BAB V. IMPLEMENTASI dan PENGUJIAN

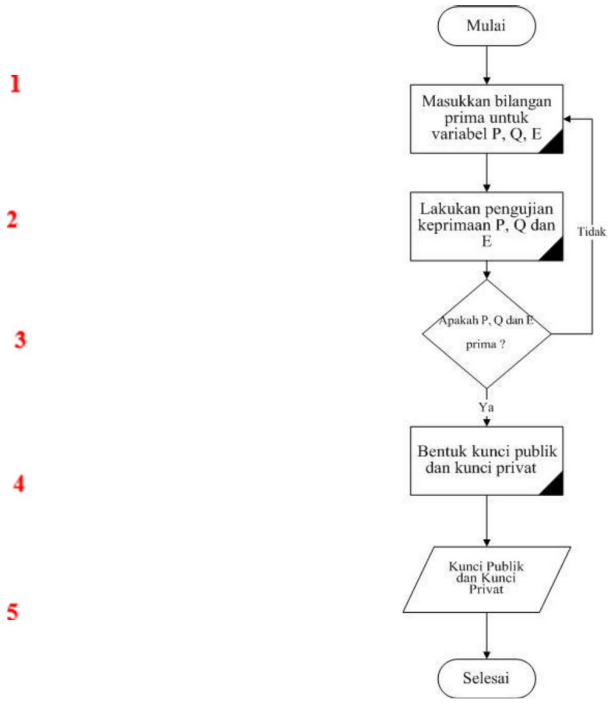
5.1 Implementasi

Pada tahap ini, penulis akan mengimplementasiakan algoritma kriptografi RSA *(Rivest-Shamir-Adleman)* pada sistem aplikasi berbais *file* transfer dengan *socket tcp/ip* berdasarkan desain sistem yang telah di buat.

5.1.1 Implementasi Pembangkit Kunci

Pada Aplikasi Desain dan Implementasi Algoritma Kriptografi RSA *(Rivest-Shamir-Adleman)* pada Telkom Bojonegoro untuk Meningkatkan Keamanan Sistem Jaringan Proses pembangkit kunci dilakukan terlebih dahulu oleh klien. Proses ini akan menghasilkan kunci privat yang digunakan untuk proses dekripsi dan kunci publik yang akan digunakan pihak perusahaan untuk proses enkripsi. Berdasarkan *flowchart* tersebut dapat di jelaskan bahwa *point* 1 pengguna dapat memilih dua buah bilangan prima acak untuk mengisi atau memenuhi beasaran P dan Q dan E sebagai kunci publik. Pada point 2 pengguna dapat melakukan pengujian E atau kunci publik yang merupakan *coprime* atau keprimaan pada aplikasi. *Point* 3 Jika FPB E adalah 1 atau *coprime* maka algoritma akan di teruskan jika tidak maka akan kembali ke *point* 1. Pada *point* 4 bentuk pasangan kunci atau *keypare* yang di simpulkan dengan besaran E dan D, dan *point* 5 Kunci publik dan privat telah terbentuk.

Berikut *flowchat* yang akan menggambarkan proses pembangkit kunci dari sebuah metode RSA dapat dilihat pada gambar 5.1:



Gambar 5. 1 *Flowchart* Pembangkit Kunci

5.1.2 Proses Pembangkit Kunci

Pada proses 1, 4 dan 5 dijelaskan dalam bentuk algortima berikut dapat dilihat pada tabel 5.1:

Tabel 5. 1 Algoritma Pembangkit Kunci RSA

Tahap	**Algoritma** Pembangkit Kunci RSA
1	Bangkitkan bilangan prima besar *p* dan *q*.
2	n □ p * q
3	Φ(n) □ (p - 1) * (q - 1)
4	e □ $Z_{\Phi(n)}$ dengan gcd(e, Φ(n)) = 1
5	d □ e^{-1} pada $Z_{\Phi(n)}$
6	dP □ d mod (p - 1)
7	dQ □ d mod (c - 1)
8	qInv = q^{-1} pada Zp Kpublik = (e, n),K privat = (dP, dQ, qInv, p,q)

Uraian untuk proses pembakit kunci RSA adalah sebagai berikut:

1. Membangkitkan bilangan prima besar untuk nilai p dan q (tahap 1) Bilangan prima merupakan bilangan asli (bilangan yang lebih besar sama dengan nol, b >= 0) yang lebih besar dari 1 (p > 1) memiliki faktor pembagi yaitu 1 dan bilangan itu sendiri. Contoh bilangan prima adalah {2, 3, 5, 7, ...}. Satu (1) bukan merupakan bilangan prima karena menurut pengertian bilangan prima, bilangan prima adalah bilangan bulat lebih dari 1 [2]. Nilai p dan q tidak boleh sama ($p \neq q$) Pada proses pembangkit kunci ini diperlukan bilangan prima karena hanya bilangan prima yang dapat memenuhi prinsip sistem kongruen pada eksponensial modular. Selain itu perkalian prima besar yang akan dilakukan pada langkah 2 akan menentukan kekuatan dari proses enkripsi yang akan dilakukan. Sebagai contoh untuk menggambarkan proses analisis ini akan kita tentukan nilai p dan q sebagai berikut p = 17 dan q = 31.
2. Perkalian p dan q untuk mendapatkan nilai n (tahap 2)

   Nilai n ini akan menjadi salah satu variabel untuk melakukan proses enkripsi. Semakin besar nilai p dan q maka akan menghasilkan nilai n yang semakin besar. Nilai n yang besar maka akan semakin sulit bagi peretas untuk menentukan nilai p dan q. Nilai p dan q sendiri selain untuk menentukan nilai n tapi juga merupakan sumber bagi pembentukan nilai d yang merupakan unsur pembentuk kunci privat. Sebagai contoh untuk menggambarkan proses analisis ini akan kita tentukan nilai p dan q yang telah ditetapkan pada proses pertama akan kita lakukan perkalian.

   $n = p \times q$

   $= 17 \times 31$

   $= 527$

1. Mencari nilai ϕn (tahap 3)

   Mencari nilai ϕn sangat penting, hal ini karena nilai ϕn akan digunakan untuk menentukan nilai e yang akan dicari pada proses selanjutnya.
   Dengan nilai yang telah ditetapkan pada contoh di proses sebelumny maka nilai ϕn dapat kita tetapkan sebagai berikut :
   ϕn = (p – 1) x (q – 1)

   = (17 – 1) x (31 -1)

   = 16 x 30

   = 480

2. Menentukan nilai e (tahap 4)

   Nilai e merupakan pasangan kunci publik n (e, n). Nilai e ini juga yang akan menentukan kekuatan dari sebuah *chipertext*. Untuk mencari nilai e akan digunakan prinsip faktor persekutuan terbesar. Hal ini untuk mencari nilai prim.a relatif.
   Faktor persekutuan terbesar adalah elemen terbesar pada himpunan *divisor* (himpunan bilangan integer yang membagi habis sebuah bilangan integer) dua bilangan integer [2]. Misalnya 18 memiliki *divisor* {1, 2, 3, 6, 9,18} dan 12
   memiliki *divisor* {1, 2, 3, 4, 6, 12} maka himpunan *divisor* bersamanya adalah:

   {1, 2, 3, 6} dan yang terbesar adalah 6. Dinotasikan sebagai gcd(18, 12) = 6.

   Dalam tahapan ke-4 algoritma pembangkit kunci RSA dituliskan bahwa "$e \leftarrow Z_{\phi(n)}$ *dengan* $gcd(\phi(n), e) = 1$". Hal ini berarti bahwa untuk setiap nilai e dan ϕ(n) akan memiliki faktor persekutuan terbesar 1. Untuk kondisi seperti gcd(ϕ(n), e) =1 biasa disebut juga dengan "Prima Relatif".
   Untuk memenuhi kondisi tahapan tersebut maka bisa gunakan cara sederhana yaitu menggunakan bilangan prima untuk mengisi nilai *e*. Bilangan prima yang digunakan haruslah memiliki nilai yang bukan

merupakan divisor dari nilai ϕ(n). Sebagai contoh :

Diberikan suatu nilai untuk P = 17 dan Q = 31. Maka kita bisa saja mengambil nilai prima sembarang untuk nilai e. Jika P =17 dan Q = 31 maka nilai ϕ(n) sebagai berikut :

ϕ(n) = (p – 1) x (q – 1)

= (17 – 1) x (31 – 1)

= 16 x 30

= 480

Maka langkah selanjutnya adalah menentukan nilai e, nilai e akan diberikan adalah bilangan prima yang bukan merupakan *divisor* dari 480. Penggunaan bilangan prima ini tidaklah mutlak karena bisa saja kita menggunakan bukan bilangan prima selama memenuhi syarat gcd(ϕ(n), e) = 1. Sebagai contoh bisa saja kita menggunakan nilai 77. Bisa kita lihat nilai gcd(480, 77) dengan algoritma *Euclid* yang bersifat iteratif [2] seperti berikut ini dapat dilihat pada tabel 5.2:

