Terakreditasi SINTA Peringkat 2
Surat Keputusan Dirjen Penguatan Riset dan Pengembangan Ristek Dikti No. 10/E/KPT/2019

Terbit online pada laman web jurnal: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 5 No. 4 (2021) 768 - 773 ISSN Media Elektronik: 2580-0760

Analisis dan Implementasi Algoritma Asimetris Dual Modulus RSA (DM-RSA) pada Aplikasi Chat

Aminudin[1], Ilyas Nuryasin[2]
[1,2]Program Studi Informatika, Teknik, Universitas Muhammadiyah Malang
[1]aminudin2008@umm.ac.id, [2]ilyas@umm.ac.id

Abstract

The RSA algorithm is one of the cryptographic algorithms with an asymmetric model where the algorithm has two keys, namely the public key and the private key. However, as time goes on, these algorithms are increasingly exposed to security holes and make this algorithm vulnerable to being hacked by people who do not have authority. The vulnerability stems from the algorithm's public keys (e and n). The strength of the RSA algorithm is based on the difficulty of factoring two prime numbers that are generated during the key generation process, if these values can be known using certain methods, the public key and private key values will be found. Therefore, there are many studies that improvise the RSA algorithm, one of which is the Dual Modulus RSA (DM-RSA) algorithm. The algorithm uses four prime numbers which produce 2 modulus and 4 keys (2 public keys and 2 private keys). From the results of the Kraitchik factorization test, it was found that the DM-RSA algorithm was proven to be more resistant up to 2 times or even more than the standard RSA algorithm. This is evidenced by the fact that the value of n is 24 bits, the RSA algorithm can last up to 63204 ms (1 minute 22 seconds) while the Dual Modulus RSA algorithm lasts up to 248494123 ms (142 minutes 47 seconds).

Keywords: RSA algorithm, Dual Modulus RSA, Kraitchik Factorization

Abstrak

Algoritme RSA merupakan salah satu algoritme kriptografi dengan model asimetris di mana algoritme tersebut mempunyai dua buah kunci yaitu kunci publik dan kunci privat. Tetapi, dengan seiring bertambahnya waktu, algoritme tersebut semakin terlihat celah keamananya dan membuat algoritme ini rentan dibobol oleh orang yang tidak mempunyai otoritas. Kerentanan tersebut berasal dari kunci publik algoritme (e dan n). Kekuatan algoritme RSA didasarkan atas sulitnya memfaktorisasi dua buah bilangan prima yang dibangkitkan ketika proses pembangkitan kunci, apabila nilai tersebut dapat diketahui dengan menggunakan metode tertentu maka nilai kunci publik dan kunci privat akan ditemukan. Maka dari itu banyak sekali kajian yang membuat improvisasi algoritme RSA salah satunya adalah algoritme Dual Modulus RSA (DM-RSA). Algoritme tersebut menggunakan empat buah bilangan prima yang menghasilkan 2 buah modulus dan 4 buah kunci (2 kunci publik dan 2 kunci privat). Dari hasil pengujian faktorisasi kraitchik didapatkan bahwa algoritme DM-RSA ini terbukti lebih tahan hingga 2 kali lipat bahkan lebih dibandingkan algoritme RSA standar. Hal ini dibuktikan dengan didapatkanya nilai n yang berjumlah 24 bit algoritme RSA dapat bertahan hingga 63204 ms (1 menit 22 detik) sedangkan algoritme Dual Modulus RSA bertahan hingga 248494123 ms (142 menit 47 detik).

Kata kunci: Algoritme RSA, Dual Modulus RSA, Faktorisasi Kraitchik

1. Pendahuluan

Algoritme RSA merupakan salah satu algoritme kriptografi asimetris yang memiliki sepasang kunci di mana prosesnya didasarkan atas model pemfaktoran dengan membangkitkan bilangan prima. Jadi, untuk memecahkan kunci di dalam algoritma RSA harus mengetahui faktor dari perkalian di dalam bilangan prima [1] [2]. Keamanan pada algoritme RSA terletak pada bilangan prima yang dibangkitkan pada proses pembangkitan kunci. Bilangan prima yang dibangkitkan diharapkan memiliki nilai yang besar sehingga algoritme tidak mudah ditembus [3]. Sebaliknya apabila bilangan prima yang digunakan kecil maka semakin kecil pula keamanan dan mengakibatkan algoritme ini condong untuk mudah ditembus [4] [5]. Maka dari itu, beberapa kajian keilmuan yang membahas tentang modifikasi algoritme RSA dengan tujuan agar hasil yang dihasilkan dapat meningkatkan keamanan dari beberapa

