**DAFTAR ISI**

[**1.** **PENDAHULUAN** 2](#_Toc419401277)

[**1.1** **Latar Belakang** 2](#_Toc419401278)

[**1.2** **Rumusan Masalah** 3](#_Toc419401279)

[**1.3** **Batasan Masalah** 3](#_Toc419401280)

[**1.4** **Tujuan** 3](#_Toc419401281)

[**2** **TINJAUAN PUSTAKA** 4](#_Toc419401282)

[**2.1** **RSA** 4](#_Toc419401283)

[**2.1.1** **Relatif Prima** 4](#_Toc419401284)

[**2.1.2** **Algoritma Membangkitkan Pasangan Kunci** 4](#_Toc419401285)

[**2.1.3** **Algoritma Enkripsi/Dekripsi** 6](#_Toc419401286)

[**2.1.4** **MPI** 7](#_Toc419401287)

[**3** **PEMBAHASAN** 8](#_Toc419401288)

[**3.1** **RSA Secara Serial** 8](#_Toc419401289)

[**3.2** **RSA secara Paralel** 12](#_Toc419401290)

[**4** **KESIMPULAN** 17](#_Toc419401291)

[**5** **Daftar Pustaka** 17](#_Toc419401292)

# **PENDAHULUAN**

## **Latar Belakang**

Saat ini internet merupakan penyedia komunikasi yang sangat penting diantara jutaan orang penggunanya. Internet sering digunakan untuk meng-upload halaman web dan dokumen lainnya dari website pribadi ke server webhosting publik*.* Keamanan dan kerahasiaan data pun menjadi isu yang sangat penting untuk mengamankan suatu data, pesan atau informasi, mengingat cara pandang masyarakat yang berkembang sekarang ini. Salah satu perkembangan yang paling signifikan adalah penggunaan email untuk pertukaran informasi atau pesan melalui jaringan internet yang merupakan infrastruktur telekomunikasi dengan standar terbuka yang dapat digunakan oleh banyak pihak.

Penyadapan atau pencurian informasi merupakan hal yang sangat merugikan bagi pengguna jaringan komunikasi. Maka dari itu keamanan dan kerahasiaan data menjadi isu yang penting dan terus berkembang. Salah satu solusi untuk mengamankan komunikasi antar pengguna adalah dengan kriptografi. Kriptografi bertujuan menjaga kerahasiaan informasi yang terkandung dalam data sehingga informasi tersebut tidak dapat diketahui oleh pihak yang tidak sah. Dalam menjaga kerahasiaan data, kriptografi mentransformasikan data jelas (plaintext) ke dalam bentuk data sandi (ciphertext) yang tidak dapat dikenali. Ciphertext inilah yang kemudian dikirimkan oleh pengirim (sender) kepada penerima (receiver). Setelah sampai di penerima, ciphertext tersebut ditransformasikan kembali ke dalam bentuk plaintext agar dapat dikenali.

Fungsi-fungsi yang mendasar dalam kriptografi adalah enkripsi dan dekripsi. Enkripsi adalah proses mengubah suatu pesan, data atau informasi asli (plaintext) menjadi suatu pesan, data atau informasi dalam bahasa sandi (ciphertext). Sedangkan dekripsi adalah proses mengubah suatu pesan, data, atau informasi dalam suatu bahasa sandi kembali menjadi suatu data, pesan, atau informasi asli. Dalam kriptografi simetris s*ingle key* digunakan untuk enkripsi dan dekripsi misalnya Data Encryption Standard (DES) dan Advanced Encryption Standard (AES). Dalam algoritma asimetris kunci yang berbeda digunakan untuk mengenkripsi dan mendekripsi data.

RSA (Rivest, Shamir Adleman &) adalah algoritma kriptografi asimetris yang dikembangkan pada tahun 1977. RSA secara luas digunakan dalam protokol e-commerce elektronik dan diyakini benar-benar aman. RSA sendiri menghasilkan dua kunci, yaitu kunci publik untuk enkripsi dan kunci pribadi untuk mendekripsi pesan. RSA terdiri dari tiga fase, fase pertama adalah *key generation* yang adalah untuk digunakan sebagai kunci untuk mengenkripsi dan mendekripsi data, tahap kedua adalah enkripsi, di mana sebenarnya proses konversi plaintext untuk cipher teks yang sedang dilakukan dan tahap ketiga adalah dekripsi, di mana teks terenkripsi dikonversi ke teks biasa di sisi lain. Sebagai kunci publik digunakan untuk enkripsi dan dikenal untuk semua orang dan dengan bantuan kunci publik.