Tabel 5. 2 Algoritma Euclid Iteratif

	Algoritma Euclid Iteratif
1	A ← e
2	B ← ϕ(n)
3	**While** B > 0 **do**
4	Q ← A/B
5	R ← A – Q * B
6	A ← B
7	B ← R
8	**Endwhile**
9	**Return** A

Tabel 5. 3 Contoh gcd

A	B	Q = A / B	R = A mod B
480	77	6	18
77	18	4	5
18	5	3	3
5	3	1	2
3	2	1	1

A	B	Q = A / B	R = A mod B
2	1	2	0
1	0		

Dari hasil diatas terlihat bahwa gcd (480, 77) = 1 hal ini sesuai dengan syarat gcd(ϕ(n), e) = 1. Namun penggunaan bilangan acak ini akan memberikan beban tersendiri terhadap memori karena harus melakukan pengujian terhadap nilai "*prima relatif*" yang belum tentu bernilai benar. Sedangkan dengan menggunakan bilangan prima kita hanya akan melakukan pengujian nilai dengan cara membagi nilai $\phi(n)$ dengan *e*. Hal ini bisa dilakukan karena bilangan prima hanya memiliki faktor pembagi bilangan itu sendiri dan 1, yang berarti bahwa yang mungkin membagi bilangan $\phi(n)$ hanya nilai *e* itu sendiri. Sebagai contoh kita akan menggunakan nilai *e* = 73 dengan nilai $\phi(n)$ tetap 480. Berikut terlihat di tabel berikut:

Tabel 5. 4 Contoh gcd

	B	Q = A /	R = A mod B
480	73	6	42
	42	1	31
	31	1	11
	11	2	9
	9	1	2
9	2	4	1
2	1	2	0
1	0		

Dari tabel diatas terlihat bahwa nilai untuk gcd (480, 73) = 1. Hal ini membuktikan bahwa nilai prima dapat digunakan untuk nilai dari *e*. Sehingga nilai *e* dalam aplikasi ini hanya bisa dimasukkan dengan nilai prima.

Nilai e dapat sembarang dengan syarat bahwa nilai e hanya memiliki faktor persekutuan terbesar dengan ϕn yaitu 1. Untuk mempercepat proses yang ada maka dapat kita asumsikan bahwa nilai e merupakan bilangan prima yang tidak habis membagi nilai ϕn. Dengan hal tersebut maka kita tentukan nilai e = 73. Dengan demikian sampe tahap ke- 4 ini telah kita dapatkan kunci publik yaitu dapat dilihat pada tabel 5.5:

Tabel 5. 5 Algoritma *Extend Euclid*

	Algoritma Extend Euclid
1	A ← e ; B ← b;
2	T1 ← 0 ; T2 ← 1;
3	**While** B > 0 **do**
4	Q ← A/B
5	R ← A – Q * B
6	A ← B ; B ← R
7	T ← T1 – Q * T2;
8	T1 ← T2 ; T2 ← T
9	**Endwhile**
10	**Return** t ← T1

Berikut contoh untuk mencari nilai *d* dengan nilai *e* = 73 dan *ϕ(n)* = 480 menggunakan algoritma *extend euclid*:

Tabel 5. 6 Contoh Mencari Nilai *Invers* (e^{-1})

A	B	Q	R	T1	T2	T
480	73	6	42	0	1	-6
73	42	1	31	1	-6	7
42	31	1	11	-6	7	-13
31	11	2	9	7	-13	33
11	9	1	2	-13	33	-46

9	2	4	1	33	-46	217
2	1	2	0	-46	217	-480
1	0			**217**	-480	

Dari hasil perhitungan diatas didapatkan nilai T1 = 217 dimana t = T1 maka t = 217. Nilai t merupakan nilai yang didapatkan dari 73^{-1} pada Z_{480} sehingga dapat disimpulkan bahwa nilai $d = t$ yang berarti d = 217. Untuk nilai *qInv* juga dilakukan dengan cara yang sama dengan mencari nilai *d*.

Dengan teorema *invers* didapatkan nilai d yaitu d = 217.

