# Algoritma Kriptografi Kunci-publik RSA menggunakan Chinese Remainder Theorem

**Muhamad Reza Firdaus Zen – NIM : 13504048**
Sekolah Teknik Elektro dan Informatika ITB, Bandung , email: if14048@students.if.itb.ac.id

***Abstrak*** **–** Algoritma RSA merupakan salah satu jenis algoritma dalam sistem kriptografi kunci-publik, atau dikenal juga dengan kriptografi asimetris yang adalah bentuk kriptografi dimana pengguna memiliki pasangan kunci kriptografi yaitu kunci publik dan kunci privat. Kunci privat dirahasiakan, sedangkan kunci publik dapat disebarluaskan. Sebuah pesan yang dienkripsi dengan kunci publik hanya dapat didekripsi dengan kunci privat yang berkoresponden. Algoritma RSA merupakan algoritma pertama yang diketahui cocok untuk *digital signature* maupun untuk enkripsi, dan merupakan salah satu dari kemajuan besar dari sistem kriptografi kunci publik. Algoritma RSA secara luas digunakan pada protokol *electronic commerce,*. dan dipercaya aman jika diberikan kunci yang panjang dan penggunaan implementasi yang *up-to date.*

Pada abad pertama, seorang matematikawan China yang bernama Sun Tse mengajukan pertanyaan sebagai berikut yang akan dikenal sebagai Chinese Remainder Problem (CRT) : ” Tentukan sebuah bilangan bulat yang bila dibagi dengan 5 menyisakan 3, bila dibagi 7 menyisakan 5, dan bila dibagi 11 menyisakan 7 “; dan mempelopori Chinese Remainder Theorem yang dapat digunakan untuk menyelesaikan masalah diatas yaitu : Misalkan m1, m2, ...,mn adalah bilangan bulat positif sedemikian sehingga FPB(mi, mj)=1 untuk $i \neq j$. Maka sistem kongruen lanjar $x \equiv ak \pmod{mk}$ mempunyai sebuah solusi unik modulo $m=m1 \cdot m2 \cdot ... \cdot mn$

Dalam karya tulis ini akan dibahas tentang algoritma RSA standar antara lain proses enkripsi dan dekripsinya. Kemudian akan dibahas Chinese Remainder Theorem. Setelah itu akan dibahas bagaimana implementasi Chinese Remainder Theorem pada algoritma RSA antara lain dalam pembangkitan kuncinya.

***Kata Kunci****: RSA, Chinese Remainder Theorem (CRT), digital signature.*

## 1. RSA

Algoritma RSA merupakan singkatan dari nama belakang para penemunya, yaitu antara lain Ron Rivest, Adi Shamir, dan Len Adleman. Algoritma RSA ini merupakan algoritma kriptografi kunci publik yang populer digunakan untuk otentikasi keaslian suatu data digital.. Algoritma RSA termasuk bagian dari web browser dari Microsoft dan Netscape dan digunakan oleh SSL (Secure Socket Layer) yang menjamin keamanan dan privasi di internet. Metode ini didasarkan pada ide bahwa mengalikan dua bilangan dapat mudah dilakukan, khususnya dengan perangkat komputer. Tetapi memfaktorkan bilangan dapat jadi sulit dilakukan”. Contohnya, mudah dilakukan untuk mengambil dua bilangan prima misalnya x dan y dan menghitung hasil operasi kalinya N =xy. Tetapi jika diberikan nilai N, akan sulit untuk menemukan faktor-faktornya yaitu x dan y, terutama untuk bilangan N yang besar. Enkripsi menggunakan nilai atau kunci publik (public key) yang disebarluaskan dan diketahui semua orang yang ingin mengirim pesan. Sedangkan dekripsinya menggunakan sebuah kunci pribadi (private key) yang dijaga kerahasiannya oleh penerima dan tidak dapat dideduksi dari kunci publik. Sistem kriptografi dengan kunci publik seperti halnya algoritma RSA ini bekerja tanpa mengharuskan kedua pihak menjaga kerahasiaan, kunci pribadi tidak perlu diberitahu ke pengirim pesan. Keamanan enkripsi dan dekripsi data dari algoritma RSA terletak pada kesulitan untuk memfaktorkan modulus N yang sangat besar. Besarnya bilangan yang digunakan mengakibatkan lambatnya operasi yang melibatkan algoritma RSA ini.