Diterima Redaksi: 15-07-2021 | Selesai Revisi: 17-08-2021 | Diterbitkan Online: 25-08-2021

metode serangan algoritme kriptografi [6]. Seperti Penelitian yang dilakukan oleh Kamardan et.al dengan memodifikasi algoritme RSA pada pembangkitan kunci dengan menggunakan banyaknya bilangan prima [7]. Metode yang dipakai di dalam penelitian tersebut adalah *Chinese Remainder Theorem* dengan menambahkan variable $(r-1)$ ke dalam perkalian $\emptyset(n)$ pada proses pembangkitan kunci. Hasil yang didapatkan bahwa penelitian ini mampu menangkal serangan faktorisasi, Attacks *Small Private Exponent* dan serangan *Chinese Remainder Theorem*. Tetapi, untuk serangan model faktorisasi agar pengacakan kunci lebih bagus dapat menggunakan adalah Metode Kurva Elliptik. Penelitian kedua berkaitan dengan kajian modifikasi algoritme RSA adalah penelitian yang dilakukan oleh Cavusoglu et.al dengan memodifikasi algoritme RSA menggunakan kombinasi antara RNG (Random Number Generator) yang disebut dengan CSRA (*Chaos Based Hybrid RSA*) [8]. Metode algoritme yang dipakai didasarkan pada proses pemilihan dua bilangan prima besar, menggunakan nilai modulus tinggi yang diperoleh dari bilangan-bilangan faktorisasi. Hasil yang didapatkan dari pengujian bahwa algoritma enkripsi CRSA menawarkan distribusi bit yang sangat baik, distribusi korelasi dan nilai NPCR (*Number of Pixel Change rate*) mendekati nilai optimum yaitu 100. Hasil analisis sensitivitas kunci juga menunjukkan sangat sensitif terhadap perubahan kunci yang sangat signifikan.

Salah satu sumbangsih yang berkaitan dengan kajian modifikasi keamanan algoritma RSA maka dalam penelitian ini akan dibahas berkaitan dengan Algoritme kriptografi Dual Modulus RSA atau yang disebut dengan DM-RSA. Algoritme ini merupakan pengembangan dari RSA yang yang digunakan untuk meningkatkan keamanan dari algoritme RSA. Algoritme ini disebut Dual Modulus karena memiliki 2 buah modulus yang digunakan untuk proses pembangkitan kunci, proses enkripsi dan proses dekripsi, sehingga algoritma ini memiliki 4 buah bilangan prima p_1, p_2, q_1, q_2. Teknik semacam ini diharapkan dapat meningkatkan keamanan algoritme RSA dikarenakan menggunakan dual modulus atau 2 buah modulus dan dalam proses enkripsi dan dekripsi menggunakan kunci dobel (2 kunci publik dan 2 kunci pribadi).

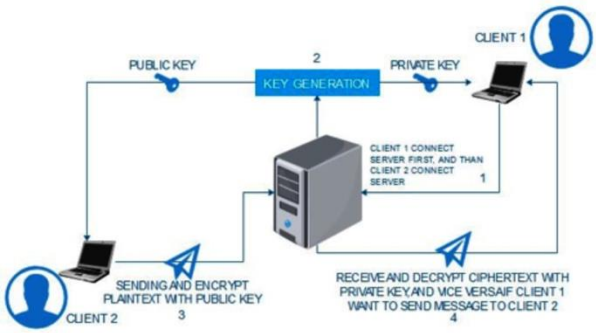
Algoritme RSA memiliki keamanan yang sulit ditembus apabila menggunakan bilangan prima yang besar, tetapi apabila kecil keamanan algoritme tersebut juga menjadi semakin kecil dan rentan. Algoritme DM-RSA memiliki empat buah bilangan prima dan apabila bilangan prima yang digunakan kecil tidak mempengaruhi karena menggunakan 4 buah bilangan prima sekaligus yang artinya 2 kali lipat dari RSA. Diharapkan algoritme yang akan dibahas mampu memiliki ketahanan yang lebih kuat jika dibandingkan dengan algoritme RSA, dikarenakan jika ada attacker yang mendapatkan 1 kunci pribadi tidak akan bisa untuk mendapatkan hasil dekripsi (pesan asli) karena masih ada 1 buah kunci pasangan untuk dekripsi pesan tersebut.

2. Metode Penelitian

Tujuan diimplementasikan algoritme DM-RSA adalah untuk memperbaiki performa algoritme RSA dengan melakukan perbandingan performa dari kedua algoritme tersebut (RSA & DM-RSA) melalui serangkaian pengujian yang dilakukan yaitu pembangkitan kunci, enkripsi, dekripsi pesan dan proses penyerangan yang akan dilakukan pada kedua algoritme tersebut. Untuk menguji performa kedua buah algoritma tersebut diimplementasikan kedalam aplikasi chatting.

