SinkrOn
JURNAL & PENELITIAN TEKNIK INFORMATIKA
Jurnal & Penelitian Teknik Informatika
Volume 2 Nomor 2, Oktober 2017
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Teknik Pemecahan Kunci Algoritma *Rivest Shamir Adleman* (RSA) dengan Metode *Kraitchik*

Budi Satria Muchlis
Program Studi S-1 Ilmu Komputer
vandeboeds@gmail.com

M. Andri Budiman
Fakultas Ilmu Komputer dan Teknologi Informasi
mandrib@usu.ac.id

Dian Rachmawati
Universitas Sumatera Utara
dian.rachmawati@usu.ac.id

Abstrak—Penelitian ini bertujuan memecahkan kunci privat algoritma RSA dengan memfaktorkan kunci publik *n* menggunakan metode *Kraitchik* dan melihat efisiensi waktu pemfaktorannya. Kriptanalisis dengan pemfaktoran (*factoring*) menggunakan kunci publik *n* yaitu $n = p \cdot q$ yang tidak dirahasiakan untuk memecahkan kunci privat RSA. Jika kunci publik *n* berhasil difaktorkan menjadi *p* dan *q* maka $\phi(n) = (p - 1)(q - 1)$ dapat dihitung dan dengan menggunakan kunci publik *e*, kunci privat *d* pun akan dapat terpecahkan. Metode *Kraitchik* yang mengawali munculnya algoritma pemfaktoran yang paling modern menyatakan bahwa untuk menemukan faktor *x* dan *y* dari bilangan bulat *n* sedemikian rupa sehingga $x^2 \equiv y^2 \pmod{n}$. Hasil penelitian memperlihatkan bahwa efisiensi waktu pemfaktoran kunci publik *n* metode *Kraitchik* sangat dipengaruhi oleh selisih faktor kunci ($p - q$) yaitu semakin besar selisih antara *p* dan *q* maka semakin lama waktu pemfaktorannya. Pemfaktoran panjang kunci publik *n* sebesar 19 digit atau 152 bit dengan selisih faktor kunci (p – q) = 22641980 membutuhkan waktu selama 93,6002 *ms* lebih cepat jika dibandingkan dengan panjang kunci sebesar 15 digit atau 120 bit dengan selisih faktor kunci (p – q) = 23396206 yang membutuhkan waktu selama 5850,0103 *ms*. Faktor lainnya yang mempengaruhi efisiensi waktu pemfaktoran metode *Kraitchik* adalah *Gcd* (p – 1, q – 1), panjang kunci dan faktor prima (p – 1), (q – 1).

Kata kunci — RSA, Kriptanalisis, Pemfaktoran (*Factoring*), Metode *Kraitchik.*

I. PENDAHULUAN

A. Latar Belakang

Rivest Shamir Adleman (RSA) adalah salah satu algoritma kriptografi asimetris (kriptografi kunci-publik) yaitu menggunakan dua kunci yang berbeda (*private key* dan *public key*). Kekuatan algoritma RSA tidak hanya terletak pada panjang kuncinya (semakin panjang kunci, maka semakin lama waktu kerja) dan penggunaan kunci-publik dan kunci privat pada umumnya. Kekuatan utama dari algoritma RSA didasarkan pada sulitnya memfaktorkan bilangan besar menjadi faktor-faktor primanya : faktorkan *n* menjadi dua faktor primanya, *p* dan *q*, sedemikian sehingga $n = p \cdot q$.

Walaupun demikian, RSA yang populer sampai saat ini dan masih sulit untuk dipecahkan memiliki celah keamanan dengan hanya mengetahui *ciphertext* dan kunci publiknya. Salah satu teknik serangan pada RSA adalah dengan memanfaatkan celah pada 'keunggulannya' yaitu dengan memfaktorkan kunci publik *n* (seperti penjelasan di atas). Kelemahan inilah yang dimanfaatkan untuk melakukan serangan atau *hacking* untuk menguji keamanan algoritma RSA dalam dalam merahasiakan pesan dan mengetahui kelemahan-kelemahannya. Beberapa penelitian yang menerapkan teknik pemfaktoran adalah penelitian dari Mahadi Z. berjudul "*Teknik Hacking Algoritma Rivest Shamir Adleman (RSA) 512-Bit Dengan Menggunakan Metode Difference of Squares*" dan Zeni Fera Bhakti berjudul

"*Kriptanalisis RSA Dengan Kurva Eliptik (Cryptanalysis of RSA with Elliptic Curves)*". Kedua penelitian tersebut menjadi inspirasi penulis untuk membuat penelitian mengenai teknik pemecahan kunci RSA untuk memfaktorkan kunci publik *n* dengan metode yang berbeda yaitu pemfaktoran metode *Kraitchik* dengan judul **"Teknik Pemecahan Kunci Algoritma *Rivest Shamir Adleman (RSA*) Dengan Metode *Kraitchik*"**. Besar harapan penulis dengan mengetahui kelebihan dan kekurangan algoritma RSA maupun metode *Kraitchik* dari penelitian ini akan menjadi perbaikan dan pengembangan penelitian-penelitian sebelumnya bahkan dapat menambah literatur dan wawasan dunia kriptanalisis.

B. Rumusan Masalah

Berdasarkan latar belakang di atas, maka dapat dirumuskan masalah yang akan diselesaikan dalam penelitian ini adalah apakah kunci privat algoritma RSA dapat dipecahkan secara efisien dengan memfaktorkan kunci publik *n* menggunakan metode *Kraitchik* ditinjau dari parameter waktu.

C. Batasan Masalah

Adapun batasan-batasan penelitian agar tidak menyimpang dari rumusan masalah adalah:

1. Ada dua proses utama pada sistem, yaitu:
   a. Pemfaktoran kunci publik *n* untuk menemukan nilai *p* dan *q* sehingga mendapat kunci privatnya dengan metode *Kraitchik*.
   b. Proses dekripsi *ciphertext* menjadi *plaintext* dengan menggunakan kunci privat yang telah dipecahkan.
2. Algoritma kriptografi yang diuji adalah RSA dengan panjang kunci maksimum 512 bit dan menggunakan uji bilangan prima *Fermat's Little Theorem*.
3. Parameter efisiensi pemecahan kunci dihitung dari lamanya *running time* program dalam satuan milisekon (*ms*).
4. *Ciphertext* berupa numerik (angka) dan *plaintext*-nya diubah menjadi karakter ASCII.

D. *Tujuan Penelitian*

Sesuai dengan rumusan masalah di atas, tujuan dari penelitian ini adalah:

1. Memecahkan kunci privat algoritma RSA dengan memfaktorkan kunci publik *n* menggunakan metode *Kraitchik.*
2. Melihat efisiensi waktu pemfaktoran kunci publik *n* dengan metode *Kraitchik*.

E. Manfaat Penelitian

Manfaat dari penelitian ini antara lain:

1. Mengetahui efisiensi metode *Kraitchik* dalam memecahkan kunci algoritma RSA sebagai acuan untuk mengembangkan metode kriptanalisis yang lebih efisien.
2. Mengetahui kemampuan algoritma RSA untuk bertahan dari serangan sebagai bahan evaluasi untuk memperbaiki dan mengembangkan tingkat keamanan algoritma RSA.

F. Metodologi Penelitian

Tahapan yang dilakukan dalam penelitian ini adalah:

1. Studi Literatur
   Tahap ini dilakukan dengan mengumpulkan informasi dan mempelajari literatur pada sejumlah buku, artikel, *paper*, jurnal, makalah, maupun situs internet khususnya berkaitan dengan konsep kriptografi, algoritma RSA, teknik-teknik kriptanalisis RSA, serta landasan matematika yang berkaitan seperti teori bilangan dan pemfaktoran bilangan bulat metode *Kraitchik*.
2. Analisis dan Perancangan Sistem
   Tahap ini akan dilakukan analisis masalah yang dihadapi dengan mengidentifikasi masalah, memodelkan masalah secara konseptual dengan UML, *flowchart*, tujuan, dan kandidat solusi yang ditawarkan. Kemudian dilakukan perancangan *interface* aplikasi berisikan tahapan-tahapan operasi dalam proses pengolahan data dan prosedur untuk mendukung aplikasi tersebut.

3. Implementasi Sistem
   Tahap ini akan menerapkan hasil konseptual analisis dan perancangan sistem dengan pengkodean (*coding*) program.
4. Pengujian Sistem
   Tahap ini akan dilakukan pengujian terhadap sistem dengan tujuan untuk melihat semua kesalahan dan kekurangan yang ada pada sistem serta semua data mengenai efisiensi waktu sistem. Pengujian yang dilakukan dengan menjalankan sistem dan memasukkan berbagai *input* pada tiap-tiap fungsi dan fasilitas yang dimiliki sistem dan melihat hasil *output*-nya.
5. Dokumentasi
   Tahap ini akan dilakukan penyusunan laporan dari hasil analisis, perancangan, implementasi hingga pengujian sistem dalam format penulisan penelitian.

