BAB II
LANDASAN TEORI

Pada bab ini akan menjelaskan mengenai teori-teori yang relevan dengan penelitian yang dilakukan, mengenai latar belakang kriptografi, algoritma *Diffie-Hellman*, algoritma *RSA* dan algoritma *RSA-CRT*.

2.1. Kriptografi

Kata kriptografi berasal dari dua kata Yunani yang memiliki arti yaitu, *crypto* yang memiliki arti (rahasia) dan *grapho* yang memiliki arti menulis (Hasugian, 2017), Menurut (Hidayatullah et al., 2016). Kriptografi adalah ilmu yang mempelajari bagaimana cara menjaga agar data atau pesan tetap aman saat dilakukannya proses pengiriman, dari pengirim ke penerima tanpa mengalami gangguan oleh pihak ketiga yang tidak memiliki hak. Menurut (Siswanto & Syukur, 2018) kriptografi adalah sebuah ilmu dan seni yang digunakan untuk menjaga keamanan pesan pada saat pesan dikirim dari suatu tempat ke tempat lain.

Menurut (Santomo, 2017) kriptografi merupakan ilmu yang mempelajari tentang pengamanan data dan langkah-langkah yang dilakukan dalam kriptografi disebut dengan algoritma kriptografi, langkah yang dilakukan untuk mengamankan suatu pesan atau data dalam kriptografi disebut dengan melakukan proses enkripsi, dimana enkripsi adalah proses yang dilakukan untuk mengubah pesan atau data (*plaintext*) asli menjadi bentuk susunan kode – kode yang tidak dapat dipahami atau disebut (*chipertext*), hal ini dilakukan untuk mengamankan data atau pesan yang sedang dalam proses pengiriman akan terjadinya pencurian atau diketahui oleh orang yang tidak berhak, dan proses pengembalian *chipertext* kembali menjadi *plaintext* disebut dengan proses dekripsi (Hasugian, 2017).

2.1.1. Tujuan Kriptografi

Menurut (Hidayatullah & Insanudin, 2016). Terdapat empat tujuan utama dalam kriptografi yaitu:

1. Kerahasiaan (*Confidelity*)

Memastikan selama proses pengiriman pesan atau data, tetap menjadi rahasia atau tidak diketahui oleh orang yang tidak memiliki hak.

2. Keutuhan data (*Data integrity*)

   Memastikan selama proses pengiriman pesan atau data yang dikirim memiliki isi atau nilai yang tetap sama dari awal pengiriman sampai penerimaan pesan atau data.

3. Keotentikasian (*Authentication*)

   Memastikan selama proses pengiriman pesan atau data, keaslian tetap terjaga dan memastikan otentikasi antara pihak yang terkait.

4. Anti penyangkalan (*Non-repudiation*)

   Memastikan tidak adanya penyangkalan pengiriman atau penerimaan pesan atau data yang diterima ataupun dikirim oleh dirinya.

2.1.2. Algoritma Kriptografi

Algoritma kriptografi adalah langkah-langkah yang dilakukan selama proses pengiriman algoritma kriptografi, menurut (Hasugian, 2017). Algoritma kriptografi terbagi menjadi dua jenis yaitu:

1. Algoritma simetris

   Algoritma ini bekerja dengan menggunakan *secret key* yang sama pada proses enkripsi dan dekripsinya, Adapun contoh algoritma kriptografi simetris yaitu, *OTP, DES, RC2, RC4, RC5, RC6, IDEA, Twofish, Magenta, FEAL, SAFER, LOKI, CAST, Rijndael* (*AES*), *Blowfish, GOST, A5, Diffie-Hellman* dan lain-lain

2. Algoritma asimetris

   Algoritma ini bekerja dengan menggunakan dua jenis kunci yang berbeda pada prosesnya yaitu:

   - Kunci public (*Public key*)

     Kunci ini berfungsi untuk melakukan enkripsi atau pengubahan pesan atau data (*Plaintext*) menjadi kode kode yang tidak dipahami (*Ciphertext*).