2.1 Arsitektur Aplikasi Chatting

Implementasi RSA dan DM-RSA diterapkan di dalam aplikasi chatting dengan menggunakan Teknik client dan server. Server berfungsi untuk menyambungkan antar client dan untuk memantau proses yang dilakukan oleh client seperti proses koneksi client, pembangkitan kunci, perubahan kunci, pengiriman pesan (*ciphertext*), hingga memutus sambungan client. Arsitektur aplikasi chatting digambarkan pada Gambar 1.



Gambar 1. Arsitektur Aplikasi Chatting (Sumber : Aminudin, 2018 [5])

2.2 Rancangan Algoritme RSA

Algoritme RSA memiliki 3 buah proses yang berbeda. Proses pertama diawali dengan proses pembangkitan kunci kemudian enkripsi dan yang terakhir adalah dekripsi. Semua proses tersebuh harus dilakukan sesuai urutan dan proses tersebut dilakukan secara terpisah [9]. Secara dasar proses algoritme RSA adalah sebagai berikut.

1. Membangkitkan bilangan prima p dan q untuk proses pembangkitan kunci di mana nilai p dan q adalah bilangan prima.
2. Menghitung nilai $n = p.q$ (1)
3. Menghitung nilai $\emptyset(n) = (p-1)(q-1)$ (2)
   $\emptyset(n)$ adalah fungsi yang menentukan berapa banyak bilangan - bilangan 1,2,3, . . ., n yang relative prima terhadap n.
4. Pilih sebuah bilangan bulat untuk kunci publik yaitu e, yang relatif prima dengan syarat
   $\big(e, \emptyset(n)\big) = 1$ (3)

DOI: https://doi.org/10.29207/resti.v5i4.3297
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

5. Hitung kunci dekripsi (kunci pribadi) d melalui
   $d\ x\ e\ mod\ (\emptyset(n)) = 1$ (4)
   d adalah bilangan acak yang akan digunakan sebagai kunci pribadi.
6. Enkripsi pesan $c = m^e\ mod\ n$ (5)
7. Dekripsi pesan $m = c^d\ mod\ n$ (6)

2.2.1 Rancangan Pembangkitan Kunci Algoritme RSA

Mekanisme pembangkitan kunci pada algoritme RSA adalah membangkitkan dua bilangan prima (p dan q). Pembangkitan bilangan prima ini menggunakan algoritme pengujian prima seperti algoritme miller rabin. Setelah dilakukan pembangkitan 2 buah bilangan prima dilakukan penghitungan nilai n yaitu $p\ x\ q$ yang menghasilkan n (modulus) lalu hitung $\emptyset(n)$ dengan cara $(p-1)(q-1)$. Selanjutnya memilih sebuah bilangan bulat e secara acak dengan syarat (e, ϕ (n)) memiliki GCD (Greatest Common Divisor) = 1. Pada java fungsi ini dapat menggunakan package java.math.BigInteger. setelah nilai *e* ditemukan dan cocok dengan syarat yang ditentukan dilanjutkan dengan menghitung nilai d melalui algoritme etended Euclid yang memiliki syarat $d\ x\ e\ mod\ (\emptyset(n)) = 1$. Pada java nilai d dapat dicari dengan menggunakan fungsi modInverse pada package java.math.BigInteger.

```
Pseudocode Generate Key RSA
Input: nilai_p, nilai_q
Output: private, publik
  Initialization p, q, phi_n
    while(true)
      p = pembangkitPrimaAcak(panjangBit);
      q = pembangkitPrimaAcak(panjangBit);
      n = p.q;
      phi_n = p-1.q-1;
      while(true
        publik = new int(length,acak());
        if((publik.gcd(phi_n)) = 1)
          break;
    private = kuncipublik.modInverse(phi_n));
```

2.2.2 Rancangan Enkripsi Algoritme RSA

Enkripsi digunakan untuk menyandikan sebuah pesan sebelum pesan tersebut dikirim melalui sebuah jaringan. Untuk menyandikan pesan tersebut menggunakan kunci publik yang didapat dari proses pembangkitan kunci. Proses enkripsi pesan dalam algoritme RSA adalah mengubah pesan menjadi nilai desimal terlebih dahulu. Nilai desimal dari tiap tiap karakter diperoleh berdasarkan nilai ASCII. Kemudian nilai desimal yang diperoleh disebut nilai C. Nilai C dari tiap-tiap karakter digabungkan menjadi sebuah pesan yang dienkripsi kemudian dikirim ke user sesuai tujuan.

```
Pseudocode Proses Enkripsi RSA
Input: publik, n
Output: cipertext
for(int i=0 to plaintext.length)
   int ASCII = (int)plaintext.charAt(i);
   c =int.valueOf(ASCII).pangkat(publik, n)
   ciphertext = ciphertext+C.toString()+" ";
return ciphertext;
```