Pada tugas besar mata kuliah HPC ini, kelompok kami mengangkat topik Kriptografi dengan penerapan Algoritma RSA sebagai pengamanan data dan melakukan komputasi paralel pada proses enkripsi dari algoritma tersebut. Berikut ini akan dijelaskan bagaimana penerapan algoritma RSA pada proses enkripsi yang diparalelkan.

## **Rumusan Masalah**

Berdasarkan latar belakang yang telah diuraikan diatas, maka akan dibahas pada tugas besar ini adalah bagaimana penerapan algoritma paralel pada proses enkripsi RSA.

## **Batasan Masalah**

Batasan masalah pada tugas besar ini adalah proses parale hanya diterapkan pada proses enkripsi RSA.

## **Tujuan**

Tujuan dari tugas besar ini adalah untuk mengetahui penerapan algoritma RSA pada proses enkripsi yang diparalelkan.

# **TINJAUAN PUSTAKA**

## **RSA**

RSA merupakan algoritma kriptografi kunci-publik yang paling popular. Algoritma RSA dibuat oleh 3 orang peneliti dati MIT pada 1976 yaitu : Ron (R)ivest, Adi (S)hamir dan Leonard (A)dleman. Pada algoritma ini dilakukan pemfaktoran untuk memperolh kunci privat.

Algoritma RSA memilki besaran-besaran sebagai berikut :

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

### **Relatif Prima**

Dua buah bilangan bulat *a* dan *b* dikatakan relative prima jika PBB(*a,b*)=1. Jika *a* dan *b* relative prima, maka terdapat bilangan bulat *m* dan *n* sedemikian sehingga

*ma + nb* = 1 (2.1)

Untuk mengecek apakah PBB(*a,b*)=r dapat menggunakan algorita Euclidean yaitu :

1. Jika *b* = 0 maka *b* adalah PBB(*a,b*); stop.

Jika tidak (yaitu *b*0) lanjutkan ke langkah 2.

1. Bagilah *a* dengan *b* dan misalkan  *r* sebagai sisanya.
2. Gantilah nilai *a* dengan nilai *b* dan nilai *b* dengan nilai *r*, lalu kembali ke langkah 1.

**Contoh 2.1**

20 dan 3 relatif prima sebab PBB(20,3)=1. Dihung dengan algoritma Eucilidean sbb :

20=63 + 2

3=12 +1

2=2+0

Sisa pembagian terakhir sebelum 0 adalah 1, maka PBB(20,3)=1.

### **Algoritma Membangkitkan Pasangan Kunci**

1. Pilih dua buah bilangan prima sembarang *p* dan *q*.

2. Hitung *n=p* (Sebaiknya *pq*, sebab jika sama makan sehingga *p* dapat diperoleh dengan menarik akar dua dari *n*).

3. Hitung (*n*) = (*p*

4. Pilih kunci public, *e,* yang relative prima terhadap (*n)*

5. Bangkitkan kunci privat dengan menggunakan prsamaan *e*(*n*)). Perhatikan bahwa *e*(*n*)) ekivalen dengan *e*(*n*), sehingga secara sederhaa *d* dapat dihitung dengan

(2.2)

Pseudo code menbangkitkan kunci privat adalah :

d,ketemu,k : interger

k=0;

ketemu =0; // 0 false 1 true

while (ketemu=0 and k<kunci enkripsi) do

d=(1+(k(*n*))/kunci enkripsi

if ((de) mod (*n*) ==1) then

ketemu =1 //true

else

k=k+1

endif

endwhile

* d //kunci rahasia

Hasil dari algoritma diatas adalah :

* Kunci public adalah pasangan (*e,n*)
* Kunci privat adalah pasangan (*d,n*)

**Contoh 2.2**

Misalkan Alice akan membangkitan kunci public dan kunci privat miliknnya. Alice memilih *p=*47 dan *q=*71. Selanjutnya, Alice menghitung

*n=p*=3337

dan

(*n*) = (*p* =3220

Alice memilih kunci public *e* =79 karena, relative prima dengan 32220. Alice mengumumkan nilai *e* dan *d* kepada public. Selanjutnya Alice menghitung kunci deskripsi *d* seperti yang dituliskan pada langkan 5 dengan menggunakan persamaan 2.2,

Dengan mencoba nilai-nilai *k=*1,2,3,…, diperoleh nilai *d* yang bulat adalah 1019. Ini adalah kunci privat untuk mendeskripsi pesan. Kunci privat ini harus dirahasiakan oleh Alice. Jasi, perhitungan kunci ini menghasilkan pasangan kunci privat dan kunci public.

