Jurnal Penelitian Ilmu Komputer, System Embedded & Logic
p-ISSN: 2303-3304, e-ISSN: 2620-3553
7 (1): 43 – 54 (Maret 2019)

Perangkat Lunak Kriptografi Algoritma *Rivest Shamir Adleman* (RSA) Untuk Mengenkripsi Data *Password* Pada Aplikasi Login

Sugiyatno [1,*], Prima Dina Atika [1]

[1] Teknik Informatika; Universitas Bhayangkara Jakarta Raya; Jl. Raya Perjuangan, Marga Mulya, Bekasi Utara, Marga Mulya, Bekasi Utara, Kota Bks, Jawa Barat 17121 (021) 88955882; e-mail: sugiyatno@dsn.ubharajaya.ac.id; e-mail: primadina@dsn.ubharajaya.ac.id

* Korespondensi: e-mail: sugiyatno@dsn.ubharajaya.ac.id

Diterima: 8 Februari 2019; Review: 22 Februari 2019; Disetujui: 8 Maret 2019

Abstract

Nowadays, information is a basic need for everyone, but the information obtained is not guaranteed right. One solution to data security techniques is cryptography. With cryptography, messages or documents are safe and cannot be read by unauthorized parties. This article discusses about the implementation of the Rivest Shamir Adleman (RSA) Algorithm software in the login functiion. This function has two menus, namely login and registration. The purpose of this paper is to encrypt password data that goes into the database, so that data is protected from abuse by irresponsible parties. The encryption process occurs when a user saves data during registration. The workings of the encryption process are as follows, first take the recipient's public key, e, and modulus n, then declare plaintext m to be blocks m1, m2, m3, ... so that each block represents the value in interval [0, n- 1], then each block m, encrypted into ci blocks with formulas (ci = mic mod n). Each ciphertext c block is decrypted back into a mi block with a formula (mi = cid mod n).

***Keywords*:** *Enkripsi, Rivest Shamir Adleman (RSA), Plainteks, Cipherteks*

Abstrak

Saat ini informasi menjadi kebutuhan pokok setiap orang, namun informasi yang didapatkan belum tentu terjamin. Salah satu solusi teknik pengamanan data adalah menggunakan kriptografi. Dengan kriptografi, pesan atau dokumen aman dan tidak dapat dibaca oleh pihak yang tidak berhak Artikel ini akan membahas implementasi perangkat lunak Algoritma *Rivest Shamir Adleman* (RSA) dalam fungsi login. Fungsi ini memiliki dua menu yaitu login dan registrasi. Proses enkripsi terjadi pada saat *user* menyimpan data ketika registrasi. Tujuan dari penulisan ini untuk mengenkripsi data *password* yang masuk ke dalam *database,* sehingga data terhindar dari penyalahgunaan oleh pihak yang tidak bertanggung jawab. Cara kerja proses enkripsi sebagai berikut, pertama mengambil kunci publik penerima pesan, *e,* dan modulus n, lalu nyatakan *plainteks* m menjadi blok-blok m1, m2, m3,…, sedemikian sehingga setiap blok merepresentasikan nilai di dalam selang [0, n-1], kemudian setiap blok m, dienkripsi menjadi blok c_i dengan rumus ($c_i=m_i^c$ mod n). Setiap blok *cipherteks* c, didekripsi kembali menjadi blok m_i dengan rumus ($m_i=c_i^d$ mod n).

Kata kunci: Enkripsi, *Rivest Shamir Adleman (RSA), Plainteks, Cipherteks*

1. Pendahuluan

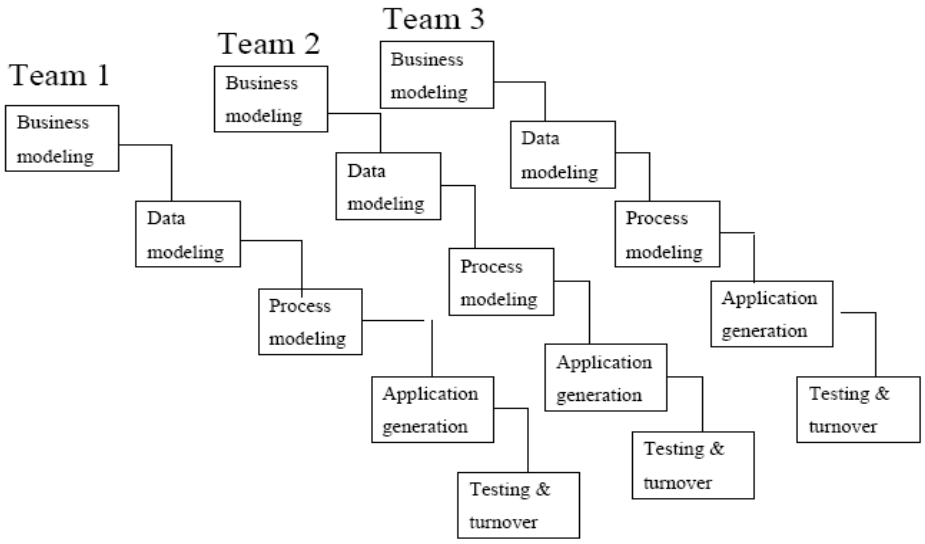
Teknik Enkripsi merupakan diperlukan untuk mengkodekan data (*plainteks*) sehingga keamanan data dijamin keamanan informasinya [Rahardjo, 2002], dan tidak dapat baca tanpa