3. Mencari nilai dP dan dQ (tahap 6 dan 7)

   Nilai dP dan dQ merupakan bagian kunci yang digunakan untuk proses dekripsi (kunci privat). Untuk mencari nilai dP dan dQ cukup mengikuti rumus yang ada pada algoritma. Berikut adalah proses mencari dP dan dQ jika nilai yang digunakan adalah nilai yang telah ditetapkan pada contoh sebelumnya:

   dP = d mod (p - 1)
   = 217 mod (17 – 1)
   = 9

   dQ = d mod (q – 1)
   = 217 mod (31 – 1)
   = 7

4. Mencari nilai qInv (tahap 8)

   Nilai qInv sama dengan dP dan dQ karena merupakan bagian dari kunci private. Nilai ini akan digunakan pada proses dekripsi. Untuk mencari nilai qInv kita menggunakan teorema invers seperti yang ada pada pencarian nilai d. Dengan nilai yang telah ditetapkan pada contoh sebelumnya maka akan kita cari nila qInv sebagai berikut :

   qInv = q^{-1} pada Z_p

Dengan rumus yang digunakan maka akan kita dapatkan nilai qInv = 11. Sehingga sampai tahapan ini kita telah mendapatkan kunci privat K_{privat} = (dP, dQ, qInv, P, Q) = (9, 7, 11, 17, 31).

Semua tahapan proses pembangkit kunci diatas menghasilkan kunci sebagai berikut:

Kunci publik : (73, 527)

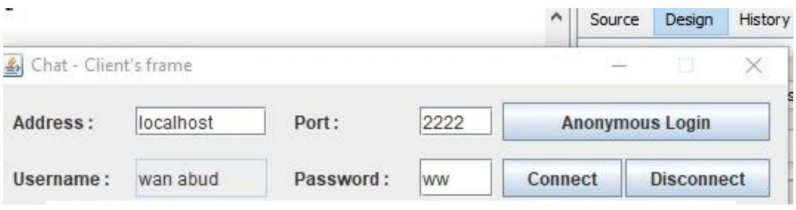
Kunci privat : (9, 7, 11, 17, 31)

5.1.3 Implementasi Sistem

Suatu aplikasi dibuat dengan melalui tahapan-tahapan salah satunya tahapan perancangan aplikasi, jika tahapan perancangan telah disusun maka pada bab ini akan dibahas tentang proses dalam tahapan implementasi, dimana merealisasikan perancangan yang telah dibuat menjadi nyata. Pada bagian bab ini meliputi implementasi pembuatan aplikasi. Tahap selanjutnya Desain dan Implementasi Algoritma Kriptografi RSA pada Telkom Bojonegoro untuk Meningkatkan Keamanan Sistem Jaringan ini menggunakan perangkat lunak yaitu *Java Netbeans*. Pembuatan aplikasi dan komponen-komponen pendukung lainnya. Aplikasi tersebut adalah sebagai berikut:

3.1.3.1 Implementasi *Form Login*

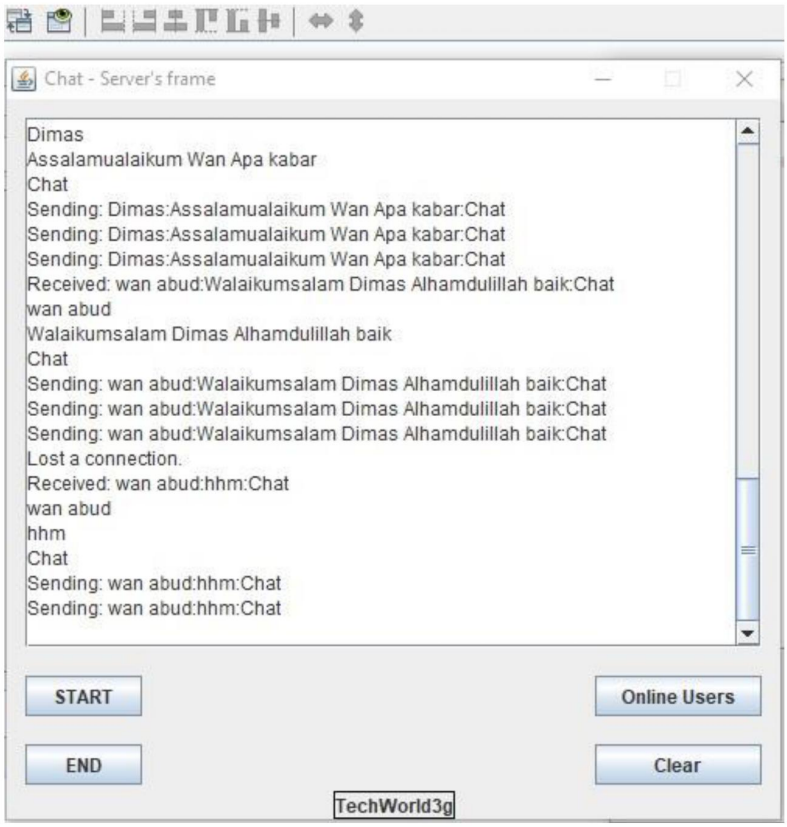
Pada tampilan form login dari aplikasi Desain dan Implementasi Algoritma Kriptografi RSA pada Telkom Bojonegoro terdapat Address dan username yang di inputkan oleh Client, kemudian button Login apabila client berhasil login pada sistem. Tampilan menu *Form Login* dapat dilihat pada gambar 5.2:



Gambar 5. 2 *Form Login*

3.1.3.2 Implementasi Menu *Server*

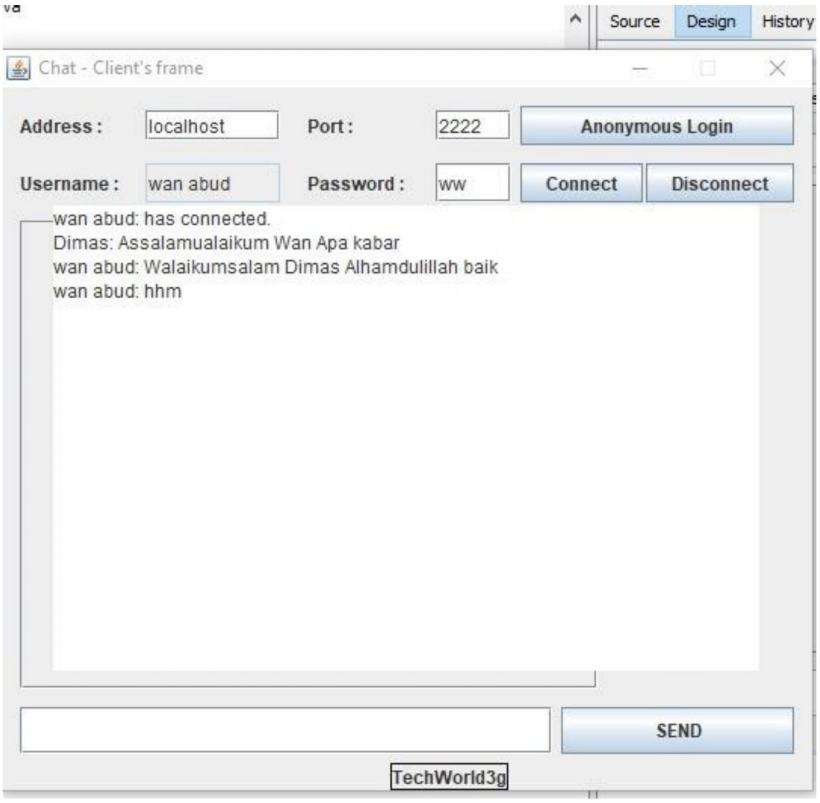
Pada tampilan menu *server* terdapat agar jaringan dapat terkoneksi pada sistem alikasi yang telah di buat maka *server* harus di aktifkan terlebih dahulu agar antar *client* bisa berkomunikasi. Sehingga mampu saling mengirim data yang mana data tersebut sudah terenkripsi. Tampilan menu *server* dapat dilihat pada gambar 5.3:



Gambar 5. 3 Menu *Server*

3.1.3.3 Implementasi Menu *Client* 1

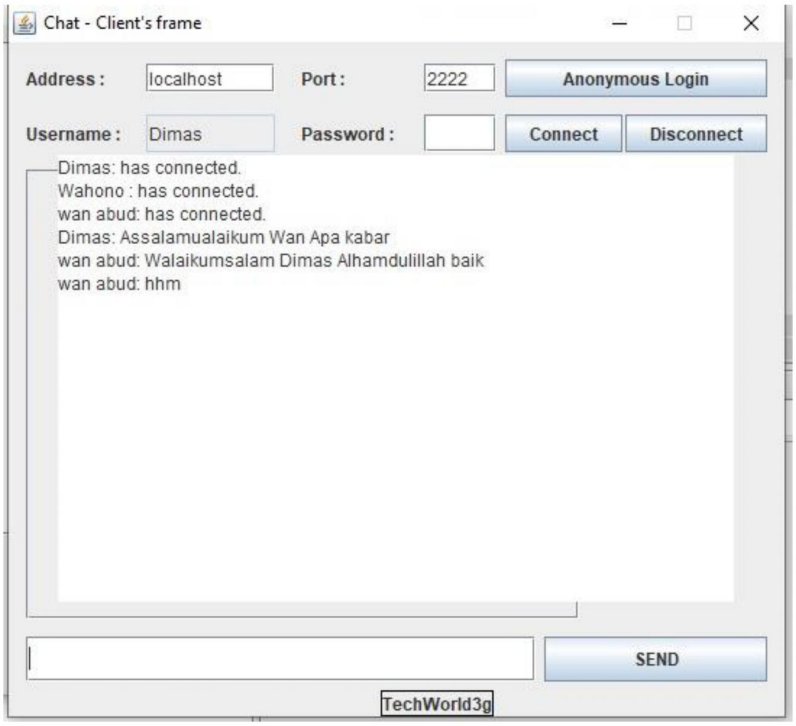
Pada tampilan menu *Client* 1 pengguna melakukan proses *chat* dengan aman, dengan menjalankan sistem kriptografi RSA untuk mengkomunikasikan pesan yang di kirim kepada *client* 2. Proses enkrisi tersebut dilakukan di balik layar sehingga memudahkan pengguna untuk melakukan proses yang ada pada sistem. Artinya *client* 1 tidak perlu meng*input*kan pasangan kunci secara manual kepada *client* 2 karena sistem telah melakukan *generate keypare* secara otomatis. Tampilan menu *client* 1 dapat dilihat pada gambar 5.4:



Gambar 5. 4 Menu *Client* 1

3.1.3.4 Implementasi Menu *Client* 2

Pada tampilan menu *Client* 2 berfungsi sebagai penerima pesan yang telah di kirmkan *client* 1. *Client* 2 menjalankan sistem untuk mendekripsikan pesan yang di terima dengan berbentuk *cyeprteks*. Sehingga penerima dapat melihat pesan asli yang dikirim oleh pengirim yaitu *client* 1 dengan bermodalkan kunci privat yang telah dimiliki. Tampilan menu *client* 2 dapat dilihat pada gam*bar* 5.5:



Gambar 5. 5 Menu *Client* 2

5.2 Pengujian

5.2.1 Pengujian Algoritma pada Plainteks pada Bilangan Kecil

Setelah sistem selesai dibangun maka harus diuji apakah sistem dapat berjalan dengan baik dan mudah dioperasikan dari performasi waktu dekrip dan enkrip. Pengujian dilakukan dengan melakukan langkah–langkah:

1. Mengisi kunci pengaman dengan membangkitkan kunci random dan mengisi kata kunci.
2. Menuliskan dokumen yang akan disimpan pada menu *Encrypt*.
3. Menuliskan hasil enkripsi untuk diubah menjadi dokumen semula pada menu Dekripsi.

p = 3

q = 7

n = p.q

= 3 x 7

= 21

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

= (3 -1)(7 – 1)

= 12

e * d mod 12 = 1

e = 5

d = 17

public key = (e,n) = (5,21)

private key = (d,n) = (17,21)

$C = M^e \bmod n$ (fungsi enkripsi)

$M = C^d \bmod n$ (fungsi dekripsi)

Dengan e= 5 dan d = 21

T	O	M	I
84	79	77	73

Tabel 5. 7 Proses Enkripsi dan Dekripsi T = 84

T = 84

Enkripsi	Dekripsi
C = M ^ e mod n	M = C ^ d mod n
8 ^ 5 mod 21 = 8	8 ^ 17 mod 21 = 8
4 ^ 5 mod 21 = 16	16 ^ 17 mod 21 = 4

Tabel 5. 8 Proses Enkripsi dan Dekripsi O = 79

O = 79

Enkripsi	Dekripsi
C = M ^ e mod n	M = C ^ d mod n
7 ^ 5 mod 21 = 7	7 ^ 17 mod 21 = 7
9 ^ 5 mod 21 = 18	18 ^ 17 mod 21 = 9

Tabel 5. 9 Proses Enkripsi dan Dekripsi M = 77

M = 77

Enkripsi	Dekripsi
C = M ^ e mod n	M = C ^ d mod n
7 ^ 5 mod 21 = 7	7 ^ 17 mod 21 = 7
7 ^ 5 mod 21 = 7	7 ^ 5 mod 21 = 7

Tabel 5. 10 Proses Enkripsi dan Dekripsi I = 73

I = 73

Enkripsi	Dekripsi
C = M ^ e mod n	M = C ^ d mod n
7 ^ 5 mod 21 = 7	7 ^ 17 mod 21 = 7
3 ^ 5 mod 21 = 12	12 ^ 17 mod 21 = 3

5.2.2 Pengujian Bilangan Prima

Dalam aplikasi ini diperlukan pengujian nilai bilangan prima, hal ini untuk memenuhi tahap 2 dan 3 *flowchart* 3.4. Berikut untuk melakukan pengujian bilangan prima:

Tabel 5. 11 Algoritma Pengujian Bilangan Prima

	Algoritma Uji Bilangan Prima
1	BP ← Bilangan Prima.
2	**If** BP <= 1 **Then**
3	**Output** (“Bukan Bilangan Prima”)
4	**Else**
5	**For** i=2 to BP-1 **do**
6	**If** (BP mod i = 0) **then**
7	**Output** (“Bukan Bilangan Prima”)
8	**Else**
9	**Output** (“Bilangan Prima”)
10	**Endif**
11	**Endfor**
12	**Endif**