2.2.3 Rancangan Dekripsi Algoritme RSA

Dekripsi merupakan proses mengubah pesan yang telah di enkripsi yaitu ciphertext (c) menjadi pesan asli yaitu plaintext (m) agar pesan tersebut dapat dibaca. Untuk proses dekripsi menggunakan kunci privat yang didapat dari proses pembangkitan kunci yang telah dilakukan sebelumnya. Nilai m dihitung dari tiap nilai c dengan cara menggunakan kunci privat yang menghasilkan nilai desimal teks. Teks yang menjadi nilai desimal m dikembalikan sesuai dengan ASCII. Proses ini sesungguhnya proses kebalikan dari enkripsi diatas.

```
Pseudocode Proses Dekripsi RSA
Input: c, privat, n
Output: p
for(int i=0 to c.length){
   if((c.charAt(i)+"").equals(" ")){
      int P = (new int(p)).pangkat(privat, n)
      byte              ASCII              =
      Byte.parseByte(P.toString());
   p = p+(char)ASCII;
return p;
```

2.3 Rancangan Algoritme DM-RSA

Rancangan algoritme DM-RSA pada penelitian ini mengikuti perkembangan dari algoritme RSA yang digagas oleh Swami Balram dengan metode pembangkitan kunci, enkripsi, dekripsi yang berbeda [10].

1. Bangkitkan empat bilangan prima dengan ukuran yang sama p_1, p_2, q_1, q_2 untuk proses pembangkitan kunci.
2. Menghitung nilai $n_1 = p_1\ x\ p_2$ (7)
   n_1 adalah nilai modulus pertama
3. Menghitung nilai $n_2 = q_1\ x\ q_2$ (8)
   n_2 adalah nilai modulus kedua
4. Menghitung nilai $\emptyset n_1 = (p_1 - 1)\ x\ (p_2 - 1)$ (9)
   $\emptyset n_1$ adalah fungsi yang menentukan berapa banyak bilangan- bilangan 1,2,3, . . ., n_1 yang relatif prima terhadap n_1
5. Menghitung $\emptyset n_2 = (q_1 - 1)\ x\ (q_2 - 1)$ (10)
   $\emptyset n_2$ adalah fungsi yang menentukan berapa banyak bilangan- bilangan 1,2,3, . . ., n_2 yang relatif prima terhadap n_2
6. Pilih dua buah bilangan bulat e_2 dan e_2 untuk proses enkripsi (kunci publik 1 dan 2).
   Syarat untuk menghitung nilai e_1 dan e_2 adalah
   $GCD\ (e_1, \emptyset(n_1)) = 1$ dan $GCD\ (e_2, \emptyset(n_2)) = 2$ (11)
   e_1 dan e_2 adalah bilangan acak untuk yang berfungsi untuk kunci publik
7. Hitung d_1 dan d_1 untuk proses dekripsi (kunci pribadi 1 dan 2) dengan syarat adalah
   $e_1 x\ d_1\ mod\ (\emptyset n_1) = 1$ (12)
   $e_2 x\ d_2\ mod\ (\emptyset(\emptyset n_2)) = 1$ (13)
   Di mana d_1 dan d_2 adalah bilangan acak yang akan digunakan sebagai kunci pribadi
8. Enkripsi pesan $c = (m^{e_1}\ mod\ n_1)^{e_2}\ mod\ n_2$ (14)
9. Dekripsi pesan $m = (m^{d_2}\ mod\ n_2)^{d_1}\ mod\ n_1$ (15)

DOI: https://doi.org/10.29207/resti.v5i4.3297
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