(*cipherteks*). *Encryption* berasal dari bahasa Yunani *Krypstos* yang artinya tersembunyi atau rahasia. Ketika pengguna aplikasi melakukan *login,* maka pengguna aplikasi memasukkan identitas dari akun pengguna dan password untuk mendapatkan hak akses sumber daya komputer. Dalam aplikasi *login* pada umumnya data *password* yang masuk ke dalam *database* akan tampil sama dengan apa yang di input oleh *user.* Hal ini menjadi sebuah kelemahan yang akan diincar oleh seorang *hacker* untuk mencuri data *password* dari sebuah *database* penampung data *password user* tersebut.

Ada banyak metode-metode enkripsi yang ada, salah satunya adalah metode enkripsi *Rivest Shamir Adleman* (RSA). Algoritma RSA dibuat oleh tiga orang peneliti dari MIT (*Massachussets Institute of Technology*) pada tahun 1976, yaitu Ron Rivest, Adi Shamir, dan Leonard Adleman. Kekuatan algoritma RSA terletak pada ketahanannya terhadap berbagai bentuk serangan, terutama serangan *brute force*. Hal ini dikarenakan kompleksitas dekripsinya yang dapat ditentukan secara dinamis dengan cara menentukan nilai *p* dan *q* yang besar pada saat proses pembangkitan pasangan kunci, sehingga dihasilkan sebuah *key space* yang cukup besar, sehingga tahan terhadap serangan.

2. Metode Penelitian

Prosedur penelitian ini dilakukan dengan menggunakan metode *Rapid Application Development* (RAD). RAD merupakan model proses perkembangan perangkat lunak sekuensial linier yang menekankan siklus perkembangan yang sangat pendek [Martin J. , 1991].



Sumber: [Kerr and Hunter, 1993]

Gambar 1. Model RAD

Tahapan Model Rapid Application Development (RAD)

Pertama Pemodelan Bisnis; Fungsi bisnis dimodelkan untuk menjawab pertanyaan sebagai berikut: Informasi apa yang mengendalikan proses bisnis? Informasi apa yang dimunculkan? Siapa yang memunculkannya? Kemana informasi itu pergi? Siapa yang memprosesnya?

Kedua Pemodelan Data; Didefinisikan sebagai bagian dari fase *business modelling* disaring kedalam serangkaian objek data yang dibutuhkan untuk mendukung bisnis.

Ketiga Pemodelan Proses; Fase data *modelling* ditransformasikan untuk mencapai aliran informasi untuk implementasi sebuah fungsi bisnis dengan menciptakan proses menambah, memodifikasi, menghapus atau mendapatkan kembali sebuah objek data.

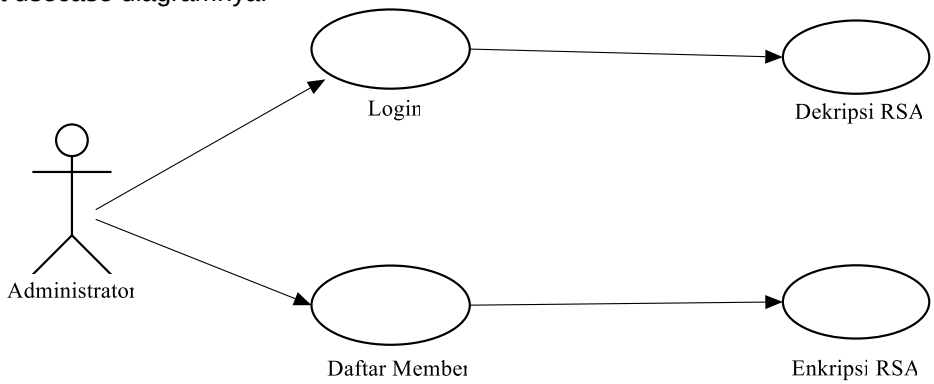
Keempat Pembentukan Aplikasi; RAD mengasumsikan pemakaian teknik geenerasi keempat. Selain menciptakan perangkat lunak dengan menggunakan bahasa pemrograman generasi ketiga yang konvensional, RAD lebih banyak memproses kerja untuk memakai lagi komponen program yang ada atau diciptakan komponenvvn yang bisa dipakai lagi.