5.2.3 Pengujian Sistem

5.2.3.1 Pengujian Alpha

Metode pengujian *alpha* yaitu dilakukan untuk menguji sistem yang telah dibangun. Metode yang digunakan dalam pengujian ini adalah pengujian *blackbox* dimana pengujian ini berfokus pada persyaratan fungsional dari sistem yang telah dibangun.

a. Pengujian *Login*

Berikut ini merupakan pengujian *login* pada Desain dan Implementasi Algoritma Kriptografi RSA pada Telkom Bojonegoro untuk Meningkatkan Keamanan Sistem Jaringan. Dapat di lihat pada tabel 5.12:

Tabel 5. 12 Pengujian *Login*

No.	Data Masukan	Hasil yang diharapkan	Hasil pengujian	Keterangan
1.	*Input username* dan *password* secara benar	Ketika data *login* dimasukkan dan tombol *Anonymouse Login* diklik maka dilakukan proses validasi data *login*. Apabila *valid* maka pengguna bisa mengakses halaman utama.	Pengguna dapat *login* ke dalam sistem dan mengakses menu utama	**Berhasil** / Tidak
2.	Input *username* dan *password* yang salah	Sistem menampilkan pesan kesalahan karena data tidak *valid* "*Username* dan *Password* tidak benar"	Sistem menampilkan pesan kesalahan "*Username* dan *Password* salah"	**Berhasil** / Tidak

b. Pengujian *Generte Key*

Berikut ini merupakan pengujian *Generte Key* pada Desain dan Implementasi Algoritma Kriptografi RSA pada Telkom Bojonegoro untuk Meningkatkan Keamanan Sistem Jaringan. Dapat di lihat pada tabel 5.13:

Tabel 5. 13 Pengujian *Generte Key*

No.	Data Masukan	Hasil yang diharapkan	Hasil pengujian	Keterangan
1.	Mengisi nilai p dan nilai q	Setelah nilai p dan q dimasukkan dan tombol lihat hasil ditekan. Maka sistem akan menampilkan hasil berupa kunci privat dan publik.	Sistem menampilkan kunci privat dan publik	**Berhasil** / Tidak
2.	Pengguna dapat menyimpan kunci	Sistem menyimpan kunci ke Server	System menyimpan kunci ke dalam server	**Berhasil** / Tidak

c. Pengujian *Encrypt*

Berikut ini merupak pengujian *Encrypt* pada Desain dan Implementasi Algoritma Kriptografi RSA an pada Telkom Bojonegoro untuk Meningkatkan Keamanan Sistem Jaringan. Dapat di lihat pada tabel 5.14:

Tabel 5. 14 Pengujian *Encrypt*

No.	Data Masukan	Hasil yang diharapkan	Hasil pengujian	Keterangan
1.	Pengguna memilih tujuan pengiriman memasukkan kunci publik (nilai n dan nilai e) dan menekan tombol kirim	Sistem menyimpan data ke server	Sistem menyimpan pesan ke server dan mengirim kepada penerima.	**Berhasil** / Tidak

d. Pengujian *Decrypt*

Berikut ini merupakan pengujian *Decrypt* pada Desain dan Implementasi Algoritma Kriptografi RSA pada Telkom Bojonegoro untuk Meningkatkan Keamanan Sistem Jaringan. Dapat di lihat pada tabel 5.15:

Tabel 5. 15 Pengujian *Decrypt*

No.	Data Masukan	Hasil yang diharapkan	Hasil pengujian	Keterangan
1.	Pengguna memilih pesan dan memasukkan kunci privat (nilai p, q dan nilai d)	Sistem akan menampilkan pesan yang sudah didekripsi	Sistem menampilkan pesan yang sudah didekripsi	**Berhasil** / Tidak

5.2.3.2 Pengujian *Betha*

Pengujian *betha* merupakan pengujian yang dilakukan dengan tujuan untuk mengetahui bagaimana kualitas sistem yang telah dibuat. Dalam pengujian *betha* dilakukan terhadap responden atau calon pengguna sistem dengan menggunakan kuesioner atau angket. Kuesioner dibuat menggunakan skala jawaban 1 sampai 5. Daftar pertanyaan kuesioner akan dijelaskan pada Tabel 5.16:

Tabel 5. 16 Daftar Pertanyaan Pada Kuesioner

No	URAIAN	SKORE				
		1	2	3	4	5
1	Tampilan sistem nyaman dan sesuai keinginan					
2	Kesesuaian nama tombol dan layanan					
3	Mudah memahami sistem dan kegunannya					
4	Keamanan sistem teruji dengan baik					
5	Hasil dekrip sesuai dengan *plaintext*					
6	Apakah dengan adanya sistem ini dapat membantu perusahaan anda?					