### **Algoritma Enkripsi/Dekripsi**

#### **Enkripsi**

1. Ambilah kunci public penerima pesan, *e,* dan modulus *n*.
2. Nyatakan plaintext m menjadi array interger (blok-blok) *m1, m2, …,* sedemikian sehingga setiap blok merepresntasikan nilai didalan selang [0,*n*-1].
3. Setiap blok *mi* dienkripsi menjadi blok *ci* dengan rumus *ci =* mod *n.*

#### **Deskripsi**

Setiap blok chipertext *ci* dideskripsi kembali menjadi blok *mi* dengan rumus *mi =* mod *n.*

Contoh 2.3

Misalkan Bob mengirim pesan kepada Alice. Pesan(plaintext) yang akan dikirm oleh Alice adalah

*m =* HARI INI

kemudia dikonversi ke ASCII adalah

7265827332737873

Bob menyimpan ASCII tersebut ke dalam array atau memecah *m* menjadi blok

*m0 =* 72 *m4 =* 32

*m1 =*65 *m5* = 73

*m2 =82 m6* = 78

*m3 = 73 m7* = 73

|  |
| --- |
| 72 |
| 65 |
| 82 |
| 73 |
| 32 |
| 73 |
| 78 |
| 73 |

Bob mengetahui kunci public Alice adalah *e* = 79 dan *n =* 3337. Alice dapat mengenkripsikan setiap blok plaintext sebagai berikut :

*c0*= 7279 mod 3337 = 2628; *c1*= 6579 mod 3337 = 2103;

*c2*= 8279 mod 3337 =984 ; *c3*= 7379 mod 3337 =583 ;

*c4*= 3279 mod 3337 =2018 ; *c5*= 7379 mod 3337 =583 ;

*c6*= 7879 mod 3337 =1609 ; *c7*= 7379 mod 3337 =583 ;

Jadi chiperteks *c* yang dihasilkan adalah :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2628 | 2103 | 984 | 583 | 2018 | 583 | 1609 | 583 |

Deskripsi dilakukan dengan menggunakan kunci privat

*d=*1019

blok-blok chipertext dideskripsikan sebagai berikut

*m0*= 26281019 mod 3337 = 72; *m1*= 21031019 mod 3337 = 65;

*m2*= 9841019 mod 3337 = 82 ; *m3*= 5831019 mod 3337 = 73;

*m4*= 20181019 mod 3337 = 32; *m5*= 5831019 mod 3337 = 73;

*m6*= 16091019 mod 3337 = 78; *m7*= 5831019 mod 3337 = 73;

Blok plaintext yang lain dikembalikan dengan cara yang serupa. Sehingga diperoleh kembali plaintext semua :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 72 | 65 | 82 | 73 | 32 | 73 | 78 | 73 |

yang dalam sistem pengkodean ASCII adalah :

*m =* HARI INI

### **MPI**

Algoritma paralel merupakan algoritma yang dapat dieksekusi dalam satu waktu pada banyak perangkat processing yang berbeda, dan pada akhirnya akan digabungkan kembali untuk mendapatkan hasil yang benar. Beberapa algoritma mudah untuk dibagi ke dalam cara ini. Contohnya, memisahkan pekerjaan dengan mengecek semua nomor dari 1sampai 100000 untuk melihat pernyataan mana yang akan dijadikan landasan untuk menetapkan subset dari nomor setiap prosesor yang ada, dan kemudian menaruh data dengan hasil yang akan dkembalikan bersama-sama.