   - Kunci rahasia (*Private key*)

Kunci jenis ini berfungsi untuk melakukan dekripsi atau pengembalian *ciphertext* menjadi pesan atau data (*plaintext*) yang dapat dipahami.

Adapun contoh algoritma yang menggunakan jenis ini yaitu, *Digital Signature Algorithm* (*DSA*), *RSA* (*Rivest Shamir Adleman*), *Elliptic Curve Cryptography* (*ECC*), Kriptografi Quantum dan lain-lain.

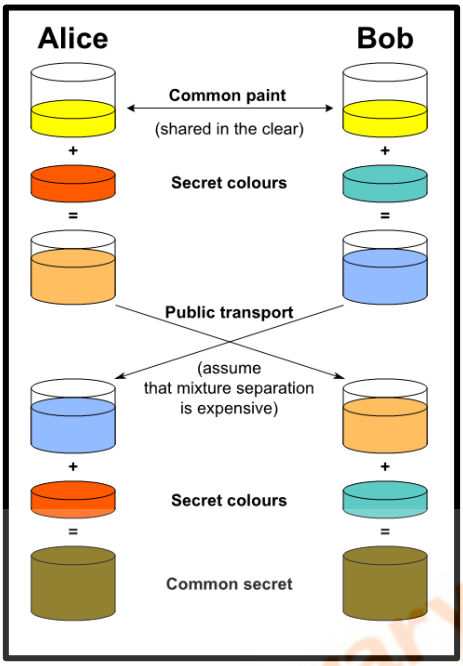
2.2. *Diffie-Hellman*

Menurut (Suryani et al., 2015) nama dari algoritma *Diffie-Hellman* berasal dari duan nama penciptanya yaitu *Whitfield Diffie* dan *Martin Hellman*, yang pertama kali dipublikasikan pada tahun 1976, dan algoritma ini termasuk ke-dalam algoritma dengan jenis kunci simetris, dimana pada kedua pengguna akan memiliki *secret key* yang sama pada hasil pertukaran data yang dilakukan. Menurut (Stallings, 2005), tujuan utama dari algoritma ini, bekerja untuk membuat dua pengguna dapat melakukan pertukaran kunci dengan aman dan dapat digunakan untuk melakukan proses enkripsi pesan selanjutnya. Algoritma *Diffie-Hellman* ini terbatas hanya pada pertukaran *secret key* saja, dan keamanan dari algoritma ini bergantung pada sulitnya komputasi logaritma diskrit, adapun besaran yang digunakan pada algoritma ini yaitu seperti pada tabel 2.

Tabel 2 Variabel dan aturan *Diffie-Hellman*

Nama	Aturan	Jenis
P (*global public element*)	Merupakan bilangan prima	*public*
q (*global public element*)	q < p dan q merupakan primitive root dari p	*public*
pn1 (private number client 1)	0 < pn1 < p	*privat*
pn2 (private number client 2)	0 < pn1 < p	*privat*
pb1 (public key client 1)	-	*public*
pb2 (public key client 2)	-	*public*
k1 (secret key client 1)	-	*privat*
k2 (secret key client 2)	-	*privat*
k (secret key)	-	*privat*

Adapun penggambaran dari alur sistem algoritma pada penelitian ini dapat dilihat pada gambar 1.



Gambar 1 Alur kerja *Diffie-Hellman*

(Sumber : https://upload.wikimedia.org/)

Pada gambar 1, menjelaskan bagaimana cara algoritma *Diffie-Hellman* melakukan proses pertukaran kunci dan bagaimana membangkitkan *secret key* dengan menggunakan cara penggabungan warna sebagai contoh proses algoritma *Diffie-Hellman*, adapun keterangan warna pada tabel 3 yaitu :

Tabel 3 Keterangan warna

Warna	Variabel
Kuning	q dan q
Jingga	pn1
Biru muda	pn2
Cream	pb1
Biru tua	pb2
Coklat emas	k