II. TINJAUAN PUSTAKA

A. Kriptografi

The Concise Oxford Dictionary (2006) mendefinisikan kriptografi sebagai seni menulis atau memecahkan kode. Definisi ini tidak sesuai dengan hakikat dari kriptografi modern. *Pertama*, hanya berfokus pada masalah komunikasi rahasia hanya sebatas sebuah kode. *Kedua*, definisi tersebut mengacu pada kriptografi sebagai bentuk seni. Memang benar sampai abad 20 (dan bisa dibilang sampai di akhir abad itu), kriptografi adalah sebuah seni. Namun, pada akhir abad ke-20 hingga sekarang, banyaknya teori yang bermunculan menjadikan kriptografi sebagai bidang keilmuan. Kriptografi sekarang digunakan di berbagai tempat yang terintegrasi dengan sistem komputer. Ruang lingkupnya meliputi lebih dari komunikasi rahasia, namun termasuk otentikasi pesan, tanda tangan digital, protokol untuk bertukar kunci rahasia bahkan masalah-masalah yang mungkin timbul dalam komputasi terdistribusi baik serangan internal atau eksternal. [5]

Beberapa buku mendefinisikan istilah kriptografi sebagai berikut:

1. Kriptografi adalah teknik-teknik studi ilmiah untuk mengamankan informasi digital, transaksi dan komputasi terdistribusi [5].
2. Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan [12].
3. Kriptografi adalah ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerhasiaan, integrasi data, otentikasi entitas dan otentikasi asal usul data [8].

Jadi, kriptografi adalah bidang ilmu yang mempelajari teknik-teknik pengamanan dalam melakukan transaksi informasi dan komputasi terdistribusi untuk memenuhi aspek keamanan informasi.

Kita dapat merangkum bahwa kriptografi bertujuan untuk memberi layanan keamanan (mencapai tujuan keamanan informasi) sebagai berikut:

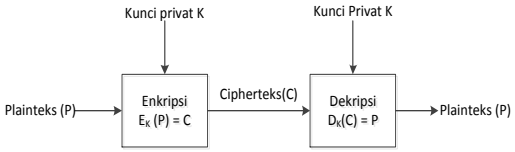
1. Kerahasiaan (*Confidentiality*), adalah layanan yang ditujukan untuk menjaga agar pesan tidak dapat dibaca oleh pihak-pihak yang tidak berhak. Istilah lainnya adalah *secrecy* dan *privacy*.
2. Integritas Data (*Data Integrity*), adalah layanan yang menjamin bahwa pesan masih asli/utuh atau belum pernah dimanipulasi selama pengiriman.
3. Otentikasi (*Authentication*), adalah layanan yang berhubungan dengan identifikasi baik mengidentifikasi kebenaran pihak-pihak yang berkomunikasi (*entity authentication*) maupun mengidentifikasi asal usul pesan (*data origin authentication*).
4. Penyangkalan (*Non-Repudiation*), adalah layanan untuk mencegah entitas yang berkomunikasi melakukan penyangkalan yaitu pengirim pesan telah menyangkal melakukan pengiriman atau penerima pesan menyangkal telah menerima pesan. [8] [12]

B. Sistem Kriptografi (Cryptosystem)

Sistem kriptografi (*cryptosystem*) sering disebut juga dengan sistem cipher (*cipher system*) adalah sistem yang terdiri dari algoritma enkripsi, algoritma dekripsi dan tiga komponen teks (*plaintext*, *ciphertext* dan kunci)

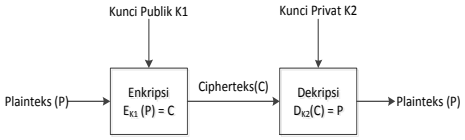
[15]. Secara umum ada dua jenis sistem kriptografi berbasis kunci:

1. Sistem kriptografi simetris (*Symmetric Cryptosystem*), sering disebut algoritma konvensional, adalah algoritma di mana kunci enkripsi dapat dihitung dari kunci dekripsi dan sebaliknya, artinya kunci enkripsi sama dengan kunci dekripsi. Keamanan algoritma simetris terletak pada kunci, kebocoran kunci berarti siapa pun bisa mengenkripsi dan mendekripsi pesan. Contoh sistem ini adalah DES, Blowfish, RC5, GOST, dsb.



Gambar 1. Skema *symmetric cryptosystem*

2. Sistem kriptografi kunci-publik (*Public-key Cryptosystem*), sering disebut algoritma asimetris (*asymmetric cryptosystem*), adalah algoritma di mana kunci yang digunakan untuk enkripsi berbeda dengan kunci yang digunakan untuk dekripsi. Selain itu, kunci dekripsi tidak dapat (setidaknya dalam jumlah waktu yang wajar) dihitung dari suatu kunci enkripsi. Algoritma ini disebut "kunci-publik" karena kunci enkripsi dapat dibuat publik yaitu pihak luar dapat menggunakan kunci enkripsi untuk mengenkripsi pesan, tetapi hanya orang tertentu dengan kunci dekripsi yang sesuai dapat mendekripsi pesan. Ada dua masalah matematika yang sering dijadikan dasar pembangkitan sepasang kunci pada algortima kunci-publik, yaitu:
   a. Pemfaktoran
      Diberikan bilangan bulat *n*. faktorkan *n* menjadi faktor primanya. Semakin besar *n*, semakin sulit memfaktorkannya (butuh waktu sangat lama). Algoritma yang menggunakan prinsip ini adalah RSA.
   b. Logaritma diskrit
      Temukan *x* sedemikian sehingga $a^x \equiv b(mod\ n)$ sulit dihitung. Semakin besar *a*, *b* dan *n* semakin sulit memfaktorkannya. Algoritma yang menggunakan prinsip ini adalah ElGamal dan DSA. [10]



Gambar 2. Skema *public-key cryptosystem*

C. Rivest Shamir Adleman (RSA)

RSA adalah salah satu dari sistem kriptografi kunci-publik (*public-key criptosystem)*. Tahun 1978, Len Adleman, Ron RSA menjadi sistem kriptografi RSA kunci-publik yang terpopuler karena merupakan sistem pertama yang sekaligus dapat digunakan untuk *confidentiality, key distribution* dan *digital signature*. Boleh dikatakan semua standar sistem kriptografi memperbolehkan penggunaan RSA, termasuk SSL/TLS (untuk pengamanan http) dan SSH (*secure shell*) [6]. Algoritma RSA memiliki besaran-besaran sebagai berikut:

1. *p* dan *q* bilangan prima (rahasia)
2. $n = p.\ q$ (tidak rahasia)
3. $\phi(n) = (p-1)\ (q-1)$ (rahasia)
4. *e* (kunci enkripsi) (tidak rahasia)
5. *d* (kunci dekripsi) (rahasia)
6. *m* (*plaintext*) (rahasia)
7. *c* (*ciphertext*) (tidak rahasia)

1. Pembangkitan Kunci

Dalam membuat suatu sandi, RSA mempunyai cara kerja dalam membuat kunci publik dan kunci privat adalah sebagai berikut:

a. Pilih dua bilangan prima sembarang, *p* dan *q*.
b. Hitung $n = p.\ q$ (sebaiknya $p \neq q$, sebab jika $p = q$ maka $n = p^2$ sehingga *p* dapat diperoleh dengan menarik akar pangkat dua dari *n*).
c. Hitung $\phi(n) = (p-1)(q-1)$.
d. Pilih kunci publik *e*, yang relatif prima terhadap $\varphi(n)$ yaitu $1 < e < \phi(n)$ dan $gcd(e, \phi(n)) = 1$.
e. Bangkitkan kunci privat dengan menggunakan persamaan
   $d \equiv 1(mod\ \phi(n))$ …..……..(1)
   $(0 \leq d \leq n)$

e-ISSN : 2541-2019
p-ISSN : 2541-044X

Sehingga hasil dari algoritma di atas adalah:

a. Kunci publik adalah pasangan (*e*, *n*)
b. Kunci privat adalah pasangan (*d*, *n*)

Contoh: Misalkan A akan membangkitkan kunci publik dan kunci privat miliknya. A memilih $p = 47$ dan $q = 71$ (keduanya prima). Selanjutnya A menghitung :

$$n = p.\ q = 3337 \text{ dan } \phi(n) = (p-1)\ (q-1) = 3220$$

A memilih kunci publik $e = 79$ karena relatif prima dengan 3320. A mengumumkan nilai *e* dan *n*. Selanjutnya A menghitung kunci dekripsi *d*, sehingga dituliskan berdasarkan persamaan:

$$79.\ d \equiv 1(mod\ 3220)$$

Dengan mencoba nilai-nilai $d = 1, 2, 3, \ldots$, diperoleh nilai *d* memenuhi persamaan (1) yaitu 1019.