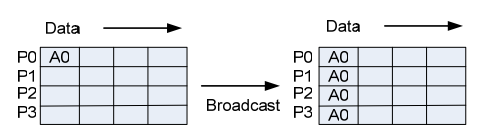
Platform algoritma parallel yang digunakan pada tugas besar ini adalah MPI (Message Passing Interface) . MPI merupakan library yang dapat digunakan untuk membuat program parale di C.Program akan di compile si server HPC dengan IP 10.5.1.254. Tools yang digunakan adalah putty atau bisa menggunakan ssh. Putty merupakan sebuah program open source remote terminal yang digunkan untuk meremote computer.

Pada penelitian ini sismtem komunikasi MPI yang digunakan adalah :

1. ***Broadcast***

*Broadcast* adalah sebuah operasi komunikasi kolektif dimana sebuah proses dalam suatu kelompok jaringan mengirimkan data yang sama kepada semua proses lain pada kelompok jaringan tersebut. Fingksi MPI yang digunakan untuk melakukan broadcast baik di induk maupun di anak adalah :

MPI\_Bcast(void \*pesan, int jumlah\_pesan, MPI\_Datatype tipe\_data, int root, MPI\_Comm komunikasi)



Gambar 2.1

1. ***MPI\_Send* dan *MPI\_Recv***

MPI\_Send berfungsi untuk mengirimkan suatu pesan dari suatu proses tertentu ke proses yang lain. Sintaksnya adalah :

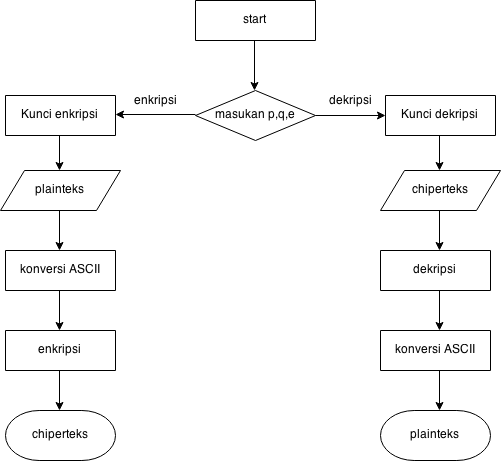
MPI\_Send (void\* message, int count, MPI\_Datatye datatype, int dest, int tag, MPI\_Comm comm).

Sedangkan MPI\_Recv berfungsi untuk menerima pesan(data) dari suatu proses tertentu Sintasnya adalah :

MPI\_Recv(void\* message, int count, MPI\_Datatype datatype, int source, int tag, MPI\_Comm comm, MPI\_Status\* status).

# **PEMBAHASAN**

## **RSA Secara Serial**



**Gambar 3.1** Flowchart RSA

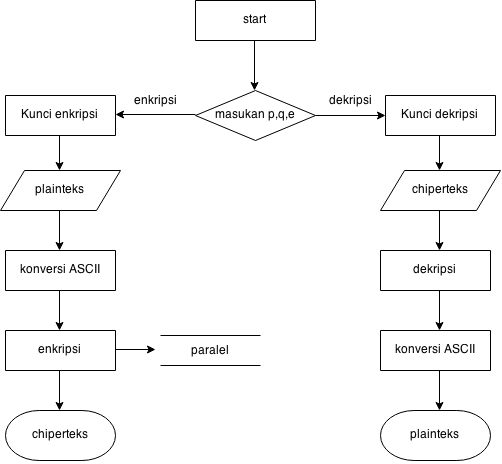
Penjelasan program RSA secara Serial

* ***Enkripsi***
* Masukkan bilangan prima p, q, dan e
* Cek apakah e relative prima terhadap
* Plainteks dikonversi ke ASCII dan dibagi ke bebearapa blok (diamasukkan pada suatu tabel)
* Proses enkripsi dengan *cie* **mod** *n*
* Chiperteks diperoleh dari hasil modulo
* ***Dekripsi***
* Mencari *d* privat key (kunci dekripsi)
* Chiperteks dibagi ke beberapa blok (dimasukkan dalam suatu tabel)
* Proses dekripsi dengan *mie* **mod** *n*
* Hasil module dikonversi dengan ASCII, kemudian diperoleh plainteks