Pada gambar 1, terlihat terdapat dua pengguna yang akan melakukan proses pertukaran dan pembangkitan kunci pada algoritma ini, yaitu alice dan bob, pada tahap pertama alice dan bob memiliki nilai kuning yang sudah bersama sama disepakati dan bersifat tidak rahasia. Alice memiliki *privat*

number yang berwarna jingga dan bob juga memiliki *privat number* yang berwarna biru muda, antara alice dan bob tidak saling mengetahui warna rahasia mereka masing masing, lalu dilanjutkan dengan melakukan pembangkitan *public key* pada kedua sisi pengguna, alice akan membangkitkan *public key* dengan cara menggabungkan warna kuning dan warna jingga milik alice, dari hasil pencampuran warna ini maka akan menghasilkan warna *cream*, pada sisi bob juga melakukan pembangkitan *public key* yang kedua, dengan cara yang sama, yaitu dengan mencampur warna kuning dengan warna rahasia milik bob, sehingga dari proses pencampuran warna pada bob akan menghasilkan warna biru tua.

Setelah alice dan bob berhasil mendapatkan *public key* masing masing, maka dilanjutkan dengan melakukan pertukaran *public key* dimana, alice akan mengirimkan *public key* nya kepada bob, dan begitu juga kepada bob akan mengirimkan warna *public key* nya kepada alice, setelah proses pertukaran *public key* dilakukan maka masing-masing pengguna akan melakukan penggabungan warna untuk menghasilkan warna *secret key*, dapat dilihat pada gambar 1, bahwa pada sisi alice dan bobo memiliki hasil akhir warna yang sama dari proses percampuran warna mereka, yaitu warna coklat emas.

Adapun proses perhitungan matematis yang dilakukan pada algoritma ini yaitu, sebagai berikut :

Langkah ke-1: Pengguna satu dan dua memilih nilai p dan q sesuai dengan aturan yang ada pada table 2.

Keterangan :

p = nilai masukan bilangan prima

q = nilai masukan dan merupakan *primitive root modulo* dari p

Langkah ke-2: Masing masing pengguna memilih nilai pn1 dan pn2 sesuai dengan aturan yang ada pada tabel 2,

Keterangn :

$pn1$ = *privat number client* satu

$pn2$ = *privat number client* dua.

dan pada tahap ini kedua client saling tidak memberi tahu *privat number* mereka masing masing.

Langkah ke-3: Pada langkah ini akan melakukan proses perhitungan untuk mendapatkan nilai pb1 oleh *client 1*, dengan menggunakan rumus menggunakan rumus (1).

$$pb1 = q^{pn1} \ mod \ p \quad \text{.....(1)}$$

Keterangan :

$pb1$ = *public key client* satu

q = nilai masukan *primitive root modulo* dari p

$pn1$ = *privat number client* satu

p = nilai masukan bilangan prima

Setelah nilai pb1 didapat, maka selanjutnya nilai pb1 akan dikirimkan kepada *client* 2.

Langkah ke-4: Pada langkah ini akan melakukan proses perhitungan untuk mendapatkan nilai pb2 oleh *client* 2, dengan menggunakan rumus (2).

$$pb2 = q^{pn2} \ mod \ p \quad \text{.....(2)}$$

Keterangn :

$pb2$ = *public key client* 2

q = nilai masukan *primitive root modulo* dari p

$pn2$ = *privat number client* 2

p = nilai masukan bilangan prima

Setelah nilai pb2 didapat, maka nilai pb2 dikirimkan kepada *client* 1

Langkah ke-5: Melakukan penghitungan *secret key* oleh *client* 1 menggunakan rumus (3)

$$k1 = pb2^{pn1} \ mod \ p$$ ……(3)

Keterangn :

$k1$ = *secret key client* 1

$pb2$ = *public key client* 2

$pn1$ = *privat number client* 1

p = nilai masukan bilangan prima

Langkah ke-6: Melakukan penghitungan *secret key* oleh *client* 2 menggunakan rumus (4)

$$k2 = pb1^{pn2} \ mod \ p$$ ……(4)