Berikut skema algoritma RSA

**Algoritma Pembangkitan Kunci**

1. Ambil dua buah bilangan prima sembarang, x dan y
2. Hitung n = xy dan m = (x-1)*(y-1).
3. Pilih bilangan integer e, $1 < e < m$, yang relative prima terhadap m yaitu FPB(e,m) = 1.
4. Hitung eksponen rahasia d, $1 < d < m$, sehingga $ed \equiv 1 \pmod{m}$.
5. Kunci publik adalah (n,e) dan kunci privat adalah (n,d). Nilai x, y dan m juga harus dirahasiakan.

- n disebut juga modulus
- e disebut juga public exponent atau exponent enkripsi
- d disebut juga secret exponent atau exponent dekripsi

**Enkripsi**

1. Ambil kunci public (n,e).

2. Nyatakan plainteks dalam integer positif m
3. Enkripsi menjadi cipher teks $c = m^e \bmod n$

**Dekripsi**

1. Menggunakan kunci privat (n, d) untuk dekripsi dengan rumus $m = c^d \bmod n$.

**Contoh**

Berikut contoh pemakaiannya :
Misalkan x dan y bilangan prima, x = 47 dan y = 71 maka dapat dihitung

$$n = x\ y = 3337 \text{ dan}$$
$$m = (x - 1)(y - 1) = 3220.$$

Pilih kunci publik e = 79 (yang relatif prima dengan 3220). Nilai e dan m dapat dipublikasikan ke umum. Selanjutnya akan dihitung kunci dekripsi d

$$e\ d \equiv 1 \pmod m$$

Kunci dekripsi d sebagai berikut:

$$d = (1 + (k\ m)) / e$$
$$d = (1 + (k\ 3220)) / 79$$

Dengan mencoba nilai-nilai k = 1, 2, 3, …, diperoleh nilai d yang bulat adalah 1019. Ini adalah kunci dekripsi.

Misalkan terdapat plainteks yang sudah dikonversi ke ASCII :

P = 7265827332737873

Pecah P menjadi blok yang lebih kecil (3 digit):

| | |
|---|---|
| p1 = 726 | p4 = 273 |
| p2 = 582 | p5 = 787 |
| p3 = 733 | p6 = 003 |

Blok pertama dienkripsikan sebagai $726^{79} \bmod 3337 = 215 = c1$.
Blok kedua dienkripsikan sebagai $582^{79} \bmod 3337 = 776 = c2$.
Dengan melakukan proses yang sama untuk sisa blok lainnya, dihasilkan chiperteks

C = 215 776 1743 933 1731 158.

Proses dekripsi dilakukan dengan menggunakan kunci rahasia d = 1019.
Blok c1 didekripsikan sebagai $215^{1019} \bmod 3337 = 726 = p1$,
Blok c2 didekripsikan sebagai $776^{1019} \bmod 3337 = 582 = p2$.
Blok plainteks yang lain dikembalikan dengan cara yang serupa. Akhirnya kita memperoleh kembali plainteks semula

P = 7265827332737873

Aplikasi RSA digunakan antara lain pada :

- *Electronic mail*
- *Electronic data transfer*
- *Electronic data interchange*
- Distribusi *software*
- *Data storage*
- Aplikasi yang membutuhkan jaminan integritas data dan otentikasi data asli
- *Digital signature*

**2. Chinese Remainder Theorem**

Sebuah permasalahan berikut dikemukakan oleh Sun Tse dalam bukunya Sunzi Suanjing :
" Tentukan sebuah bilangan bulat yang bila dibagi dengan 5 menyisakan 3, bila dibagi 7 menyisakan 5, dan bila dibagi 11 menyisakan 7 ".
Persoalan jenis ini merupakan suatu contoh permasalahan yang secara luas dikenal sebagai Chinese Remainder Theorem.

**Teorema *Chinese Remainder Theorem***

*Misalkan m1, m2, ...,mn adalah bilangan bulat positif sedemikian sehingga FPB(mi, mj)=1 untuk $i \neq j$. Maka sistem kongruen lanjar*

$$x \equiv ak \pmod{mk}$$

*mempunyai sebuah solusi unik modulo*
$m = m1 \cdot m2 \cdot \ldots \cdot mn$.

Contoh : Tentukan solusi persoalan dari pernyataan Sun Tse diatas !