2.3.1 Rancangan Pembangkitan Algoritme DM-RSA

Pembangkitan kunci algoritme DM-RSA diawali dengan membangkitkan bilangan prima p_1, p_2, q_1, q_2. Pembangkitan bilangan prima ini menggunakan algoritme miller rabin. Selanjutnya memilih 2 bilangan bulat e_1 dan e_2 dengan menggunakan package library java.math.BigInteger. Setelah nilai e_1 dan e_2 ditemukan dan cocok dengan syarat yang ditentukan. Kemudian proses dilanjutkan dengan menghitung nilai d_1 dan d_1 menggunakan algoritme extended euclid yang memiliki syarat $e_1 x\ d_1\ mod\ (\emptyset n_1) = 1$ dan $e_2 x\ d_2\ mod\ \left(\emptyset(\emptyset n_2)\right) = 1$ Pada java perhitungan nilai tersebut dapat dicari dengan menggunakan fungsi modInverse pada package java.math.BigInteger. Nilai e_1 dan e_2 adalah kunci publik digunakan sebagai kunci enkripsi, sedangkan d_1 dan d_2 adalah kunci privat yang digunakan untuk dekripsi pesan yang telah di enkripsi menggunakan kunci publik tersebut. pseudocode alur dari pembangkitan kunci dapat dijelaskan algoritme DM-RSA adalah sebagai berikut,

```
Pseudocode Pembangkitan Kunci DM-RSA
Input: c, privat, n
Output  : kp1,kp2,kv1,kv2
Initialization integer p1, p2, q1, q1, phi_n1,
phi_n2;
while(true)
    p1 = pembangkitPrimaAcak(panjangBit);
    p2 = pembangkitPrimaAcak(panjangBit);
    q1 = pembangkitPrimaAcak(panjangBit);
    q2 = pembangkitPrimaAcak(panjangBit);
n1 = p1.p2;
n2 = q1.q2;
if(n1<n2)
phi_n1 = p1-1.p2-1;
phi_n2 = q1-1.q2-1;
while(true)
    kp1 = new integer(panjangBit, acak());
    kp2 = new integer(panjangBit, acak());
    if(kv1.gcd(phi_n1).equals=1                 &&
      kp2.gcd(phi_n2).equals=1 )
      if(kp1!=kp2)
        break:
kv1 = kunciPublik1.modInverse(phi_n2);
kv1 = kunciPublik1.modInverse(phi_n2);
```

2.3.2 Rancangan Enkripsi Algoritme DM-RSA

Metode enkripsi yang digunakan dalam algoritme DM-RSA ini tidak jauh berbeda dengan algoritme RSA. Perbedaanya terletak pada penghitungan modulus karena algoritme ini menggunakan 2 buah modulus maka penghitungan yang semula satu kali (n) menjadi dua kali (n_1 dan n_2). Enkripsi ini juga menggunakan dua buah kunci public yang didapat dari pembangkitan kunci yaitu e_1 dam e_2. Proses enkripsi pesan dalam algoritme DM-RSA sama dengan RSA, yaitu mengubah pesan menjadi nilai desimal. Nilai desimal dari tiap tiap karakter diperoleh berdasarkan nilai ASCII. Kemudian nilai desimal yang diperoleh disebut nilai C. Nilai C dari tiap-tiap karakter digabungkan menjadi sebuah pesan yang dienkripsi kemudian dikirim ke user sesuai tujuan. Pseudocode algoritma enkripsi untuk DM-RSA ditunjukan sebagai berikut.

```
Pseudocode Proses Enkripsi DM-RSA
input : kp1, kp2, n1, n2
output : ciphertext
for(int i=0 to plaintext.length)
  ASCII = (int)plaintext.charAt(i);
  C     =     int.valueOF(ASCII).pangkat(kp1,
  n1).pangkat(kp2, n2);
  ciphertext = ciphertext+C.toString()+" ";
```

2.3.3 Rancangan Dekripsi Algoritme DM-RSA

Proses dekripsi pada DM-RSA juga tidak jauh berbeda dengan algoritme RSA. Perbedaanya terletak pada penghitungan modulus karena algoritme ini menggunakan 2 buah modulus maka penghitungan yang semula satu kali (n). Dekripsi ini juga menggunakan dua buah kunci privat yang didapat dari pembangkitan kunci yaitu d_1 dan d_2. Seperti yang dijelaskan pada proses dekripsi RSA, nilai m dihitung dari tiap nilai c dengan cara menggunakan 2 buah kunci privat yang menghasilkan nilai desimal teks. Teks yang menjadi nilai desimal m dikembalikan sesuai dengan ASCII. Proses ini sesungguhnya proses kebalikan dari enkripsi.

```
Pseudocode Proses Dekripsi DM-RSA
input : c, kv1, kv2, n1, n2
output : plaintext
for(int i=0 to plaintext.length){
  if((ciphertext.charAt(i)+"").equals(" ")){
    int P = (new int(plaintext)).pangkat(kv1,
  n2).modPow(kv2, n1)
    byte ASCII = Byte.parseByte(P.toString());
    plaintext = plaintext+(char)ASCII;
  }
  return plaintext;
```

3. Hasil dan Pembahasan

Pengujian dilakukan untuk mengetahui performansi dari masing masing algoritme tersebut. Pengujian tersebut meliputi waktu pembangkitan kunci, waktu enkripsi dan waktu dekripsi.