Keterangn :

$k2$ = *secret key client* 2

$pb1$ = *public key client* 1

$pn2$ = *privat number client* 2

p = nilai masukan bilangan prima

Langkah ke-7: Setelah nilai k1 dan k2 didapatkan maka nilai k akan menjadi seperti pada rumus (5).

$$k = k1 = k2$$ ……..(5)

Keterangan :

k = *secret key*

$k1$ = *secret key client* 1

$k2$ = *secret key client* 2

Dimana pada rumus 5, nilai k akan bernilai sama dengan nilai dari k1 dan k2, dan pada prosesnya apabila nilai

masukkan yang digunakan telah memenuhi persyaratan, maka nilai k1 dan k2 akan selalu bernilai serupa.

2.3. *RSA* (*Rivest Shamir Adleman*)

Menurut (Panjaitan et al., 2019), algoritma *RSA* adalah algoritma kriptografi yang memiliki jenis kunci asimetris, dimana algoritma ini akan menggunakan *key* yang berbeda pada saat proses enkripsi dan dekripsinya. Menurut (Ginting et al., 2015), algoritma *RSA* adalah algoritma kriptografi kunci *public* atau yang disebut juga dengan asimetris. Pada algoritma ini akan menggunakan kunci yang pada saat melakukan proses enkripsi dan proses dekripsi, baik kunci enkripsi atau dekripsinya, keduanya merupakan bilangan bulat, kunci untuk melakukan enkripsi bersifat tidak rahasia atau disebut dengan *public key*, sedangkan kunci dekripsinya bersifat rahasia, atau yang disebut *privat key*.

Sedangkan menurut (Stallings, 2005) nama dari algoritma *RSA* ini sendiri berasal dari tiga nama pencipta algoritma ini yaitu *Rivest, Adi Shamir,* dan *Len Adleman* di *MIT.* Algoritma ini pertama kali diterbitkan pada tahun 1978, sejak saat itu juga algoritma *RSA* menjadi algoritma *public key* yang paling banyak diterima dan diterapkan. Dari pendapat diatas disimpulkan bahwa, algoritma *RSA* adalah algoritma yang memiliki jenis kunci asimetris, dimana pada proses enkripsi dan dekripsinya algoritma ini menggunakan kunci yang berbeda yaitu disebut dengan *public key* dan *privat key*, dan *private key* digunakan untuk melakukan proses enkripsi pesan atau *plaintext* dan kunci ini bersifat tidak rahasia, sedangkan *privat key* digunakan untuk melakukan proses dekripsi dari *ciphertext* dan kunci ini bersifat rahasia dalam prosesnya.

Kekuatan dari algoritma ini berada pada pemecahan nilai *N* dimana dalam memecahkan nilai tersebut dapat dilakukan dengan memfaktorkan bilangan bulat tersebut menjadi faktor-faktor prima, dimana untuk melakukan hal tersebut, belum ditemukan algoritma yang efisien untuk melakukan proses tersebut, dan cara lain yang dapat dilakukan yaitu dengan melakukan pemfaktoran bilangan bulat tersebut menggunakan pohon faktor, tetapi hal ini

tetap menjadi tidak efisien, mengingat nilai n yang berjumlah besar, dengan begitu proses yang dibutuhkan pohon pemfaktoran akan semakin lama pula, dengan kata lain apabila nilai *N* pada algoritma *RSA* bernilai besar, maka akan semakin kuat algoritma tersebut (Ginting et al., 2015). Adapun variabel dan syarat yang digunakan pada algoritma ini dapat dilihat pada tabel 4.