Keterangan :

5 : Sangat Setuju
4 : Setuju
3 : Cukup Setuju
2 : Tidak Setuju
1 : Sangat Tidak Setuju

5.2.3.3 Pengujian Metode RSA

Dalam pengujian metode *(Rivest-Shamir-Adleman)* RSA pada aplikasi Desain dan Implementasi Algoritma Kriptografi RSA pada Telkom Bojonegoro untuk Meningkatkan Keamanan Sistem Jaringan ini berjalan sesuai yang diharapkan, dengan itu dilakukan pengujian dengan cara membandingkan *plaintext* dengan hasil dekrip yang ada pada aplikasi. Perbandingan *plaintext* dengan hasil dekrip menggunakan algoritma kriptografi RSA dapat dilihat pada gam*bar* 5.17:

Tabel 5. 17 Pengujian Metode RSA

Original *message*	Dimas Wahono
Karakter Pesan	11
plaintext	68737765833287657279787 9
Nilai p	12879080168749053604050825711099984777829235310347429231391828444482813927020347822534110041411378846219632123022330078854084487599970282427108015095432601
Nilai q	81988512529072384621504701583791162847379135850763706617163507896914499328383721547888255069415339607967411152652201804400453961941183310445198134328453 01
Nilai n	10559366257784094630159573374154953436342445381997767228952303978068987271210865381879510087463302051555157972295776860105809299964364252481694760412923701163278965609825731723969325694681493564639669184142189969657338153150762389118560239239711896952598068963706950398206463925511768017534690411868090505790 1
Encrypted	65040252233297037479598591506638714111407889445590394031258741202088639681185977023902098257307126684658888742120924717227443809395334857476068019345440962847565469022681277483964387909128850493879427896213774581318657852417106922954451460876700592819974050303146150584358196511638316704137490806875271919100
Decrypted	68737765833287657279787 9
Ukuran *File* Hasil Enkrip	308 *bytes*
Ukuran *File* Hasil Dekrip	12 *bytes*
Original *message*	Tommy
Karakter Pesan	5
plaintext	8479777789
Nilai p	95234784676314382280703756496136815498625324859391996626295723046550472561955085722081551695926030281420745486570033691589218505630979327633577864373327 09

Nilai q	1340727956660866861233431970219152994838561219998033948697560365500750948215005177301441757235223744420538370786099599928783661707979107388646659047517762 1
Nilai n	1276839382621126165978881959759788138360689343454718352254128886839203892759601772722904601707467704378641994023955846280084055695529977576621600067363129243133299428823666238802195312360130507743155099086711975117148375833985362819571813819075961029686051929900216399120534930185170826676411738101322481052 89
Encrypted	4423387306703766300713140641652179362422222885277489498244369043557478557567848463796266986197004927299262888448088529299525994167417231714287123896934674069120598397461030845891828104568628359144921070148256776412293012234697079085720116476904211538766171164068950200126906469247234555644612329036214121189
Decrypted	8479777789
Ukuran *File* Hasil Enkrip	307 *bytes*
Ukuran *File* Hasil Dekrip	5 *bytes*
Original *message*	Bojonegoro
Karakter Pesan	10
plaintext	3266797479786971798279
Nilai p	8660626391192400210647665822696587948245285614630467880841519203482777445824373807721192499160276199100005160604349773594728900587592356508828637274045057
Nilai q	1264578701034610235830414114176433951947079806398947439257680383034903345605228136166023161738618609216431240455956961172585131034747552612043696160993259 1
Nilai n	1095204367192014962129122890910739568604292167103490218374072649481879910772665510498873183578566994569140522504718330361502505474408329900110207458257609231959872400172462176812534657893217899177439941936697823797851811919990592797761058712400096418839958556495547088651350058602015290578579497092556667526 87

Encrypted	477140639863046677557614280885257322774863126049308991438184138695 530868350159394441006822505252869741779629056649004497087460131567 555126113987747294470871127180074985152611631750470591565785370462 340724686410901965136065379115484389346340572438401840222543145404 361946181481855415056989419136901893279871 60
Decrypted	3266797479786971798279
Ukuran *File* Hasil Enkrip	308 *bytes*
Ukuran *File* Hasil Dekrip	11 *bytes*