***Source Code* RSA secara serial**

|  |
| --- |
| #include<stdio.h>  #include<conio.h>  #include<string.h>  #include<math.h>  int komprima(int prima, int sigma\_N); //MENENTUKAN BILANGAN PRIMA  long int BigMod(int b,int p,int m);  long int square(long long a);  void loadStringFromFile(char \*filename, char \*s);  void enkripsi(long int k\_ascii[],int e,long N,int count,char text[6]);  int kunci\_rahasia(int e, int sigma\_N);  void dekripsi(int d,long N,int count,char text[20]);  int i;  int z[100000];  //19 17 23  int main(){  int p,q,d,sigma\_N,e,found;  long N;  long int k\_ascii[100000];  int count=0;  char kata[10000];  loadStringFromFile("text.txt",kata);  int strlength;  strlength=strlen(kata);  printf("banyak karakter : %i\n",strlength);  printf("tugas besar\n");  printf("masukkan nilai p : ");  scanf("%i",&p);  printf("masukkan nilai q : ");  scanf("%i",&q);  N=p\*q;  sigma\_N=(p-1)\*(q-1);  printf("nilai N adalah %d dan sigma\_N adalah %d\n",N,sigma\_N);  printf("masukkan e : ");  scanf("%i",&e);  found=koprima(e,sigma\_N);  printf("e termasuk prima %d\n",found);  d=kunci\_rahasia(e,sigma\_N);  printf("kunci rahasia %d\n",d);  if(found==1){  for(i=0;kata[i]!=NULL;i++){  k\_ascii[i]=kata[i];  printf("ascii %i \n",k\_ascii[i]);  count++;  }  printf("banyak karakter : %i\n",count);  enkripsi(k\_ascii,e,N,count,"hasil\_rsa.txt");  printf("\n");  printf("\n");  dekripsi(d,N,count,"rsa\_hasil.txt");  }else{  printf("kunci salah");  }  }  int koprima(int prima, int sigma\_N){  int found=0;  int i=0;  if(prima==2){  found=1;  }else{  for(i=2;i<prima;i++){  if(prima%i==0){  found=0;  break;  } else {  found=1;  }  }  }  int r;  while((r=prima % sigma\_N)!=0){  prima=sigma\_N;  sigma\_N=r;  }  return sigma\_N;  }  int kunci\_rahasia(int e, int sigma\_N){  int d;  int ketemu=0;  i=0;  int r;  while(ketemu==0 && i<e){  d=(1+(i\*sigma\_N))/e;  if ((d\*e)%sigma\_N==1) {  ketemu=1;  }else{  i++;  }  }  return d;  }  void loadStringFromFile(char \*filename, char \*s){  FILE \*fp;  int i,j;  if((fp=fopen(filename, "r"))==NULL) {  printf("Cannot open file.\n");  } else {  char x[100000];  fgets(x,100000,fp);  fclose(fp);  strcpy(s,x);  }  }  long int BigMod(int b,int p,int m){  if(p==0)return 1;  else if(p%2==0)return square(BigMod(b,p/2,m))%m;  else return ((b % m) \* BigMod(b,p-1,m)) % m;  }  long int square(long long a){  return a\*a;  }  void enkripsi(long int k\_ascii[],int e,long N,int count,char text[20]){  FILE \* data;  data = fopen(text,"wb");  FILE \* numerik;  numerik = fopen("rsa\_hasil.txt","wb");  for(i=0;i<count;i++){  z[i]=BigMod(k\_ascii[i],e,N);  printf("%i\n",z[i]);  fprintf(data,"%i",z[i]);  }  fclose(data);  for(i=0;i<count;i++){  fprintf(numerik,"%i\n",z[i]);  }  fclose(numerik);  }  void dekripsi(int d,long N,int count,char text[20]){  FILE \* data;  data = fopen(text,"rb");  for(i=0;i<count;i++){  fscanf(data,"%i",&z[i]);  //printf("%i\n",z[i]);  }  fclose(data);  printf("\n");  printf("\n");  for(i=0;i<count;i++){  z[i]=BigMod(z[i],d,N);  printf("%i",z[i]);  }  printf("\n");  printf("\n");  FILE \* dekrip;  dekrip = fopen("dekripsi.txt","wb");  for(i=0;i<count;i++){  fprintf(dekrip,"%c",z[i]);  printf("%c",z[i]);  }  } |

## **RSA secara Paralel**



**Gambar 3.2** flowchart RSA secara parallel

Pada tugas besar ini penerapan parallel terdapat diproses enkripsi, adapun langkah-langkahnya adalah :