$d = 1$	$(79.1)\ mod\ 3220 = 79$
$d = 2$	$(79.2)\ mod\ 3220 = 158$
$d = 3$	$(79.3)\ mod\ 3220 = 237$
⋮	⋮
$d = 1019$	$(79.1019)\ mod\ 3220 = 1$

Kunci privat digunakan untuk mendekripsi pesan dan harus dirahasiakan A. Jadi, perhitungan kunci ini menghasilkan pasangan kunci:

a. Kunci publik ($e = 79$, $n = 3337$)
b. Kunci privat ($d = 1019$, $n = 3337$)

Pada RSA hanya diberikan kunci publik, yaitu modulus *n* dan *e*. Sedangkan kunci privat *d* dirahasiakan. Oleh sebab itu, keamanan algoritma *RSA* terletak pada tingkat kesulitan dalam memfaktorkan bilangan non prima *n* menjadi faktor primanya, dalam hal ini $n = p.\ q$. Sekali *n* berhasil difaktorkan menjadi *p* dan *q*, maka $\phi(n) = (p-1)(q-1)$ dapat dihitung. Selanjutnya, karena kunci enkrispi *e* diumumkan (tidak rahasia), maka kunci dekripsi *d* dapat dihitung dari persamaan (1) kemudian dilakukan dekripsi *ciphertext c* menjadi *plaintext m* menggunakan persamaan (2).

Untuk menjaga keamanan tersebut, ada beberapa hal yang perlu diperhatikan dalam memilih *p* dan *q*:

a. Nilai *p* harus cukup jauh dari nilai *q*. Sebaiknya panjang dari *p* harus berbeda beberapa digit dari *q*.
b. Sebaiknya *gcd*(p – 1, q – 1) tidak terlalu besar.
c. Sebaiknya ($p - 1$) dan ($q - 1$) mempunyai faktor prima yang besar. [6]

2. Proses Enkripsi

Proses enkripsi pesan sebagai berikut:

a. Ambil kunci publik penerima pesan e dan modulus n.
b. Nyatakan *plaintext m* menjadi blok-blok m_1, m_2, ..., sedemikian sehingga setiap blok merepresentasikan nilai di dalam selang [0, $n-1$].
c. Setiap bok m_i dienkripsi menjadi blok c_i dengan rumus

$$c_i = m_i^e \bmod n \quad \ldots\ldots(2)$$

Contoh: Misalkan B mengirim pesan kepada A. Pesan (*plaintext*) yang akan dikirim ke A adalah

$$m = \text{BUDI}$$

B mengubah *m* ke dalam desimal pengkodean ASCII dan sistem akan memecah *m* menjadi blok yang lebih kecil dengan menyeragamkan masing-masing blok menjadi 3 digit dengan menambahkan digit semu (biasanya 0) karena kode ASCII memiliki panjang digit maksimal sebesar 3 digit:

$m_1 = 066$ $m_2 = 085$ $m_3 = 068$ $m_4 = 073$

Nilai-nilai m_i ini masih terletak di dalam selang [0, 3337-1] agar transformasi menjadi satu-ke-satu. B mengetahui kunci publik A adalah $e = 79$ dan $n = 3337$. B dapat mengenkripsi setiap blok *plaintext* sebagai berikut:

$c_1 = 66^{79}\ mod\ 3337 = 795$ $c_2 = 85^{79}\ mod\ 3337 = 3048$
$c_3 = 68^{79}\ mod\ 3337 = 2753$ $c_4 = 73^{79}\ mod\ 3337 = 725$

Dalam penerapannya, untuk memudahkan sistem membagi *ciphertext* menjadi blok-blok yang mewakili tiap karakter maka ditambahkan digit semu (biasanya 0) pada blok cipher sehingga tiap blok memiliki panjang yang sama sesuai ketetapan (dalam hal ini panjangnya 4 digit). Jadi, *ciphertext* yang dihasilkan adalah

$$c = 0795\ 3048\ 2753\ 0725$$

3. Proses Dekripsi

Proses dekripsi pesan sebagai berikut:

a. Ambil kunci privat penerima pesan *d*, dan modulus *n*.
   Nyatakan *plaintext c* menjadi blok-blok c_1, c_2, ..., sedemikian sehingga setiap blok merepresentasikan nilai di dalam selang [0, $n-1$].
b. Setiap blok m_i dienkripsi menjadi blok c_i dengan rumus:

$$m_i = c_i^d \bmod n \quad \text{.....(3)}$$

Contoh: Dengan kunci privat *d = 1019*, chiperteks yang telah dibagi menjadi blok-blok cipher yang sama panjang, *c* = 0795 3048 2753 0725, kembali diubah ke dalam *plaintext*: BUDI

$m_1 = 795^{1019} \bmod 3337 = 66$ $m_2 = 3048^{1019} \bmod 3337 = 85$

$m_3 = 2753^{1019} \bmod 3337 = 68$ $m_4 = 735^{1019} \bmod 3337 = 73$

Sehingga *plaintext* yang dihasilkan *m* = BUDI

D. *Kriptanalisis (Cryptanalysis)*

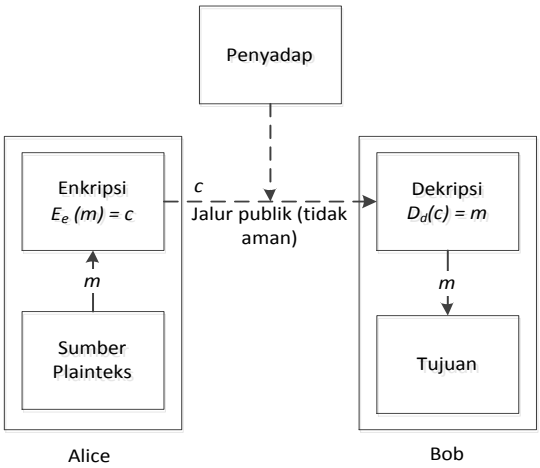
Kriptanalisis adalah bidang ilmu yang memecahkan *ciphertext* tanpa memiliki secara sah kunci yang digunakan, biasanya mendapatkan beberapa atau keseluruhan *plaintext* bahkan kuncinya. (*cryptanalyst*) [15]. Sedangkan pelakunya disebut kriptanalis.

Dalam membahas serangan terhadap kriptografi, kita selalu mengasumsikan kriptanalis mengetahui algoritma kriptografi yang digunakan, sehingga satu-satunya keamanan sistem kriptografi sepenuhnya pada kunci. Hal ini didasarkan pada Prinsip Kerckhoff (1883):

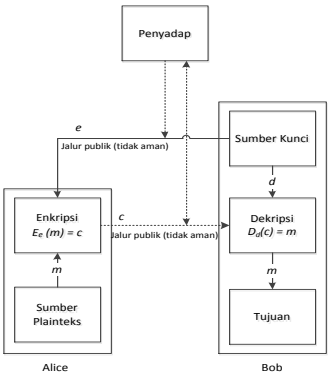
"Algoritma kriptografi tidak harus rahasia bahkan dapat diasumsikan jatuh ke tangan musuh tanpa menimbulkan masalah. Namun, kunci yang digunakan dalam algoritma tersebut harus dianggap rahasia"

Alasan pertama, jauh lebih mudah bagi para pihak untuk menjaga kerahasiaan kunci yang dibangkitkan secara acak daripada menjaga kerahasiaan dari suatu algoritma. Kedua, apabila kunci terbongkar maka lebih mudah pihak yang terkait untuk mengubah kuncinya dibandingkan mengubah algoritma [5].

Gambar 3 menjelaskan skema bahwa seorang penyadap dapat memperoleh *ciphertext* atau kunci karena pihak yang berkomunikasi akan berbagi kunci melalui jalur publik. Hal inilah yang menjadi celah bagi kriptanalis untuk membongkar *ciphertext* menjadi *plaintext*. Gambar 4 menjelaskan skema penyadap pada kriptografi kunci-publik.



Gambar 3. Skema komunikasi kriptografi



Gambar 4. Skema komunikasi kriptografi kunci-publik

E. *Metode Kriptanalisis RSA*

Secara garis besar dapat dikelompokkan menjadi tiga, yaitu Pemfaktoran (*Factoring*), Serangan Fungsi RSA (*Attacks on the RSA Function*) dan *Implementation Attack* [3].