Kelima Pengujian dan *Turnover; k*arena proses RAD menekankan pada pemakaian kembali, banyak komponen program telah diuji, hal ini mengurangi keseluruhan waktu penguji, tetapi komponen baru tersebut harus diuji dan semua *interface* harus dilatih secara penuh.

2.1. Fase Perancangan

a. *Usecase Diagram*

Berikut usecase diagramnya.



Sumber: Hasil Penelitian (2018)

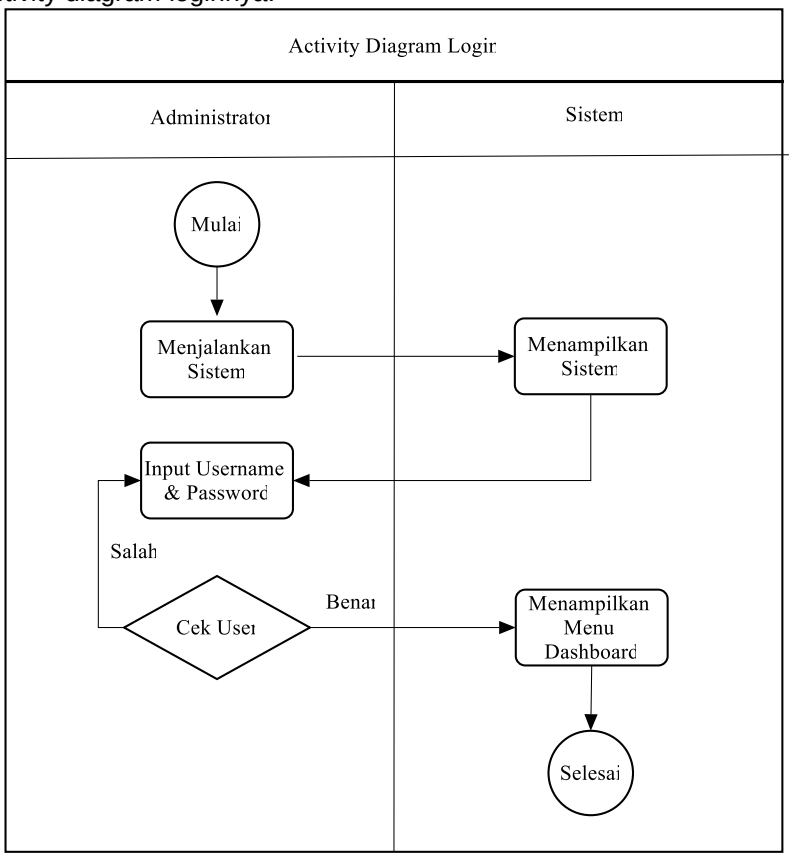
Gambar 2. *Use Case Diagram*

b. *Activity Diagram*

Activity Diagram menggambarkan aktifitas–aktifitas yang terjadi dalam sistem.

1. Activity Diagram Login

Berikut activity diagram loginnya.