Pertanyaan Sun Tse dapat dirumuskan kedalam sistem kongruen lanjar:

$$x \equiv 3 \pmod 5$$
$$x \equiv 5 \pmod 7$$
$$x \equiv 7 \pmod{11}$$

Untuk menyelesaikan persolan pertama ambil kongruen pertama, $x \equiv 3 \pmod 5$, memberikan
$x = 3 + 5k_1$ untuk beberapa nilai $k$.
Sulihkan ini ke dalam kongruen kedua menjadi $3 + 5k_1 \equiv 5 \pmod 7$, dari sini kita peroleh $k_1 \equiv 6 \pmod 7$, atau $k_1 = 6 + 7k_2$ untuk beberapa nilai $k_2$.
Jadi kita mendapatkan $x = 3 + 5k_1 = 3 + 5(6 + 7k_2) = 33 + 35k_2$ yang mana memenuhi dua kongruen pertama.
Jika $x$ memenuhi kongruen yang ketiga, kita harus mempunyai $33 + 35k_2 \equiv 7 \pmod{11}$, yang mengakibatkan $k_2 \equiv 9 \pmod{11}$ atau $k_2 = 9 + 11k_3$.
Sulihkan $k_2$ ini ke dalam kongruen yang ketiga menghasilkan
$x = 33 + 35(9 + 11k_3) \equiv 348 + 385k_3 \pmod{11}$.
Dengan demikian, $x \equiv 348 \pmod{385}$ yang

memenuhi ketiga konruen tersebut. Dengan kata lain, 348 adalah solusi unik modulo 385. Catatlah bahwa $385 = 5 \times 7 \times 11$.

Solusi unik ini mudah dibuktikan sebagai berikut. Solusi tersebut modulo $m = m_1 \times m_2 \times m_3 = 5 \times 7 \times 11 = 5 \times 77 = 11 \times 35$.
Karena $77\ 3 \equiv 1 \pmod 5$, $55 \times 6 \equiv 1 \pmod 7$, dan $35 \times 6 \equiv 1 \pmod{11}$, solusi unik dari sistem kongruen tersebut adalah

$x \equiv 3 \times 77 \times 3 + 5 \times 55 \times 6 + 7 \times 35 \times 6 \pmod{385}$
$\equiv 3813 \pmod{385} \equiv 348 \pmod{385}$

Aplikasi Chinese Remainder Theorem banyak digunakan antara lain dalam skema *secret sharing* dan modifikasi algoritma untuk percepatan seperti contohnya modifikasi pada algoritma RSA yang akan dibahas.

## 3. Aplikasi Chinese Remainder Theorem (CRT) dalam Algoritma RSA

Dalam algoritma RSA dengan CRT terdapat perbedaan dengan RSA standar dalam hal pembangkitan kunci dan dekripsi. Pemakaian CRT menghasilkan dekripsi yang jauh lebih cepat daripada RSA standar yang menggunakan eksponen modular. Nilai pangkat rahasia *d* tidak dapat dibuat pendek. Jika $d < N^{0.292}$, sistem RSA dapat dihancurkan secara menyeluruh.

### Pembangkitan Kunci

1. Misal *p* dan *q* adalah dua bilangan prima yang sangat besar dengan ukuran yang hampir sama sedemikian hingga FPB(*p*-1, *q*-1) = 2.
2. Hitung $N = p*q$.
3. Ambil dua bilangan integer secara acak *dp* dan *dq* sedemikian sehingga FPB(*dp, p*-1) = 1, FPB(*dq, q*-1) = 1, dan $dp \equiv dq \pmod 2$.
4. Cari bilangan *d* sedemikian sehingga $d \equiv d_p \pmod{p-1}$ dan $d \equiv d_q \pmod{q-1}$.
5. Hitung $e = d^{-1} \pmod{\mathrm{Phi}\ (N)}$.

Kunci publik adalah <*N,e*> dan kunci rahasia adalah <*p,q,dp,dq*>. Karena FPB(*dp, p*-1) = 1 dan $d \equiv dp \pmod{p-1}$, kita mempunyai FPB(*d,p*-1) = 1. Dengan cara yang sama, FPB(*d,q*-1) = 1. Akibatnya FPB(*d,*Phi(*N*)) = 1, dan karena langkah 5, *e* dapat dihitung nilainya.