1. Pemfaktoran (*factoring*)
   Penyerangan ini bertujuan untuk memfaktorkan nilai *n* menjadi dua buah faktor primanya yaitu *p* dan *q*. Jika *p* dan *q* berhasil difaktorkan, fungsi *euler* $\phi = (p-1)(q-1)$ akan dapat dihitung

dengan mudah dan kemudian kunci privat $d = e - 1 \, mod \, \phi(n)$ dapat segera dihitung. Beberapa metodenya adalah metode *Fermat's Difference of Squares*, metode *Euler's Factoring*, metode *Kraitchik*, *Quadratic Sieve*, Metode *Pollard's ρ-1 and ρ* dan *Continued Fractions Algorithm*.

2. Serangan fungsi RSA
   Serangan ini mengambil keuntungan dari sifat khusus dari fungsi RSA. Biasanya memanfaatkan kesalahan sistem, misalnya kesalahan dalam pemilihan eksponen privat *d* atau eksponen publik *e*, dan lain-lain. Banyak serangan ini menggunakan teknik matematika canggih. Beberapa metodenya adalah *Low Private Exponent Attack*, *Partial Key Exposure Attack*, *Broadcast and Related Message Attacks* dan *Short Pad Attack*.
3. *Implementation attacks*
   Serangan ini (disebut juga *Side-channel attacks)* ditujukan pada rincian implementasinya. Dalam kasus ini, penyerang biasanya menggunakan beberapa informasi tambahan yang bocor dari implementasi fungsi RSA atau memanfaatkan kesalahan implementasi. Pertahanan terhadap serangan sangat sulit, biasanya mengurangi jumlah informasi yang bocor atau membuatnya tidak berkaitan. Beberapa metodenya adalah *Timing Attack*, *Power Analysis*, *Fault Analysis* dan *Failure Analysis*.

F. *Pemfaktoran Metode Kraitchik*

Metode *Kraitchik* merupakan salah satu metode kriptanalisis dengan cara pemfaktoran (*factoring*) yaitu memfaktorkan nilai n menjadi dua buah faktor primanya. Pada tahun 1945, berdasarkan metode dasar *Fermat*, Maurice Kraitchik menemukan metode untuk memfaktorkan *n* menjadi *x* dan *y* sedemikian rupa sehingga $x^2 \equiv y^2 (mod \, n)$. Dia mengamati bahwa $x^2 - y^2$ merupakan kelipatan *n*. Jadi, untuk menghitung faktor dari *n* Kraitchik menggunakan persamaan

$kn = x^2 - y^2$ ……..……(4)

Untuk *k* adalah bilangan bulat positif (diasumsikan *x* > *y*). Metode *Kraitchik* ini menjadi dasar dari metode *Quadratic* Sieve, salah satu yang paling efisien yang masih digunakan sampai sekarang.

Contoh: Faktorkan 899 menggunakan metode *Kraitchik* dengan *k* adalah sebuah prima ganjil.

1. Hitung x_0 dengan mengakarkan $n = 899$
   $xo = \sqrt{n} = \sqrt{899} = 29{,}98\ldots = 29$
2. Tentukan $k = 3$ dan hitung y^2 dengan persamaan (4) (m = 1, 2, 3,…):
   $x^2 - k.\, n = y^2 \rightarrow (xo + m)^2 - k.\, n = y^2$
   $(29 + 1)^2 - 3 * 899 = -1736$
   Karena hasilnya negatif dan tidak bisa diakarkan, hitung y^2 dengan menaikkan nilai *m* sampai memberikan hasil positif dan akar sempurna:
   $(29 + 23)^2 - 3 * 899 = 7$
   $(29 + 24)^2 - 3 * 899 = 112$
   $(29 + 25)^2 - 3 * 899 = 219$
   $(29 + 26)^2 - 3 * 899 = 328$
   $(29 + 27)^2 - 3 * 899 = 439$
   $(29 + 28)^2 - 3 * 899 = 552$
   $(29 + 29)^2 - 3 * 899 = 667$
   $(29 + 30)^2 - 3 * 899 = 784 = 28^2$

   Sehingga 3 * 899 = (59 – 28)(59 +28) = 331 * 87. Jadi, *n* = 31 * (87/3) = 31 * 29. [1]

G. *Invers Modulo (Extended Euclidean)*

Jika *a* dan *m* relatif prima dan *m* > 1, maka kita dapat menemukan invers dari *a* modulo *m* adalah bilangan bulat *a* sedemikian sehingga

$aa^{-1} \equiv 1 \pmod{m}$

Bukti: Dua buah bilangan bulat *a* dan *m* dikatakan relatif prima jika *Gcd*(*a*, *m*) = 1, dan jika *a* dan *m* relatif prima, maka terdapat bilangan bulat *p* dan *q* sedemikian sehingga

$pa + qm = 1$

yang mengimplikasikan bahwa

$pa + qm \equiv 1 \pmod{m}$

Karena $qm \equiv 0 \pmod{m}$, maka

$pa \equiv 1 \pmod{m}$.

Kekongruenan yang terakhir ini berarti bahwa *p* adalah invers dari *a* modulo *m*. Pembuktian di atas juga menceritakan bahwa untuk mencari invers dari *a*

modulo *m*, kita harus membuat kombinasi lanjar dari *a* dan *m* sama dengan 1. Koefisien *a* dari kombinasi lanjar tersebut merupakan invers dari *a* modulo *m*. [10]

Contoh: Tentukan invers dari 4 mod 9 dan 17 mod 7.

1. Karena *Gcd*(4, 9) = 1, maka invers dari 4 (mod 9) ada. Dari algoritma Euclidean diperoleh bahwa

   $$9 = 2 \cdot 4 + 1$$

   Susun persamaan di atas menjadi

   $$-2 \cdot 4 + 1 \cdot 9 = 1$$

   Dari persamaan terakhir ini kita peroleh –2 adalah invers dari 4 modulo 9. Periksalah bahwa

   $$-2 \cdot 4 \equiv 1 \pmod 9$$

   (10 habis membagi $-2 \cdot 4 - 1 = -9$)
2. Karena *Gcd*(17, 7) = 1, maka invers dari 17 (mod 7) ada. Dari algoritma Euclidean diperoleh rangkaian pembagian berikut:

   $$17 = 2 \cdot 7 + 3 \text{ (i)}$$
   $$7 = 2 \cdot 3 + 1 \text{ (ii)}$$
   $$3 = 3 \cdot 1 + 0 \text{ (iii)}$$

   (yang berarti: *Gcd*(17, 7) = 1))

   Susun (ii) menjadi:

   $$1 = 7 - 2 \cdot 3 \text{ (iv)}$$

   Susun (i) menjadi

   $$3 = 17 - 2 \cdot 7 \text{ (v)}$$

   Sulihkan (v) ke dalam (iv):

   $$1 = 7 - 2 \cdot (17 - 2 \cdot 7) = 1 \cdot 7 - 2 \cdot 17 + 4 \cdot 7 = 5 \cdot 7 - 2 \cdot 17$$

   atau

   $$-2 \cdot 17 + 5 \cdot 7 = 1$$

   Dari persamaan terakhir ini kita peroleh –2 adalah invers dari 17 modulo 7.

   $$-2 \cdot 17 \equiv 1 \pmod 7$$

   (7 habis membagi $-2 \cdot 17 - 1 = -35$) [10]

H. *Uji Bilangan Prima Algoritma Lehmann*

Sebagaimana diketahui bahwa bilangan prima tidak mengikuti pola yang jelas dan untuk mengetahui bahwa suatu bilangan prima atau tidak bukanlah tugas yang mudah. Bilangan *n* disebut prima, jika tidak boleh ada pembagi *n* berada antara 2 dan $\sqrt{n}$. Jika nilai bilangan *n* kecil maka pengujian sangat mudah dihitung, tetapi kesulitannya meningkat jika nilai *n* semakin besar. Salah satu metode yang cukup cepat untuk menguji keprimaan suatu bilangan *n* adalah algoritma *Lehmann*.

Algoritma *Lehmann*:

1. Bangkitkan bilangan acak *a*, $1 \le a \le p$ (*p* adalah bilangan yang diuji keprimaannya)
2. Hitung $a^{(p-1)/2} \bmod p$
3. Jika $a^{(p-1)/2} \not\equiv 1$ atau -1 (*mod* p), maka *p* tidak prima
4. Jika $a^{(p-1)/2} \equiv 1$ atau -1 (*mod p*), maka peluang *p* bukan prima adalah 50%.
5. Ulangi pengujian di atas sebanyak *t* kali (dengan nilai *a* yang berbeda). Jika hasil perhitungan langkah 2 sama dengan 1 atau – 1, tetapi tidak selalu sama dengan 1, maka peluang p adalah bilangan prima mempunyai kesalahan tidak lebih dari $1/2^t$.