Tabel 4 Variabel dan aturan pada *RSA*

Nama	Syarat	Sifat
p,q (merupakan nilai pilihan pengguna)	merupakan bilangan prima	*privat*
k (merupakan bilangan hasil proses algoritma *Diffie-Hellman*)	-	*privat*
n	-	*not privat*
ϕ(n)	-	*privat*
e' (merupakan kunci enkripsi)	-	*not privat*
d' (merupakan kunci dekripsi)	-	*privat*
m (merupakan nilai *plaintext*)	dengan syarat $0 < m < n$ dan $m^{e\prime} > n$	*privat*
c (merupakan nilai *ciphertext*)	dengan syarat $0 < c < n$ dan $c^{d\prime} > n$	*not privat*
md (merupakan *plain text* hasil dekripsi)	-	*privat*

Pada dasarnya dalam proses algoritma ini terbagi menjadi tiga proses yaitu proses pembangkitan kunci yaitu, proses enkripsi, dan proses dekripsi, adapun penjabaran dari proses – proses berikut yaitu.

2.3.1. Proses Pembangkitan Kunci *RSA*

Pada tahapan proses ini akan dilakukan pembangkitan kunci enkripsi (*public key*) dan kunci dekripsi (*privat key*), adapun perhitungan matematis yang dilakukan pada proses ini adalah sebagai berikut:

Langkah 1: Pada langkah ini akan dilakukan pemilihan nilai dari p dan q dengan syarat seperti pada tabel 4.

Keterangan :

p = nilai masukan berupa bilangan prima

q = nilai masukan berupa bilangan prima

Setelah nilai p dan q di pilih, maka lanjut ke langkah selanjutnya.

Langkah 2: Pada langkah ini akan dilakukan perhitungan untuk mendapatkan nilai n, dimana nilai n ini akan digunakan untuk mendapatkan nilai dari *privat key* dan *public key*, adapun rumus matematis yang dilakukan pada langkah ini dapat dilihat pada rumus (6).

$$n = p\ x\ q \quad \text{.....(6)}$$

Keterangan :

n = nilai hasil perhitungan p dan q

p = nilai masukkan bilangan prima

q = nilai masukkan bilangan prima

Langkah 3: Setelah nilai n didapatkan, maka dilanjutkan untuk menghitung nilai ϕ(n), dengan menggunakan rumus (7).

$$\phi(n) = (p-1)x(q-1) \quad \text{.....(7)}$$

Keterangan :

$\phi(n)$ = nilai hasil perhitungan p dan q

p = nilai masukkan bilangan prima

q = nilai masukkan bilangan prima

Langkah 4: Setelah nilai ϕ(n) didapatkan, maka dilanjutkan untuk membangkitkan kunci enkripsi, atau disebut dengan *public key* (e'), dengan menggunakan persamaan (8).

$$GCD\left(e', \phi(n)\right) = 1 \quad \text{.....(8)}$$

Keterangan :

e' = nilai kunci enkripsi

$\phi(n)$ = *totien* n, nilai hasil perhitungan *GCD* dari e' dan ϕ(n)

Langkah 5: Setelah nilai e' didapatkan, maka dilanjutkan untuk membangkitkan kunci dekripsi, atau disebut dengan *privat key* (d'), dengan menggunakan rumus (9).

$$d'x\ e'\ \equiv 1\ (mod\ \phi(n)) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(9)$$

Keterangan :

d' = nilai kunci dekripsi

e' = nilai kunci enkripsi

$\phi(n)$ = *totien* n, nilai hasil perhitungan *GCD* dari e' dan ϕ(n)

Dan rumus (9) merupakan ekivalen dengan rumus (10)

$$d' = \frac{1+i\ \phi(n)}{e'} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(10)$$