Untuk mengaplikasikan CRT pada langkah 4, tiap bilangan modulo masing-masing (*p*-1 dan *q*-1)
harus berpasangan dengan bilangan yang relatif prima agar persoalan ini mempunyai solusi. Perhatikan bahwa *p*-1 dan *q*-1 adalah genap dan karenanya kita tidak dapat langsung mengaplikasikan CRT.
Akan tetapi, FPB((*p*-1)/2, (*q*-1)/2) = 1. Karena FPB(*dp, p*-1) = 1 dan FPB(*dq, q*-1) = 1, didapatkan *dp, dq* adalah bilangan integer ganjil dan *dp*-1, *dq*-1 adalah bilangan integer genap. Perhatikan FPB(*d,p*-1) = 1, yang menunjukkan bahwa *d* adalah bilangan ganjil dan *d*-1 adalah bilangan genap.

Untuk memperoleh solusi
$d \equiv dp \pmod{p-1}$,
$d \equiv dq \pmod{q-1}$
kita mencari solusi dari
$d\text{-}1 \equiv dp - 1 \pmod{p-1}$,
$d\text{-}1 \equiv dq - 1 \pmod{q-1}$.
Dengan menggunakan hukum kanselasi (*cancellation law*) dan menarik faktor 2 keluar, kita mempunyai
$x=d' \equiv (d-1)/2 == (dp-1)/2 \pmod{(p-1)/2}$,
$x=d' \equiv (d-1)/2 == (dq-1)/2 \pmod{(q-1)/2}$.
Dengan menggunakan CRT didapatkan nilai *d* sedemikian sehingga $d = (2 * d') + 1$.

### Enkripsi

Proses enkripsi RSA dengan CRT sama dengan proses enkripsi RSA standar.

### Dekripsi

Misal *M* adalah *plaintext* dan *C* adalah *ciphertext.* Teorema **:**

*Jika C tidak habis dibagi oleh p dan* $dp \equiv d \pmod{p-1}$, *maka* $C^{dp} \equiv C^d \pmod p$.

Untuk dekripsi kita dapatkan :
1. $Mp = C^{dp} \pmod p = C^d \pmod p$ dan
   $Mq = C^{dq} \pmod q = C^d \pmod q$.
2. Kemudian dengan menggunakan CRT didapatkan solusi untuk
   $M = Mp \pmod p = C^d \pmod p$,
   $M = Mq = C^{dq} \pmod q = C^d \pmod q$.

### Contoh

Ambil $p = 7$, $q = 11$, FPB(*p*-1, *q*-1) = 2, $N = p*q = 7*11 = 77$, Phi $(N) = (p-1)*(q-1) = 6*10 = 60$. Misal $dp = 5$, FPB(*dp* , *p*-1) = FPB(5,6) = 1.
$dq = 3$, FPB(*dq* , *q*-1) = FPB(3,10) = 1.
Kita akan mencari nilai *d* sedemikian sehingga

$d \equiv 5 \pmod 6$,
$d \equiv 3 \pmod{10}$.

Kita tidak dapat langsung menggunakan CRT karena FPB(6,10) ≠ 1, oleh karena itu kita mengubah sistem kekongruenan sedemikian dengan cara agar hukum kanselasi (*cancellation law*) dapat diaplikasikan.
Setelah itu, kita mempunyai

$d\text{-}1 \equiv 5\text{-}1 \bmod 6$,
$d\text{-}1 \equiv 3\text{-}1 \bmod 10$.

Dengan mengaplikasikan *cancellation law,*

($d$-1)/2 ≡ (5-1)/2 mod (6/2),
($d$-1)/2 ≡ (3-1)/2 mod (10/2).
$x = d' = (d-1)/2 \equiv 2 \bmod 3$,
$x = d' = (d-1)/2 \equiv 1 \bmod 5$.

Selesaikan dengan menggunakan CRT

$M = 3*5 = 15$,
$M1 = 15/3 = 5$,
$M2 = 15/5 = 3$.
$5*N1 \equiv 1 \bmod 3$, $N1 = 2$,
$3*N2 \equiv 1 \bmod 5$, $N2 = 2$.

Kita punya ,

$d' = x = 2*5*2 + 1*3*2 = 26 \pmod{15} = 11$.

Oleh karena itu

$d' = 11$ dan
$d = (2*d')+1 = (2*11) +1 = 23$

Sekarang kita mencari nilai $e$ sedemikian sehingga

$e*d \equiv 1 \bmod \text{phi}(N)$,
$e*23 \equiv 1 \bmod 60$, $e = 47$

Misalkan *plaintext* $M = 5$.

$C = 5^{47} \pmod{77} = 3$.