Contoh 1: Ujilah keprimaan bilangan p = 7 dengan algoritma *Lehmann*

Hitung $a^{(7-1)/2} \bmod 7$ dengan mencoba nilai-nilai *a* ($1 \le a \le 7$)

$a = 1$	$1^{(7-1)/2} \bmod 7 \equiv 1 (\bmod 7)$
$a = 2$	$2^{(7-1)/2} \bmod 7 \equiv 1 (\bmod 7)$
$a = 3$	$3^{(7-1)/2} \bmod 7 \not\equiv 1 (\bmod 7)$
$a = 4$	$4^{(7-1)/2} \bmod 7 \equiv 1 (\bmod 7)$
$a = 5$	$5^{(7-1)/2} \bmod 7 \not\equiv 1 (\bmod 7)$
$a = 6$	$6^{(7-1)/2} \bmod 7 \not\equiv 1 (\bmod 7)$
$a = 7$	$7^{(7-1)/2} \bmod 7 \not\equiv 1 (\bmod 7)$

Jadi, 7 adalah **Bilangan Prima** karena memenuhi syarat $a^{(p-1)/2} \equiv 1$ atau -1 (*mod p*) sebanyak tiga kali yaitu sama dengan ½ dari banyaknya penghitungan ($1 \le a \le 7$) sebanyak 7/2 = 3,5 = 3 dengan tingkat kesalahan sebesar $1/2^7$ = 0,78125% atau tingkat kebenarannya mencapai 99, 21875%.

Contoh 2: Ujilah keprimaan bilangan p = 9 dengan algoritma *Lehmann*

Hitung $a^{(9-1)/2} \bmod 9$ dengan mencoba nilai-nilai *a* ($1 \le a \le 9$)

$a = 1$	$1^{(9-1)/2} \bmod 9 \equiv 1 (\bmod 9)$
$a = 2$	$2^{(9-1)/2} \bmod 9 \not\equiv 1 (\bmod 9)$
$a = 3$	$3^{(9-1)/2} \bmod 9 \not\equiv 1 (\bmod 9)$

$a = 4$ $4^{(9-1)/2} \ mod\ 9 \not\equiv 1 (mod\ 9)$
$a = 5$ $5^{(9-1)/2} \ mod\ 9 \not\equiv 1 (mod\ 9)$
$a = 6$ $6^{(9-1)/2} \ mod\ 9 \not\equiv 1 (mod\ 9)$
$a = 7$ $7^{(9-1)/2} \ mod\ 9 \not\equiv 1 (mod\ 9)$
$a = 8$ $8^{(9-1)/2} \ mod\ 9 \equiv 1 (mod\ 9)$
$a = 9$ $9^{(9-1)/2} \ mod\ 9 \not\equiv 1 (mod\ 9)$

Jadi, 9 adalah **Bukan Bilangan Prima** karena banyaknya penghitungan $a(1 \leq a \leq 9)$ yang memenuhi syarat $a^{(p-1)/2} \equiv 1$ atau $-1\ (mod\ p)$ hanya sebanyak dua kali yaitu *tidak* mencapai ½ dari banyaknya penghitungan $a\ (1 \leq a \leq 7)$ sebanyak 9/2 = 4,5 = 4. [7] [10]

Berikut ini adalah perbandingan uji keprimaan metode Lehmann dengan beberapa metode keprimaan lainnya:

1. **Metode *Trial-Division***: mudah diterapkan dan akurat (tidak ada bilangan yang akan terlewati hanya bergantung pada definisi keprimaannya) tetapi akan melambat karena mencoba setiap angka hingga $\sqrt{n}$ dan memeriksanya satu per satu.
2. **Tes keprimaan *Fermat***: sangat mudah dan cepat, tetapi tidak terlalu akurat (Adanya bilangan *Carmichael* yaitu angka komposit akan dilaporkan sebagai bilangan prima).
3. **Tes *Solovay-Strassen* dan tes *Miller-Rabin***: cepat dan akurat, tapi tidak mudah untuk menerapkan karena mengharuskan memahami Simbol Legendre dan Jacobi (konsep-konsep matematika rahasia) dan membutuhkan algoritma faktorisasi yang efisien. Kemungkinan *error* tes Solovay-Strassen adalah $(1/2)^t$, sedangkan kemungkinan *error* tes Miller-Rabin adalah terbatas di atas oleh $(1/4)^t$.
   [8]

III. ANALISIS DAN PERANCANGAN

A. *Analisis Masalah*

Cause and Effect Diagram menunjukkan akar permasalahan yang paling mempengaruhi efisiensi waktu pemfaktoran kunci publik dalampemecahan kunci privat RSA adalah selisih faktor kunci $(p - q)$, $Gcd(p - 1, q - 1)$ dan faktor prima $(p - 1)$, $(q - 1)$.

B. *Analisis Kebutuhan Sistem*

Ada dua tipe kebutuhan yang akan dianalisis, yaitu:

1. Kebutuhan Fungsional

   Kebutuhan fungsional adalah jenis kebutuhan yang berisi proses-proses yang harus dilakukan oleh sistem dan informasi-informasi yang harus ada di dalam sistem.

a. Sistem akan melakukan dua proses, yaitu pemecahan kunci privat RSA dan dekripsi *ciphertext* menjadi *plaintext*
b. Pada proses pemecahan kunci privat RSA, pengguna meng-*input* kunci publik *n* bertipe data *BigInteger* -2^{63}(-9,223,372,036,854,775,808) sampai dengan 2^{64}(18,446,744,073,709,551,616) untuk difaktorkan menjadi faktor kunci (*p*,*q*) menggunakan metode *Kraitchik*. *Output* berupa kunci privat *d* menggunakan faktor kunci (*p*,*q*) dengan invers modulo (*Extended Euclidean*)
c. Pada proses dekripsi, pengguna meng-*input* pasangan kunci privat (*d*, *n*) dan *ciphertext* dengan *ouput* berupa *plaintext*
d. *Ciphertext* dan *plaintext* adalah *file* berekstensi *.doc dan *.txt dengan *ciphertext* berupa numerik (angka) dan *plaintext*-nya diubah menjadi karakter ASCII
e. Sistem dilengkapi fungsi *timer* untuk menghitung waktu proses kerja sistem
f. Sistem dapat menyimpan *file* hasil dekripsi (*plaintext*) pada direktori penyimpanan

2. Analisis Kebutuhan Nonfungsional

   Kebutuhan nonfungsional adalah jenis kebutuhan yang berisi hal-hal atau fitur-fitur lain untuk menunjang fungsionalitas dan utilitas sistem.

a. *Performance* (Kinerja)
   Menggunakan algoritma yang efektif dan efisien membatasi panjang kunci maksimal yang masih dalam jangkauan kinerja sistem, tidak terlalu

e-ISSN : 2541-2019
p-ISSN : 2541-044X

banyak informasi dan kontrol atau pengendalian tidak berlebihan

b. *Information* (Informasi)
Menampilkan informasi mengenai proses atau situasi yang sedang berjalan, proses yang telah selesai dan langkah-langkah selanjutnya dan menampilkan informasi prosedur atau langkah langkah sistem

c. *Control* (Kontrol)
Menangani masalah di luar kondisi normal, misalnya jika *input* tidak sesuai dengan kriteria sistem, salah melakukan prosedur dan lain-lain.

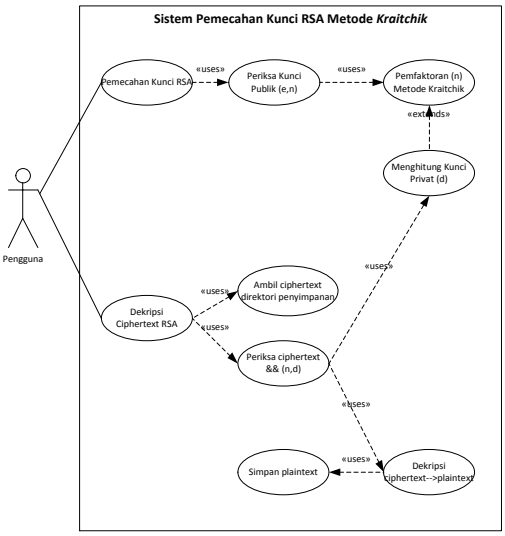
d. *Service* (Pelayanan)
Akurasi dalam pengolahan data, kehandalan terhadap konsistensi dalam pengolahan *input* dan *output*-nya serta kehandalan dalam menangani pengecualian dan sistem *user friendly* dalam penggunaan dan *interface.*

C. Perancangan Sistem

Sistem akan dikembangkan menggunakan teknologi *Ms. Visual C#* yang menerapkan paradigma pengembangan sistem berorientasi objek, oleh karena itu dalam proses perancangan ini akan digunakan *Unified Modelling Language* (UML). Diagram UML yang digunakan adalah *use case diagram* dan *activity diagram.*

1. *Use case diagram*

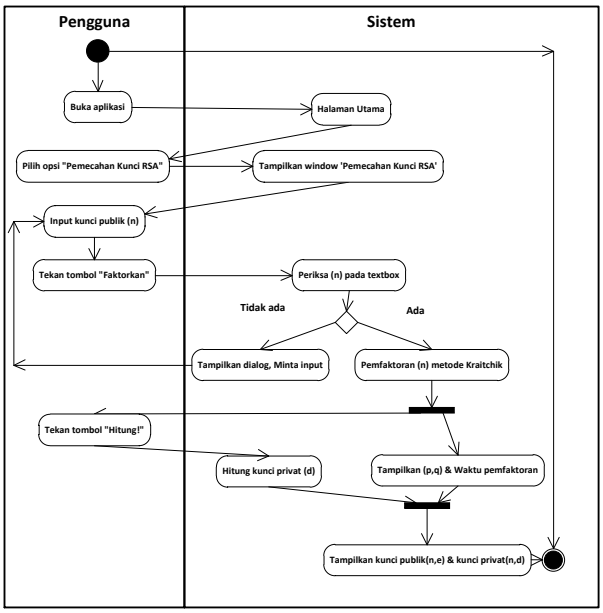
Gambar 5 menunjukkan interaksi antara pengguna dan Sistem Pemecahan Kunci RSA. Ada dua proses yang dilakukan sistem yaitu pemecahan kunci RSA dan dekripsi RSA.