* ***Enkripsi***
* *p,q,n,e,* di rank-0
* mengecek apakah *e* relative prima terhadap
* plaintexs konversi menjadi ASCII dan dimasukkan ke sebuah tabel di rank-0
* induk(rank-0) mengirim variable-variabel umum seperti *e* dan *n* ke semua anak dengan menggunakan MPI\_Bcast.
* Induk mengirim tabel plaintext yang telah dikonversi menjadi ASCII ke semua anank dengan menggunakan MPI\_Bcast
* Induk membagi pekerjaan ke setiap anak mengirim indeks tabel ke anak menggunkan MPI\_Send
* Anak menerima variable *e*  dan *n*  dan tabel dengan MPI\_bcast
* Anak menerima indeks tabel dengan MPI\_Recv
* Anak melakukan proses enkripsi dengan module
* Hasil modulo akan dikirimkan ke induk
* Induk menerima hasil module dan menggabungkannya menjadi chiperteks

**Source code RSA secara Pararel**

|  |
| --- |
| #include<stdio.h>  #include<mpi.h>  #include<string.h>  #include<math.h>  int komprima(int prima, int sigma\_N); //MENENTUKAN BILANGAN PRIMA  long int BigMod(int b,int p,int m);  long int square(long long a);  void loadStringFromFile(char \*filename, char \*s);  int kunci\_rahasia(int e, int sigma\_N);  void dekripsi(int c[], int d,long N,int strlength,char text[20]);  const int SIZE = 10000;  int main (int argc, char\* argv[]){  int i,start,end, strlength;  int m[SIZE], c[SIZE];  char text[SIZE];  int rank, jumlahProc;  int N, sigma\_N, found;  int p,q,e,d;  int status\_prima;  loadStringFromFile("text.txt",text); //membaca text  strlength=strlen(text);  printf("banyak karakter : %i\n",strlength);  MPI\_Init (&argc, &argv);  MPI\_Status status;  MPI\_Comm\_rank (MPI\_COMM\_WORLD, &rank); // baca id prosesor  MPI\_Comm\_size (MPI\_COMM\_WORLD, &jumlahProc); // baca banyak prosesor  int RANKSIZE = strlength/(jumlahProc-1);  int modRANKSIZE=strlength%(jumlahProc-1);    if(modRANKSIZE!=0){  printf("tidak diproses");  MPI\_Abort(MPI\_COMM\_WORLD,911);  }  if(rank==0){  p=19;  q=17;  e=23;  N=p\*q;  sigma\_N=(p-1)\*(q-1);  status\_prima=koprima(e,sigma\_N);  if(status\_prima==1){  for(i=0;i<SIZE;i++){  m[i]=text[i];  }  MPI\_Bcast (&e, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);  MPI\_Bcast (&N, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);  MPI\_Bcast (&m[0],strlength, MPI\_INT, 0, MPI\_COMM\_WORLD);    for (i=1; i<jumlahProc; i++){  start= (i-1)\*RANKSIZE;  end=(i\*RANKSIZE)-1;  MPI\_Send(&start,1,MPI\_INT,i,0,MPI\_COMM\_WORLD);  MPI\_Send(&end,1,MPI\_INT,i,0,MPI\_COMM\_WORLD);  }//pembagian tugas  for (i=1; i<jumlahProc; i++){  start= (i-1)\*RANKSIZE;  MPI\_Recv(&c[start],RANKSIZE,MPI\_INT,i,0,MPI\_COMM\_WORLD,&status);  printf("%d receive dari array enkripsi . Messagesource = %d\n", rank, status.MPI\_SOURCE);  }//menerima hasil tugas dari anak    FILE\* data;  data=fopen("ENKRIPSI\_RSA.txt","wb");  for(i=0;i<strlength;i++){  fprintf(data,"%i\n",c[i]);  }  fclose(data);    d=kunci\_rahasia(e,sigma\_N);  dekripsi(c,d,N,strlength,"DESKRIPSI\_RSA.txt");    }    }  else {  MPI\_Status status;  MPI\_Bcast (&e, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);  MPI\_Bcast (&N, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);  MPI\_Bcast (&m[0], strlength, MPI\_INT, 0, MPI\_COMM\_WORLD);  MPI\_Recv(&start,1,MPI\_INT,0,0,MPI\_COMM\_WORLD,&status);  MPI\_Recv(&end,1,MPI\_INT,0,0,MPI\_COMM\_WORLD,&status);  printf("%d receive vaiabel e = %d dan N = %d dari induk\n",e,N,rank);  printf("start : %d",start);  printf("end : %d",end);  for(i=start;i<=end;i++){  c[i]=BigMod(m[i],e,N);  printf("rank ke%i %i %i\n ",rank,i,m[i]);  }  MPI\_Send(&c[start],RANKSIZE,MPI\_INT,0,0,MPI\_COMM\_WORLD);  }  MPI\_Finalize();  return 0;  }  void loadStringFromFile(char \*filename, char \*s){  FILE \*fp;  int i,j;  if((fp=fopen(filename, "r"))==NULL) {  printf("Cannot open file.\n");  } else {  char x[10000];  fgets(x,10000,fp);  fclose(fp);  strcpy(s,x);  }  }  int koprima(int prima, int sigma\_N){  int found=0;  int i=0;  if(prima==2){  found=1;  }else{  for(i=2;i<prima;i++){  if(prima%i==0){  found=0;  break;  } else {  found=1;  }  }  }  int r;  while((r=prima % sigma\_N)!=0){  prima=sigma\_N;  sigma\_N=r;  }  return sigma\_N;  }  long int BigMod(int b,int p,int m){  if(p==0)return 1;  else if(p%2==0)return square(BigMod(b,p/2,m))%m;  else return ((b % m) \* BigMod(b,p-1,m)) % m;  }  long int square(long long a){  return a\*a;  }  void dekripsi(int c[], int d,long N,int strlength,char text[20]){  FILE \* data2;  data2 = fopen(text,"wb");  int z[10000];  int i;  for(i=0;i<strlength;i++){  z[i]=BigMod(c[i],d,N);  fprintf(data2,"%c",z[i]);  }  fclose(data2);  }  int kunci\_rahasia(int e, int sigma\_N){  int d;  int ketemu=0;  int i=0;  int r;  while(ketemu==0 && i<e){  d=(1+(i\*sigma\_N))/e;  if ((d\*e)%sigma\_N==1) {  ketemu=1;  }else{  i++;  }  }  return d;  } |