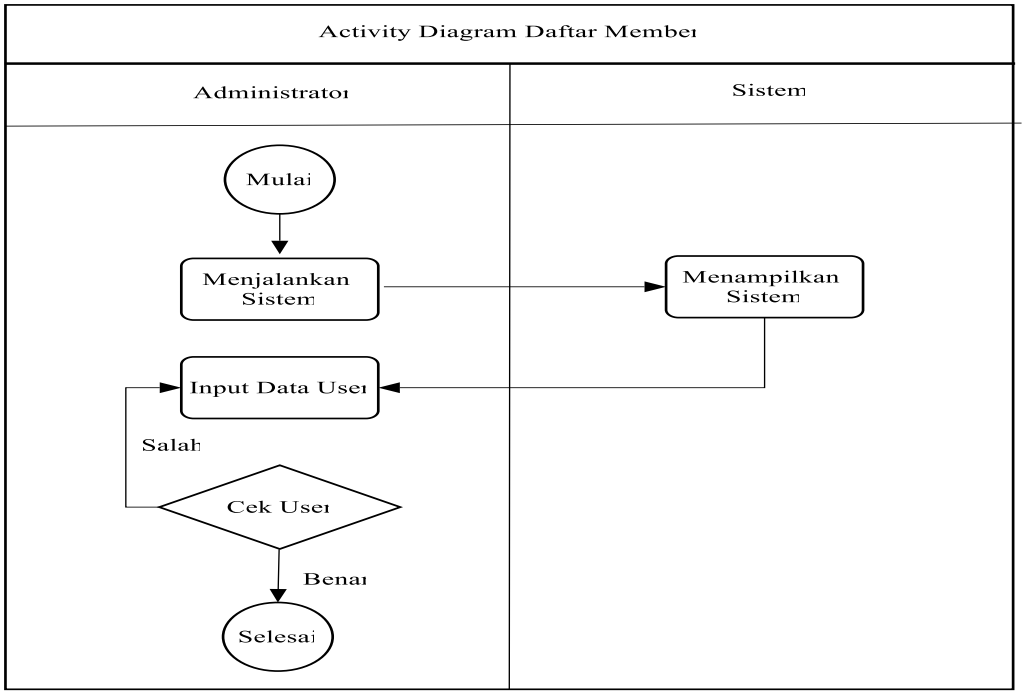
Sumber: Hasil Penelitian (2018)

Gambar 3. *Activity Diagram Login*

User Administrator dapat mengakses sistem dengan memasukkan username dan password Administrator

2. Activity Diagram Daftar Member

Berikut activity diagram daftar member.

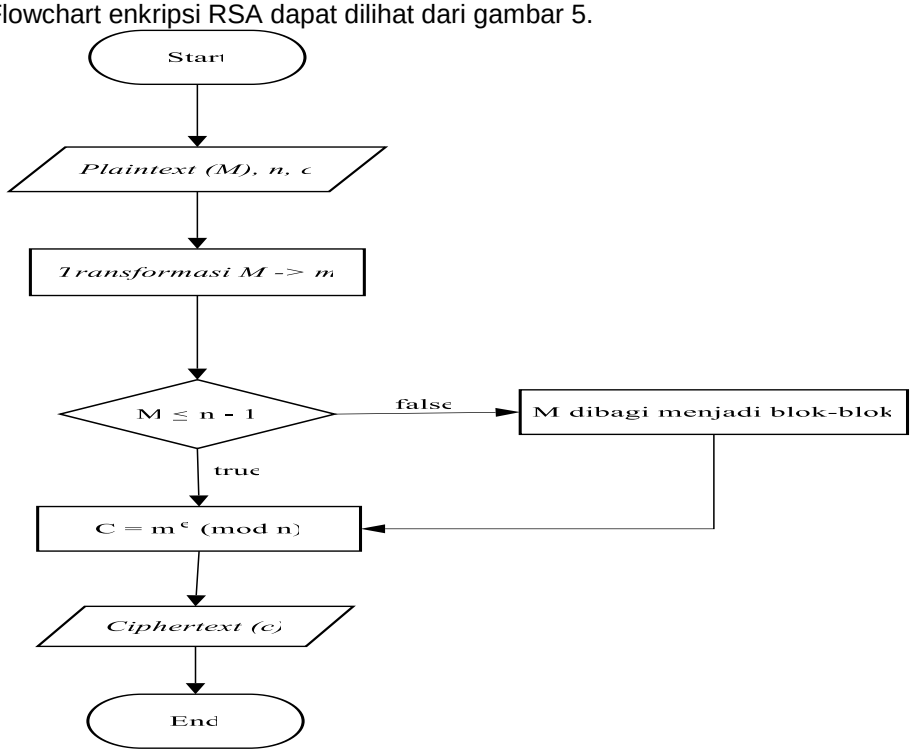


Sumber: Hasil Penelitian (2018)

Gambar 4. *Activity Diagram* Data Member

a. Flowchart Enkripsi RSA

Flowchart enkripsi RSA dapat dilihat dari gambar 5.



Sumber: Hasil Penelitian (2018)

Gambar 5. *Flowchart Enkripsi RSA*

Dari flowchart gambar 5 dapat dijelaskan bahwa di mulai dengan menyusun plaintext menjadi blok-blok m1, m2, …, mn, Hitung ciphertext ci dengan rumus : Ci = M^emod n, dan menghasilkan ciphertext.