Gambar 5. *Use case diagram* Sistem Pemecahan Kunci RSA Metode *Kraitchik*

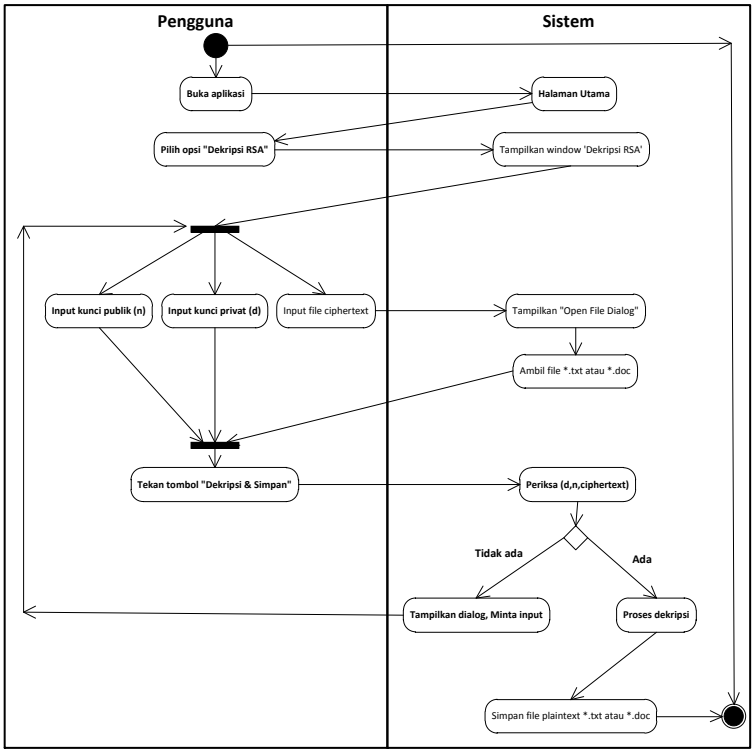
2. *Activity diagram*

Activity diagram untuk *use case* Pemecahan Kunci RSA dan *use case* Dekripsi RSA ditunjukkan pada Gambar 6 dan Gambar 7.



Gambar 6. *Activity diagram* untuk *use case* Pemecahan Kunci RSA

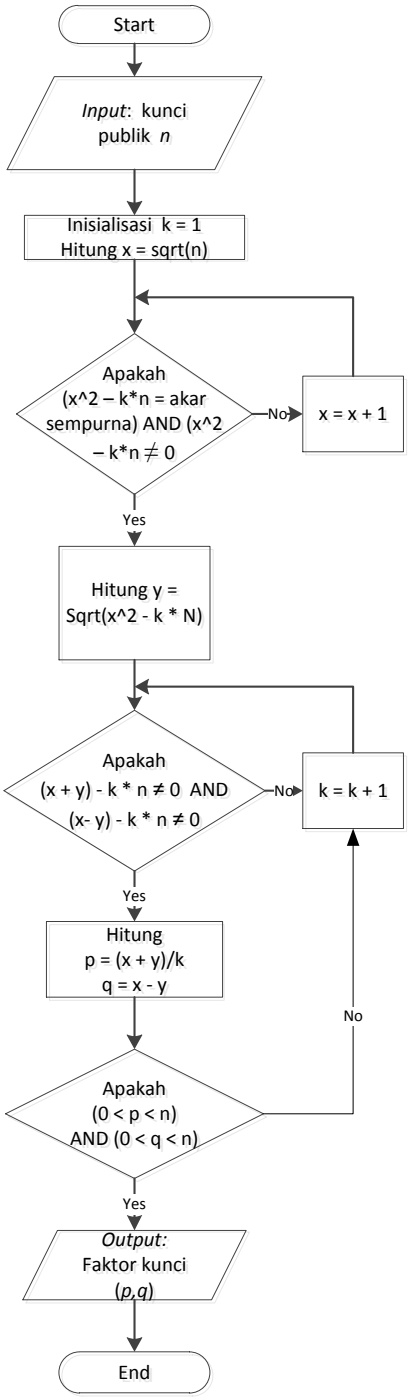
Jurnal & Penelitian Teknik Informatika
Volume 2 Nomor 2, Oktober 2017
e-ISSN : 2541-2019
p-ISSN : 2541-044X



Gambar 7. *Activity diagram* untuk *use case* Dekripsi RSA

D. Flowchart

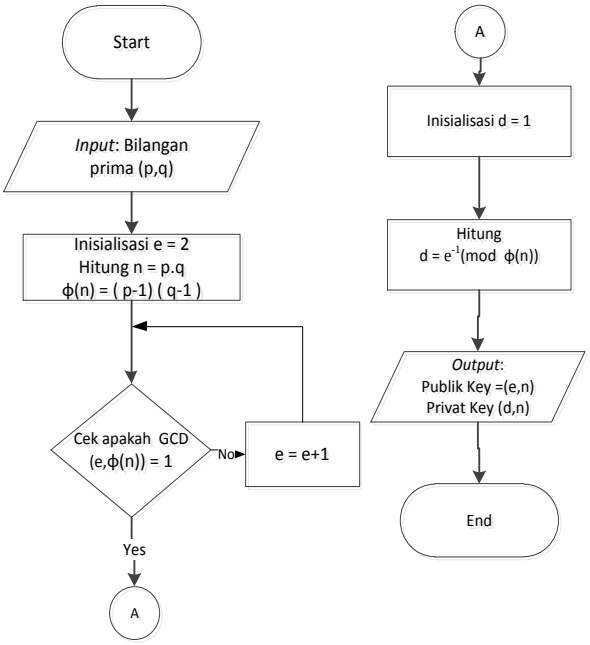
1. Flowchart Pemecahan Kunci RSA Metode *Kraitchik*



Gambar 8. *Flowchart* pemfaktoran metode *Kraitchik*

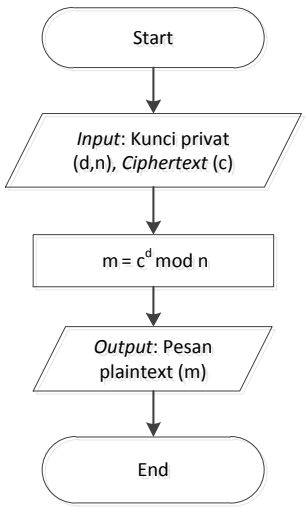
2. Flowchart Menghitung Kunci Publik dan Kunci Privat

e-ISSN : 2541-2019
p-ISSN : 2541-044X



Gambar 9. *Flowchart* menghitung kunci publik dan kunci privat

3. *Flowchart* Dekripsi *RSA*



Gambar 10. *Flowchart* proses dekripsi RSA

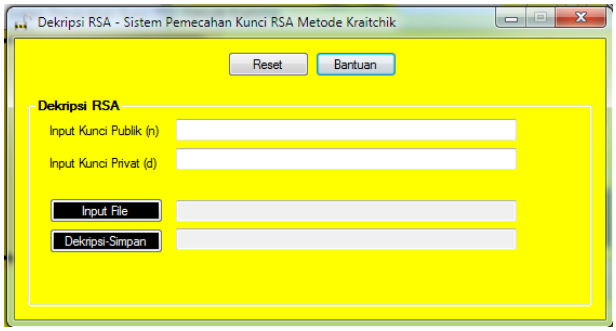
IV. IMPLEMENTASI DAN PENGUJIAN SISTEM

A. Implementasi Sistem

Aplikasi Sistem Pemecahan Kunci RSA Metode *Kraitchik* terdiri dari lima *form interface* yaitu *form* ’Halaman Utama’, *form* ‘Pemecahan Kunci RSA’, *form* ‘Dekripsi RSA’, *form* ‘Bantuan’ dan *form* ‘Tentang’.