Pada tugas besar ini, pembagian pekerjaan oleh induk ke anak dilakukan berdasarkan hasil modulo banyak karakter dengan banyak prosesor anak. Jika hasil modulonya adalah nol, maka proses dapat dikerjakan. Induk akan mengirimkan tabel m[i] ke semua anak dengan MPI\_Bcast. Banyaknya size pekerjaan tiap anak ditentukan berdasarkan hasil pembagian banyak karakter dengan banyak prosesor anak, dimana induk akan mengirikan indeks yang akan dikerjakan oleh setiap anak.Digunakan variable bantu start dan end sebagai indeks awal dan akhir yang dikirimkan ke setiap anak. Algoritma pembagian pekerjaan pada tugas besar ini adalah:

RANKSIZE = strlength/(jumlahProc-1);

{banyak karakter dibagi dengan jumlah anak}

for (i=1; i<jumlahProc; i++){

start= (i-1)\*RANKSIZE; {indeks awal}

end=(i\*RANKSIZE)-1; {indeks akhir}

MPI\_Send(&start,1,MPI\_INT,i,0,MPI\_COMM\_WORLD);

MPI\_Send(&end,1,MPI\_INT,i,0,MPI\_COMM\_WORLD);

}

Jika hasil modulo banyak karakter dengan banyak prosesor anak tidak samadengan nol maka proses tidak bisa dilanjutkan.

# **KESIMPULAN**

Sistem komunikasi pada MPI dapat menggunkanan MPI\_Bcast dan MPI\_Send MPI\_Recv. MPI\_Bcast kounikasi dengan induk mengirimkan data ke semua anak. Sedangkan MPI\_Send dan MPI\_Recv merupakan komunikasi poin to point dimana induk mengirimkan data hanya kepada anak tertentu. Pada proses parallel pembagian pekerjaan dapat ditentukan berdasarkan ukuran array dibagi dengan banyak prosesor anak.

# **Daftar Pustaka**

[1] mpitutorial.com

[2] <http://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/Algoritma%20RSA.pdf>

[3] <http://id.wikipedia.org/wiki/RSA>