3. Hasil dan Pembahasan

3.1. Perhitungan Algoritma Enkripsi RSA pada aplikasi Login RSA

Perhitungan algoritma RSA terjadi pada saat user melakukan proses registrasi. Berikut adalah salah satu proses dan perhitungan enkripsi dan deksripsi pada aplikasi Login RSA.

1. Perhitungan Kunci Enkripsi RSA

Kunci enkripsi (e) = 5
faktor persekutuan terbesar (e, Q(n)) = 1
Q(n) = (p-1) x (q-1)
p = misalkan 113
q = misalkan 157

maka :
Q(n) = (113-1) x (157-1) = 17472
faktor persekutuan terbesar (e, 17472) = 1

Misalkan e = 5
Faktor dari 5 adalah : (1,5)
Sedangkan 17472 adalah : (1,2,3,5,6,.....)
FPB (e, (Q(n))) = 1
Maka Nilai e = 5 bisa digunakan

2. Perhitungan Kunci Dekripsi RSA

Kunci Deskripsi (d) >>> e = 5 & Q(n) = 17472
e * d = 1 Mod (Q(n))
artinya
5 * d = 1 Mod (17472)
Misalkan d = 6989
5 * 6989 = 1 Mod (17472)
34945 Mod (17472) = 1
Karena e * d = 1 Mod (Q(n)),
Maka Nilai d = 6989 Bisa digunakan

3. Proses Enkripsi (Algoritma Enkripsi RSA)

a) Konvert huruf menjadi Kode ASCII (desimal)
Contoh : admin123
a = 97
d = 100
m = 109
i = 105
n = 110
1 = 49
2 = 50
3 = 51

b) C = Pe Mod n
n = p x q
n = 113 x 157
= 17741

C = P pangkat e Mod 17741 >>>> e = 50
a = 97 pangkat 5 Mod 17741 = 8587340257 mod 17741 = 4358
d = 100 pangkat 5 Mod 17741 = 10000000000 mod 17741 = 494
m =109 pangkat 5 Mod 17741 = 15386239549 mod 17741 = 2479
i = 105 pangkat 5 Mod 17741 = 12762815625 mod 17741 = 11189
n = 110 pangkat 5 Mod 17741 = 16105100000 mod 17741 = 15351
1 = 49 pangkat 5 Mod 17741 = 282475249 mod 17741 = 3047
2 = 50 pangkat 5 Mod 17741 = 312500000 mod 17741 = 10026
3 = 51 pangkat 5 Mod 17741 = 345025251 mod 17741 = 16024

hasil enkripsi= 4358 + 1494 + 2479 + 11189 + 15351 + 3047 + 10026 + 16024
a d m i n 1 2 3

hasil di database = 4358 1494 2479 11189 15351 3047 10026 16024 //cocok

4. Proses Deskripsi (Algoritma Deskripsi)

a) P = C pangkat d Mod n >>>> d = 6989
P1 = 4358 pangkat 6989 Mod 17741 = 97
P2 = 1494 pangkat 6989 Mod 17741 = 100
P3 = 2479 pangkat 6989 Mod 17741 = 109
P4 = 11189 pangkat 6989 Mod 17741 = 105
P5 = 15351 pangkat 6989 Mod 17741 = 110
P6 = 3047 pangkat 6989 Mod 17741 = 49
P7 = 10026 pangkat 6989 Mod 17741 = 50
P8 = 16024 pangkat 6989 Mod 17741 = 51

b) Konvert Kode ASCII Menjadi Teks
P1 = 97 Konvert ke ASCII Teks= a
P2 = 100 Konvert ke ASCII Teks = d
P3 = 109 Konvert ke ASCII Teks = m
P4 = 105 Konvert ke ASCII Teks = i
P5 = 110 Konvert ke ASCII Teks = n
P6 = 49 Konvert ke ASCII Teks = 1
P7 = 50 Konvert ke ASCII Teks = 2
P8 = 51 Konvert ke ASCII Teks = 3

3.2. Pengujian Sistem

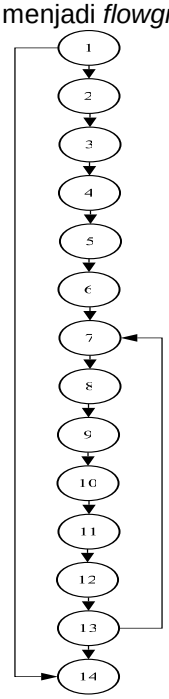
1. Pengujian Enkripsi

Pengujian Enkripsi berikut dimulai dengan merubah *pseudocode* menjadi *flowgraph*.