Gambar 11. *Form* ’Pemecahan Kunci RSA’

Gambar 11 menunjukkan *form* ’Pemecahan Kunci RSA’ yang memiliki dua fungsi yaitu pemfaktoran kunci publik *n* dan penghitungan kunci publik (*e*, *n*) dan kunci privat (*d*, *n*).



Gambar 12. *Form* ’Dekripsi RSA’

Gambar 12 menunjukkan form ’Dekripsi RSA’ yang berfungsi untuk mendekripsi *ciphertext* menjadi *plaintext*.

B. *Hasil Pengujian*

Pengujian dilakukan dengan memecahkan kunci RSA dan mendekripsi *ciphertext* dari Aplikasi RSA [17] yang akan membangkitkan kunci RSA dan mengenkripsi sebuah file teks.

1. Pengujian proses pemecahan kunci RSA dan dekripsi RSA

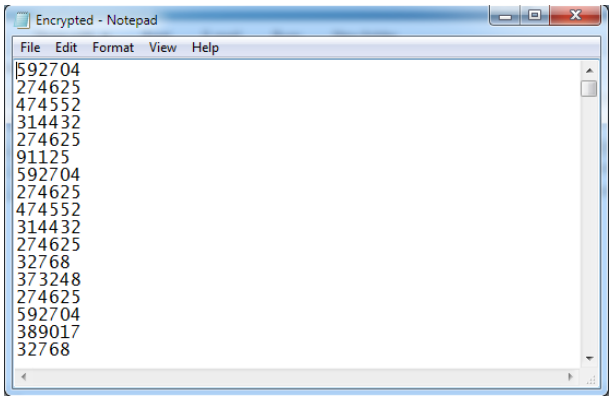
Pengujian terhadap proses pemecahan kunci RSA dilakukan dengan meng-*input* kunci publik *n* yang telah dibangkitkan pada *form* 'Pemecahan Kunci RSA'.

Input : n = 31257942904494281

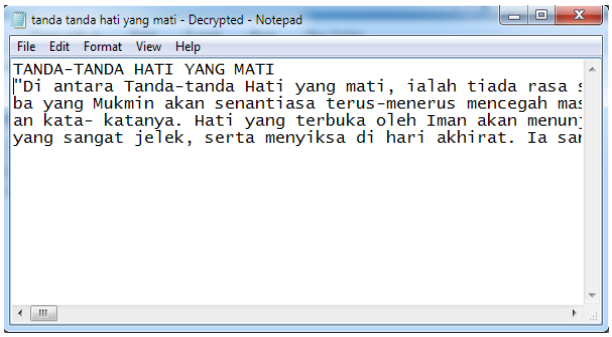
Output 1 : p = 186529753, q = 167576177, Waktu proses = 514,801 ms

Output 2 : Kunci Publik (e = 5, n = 31257942904494281)
Kunci Privat (d = 12503177020155341, n = 31257942904494281)
Panjang Kunci (n) = 17
Totien = 31257942550388352
Selisih Faktor Kunci ($p - q$) = 18953576
Gcd ($p - 1$, $q - 1$) = 8

Pengujian dilanjutkan pada proses dekripsi RSA dengan meng-input kunci publik *n* dan kunci privat *d*, serta *ciphertext* pada form 'Dekripsi RSA'. Ciphertext dan hasil dekripsi berupa *plaintext* ditunjukkan pada Gambar 13 (a) dan (b).



(a)



(b)

Gambar 13. (a) *ciphertext* atau file terenkripsi (Encrypted.txt) (b) *plaintext* atau file hasil dekripsi (tanda-tanda hati yang mati.txt)

Selanjutnya, penulis akan berfokus pada pengujian lama waktu sistem dalam memfaktorkan kunci publik *n* menjadi faktor kunci *p* dan *q* dengan metode *Kraitchik* dan faktor-faktor yang mempengaruhi efisiensi waktunya.

2. Pengujian panjang dan nilai kunci publik *n*

Pengujian dilakukan berdasarkan panjang kunci publik *n* dengan menaikkan panjang kunci publik *n* dan menaikkan nilai kunci publik *n* pada panjang kunci yang sama.

Tabel I Kunci publik *n* terhadap waktu pemfaktoran

No.	Kunci Publik *n*	Digit *n*	*p*	*q*	Waktu (*ms*)
1.	404710582450991	15	34969463	11573257	5850.0103
2.	805532433421423	15	33395099	24121277	717.6013
3.	2506398389987369	16	54589903	45913223	374.4007
4.	6522295602531389	16	83644243	77976623	109.2002
5.	31257942904494281	17	186529753	167576177	453.6008
6.	33304223962748509	17	278537687	119568107	62556.109
7.	327575949756535557	18	262476979	124801783	20623.2362
8.	549856378476824039	18	812162363	677027653	4992.0088
9.	2716693773287446741	19	2050749269	1324732289	315089.3535
10.	7877630259565883429	19	2818055467	2795413487	93.6002

Dari TABEL I menunjukkan beberapa informasi sebagai berikut:

a. Semakin besar panjang dan nilai kunci publik *n* tidak selalu menghasilkan waktu pemfaktoran yang semakin lama.
b. Panjang kunci yang lebih kecil menghasilkan waktu pemfaktoran lebih lama daripada kunci yang lebih

besar atau sebaliknya. Contohnya pada $n_1 < n_3 < n_{10}$ (15<16< 19 digit) dengan $t_1 > t_3 > t_{10}$ (5850.0103 > 374.4007 > 93.6002).

c. Menaikkan nilai kunci publik *n* pada panjang kunci yang sama dapat menghasilkan waktu pemfaktoran yang berbeda, dapat lebih lama atau lebih cepat. Contohnya pada $n_5 < n_6$ (panjang kunci = 17) menghasilkan waktu $t_5 < t_6$ (453,6008<62556,109). Pada $n_7 < n_8$ (panjang kunci = 18) menghasilkan waktu $t_7 > t_8$ (20623,2362>4992,0088).

3. Selisih kunci ($p - q$)

Pengujian dilakukan dengan menaikkan nilai kunci publik *n* (*n* = 1194221746064803, 1655784864358721, 2309511238585409, …) dengan $Gcd(p - 1, q - 1) = 2$ dan menurunkan besar selisih faktor kunci ($p - q$) tanpa mengubah panjang kunci publik *n* (panjang *n* = 16 digit).

Tabel 2 Kunci publik *n* dan selisih kunci ($p - q$) terhadap waktu pemfaktoran

No.	Kunci Publik *n*	*p*	*q*	$p - q$	Waktu(*ms*)
1.	1194221746064803	95108087	12556469	82551618	32916.0579
2.	1655784864358721	95596889	17320489	78276400	31012.8544
3.	2309511238585409	89005541	25947949	63057592	19110.0336
4.	2834892913358501	87649321	32343581	55305740	12838.8225
5.	3049851886570831	84434953	36120727	48314226	10030.8176
6.	4057997615290243	87844301	4615343	41648958	6645.6117
7.	4994320816150247	94017199	53121353	40895846	5740.8101
8.	6434939629688561	87875639	73227799	14647840	624.0011
9.	7643757470094443	94212067	81133529	13078538	483.6009
10.	8676761544315721	98459177	88125473	10333704	296.4005

Dari TABEL II menunjukkan beberapa informasi sebagai berikut:

a. Waktu tercepat (t_{10} = 296,4005 *ms*) dihasilkan oleh kunci publik n_{10} dengan dengan selisih kunci ($p - q$) yang terkecil.
b. Waktu terlama (t_1 = 32916,0579 *ms*) dihasilkan oleh kunci publik n_1 dengan dengan selisih kunci ($p - q$) yang terbesar.
c. Semakin besar nilai kunci publik *n* dan semakin kecil nilai selisih faktor kunci ($p - q$) menghasilkan waktu pemfaktoran yang semakin cepat.

4. $Gcd(p - 1, q - 1)$

Pengujian dilakukan dengan menaikkan nilai kunci publik *n* dengan besar $Gcd(p - 1, q - 1) = 2$ dan selisih faktor kunci ($p - q$) dipilih secara acak tanpa mengubah panjang kunci publik *n* (panjang *n* = 16 digit).