Untuk dekripsi kita dapatkan

$M = Mp \bmod p = c^d \bmod p$,
$M = Mq \bmod q = c^d \bmod q$.
$Mp = 3^5 \bmod 7 = 243 \bmod 7 = 5$,
$Mq = 3^3 \bmod 11 = 27 \bmod 11 = 5$.

Dengan menggunakan CRT,
$M = 7*11 = 77$,
$M1 = 77/7 = 11$,
$M2 = 77/11 = 7$.
$11*N1 \equiv 1 \bmod 7$, $N1 = 2$,
$7*N2 \equiv 1 \bmod 11$, $N2 = 8$.
$x = 5*11*2 + 5*7*8 = 390 \bmod 77 = 5$

Sehingga didapat $x = M = 5$, seperi yang diharapkan. Dalam contoh spesifik ini ($Mp$ dan $Mq$) = 5 adalah solusi umum dan tidak perlu mengaplikasikan CRT lebih jauh.

Berikut contoh potongan *source code* dalam bahasa java untuk pembangkitan kunci publik dan enkripsi. (Untuk versi lengkap semua *source code* dapat diakses di http://students.if.itb.ac.id/~if14048/rsa)

```
// RSAPublicKey: RSA public key
import java.math.*; // untuk BigInteger
public class RSAPublicKey
{
   public BigInteger n; //  modulus
publik
   public BigInteger e = new
BigInteger("3"); // exponent enkripsi
   public String Name; // memberi nama
pada pasangan kunci publik/privat
   public RSAPublicKey(String name)
   {
      Name = name;
   }
   // setN: memberi n suatu nilai
   public void setN(BigInteger newN)
   {
      n = newN;
   }
   // getN: mendapat n
   public BigInteger getN()
   {
      return n;
   }
   // RSAEncrypt: menaikkan m pada
perpangkatan e (3) mod n
   public BigInteger
RSAEncrypt(BigInteger m)
   {

      return m.modPow(e, n);
   }
}
```

Sedangkan berikut contoh potongan *source code* dalam bahasa java untuk pembangkitan kunci privat dan dekripsi dengan aplikasi CRT.

```
// RSAPrivateKeyFast: RSA private key,
dengan CRT
import java.math.*;
import java.util.*;
public class RSAPrivateKeyFast extends
RSAPublicKey
{
   public BigInteger TWO = new
BigInteger("2");
   public BigInteger THREE = new
BigInteger("3");

   private BigInteger p; // bilangan
prima pertama
   private BigInteger q; // bilangan
prima kedua
   private BigInteger d; // exponent
dekripsi

   private BigInteger p1, pM1, q1, qM1,
phiN; // untuk pembangkitan kunci

   public RSAPrivateKeyFast(int size,
Random rnd, String name)
   {
      super(name);
      generateKeyPair(size, rnd);
   }

   public void generateKeyPair(int size,
Random rnd)
   {   // size = n dalam bits
      int size1 = size/2;
```

```
      int size2 = size1;
      int offset1 =
(int)(5.0*(rnd.nextDouble()) + 5.0);
      int offset2 = -offset1;
      if (rnd.nextDouble() < 0.5)
      {
         offset1 = -offset1; offset2 = -
offset2;
      }
      size1 += offset1; size2 +=
offset2;
      // bangkitkan 2 bilangan prima
acak, sehingga p*q = n punya ukuran bits
      p1   = new BigInteger(size1, rnd);
// int acak
      p    = nextPrime(p1);

      pM1  = p.subtract(BigInteger.ONE);
      q1   = new BigInteger(size2, rnd);
      q    = nextPrime(q1);

      qM1  = q.subtract(BigInteger.ONE);
      n    = p.multiply(q);
      phiN = pM1.multiply(qM1); // (p-
1)*(q-1)
      d    = e.modInverse(phiN);
   }

   // nextPrime: bilangan prima p
setelah x, dengan p-1 and 3 relatif
prima
   public BigInteger
nextPrime(BigInteger x)
   {
      if
((x.remainder(TWO)).equals(BigInteger.ZE
RO))
         x = x.add(BigInteger.ONE);
      while(true)
      {
         BigInteger xM1 =
x.subtract(BigInteger.ONE);
         if
(!(xM1.remainder(THREE)).equals(BigInteg
er.ZERO))
            if (x.isProbablePrime(10))
break;
         x = x.add(TWO);
      }
      return x;
   }

   // RSADecrypt: fungsi dekripsi, versi
CRT
   public BigInteger
RSADecrypt(BigInteger c)
   {
     BigInteger
c1,c2,a1,a2,d1,d2,m1,m2,m,m3,p1,q1,m02;
     BigInteger TWO = new
BigInteger("2");
     int a=0;
      // return c.modPow(d, n);
      c1=c.mod(p);
      c2=c.mod(q);

d1=d.mod(p.subtract(BigInteger.ONE));
d2=d.mod(q.subtract(BigInteger.ONE));
      m1=c1.modPow(d1,p);
      m2=c2.modPow(d2,q);
      //System.out.println("m2 \n" +m2);
      a1=q.modInverse(p);
      a2=p.modInverse(q);

m=(((a1.multiply(q)).multiply(m1))).add(
(a2.multiply(p)).multiply(m2));
      m=m.mod(n);
      return m;
   }
```