a) Source code Enkripsi

```
    // Proses Enkripsi
function enkripsi($plain, $n, $e)
{
    //Pesan dikodekan menjadi kode ASCII, Kemudian di Enkripsi Per Karakter
$hasil='';
for($i=0; $i<strlen($plain); ++$i)
{
    //Rumus Enkripsi ---> ChiperTeks = <pesan>^<e>mod<n>
$hasil.=gmp_strval(gmp_mod(gmp_pow(ord($plain[$i]),$e),$n));
    //Antar Tiap Karakter dipisahkan dengan "+"
if($i!=strlen($plain)-1){
$hasil.=' '; //spasi untuk tiap2 karakter
}
}
return $hasil;
```

b) Tahap perubahan *source code* menjadi *flowgraph*
Berikut tahap *source code* pada tabel 1 menjadi *flowgraph* seperti pada gambar 6.



Sumber: Hasil Penelitian (2018)

Gambar 6. *Flowgraph* Enkripsi

c) *Cyclomatic Complexity*
Diketahui : Edge = Jumlah busur panah pada *flowgraph*, yaitu 15
Node = Jumlah simpul pada *flowgraph,* yaitu 14
$V_{(G)}$ = Edge – Node + 2
= 15 – 14 + 2
= 1 + 2
= 3
Jadi, jalur bebas (*independent path*) pada *flowgraph* enkripsi yang akan diuji adalah sebanyak 3 jalur bebas

d) *Independent Path*
Path 1 = 1-14
Path 2 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14
Path 3 = 1-2-3-4-5-6-7-8-9-10-11-12-13-7-8-9-10-11-12-13-14

e) *Graph Matriks*
Graph matriks dapat dijelaskan dalam Tabel 1.

Tabel 1. *Graph Matriks* Enkripsi

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Sum
1		1												1	1
2			1												0
3				1											0
4					1										0
5						1									0
6							1								0
7								1							0
8									1						0
9										1					0
10											1				0
11												1			0
12													1		0
13							1							1	1
14															0
						Sum + 1									2+1=3

Sumber: Hasil Penelitian (2018)

f) *Predicate Node*
$V_{(G)}$ = Jumlah node yang memiliki lebih dari 1 jalur + 1
= 2 + 1
= 3

g) Tabel Hasil Pengujian

Tabel 2**.** Tabel Hasil Pengujian Enkripsi

No	Path	Yang Diharapkan	Hasil Pengujian	Kesimpulan
1	1-14	Berhasil melakukan enkripsi password	Berhasil melakukan enkripsi password	[√] Diterima [ ] Ditolak
2	1-2-3-4-5-6-7-8-9-10-11-12-13-14	Berhasil melakukan tahap demi tahap enkripsi	Berhasil melakukan tahap demi tahap enkripsi	[√] Diterima [ ] Ditolak
3	1-2-3-4-5-6-7-8-9-10-11-12-13-7-8-9-10-11-12-13-14	Jika nilai $hasil tidak sesuai, maka kembali ke langkah sebelumnya yaitu menghitung nilai $hasil	Berhasil menghitung nilai $hasil	[√] Diterima [ ] Ditolak

Sumber: Hasil Penelitian (2018)

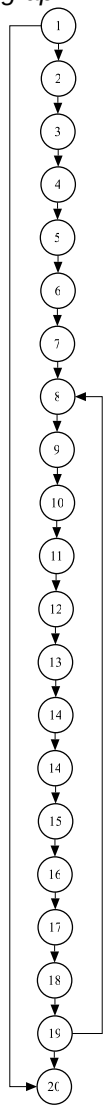
2. Pengujian Dekripsi

Pengujian *Decryption* berikut dimulai dengan merubah *source code* menjadi *flowgraph*.

a) *Source Code* Dekripsi

```
// Proses Dekripsi
    function deskripsi($chiper, $d, $n){
    $time_start = microtime(true);
    //Pesan Enkripsi dipecah menjadi array dengan batasan "+"
    $hasildekrip = '';
    $hasil = '';
    $dekrip=explode(" ", $chiper);//spasi untuk tiap2 karakter
    foreach($dekrip as $nilai){
            //Rumus Deskripsi ---> PlainTeks = <enkripsi>^<d>mod<n>
    $gm1 = gmp_pow($nilai,gmp_strval($d));
    $gm2 = gmp_mod($gm1,$n);
    $gm3 = gmp_strval($gm2);
    $hasildekrip.=chr($gm3);
    $time_end = microtime(true);
            // Waktu Eksekusi
    $time = $time_end - $time_start;
    $hasil = array($time, $hasildekrip);
    return $hasil;
    }
```

b) Tahap perubahan *source code* menjadi *flowgraph*



Sumber: Hasil Penelitian (2018)