3.1 Perbandingan Pengujian Waktu Pembangkitan Kunci RSA dan DM-RSA

Analisa perbandingan waktu pembangkitan kunci dilakukan dengan membandingkan rata-rata waktu pembangkitan kunci pada tiap waktu yang diperoleh ketika melakukan 1 sampai dengan 5 percobaan pada tiap bit yang digunakan. Panjang bit yang digunakkan adalah 8,16,24 dan 32 bit. Perbandingan ini dilakukan untuk menganalisa performa waktu pembangkitan kunci antara algoritme RSA dan DM-RSA.

Jika dilakukan pengamatan dari hasil pembangkitan kunci yang telah didapat pada Tabel 1 dan Tabel 2 maka ada beberapa hasil analisa yang dapat diperoleh diantaranya adalah (1) panjang bit 8 dan 16 pada algoritme RSA dan DM-RSA tidak terlalu berbeda pada proses pembangkitan kunci, yaitu rata rata dari bit 8 dan 16 memiliki nilai yang hampir sama atau memiliki selisih sedikit tidak naik atau turun secara drastis tetapi selalu mengalami kenaikan. (2) Panjang bit 24 dan 32 pada algoritme RSA terbukti lebih cepat dalam proses

DOI: https://doi.org/10.29207/resti.v5i4.3297
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

pembangkitan kunci, algoritme Dual Modulus RSA lebih lambat hal ini terjadi karena pada proses pembangkitan kunci pada algoritme DM-RSA membangkitkan 4 buah bilangan prima (p_1, p_2, q_1, q_1) sedangkan RSA 2 buah bilangan prima (p dan q).

Tabel 1. Waktu Pembangkitan Kunci Algoritme RSA

Bit	Hasil Pengujian RSA 1-5 (ms)					Rata (ms)
	1	2	3	4	5	
8	16	15	15	16	15	15.4
16	15	15	15	15	16	15.2
24	31	16	15	31	16	21.8
32	47	31	16	16	31	28.2

Tabel 2. Waktu Pembangkitan Kunci Algoritme DM-RSA

Bit	Hasil Pengujian DMRSA 1-5 (ms)					Rata (ms)
	1	2	3	4	5	
8	16	16	31	16	15	18.8
16	15	31	16	16	16	18.8
24	47	15	47	47	47	37.6
32	63	31	47	32	16	37.8

3.2 Perbandingan Pengujian Waktu Proses Enkripsi RSA dan DM-RSA

Analisa perbandingan waktu enkripsi dilakukkan dengan membandingkan rata-rata waktu enkripsi pada tiap waktu yang diperoleh ketika melakukan percobaan 1 sampai dengan 5 pada tiap bit yang digunakan. Panjang bit yang digunakkan adalah 8,16,24 dan 32 bit. Perbandingan ini dilakukan untuk menganalisa performa waktu enkripsi antara algoritme RSA dan DM-RSA. Rata - rata perbandingan waktu pembangkitan kunci RSA dan DMRSA terlihat pada Tabel 3 dan Tabel 4.

Tabel 3. Waktu Proses Enkripsi Algoritme RSA

Bit	Tes 200 karakter 1-5 (ms)					Rata[2] (ms)
	1	2	3	4	5	
8	37	30	22	31	43	32.6
16	26	36	38	42	27	33.8
24	50	44	73	75	67	61.8
32	76	80	53	99	61	73.8

Tabel 4. Waktu Proses Enkripsi Algoritme DM-RSA

bit	Tes 200 karakter 1-5 (ms)					Rata[2] (ms)
	1	2	3	4	5	
8	95	63	59	31	66	62.8
16	53	69	75	75	63	67
24	81	98	84	73	71	81.4
32	94	87	84	95	88	89.6

Jika dilakukan pengamatan dari hasil proses enkripsi Kondisi perangkat yang digunakan memiliki pengaruh pada waktu enkripsi sehingga waktu enkripsi menjadi tidak menentu, tetapi hal ini tidak jadi masalah karena setiap dilakukan perubahan bit selalu memiliki perbedaan waktu yang signifikan. Dan dilihat dari segi kecepatanya algoritme RSA memiliki kecepatan yang lebih cepat dibandingkan algoritme DM-RSA hal tersebut dapat dilihat pada rata-rata hasil proses enkripsi yang didapatkan. Bahwa enkripsi RSA lebih cepat hingga 50% dibandingkan algoritme DM-RSA, hal ini terjadi karena pada proses enkripsi algoritme DM-RSA menggunakan 2 buah kunci publik (e1,e2) dan 2 buah modulus (n1,n2) yang menjadikan proses enkripsi algoritme ini membutuhkan waktu 2 kali lipat dari algoritme RSA yang hanya memiliki 1 kunci publik (e) dan 1 buah modulus (n).

