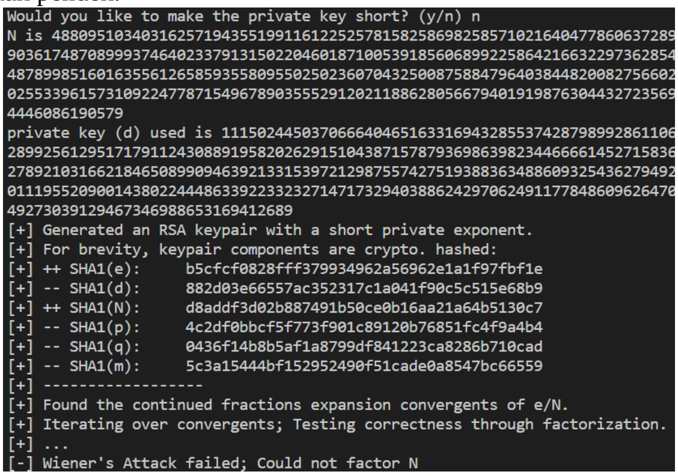
```
Would you like to make the private key short? (y/n) y
N is 25268538484129048512404857456647361949338414385651570749939524188833063
33353330624389492600203134224721899128849090335111861999277300422229510020
44622095061509850247311040458386393690629378476520140661685065540437919816
76941632031528824111149481804893693194754617800099346303000867771467289272
529923555745619
private key (d) used is 60583913708767231575588634148520482296401983601232
49193067222492147175004527
[+] Generated an RSA keypair with a short private exponent.
[+] For brevity, keypair components are crypto. hashed:
[+] ++ SHA1(e):     34ccb8a68c9b97265eb421ed7fa52713f070234c
[+] -- SHA1(d):     c594f044a0feefc2695b9c2cc247b68a8e5c75cb
[+] ++ SHA1(N):     58f68f9e858c99f57af5404642cec759fa06356e
[+] -- SHA1(p):     ee52176bf652b856025fafde092700836aef6250
[+] -- SHA1(q):     e566dd17c53d61bb2f23d31aca5368e45827cd79
[+] -- SHA1(m):     602c7d1ebd8a248663fb52afb25886603bc6d4ed
[+] -----------------
[+] Found the continued fractions expansion convergents of e/N.
[+] Iterating over convergents; Testing correctness through factorization.
[+] ...
[+] Factored N! :) derived keypair components:
[+] ++ SHA1(e):     34ccb8a68c9b97265eb421ed7fa52713f070234c
[+] ++ SHA1(d):     c594f044a0feefc2695b9c2cc247b68a8e5c75cb
[+] ++ SHA1(N):     58f68f9e858c99f57af5404642cec759fa06356e
[+] ++ SHA1(p):     e566dd17c53d61bb2f23d31aca5368e45827cd79
[+] ++ SHA1(q):     ee52176bf652b856025fafde092700836aef6250
[+] ++ SHA1(m):     602c7d1ebd8a248663fb52afb25886603bc6d4ed
```

Gambar 5.2. Hasil uji coba kunci privat pendek 2

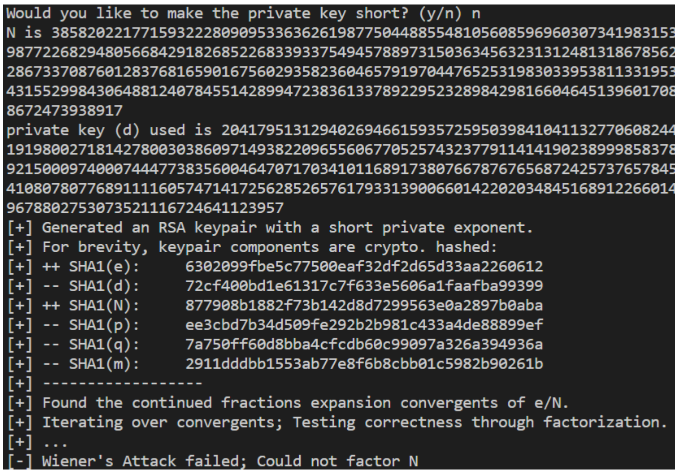


Gambar 5.3. Hasil uji coba kunci privat pendek 3

Berikut merupakan beberapa hasil dengan kunci privat yang tidak pendek.



Gambar 5.4. Hasil uji coba kunci privat tidak pendek 1



Gambar 5.5. Hasil uji coba kunci privat tidak pendek 2

Gambar 5.6. Hasil uji coba kunci privat tidak pendek 3

VI. KESIMPULAN

Berdasarkan uji coba yang telah dilakukan, dapat disimpulkan bahwa teorema Wiener terbukti benar dalam hal serangan Wiener. Jika kunci privat yang digunakan panjangnya kurang dari sebesar $\frac{1}{3}N^{\frac{1}{4}}$, maka RSA akan rentan terhadap serangan Wiener. Oleh karena itu, dalam pembuatan RSA, komponen kunci privat dari RSA perlu diperhatikan agar nilainya tidak terlalu kecil sehingga sistem RSA menjadi aman.

VII. UCAPAN TERIMA KASIH

Penulis menyampaikan puji syukur kepada Tuhan Yang Maha Esa atas berkat-Nya yang memungkinkan penulis menyelesaikan makalah ini. Penulis juga berterima kasih kepada dosen pengampu mata kuliah Matematika Diskrit Semester Ganjil 2022/2023 kelas 02, Ibu Fariska Zakhralativa Ruskanda, S.T., M.T., yang telah menyalurkan ilmu-ilmu yang digunakan dalam penulisan makalah ini. Tugas makalah ini telah menambah wawasan Matematika Diskrit yang lebih mendalam bagi penulis dan juga menambah pengalaman penulis dalam pembuatan makalah ilmiah.

REFERENCES

[1] Husni, Emir; Leksono, Bramanto; Rosa, Muhammad Ridho (2015-09). "Digital signature for contract signing in service commerce". 2015 International Conference on Technology, Informatics, Management, Engineering & Environment (TIME-E). Samosir, Toba Lake, Indonesia: IEEE: 111–116.

[2] https://www.geeksforgeeks.org/rsa-full-form/, diakses pada 10 Desember 2022.

[3] Boneh, Dan (1999). Twenty Years of attacks on the RSA Cryptosystem. Notices of the American Mathematical Society (AMS) 46.

[4] Casteivecchi, Davide, Quantum-computing pioneer warns of complacency over Internet security, Nature, October 30, 2020 interview of Peter Shor.

[5] https://sagi.io/crypto-classics-wieners-rsa-attack/, diakses pada 10 Desember 2022.

[6] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/TeoriBilangan-2020-Bagian1.pdf, diakses pada 10 Desember 2022

[7] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/TeoriBilangan-2020-Bagian2.pdf, diakses pada 10 Desember 2022

[8] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/TeoriBilangan-2020-Bagian3.pdf, diakses pada 10 Desember 2022

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 Desember 2022

Muchammad Dimas Sakti Widyatmaja - 13521160