Gambar 7. *Flowgraph* Dekripsi

c) *Cyclomatic Complexity*

Diketahui : Edge = Jumlah busur panah pada *flowgraph*, yaitu 21
Node = Jumlah simpul pada *flowgraph,* yaitu 20

$V_{(G)}$ = Edge – Node + 2
= 21 – 20 + 2
= 1 + 2
= 3

Jadi, jalur bebas (*independent path*) pada *flowgraph* dekripsi yang akan diuji adalah sebanyak 3 jalur bebas

d) *Independent Path*

Path 1 = 1-20
Path 2 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20
Path 3 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-8-9-10-11-12-13-14-15-16-17-18-19-20

e) *Graph Matriks*

Berikut graph matriks deskripsinya.

Tabel 3. *Graph Matriks* Dekripsi

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Sum
1		1																		1	1
2			1																		0
3				1																	0
4					1																0
5						1															0
6							1														0
7								1													0
8									1												0
9										1											0
10											1										0
11												1									0
12													1								0
13														1							0
14															1						0
15																1					0
16																	1				0
17																		1			0
18																			1		0
19								1												1	1
20																					0
									Sum + 1												2+1=3

Sumber: Hasil Penelitian (2018)

f) *Predicate Node*

Predicate Node dijelaskan sebagai berikut:

$V_{(G)}$ = Jumlah node yang memiliki lebih dari 1 jalur + 1
= 2 + 1
= 3

g) *Tabel Hasil Pengujian Dekripsi*

Berikut ini tabel hasil pengujian deskripsi.

Tabel 4. Tabel Hasil Pengujian Dekripsi

No	Path	Yang Diharapkan	Hasil Pengujian	Kesimpulan
1	1-20	Berhasil melakukan dekripsi password	Berhasil melakukan dekripsi password	[√] Diterima [ ] Ditolak
2	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20	Berhasil melakukan tahap demi tahap dekripsi	Berhasil melakukan tahap demi tahap dekripsi	[√] Diterima [ ] Ditolak
3	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-8-9-10-11-12-13-14-15-16-17-18-19-20	Jika nilai $hasil tidak sesuai, maka kembali ke langkah sebelumnya yaitu menghitung nilai $hasil	Berhasil menghitung nilai $hasil	[√] Diterima [ ] Ditolak

Sumber: Hasil Penelitian (2018)

3.3. Hasil Pengujian

Berikut penjelasan tentang hasil pengujian.

1. Pengujian Login

Pengujian login dapat dilihat pada Tabel 5.

Tabel 5. Pengujian data benar *Login*

Kasus dan Hasil Uji (Data Benar)			
Aktivitas yang dilakukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Mengisikan *username or email*	Berhasil mengisi *username or email*	Berhasil mengisi *username or email*	[√] Diterima [ ] Ditolak
Mengisikan *Password*	Berhasil mengisi *password*	Berhasil mengisi *password*	[√] Diterima [ ] Ditolak
Klik tombol Masuk	*User* berhasil masuk ke menu utama	*User* berhasil masuk ke menu utama	[√] Diterima [ ] Ditolak
Klik tombol Mendaftar	*User* berhasil masuk ke menu form pendaftaran *user* baru	*User* berhasil masuk ke menu form pendaftaran *user* baru	[√] Diterima [ ] Ditolak

Sumber: Hasil Penelitian (2018)

Pengujian data salah login dapat dilihat pada Tabel 6.

Tabel 6. Pengujian data salah *Login*

Kasus dan Hasil Uji (Data Salah)			
Aktivitas yang dilakukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Tidak mengisikan *username or email*	Menampilkan pesan "*username* atau *password* salah!"	Menampilkan pesan "*username* atau *password* salah!"	[√] Diterima [ ] Ditolak
Tidak mengisikan *Password*	Menampilkan pesan "*username* atau *password* salah!"	Menampilkan pesan "*username* atau *password* salah!"	[√] Diterima [ ] Ditolak
Klik tombol Masuk tapi data *username or email* salah atau belum di input	Menampilkan pesan "*username* atau *password* salah!"	Menampilkan pesan "*username* atau *password* salah!"	[√] Diterima [ ] Ditolak
Klik tombol Masuk tapi data *password* salah atau belum di input	Menampilkan pesan "*username* atau *password* salah!"	Menampilkan pesan "*username* atau *password* salah!"	[√] Diterima [ ] Ditolak

Sumber: Hasil Penelitian (2018)

2. Pengujian Daftar Member

Pengujian data benar daftar member dapat dilihat pada Tabel 7.