3.3 Perbandingan Pengujian Waktu Enkripsi Enkripsi RSA dan DM-RSA

Analisa perbandingan waktu dekripsi dilakukan dengan membandingkan rata rata waktu dekripsi pada tiap waktu yang diperoleh ketika melakukan percobaan 1 sampai dengan 5 pada tiap bit yang digunakan. Panjang bit yang digunakan adalah 8, 16, 24 dan 32 bit. Perbandingan ini dilakukan untuk menganalisa performa waktu dekripsi antara algoritme RSA dan algoritme DM-RSA.

Tabel 5. Waktu Proses Dekripsi Algoritme RSA

bit	Tes 200 karakter 1-5 (ms)					Rata (ms)
	1	2	3	4	5	
8	23	40	42	14	16	27
16	27	35	29	38	28	31.4
24	31	29	25	39	47	34.2
32	38	65	34	62	61	52

Tabel 6. Waktu Proses Dekripsi Algoritme DM-RSA

bit	Tes 200 karakter 1-5 (ms)					Rata (ms)
	1	2	3	4	5	
8	31	28	30	22	33	28.2
16	36	42	39	37	37	38.2
24	53	57	56	54	55	55
32	70	75	71	73	72	72.2

Jika dilakukan pengamatan lebih mendalam dari hasil dekripsi yang telah didapatkan makan didapatkan beberapa hasil analisa Algoritme RSA terbukti lebih cepat terlihat bahwa algoritme RSA memiliki waktu lebih cepat hingga 50% atau bahkan 2 kali lipat dari algoritme DM-RSA, hal ini terjadi sama dengan masalah pada enkripsi yaitu pada kunci pribadi atau kunci dekripsi algoritme Dual Modulus RSA memiliki 2 buah kunci pribadi (d_1, d_2) dan 2 buah modulus (n_1, n_2) yang mengakibatkan waktu dekripsi menurun 2x lipat atau lebih lambat dibandingkan algoritme RSA yang hanya menggunakan 1 buah kunci pribadi (e) dan 1 buah modulus (n).

3.4 Pengujian metode Keamanan Menggunakan Pengujian Faktorisasi Kraitchik

Pengujian faktorisasi kraitchik dilakukan sesuai dengan celah keamanan algoritme RSA dan DM-RSA yang terletak pada kunci publik e dan n. Faktorisasi kraitchik ini merupakan metode penyerangan untuk mengetahui kunci pribadi (nilai d) dengan cara memfaktorkan nilai n sehingga menjadi nilai p dan q untuk mengetahui nilai d (kunci pribadi). Serangan ini membutuhkan waktu yang sangat lama untuk mendapatkan nilai p dan q karena proses pemfaktoran nilai n hingga menemukan akar sempurna. Pengujian ini menggunakan bit 8,16,24

DOI: https://doi.org/10.29207/resti.v5i4.3297
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)

dan 32. Hasil pengujian serangan ini terlihat pada Tabel 7.

Tabel 7. Perbandingan Pengujian Faktorisasi Kraitchik

Bit	Attacking time (ms)		Percentage of success
	RSA	DMRSA	
8	16	63	100%
16	9578	13110	100%
24	63204	248494123	100%
32	-	-	-

Jika dilakukan pengamatan dari hasil dari pengujian menggunakan Faktorisasi Kraitchik didapatkan bahwa Algoritme DM-RSA terbukti lebih tahan dibandingkan algoritme RSA dalam serangan ini. Algoritme Dual Modulus RSA ini terbukti lebih tahan hingga 2 kali lipat bahkan lebih dibandingkan algoritme RSA. Penyerangan terhadap nilai n 24 bit algoritme RSA dapat bertahan hingga 63204 ms (1 menit 22 detik) sedangkan algoritme Dual Modulus RSA bertahan hingga 248494123 ms (142 menit 47 detik) dan terbukti algoritme DM-RSA lebih memiliki ketahanan yang lebih lama dibandingkan algoritme RSA karena algoritme DM-RSA memiliki memiliki dua buah modulus sedangkan algoritme RSA hanya memiliki satu, walaupun ini mempengaruhi waktu pembangkitan kunci,enkripsi dan dekripsi yang menjadi lebih lama tetapi berhasil memberi ketahanan yang lebih baik terhadap serangan faktorisasi kraitchik. Kemudian pengujian menggunakan 32 bit pada serangan ini diuji hingga memakan waktu 1 hari lebih dan belum mendapatkan nilai faktor n. hal ini membuktikan bahwa bit mempengaruhi keamanan dari kedua algoritme ini, apabila bit yang digunakan semakin besar maka semakin lama pula algoritme ini dapat bertahan dari serangan faktorisasi kraitchik ini.