Keterangan :

d' = nilai kunci dekripsi

e' = nilai kunci enkripsi

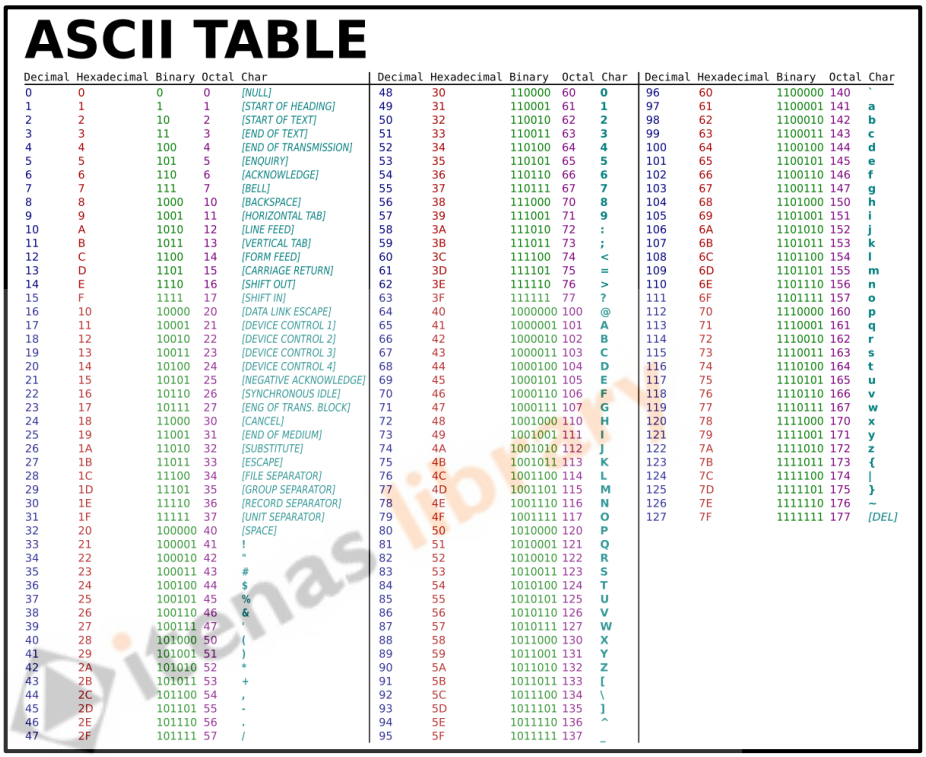
i = nilai masukan percobaan

$\phi(n)$ = *totien* n, nilai hasil perhitungan *GCD* dari e' dan ϕ(n)

Sehingga nilai dari d' dapat hitung dengan menggunakan rumus (10).

2.3.2. Proses Enkripsi *RSA*

Pada proses ini akan dilakukan enkripsi, atau melakukan proses pengubahan *plaintext* menjadi *chipertext*, dalam melakukan proses ini akan digunakan nilai yang acuan yang bersifat *universal* yaitu berdasarkan kode *ASCII*, adapun kode *ASCII* yang digunakan dapat dilihat pada gambar 2.



ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

Gambar 2 Tabel *ASCII Code*

(Sumber : https://upload.wikipedia.org/)

Adapun proses yang dilakukan pada tahapan ini yaitu sebagai berikut :

Langkah 1: Pada tahap ini *plaintext* (m) masukan akan diubah perkata menjadi nilai desimal berdasarkan kode *ASCII* yang berlaku.

Langkah 2: Setelah *plaintext* (m) diubah menjadi bentuk desimal, maka dilanjutkan dengan melakukan pemisahan nilai m menjadi nilai per *block*, agar dapat dilakukan perhitungan selanjutnya.

Langkah 3: Setelah nilai m dipisahkan menjadi *block-block*, maka selanjutnya dilakukan proses enkripsi nilai m per *block*,

untuk melakukan proses enkripsi pada nilai tersebut, dilakukan dengan menggunakan rumus (11).

$$c = m^{e'} \bmod n \quad \text{.....(11)}$$

Keterangan :

c = nilai *chipertext*

m = nilai *plaintext* masukkan

e' = nilai *public key* atau kunci enkripsi

n = nilai hasil perhitungan p dan q

Langkah 4: Setelah nilai *chipertext* (c) didapatkan, maka selanjutnya nilai tersebut digabungkan kembali sehingga menjadi sederet angka acak yang tidak dapat diketahui.