Tabel 7. Pengujian Data Benar Daftar Member

Kasus dan Hasil Uji (Data Benar)			
Aktivitas yang dilakukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Mengisikan Nama *User*	Berhasil mengisi Nama *User*	Berhasil mengisi Nama *User*	[√] Diterima [ ] Ditolak
Mengisikan *Email*	Berhasil mengisi *Email*	Berhasil mengisi *Email*	[√] Diterima [ ] Ditolak
Mengisikan *Username*	Berhasil mengisi *Username*	Berhasil mengisi *Username*	[√] Diterima [ ] Ditolak
Mengisikan *Password*	Berhasil mengisi *Password*	Berhasil mengisi *Password*	[√] Diterima [ ] Ditolak
Mengisikan kembali *Password*	Berhasil mengisi kembali *Password*	Berhasil mengisi kembali *Password*	[√] Diterima [ ] Ditolak
Klik tombol Masuk	*User* berhasil masuk ke menu *Login*	*User* berhasil masuk ke menu *Login*	[√] Diterima [ ] Ditolak
Klik tombol Daftar	*User* berhasil mendaftarkan data dan menyimpan data ke dalam *database*	*User* berhasil mendaftarkan data dan menyimpan data ke dalam *database*	[√] Diterima [ ] Ditolak

Sumber: Hasil Penelitian (2018)

Pengujian data salah daftar member dapat dilihat pada Tabel 8.

Tabel 8. Pengujian Data Salah Daftar Member

Kasus dan Hasil Uji (Data Salah)			
Aktivitas yang dilakukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Tidak mengisikan Nama *User*	Menampilkan pesan "Nama Tidak Boleh Kosong"	Menampilkan pesan "Nama Tidak Boleh Kosong"	[√] Diterima [ ] Ditolak
Tidak mengisikan *Email*	Menampilkan pesan "*Email* Tidak Boleh Kosong"	Menampilkan pesan "*Email* Tidak Boleh Kosong"	[√] Diterima [ ] Ditolak
Mengisikan *Email* tetapi format input *email* salah	Menampilkan pesan "*Email* Tidak Valid"	Menampilkan pesan "*Email* Tidak Valid"	[√] Diterima [ ] Ditolak
Tidak mengisikan *Username*	Menampilkan pesan "*Username* Tidak Boleh Kosong"	Menampilkan pesan "*Username* Tidak Boleh Kosong"	[√] Diterima [ ] Ditolak
Klik tombol Daftar tetapi belum mengisikan salah satu dari Nama Anda, Email, Username, atau email tidak valid	Menampilkan pesan "Nama Tidak Boleh Kosong", "*Email* Tidak Boleh Kosong", "*Username* Tidak Boleh Kosong", atau "*Email* Tidak Valid"	Menampilkan pesan "Nama Tidak Boleh Kosong", "*Email* Tidak Boleh Kosong", "*Username* Tidak Boleh Kosong", atau "*Email* Tidak Valid"	[√] Diterima [ ] Ditolak

Sumber: Hasil Penelitian (2018)

4. Kesimpulan

Aplikasi Login RSA tidak memerlukan proses instalasi aplikasi dan dapat dijalankan dari sistem operasi apapun karena aplikasi Login RSA ini berbasis *local area (web based).* Aplikasi Login RSA hanya dapat menggunakan maksimal 8 karakter pada *input password*, diharapkan dapat dikembangkan dengan karakter maksimal yang lebih besar.

Daftar Pustaka

Falani AZ, Zunaidy M. 2018. Sistem Pengaman File dengan Menggunakan Metode RSA Kriptografi & Digital Signature. Surabya: Jurusan Sistem Komputer, Fakultas Ilmu Komputer, Universitas Narotama.

Kerr J. M, Hunter R. 1993. Inside RAD: How to Build a Fully Functional System in 90 Days or Less. s.l.: McGraw-Hill.
Martin J. 1991. Rapid Application Development. Macmillan.. s.l.:s.n.
Munir R. 2006. Kriptografi. Bandung: ITB.
Rahardjo B. 2002. Keamanan Sistem Berbasis Internet. Bandung: PT. Insan Komunikasi Indonesia.
Samtomo. 2018. Enkripsi Dan Deskripsi Menggunakan Algoritma RSA. Tuban: Fakultas Teknik, Universitas PGRI RonggolaweTuban.