4. Kesimpulan

Dari hasil pengujian yang telah dilakukan dapat diambil kesimpulan diantaranya adalah berkaitan dengan kecepatan pemrosesan algoritme RSA maupun DM-RSA sangat dipengaruhi oleh jumlah bit dan jumlah karakter yang diinputkan. Kemudian dari pengujian ketahanan algoritme dengan menggunakan metode faktorisasi kraitchik bahwa algoritme DM-RSA terbukti memiliki ketahanan yang lebih baik dibandingkan algoritme RSA karena algoritme DM-RSA memiliki waktu yang lebih lama hingga dua kali jika dibandingkan algoritme RSA standar seperti yang telah disajikan pada Tabel 7. Hal ini disebabkan karena modulus yang digunakan pada algoritme DM-RSA memiliki 2 modulus yang memberikan efek lebih tahan terhadap serangan faktorisasi.

Daftar Rujukan

[1] A. Aminudin, G. P. Aditya, and S. Arifianto, "RSA algorithm using key generator ESRKGS to encrypt chat messages with TCP/IP protocol," *J. Teknol. dan Sist. Komput.*, vol. 8, no. 2, pp. 113–120, 2020, doi: 10.14710/jtsiskom.8.2.2020.113-120.

[2] A. Aminudin and E. Budi Cahyono, "A Practical Analysis of the Fermat Factorization and Pollard Rho Method for Factoring Integers," *Lontar Komput. J. Ilm. Teknol. Inf.*, vol. 12, no. 1, p. 33, 2021, doi: 10.24843/lkjiti.2021.v12.i01.p04.

[3] N. T. E. Hermawan, E. Winarko, and A. Ashari, "Multi prime numbers principle to expand implementation of CRT method on RSA algorithm," *AIP Conf. Proc.*, vol. 2331, no. April, 2021, doi: 10.1063/5.0041856.

[4] R. Thiyagarajan and B. Meenakshi Priya, "An enhancement of EAACK using P2P ACK and RSA public key cryptography," *Meas. J. Int. Meas. Confed.*, vol. 136, no. December, pp. 116–121, 2019, doi: 10.1016/j.measurement.2018.12.031.

[5] A. Aminudin, A. F. Helmi, and S. Arifianto, "Analisa Kombinasi Algoritma Merkle-Hellman Knapscak dan Logaritma Diskrit pada Aplikasi Chat," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 5, no. 3, pp. 325–334, 2018, doi: http://dx.doi.org/10.25126/jtiik.201853844.

[6] A. Purnomo Sidik, S. Efendi, and S. Suherman, "Improving One-Time Pad Algorithm on Shamir's Three-Pass Protocol Scheme by Using RSA and ElGamal Algorithms," *J. Phys. Conf. Ser.*, vol. 1235, no. 1, pp. 0–7, 2019, doi: 10.1088/1742-6596/1235/1/012007.

[7] M. G. Kamardan, N. Aminudin, N. Che-Him, S. Sufahani, K. Khalid, and R. Roslan, "Modified Multi Prime RSA Cryptosystem," *J. Phys. Conf. Ser.*, vol. 995, no. 1, pp. 0–6, 2018, doi: 10.1088/1742-6596/995/1/012030.

[8] Ü. Çavuşoğlu, A. Akgül, A. Zengin, and I. Pehlivan, "The design and implementation of hybrid RSA algorithm using a novel chaos based RNG," *Chaos, Solitons and Fractals*, vol. 104, pp. 655–667, 2017, doi: 10.1016/j.chaos.2017.09.025.

[9] I. Al-Barazanchi, S. A. Shawkat, M. H. Hameed, and K. S. L. Al-Badri, "Modified RSA-based algorithm: A double secure approach," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 17, no. 6, pp. 2818–2825, 2019, doi: 10.12928/TELKOMNIKA.v17i6.13201.

[10] B. Swami, R. Singh, and S. Choudhary, "Dual Modulus RSA Based on Jordan-totient Function," *Procedia Technol.*, vol. 24, pp. 1581–1586, 2016, doi: 10.1016/j.protcy.2016.05.143.

DOI: https://doi.org/10.29207/resti.v5i4.3297
Lisensi: Creative Commons Attribution 4.0 International (CC BY 4.0)