2.3.3. Proses Dekripsi

Pada proses ini akan dilakukan proses dekripsi, atau mengubah kembali *ciphertext* kembali menjadi *plaintext*. Adapun perhitungan matematis yang dilakukan pada proses ini adalah sebagai berikut:

Langkah 1: Pada langkah pertama, nilai c dari hasil proses sebelumnya akan dilakukan pemisahan kembali menjadi nilai nilai per *block*.

Langkah 2: Setelah nilai c dipisahkan, maka dilanjutkan dengan melakukan proses dekripsi untuk mengubah kembali nilai c menjadi *plaintext* hasil (md), dengan menggunakan rumus (12).

$$md = c^{d'} \bmod n \quad \text{.....(12)}$$

Keterangan :

md = nilai *plaintext* keluaran

c = nilai *chipertext*

d' = nilai *privat key* atau kunci dekripsi

n = nilai hasil perhitungan nilai p dan q

Langkah 3: Setelah nilai c diubah menjadi nilai md, maka proses selanjutnya mengubah nilai desimal tersebut menjadi karakter yang seharusnya berdasarkan kode *ASCII* yang digunakan pada proses sebelumnya juga, adapun kode *ASCII* yang digunakan dapat dilihat pada gambar 2.

Langkah 4: Setelah nilai md di ubah kedalam karakter, maka nilai tersebut disatukan kembali, sehingga menjadi sama seperti *plaintext* sebelumnya.

2.4. *RSA-CRT*

Menurut (Arief & Saputra, 2016), algoritma *RSA-CRT* merupakan algoritma *RSA* yang dimodifikasi dengan menggunakan teorema *CRT*, dimana *CRT* sendiri menurut (Panjaitan et al., 2019) adalah suatu teori matematis yang digunakan untuk melakukan penyederhanaan dari eksponensial yang berukuran besar. Dimana pada pada penelitian ini digunakan algoritma *RSA-CRT* bertujuan untuk mengatasi kekurangan pada lamanya waktu proses dari *RSA*. Masalah ini terjadi pada proses dekripsi dari algoritma ini, dimana nilai eksponensial pada proses dekripsi algoritma ini bernilai sangat besar (Prihanto, 2019), dimana pada penelitian ini menggunakan bahasa pemrograman *javascript*, sehingga pastinya program akan membutuhkan waktu yang cukup lama untuk melakukan perhitungan yang bernilai sangat besar, yang menjadikan waktu proses menjadi tidak efisien dalam segi waktu prosesnya.

Berdasarkan dari permasalahan diatas maka dalam penelitian ini dibutuhkan algoritma *RSA* yang dimodifikasi dengan algoritma *CRT* atau yang disebut *RSA-CRT* untuk meminimalisir waktu proses dekripsi dari algoritma ini. pada algoritma ini, pada tahapan proses enkripsi memiliki langkah yang sama seperti pada algoritma *RSA*, yang berbeda hanyalah pada proses perhitungan dekripsinya saja, sehingga adapun langkah – langkah yang dilakukan pada proses dekripsi dari algoritma ini adalah sebagai berikut:

Langkah 1: Setelah nilai *chipertext* (c) dari proses enkripsi didapatkan, lalu dilakukan pemisahan nilai (c) tersebut menjadi per *block*.

Langkah 2: Setelah nilai c diubah menjadi per *block*, selanjutnya akan dilakukan perhitungan terlebih dahulu untuk menghitung nilai dp dan dq, menggunakan rumus (13)

$$dp = d' \grave{m}od\ (p-1) \quad \text{.....(13)}$$

Keterangan :

dp = nilai hasil perhitungan nilai p dan d'

p = nilai masukkan bilangan prima

d' = nilai *privat key* atau kunci dekripsi

$$dq = d' \grave{m}od\ (q-1) \quad \text{.....(14)}$$

Keterangan :

dq = nilai hasil perhitungan nilai q dan d'

q = nilai masukkan bilangan prima

d' = nilai *privat key* atau kunci dekripsi

Langkah 3: Setelah nilai dp dan dq didapatkan, maka dilanjutkan untuk mencari nilai *qInv*, dengan menggunakan rumus (14).