**Uji Coba**

Uji coba dilakukan dengan menguji implementasi algoritma RSA tanpa mengaplikasikan CRT dan implementasi algoritma RSA dengan CRT. Parameter yang diuji antara lain waktu yang dibutuhkan untuk mendekripsi pesan 1024 bit dan waktu untuk *signing,* dekripsi dan *verify* pesan 1024 bit. Implementasi dilakukan dengan bahasa pemrograman java.

Terdapat 5 *file* antara lain (dapat diakses di http://students.if.itb.ac.id/~if14048/rsa) :

1. RSAPublicKey.java
2. RSAPrivateKey.java
3. RSAPrivateKeyFast.java
4. RSATest.java
5. RSATestFast.java

Untuk menjalankan program algoritma RSA tanpa CRT kompilasi file file berikut :

RSAPublicKey.java
RSAPrivateKey.java
RSATest.java

Kemudian jalankan main file nya. Dari hasil *run*-nya didapatkan informasi output sebagai berikut :

- Waktu yang dibutuhkan untuk dekripsi pesan 1024 bit : 0.094 detik
- Waktu yang dibutuhkan untuk *sign,* dekripsi dan *verify* pesan 1024 bit : 0.506 detik

Sedangkan untuk menjalankan program algoritma RSA dengan implementasi CRT kompilasi file file berikut :

RSAPublicKey.java
RSAPrivateKeyFast.java
RSATestFast.java

Kemudian jalankan main file nya. Dari hasil *run*-nya didapatkan informasi output sebagai berikut :

- Waktu yang dibutuhkan untuk dekripsi pesan 1024 bit : 0.031 detik

- Waktu yang dibutuhkan untuk *sign,* dekripsi dan *verify* pesan 1024 bit : 0.188 detik

Dari hasil uji coba diatas dapat diambil kesimpulan bahwa algoritma RSA dengan menggunakan CRT lebih cepat kira kira sebanyak 3 kali bila dibandingkan dengan algoritma RSA tanpa menggunakan CRT. Percepatan proses terletak pada proses dekripsi sedangkan pada proses enkripsi tidak terdapat perbedaan karena prosesnya sama untuk kedua algoritma tersebut sehingga proses enkripsi tidak diikut sertakan pada parameter uji coba.

## 4. Kesimpulan

Dari hasil uraian uraian diatas dapat diambil kesimpulan antara lain :

1. Algoritma RSA adalah algoritma kriptografi kunci publik yang mengandalkan eksponensial modular.
2. Chinese Remainder Problem (CRT) dapat diaplikasikan untuk modifikasi algoritma salah satunya algoritma RSA.
3. Perbedaan algoritma RSA dengan CRT dengan algoritma RSA biasa terletak pada pembangkitan kunci dan proses dekripsi.
4. Algoritma RSA dengan CRT memiliki keuntungan dalam kecepatan proses bila dibandingkan dengan algoritma RSA standar.

## 5. Daftar Pustaka


[1] Munir, Rinaldi, *Diktat Kuliah IF5054 Kriptografi*, Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, 2006.

[2] Munir, Rinaldi, *Diktat Kuliah IF2151 Matematika Diskrit*, Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung, 2005.

[3] http://neworder.box.sk/files/CRT.pdf
waktu akses 12 Januari 2008

[4]http://islab.oregonstate.edu/koc/ece575/04Project2/Raj/Report.pdf
waktu akses 12 Januari 2008

[5] http://mirror.cr.yp.to/eprint.iacr.org/2004/147.ps
waktu akses 12 Januari 2008

[6] http://en.wikipedia.org/wiki/Chinese_remainder_theorem
waktu akses 12 Januari 2008

[7] http://en.wikipedia.org/wiki/Rsa
waktu akses 12 Januari 2008


.