TABEL 3 Kunci Publik *n* dan $Gcd(p - 1, q - 1)$ Terhadap Waktu Pemfaktoran

No.	Kunci Publik *n*	*p*	*q*	$p - q$	*Gcd* (p – 1, q – 1)	Waktu(*ms*)
1.	1497663331801279	59095313	25343183	33752130	2	6786.0119
2.	2707617775411949	72671609	37258261	35413348	4	5506.8097
3.	2785517637270793	90710849	30707657	60003192	8	15584.4274
4.	3009948426489709	75717209	39752501	35964708	4	5694.01
5.	4782545980579907	78652349	60806143	17846206	2	530.401
6.	4801478951361541	90819643	52868287	37951356	6	5038.8088
7.	4961275670761321	9251853	53624657	38893896	8	4882.8086
8.	5494230208915297	82463683	66626059	15837624	6	811.2014
9.	5944323272022461	99675271	59636891	40038380	10	4773.6084
10.	972176346581351	40165471	24204281	15961190	10	1918.8034

Dari TABEL V menunjukkan beberapa informasi sebagai berikut:

a. Waktu tercepat (t_5 = 530,401 *ms*) dihasilkan oleh kunci publik n_5 dengan $Gcd(p - 1, q - 1) = 2$.
b. Waktu terlama (t_1 = 6786,0119 *ms*) dihasilkan oleh kunci publik n_1 dengan $Gcd(p - 1, q - 1) = 2$.
c. Pada kunci publik *n* tertentu yaitu pada n_1, n_4, n_6, n_7, n_9 menunjukkan bahwa semakin besar $Gcd(p - 1, q - 1)$ maka semakin lama waktu pemfaktorannya.

5. Faktor prima (p – 1) dan (q – 1)

Pengujian dilakukan dengan menaikkan nilai kunci publik *n* dengan besar $Gcd(p - 1, q - 1) = 2$ dan faktor prima ($p - 1$), ($q - 1$) dipilih secara acak tanpa mengubah panjang kunci publik *n* (panjang *n* = 16 digit).

TABEL 4 Kunci Publik n dan Faktor Prima $(p-1)$, $(q-1)$ Terhadap Waktu Pemfaktoran

No.	Kunci Publik n	p	q	$p-q$	Faktor Prima		Waktu(m
					$p-1$	$q-1$	
1.	1282096117599487	67046333	19122539	47923794	16761583	443	13119.62
2.	2195307769095113	67327943	32606191	34721752	3060361	362291	5600.409
3.	3107812708193377	66797363	46525979	20271384	76079	750419	1809.603
4.	4471760003932499	82714717	54062447	28652270	71	27031223	2823.604
5.	5484597314790731	92900609	59037259	33863350	3593	200807	3276.005
6.	6183130661088377	84604783	73082519	11522264	7577	12527	421.2008
7.	7782532222483679	92957279	83721601	9235678	751	19	265.2005
8.	8420052833146261	93283259	90263279	3019980	46641629	149939	31.2001
9.	919937261599969	89853251	10238219	79615032	3359	176521	54974.49
10.	9201780071015039	97747361	94138399	3608962	610921	5229911	46.8001

Dari TABEL IV menunjukkan beberapa informasi terhadap waktu proses pemfaktoran sebagai berikut:

a. Waktu tercepat (t_8 = 31,2001 *ms*, t_{10} = 46.8001 *ms*) dihasilkan oleh kunci publik n_8 dan n_{10} dengan faktor prima yang cukup besar.
b. Waktu terlama (t_1 = 13119,6231 *ms*, t_9 = 54974,4965 *ms*) dihasilkan oleh kunci publik n_1 dan n_9 dengan faktor prima yang cukup kecil.

V. KESIMPULAN DAN SARAN

A. *Kesimpulan*

Tahap pengujian yang telah dilakukan menghasilkan kesimpulan bahwa kunci privat algoritma RSA dapat dipecahkan dengan memfaktorkan kunci publik n menggunakan metode *Kraitchik* secara efisien ditinjau dari parameter waktu dengan beberapa penjelasan sebagai berikut:

1. Efisiensi waktu pemecahan kunci RSA dipengaruhi oleh lamanya waktu pemfaktoran kunci publik n dengan metode *Kraitchik*.
2. Panjang dan nilai kunci publik n maupun faktor prima $(p-1)$, $(q-1)$ tidak berpengaruh besar terhadap efisiensi waktu pemfaktoran metode *Kraitchik* yaitu panjang dan nilai kunci n maupun faktor prima $(p-1)$, $(q-1)$ tidak berbanding lurus dengan lama waktu pemfaktoran, dapat lebih lama atau lebih cepat.
3. Faktor yang mempengaruhi *best case* dan *worst case* efisiensi waktu pemfaktoran kunci publik n metode *Kraitchik* sesuai tingkatannya adalah:
   a. Selisih faktor kunci $(p-q)$
      Selisih faktor kunci $(p-q)$ adalah faktor yang paling berpengaruh terhadap efisiensi waktu pemfaktoran. *Best case*: Semakin kecil selisih faktor kunci $(p-q)$, semakin cepat waktu pemfaktoran.
      Worst case: Semakin besar selisih faktor kunci $(p-q)$, semakin lama waktu pemfaktoran
   b. *Gcd*$(p-1, q-1)$
      Gcd$(p-1, q-1)$ lebih berpengaruh terhadap efisiensi waktu pemfaktoran daripada dengan panjang dan nilai kunci publik n dan faktor prima $(p-1)$, $(q-1)$ walaupun tidak lebih berpengaruh jika dibandingkan dengan selisih faktor kunci $(p-q)$.
      Best case: Semakin besar *Gcd*$(p-1, q-1)$, semakin cepat waktu pemfaktoran
      Worst case: Semakin kecil *Gcd*$(p-1, q-1)$, semakin lama waktu pemfaktoran.

B. Saran

Adapun saran terkait perbaikan dan pengembangan penelitian ke depannya adalah:

1. Banyaknya metode pemfaktoran kunci publik n RSA yang telah dikembangkan, sehingga diperlukan perbandingan metode tersebut untuk mengetahui kekuatan dan kelemahan baik efektivitas dan efisiensi dari setiap metode.
2. Diperlukan penentuan spesifikasi perangkat keras yang tepat untuk melakukan pengujian sehingga menghasilkan data yang objektif untuk mendukung penelitian selanjutnya.
3. Diharapkan ke depannya dapat ditemukan cara mengatasi masalah efisiensi waktu pemecahan kunci yang dipengaruhi oleh selisih faktor kunci $(p-q)$.

REFERENSI

[1] Batten, L.M. 2013. *Public Key Cryptography: Applications and Attacks*. John Wiley & Sons, Inc: New Jersey
[2] Bhakti, Z.F. 2013. Kriptanalisis RSA Dengan Kurva Eliptik (*Cryptanalysis of RSA with Elliptic Curves*)
[3] Cid, C.F. 2003. *Cryptanalysis of RSA. A Survey*. SANS Institute. Maryland
[4] Imani, P. 2002. Analisis Keamanan Kriptografi Kunci Publik RSA. Skripsi. Institut Pertanian Bogor
[5] Katz, J. & Lindell, Y. 2007. Introduction to Modern Cryptography. CRC Press: Boca Raton

[6] Kromodimoeljo, Sentot. 2009. Teori dan Aplikasi Kriptografi. SPK IT Consulting
[7] Lehmann, D.J. 1982. On primality tests. SIAM J. COMPUT. 11(2): 374-375
[8] Menezes, A.J., Oorschot, van P.C., Vanstone, S.A. 2001. Cryptography and Network Security. Boca Raton: CRC Press
[9] Mollin, R.A. 2007. An Introduction to Cryptography, 2nd Ed. Taylor & Francis Group: LLC Boca Raton
[10] Munir, R. 2006. Kriptografi. Informatika Bandung: Bandung
[11] Rhee, M.Y. 2003. Internet Security: Cryptographic Principles, Algorithms and Protocols. John Wiley & Sons Ltd: Chichester
[12] Schneier, Bruce. 1996. Applied Cryptography, Second edition: Protocol, Algorithm and Source Code in C. Wiley Computer Publishing, John Wiley & Sons, Inc.: New York
[13] Smart, N. 2004. Cryptography: An Introduction, Third Edition. McGraw-Hill Education: New York
[14] Stallings, W. 2006. Cryptography and Network Security Principles and Practices, Fourth Edition. Pearson Education, Inc.: New Jersey
[15] Tilborg, H.C.A. van & Jajodi, S. 2011. Encyclopedia of Cryptography and Security. Springer Science+Business Media, LLC: New York
[16] Zarlis, M. & Handrizal. 2010. Pemrograman Komputer: Satu Pendekatan kepada Pendekatan Pemrograman Berorientasikan Objek Dalam C++. Edisi kedua. USU Press: Medan
[17] Z., Mahadi. 2013. Teknik Hacking Algoritma Rivest Shamir Adleman (RSA) 512-Bit Dengan Menggunakan Metode *Difference of Squares*. Skripsi. Universitas Sumatera Utara