$$qInv = q^{\backsim 1}\ mod\ p \quad \text{.....(15)}$$

Keterangan :

$qInv$ = nilang hasil perhitungan modulus invers q dan p

q = nilai masukkan bilangan prima

p = nilai masukkan bilangan prima

Langkah 4: Setelah nilai dari *qInv* didapatkan, maka dilanjutkan untuk mencari nilai dari m1 dan m2 dengan menggunakan nilai dari dp dan dq yang telah dihitung sebelumnya, dengan menggunakan rumus (15).

$$m1 = c^{dp} \ mod \ p$$ .....(16)

Keterangan :

$m1$ = nilai hasil perhitungan *chipertext*

c = nilai *chipertext*

dp = nilai hasil perhitungan nilai p dan d'

p = nilai masukkan bilangan prima

$$m2 = \ c^{d\grave{q}} \ mod \ q$$ .....(17)

Keterangan :

$m2$ = nilai hasil perhitungan *chipertext*

c = nilai *chipertext*

dq = nilai hasil perhitungan nilai q dan d'

q = nilai masukkan bilangan prima

Langkah 5: Setelah nilai dari m1 dan m2 didapatkan, maka dilanjutkan untuk mendapatkan nilai h, dengan menggunakan nilai dari qinv, m1, dan m2 yang telah dihitung sebelumnya, dengan menggunakan rumus (16).

$$h = qInv \ x \ (m1 - \grave{m2}) \ mod \ p$$ .....(18)

Keterangan :

h = nilai hasil perhitungan nilai *qInv*, m1, m2 dan p

$qInv$ = nilang hasil perhitungan modulus invers q dan p

$m1$ = nilai hasil perhitungan *chipertext*

$m2$ = nilai hasil perhitungan *chipertext*

p = nilai masukkan bilangan prima

Langkah 6: Setelah nilai h didapatkan, maka baru di dapatkan nilai hasil akhir md dengan menggunakan nilai h dan m2 dari perhitungan sebelumnya, menggunakan rumus (17).

$$md = m2\grave{} + (h\ x\ q) \quad \text{.....(19)}$$

Keterangan :

md = nilai *plaintext* keluaran

$m2$ = nilai hasil perhitungan *chipertext*

h = nilai hasil perhitungan nilai *qInv*, m1, m2 dan p

q = nilai masukkan bilangan prima

Langkah 7: Setelah nilai md didapatkan, maka selanjutnya nilai tersebut diubah kembali menjadi bentuk karakter sesuai dengan kode *ASCII*, dan digabungkan kembali, sehingga membentuk seperti nilai pada *plaintext* (m) masukannya.asdasdas

2.5. Studi Kasus

Pada bagian ini, akan dijelaskan mengenai studi kasus dari metode *Diffie-Hellman, RSA,* dan *RSA-CRT*.

2.5.1. Studi Kasus Algoritma *Diffie-Hellman*

Berikut ini adalah contoh studi kasus perhitungan dengan menggunakan algoritma *Diffie-Hellman*, proses ini bertujuan untuk mendapatkan nilai dari *secret key*. Adapun contoh perhitungan pada studi kasus ini menggunakan dua contoh nama pengguna yang akan saling bertukar kunci dan membangkitkan kunci yaitu Alice dan Bob Adapun contoh perhitungannya yaitu sebagai berikut:

Langkah 1: Pada langkah pertama, Alice dan Bob akan melakukan pemilihan nilai dari p dan q , dengan berdasarkan persyaratan pemilihan pada table 2. dimana alice dan bob sepakat menggunakan nilai yang sama, pada tahap ini alice dan bob sepakat memilih 13 dan 7 sebagai nilai p dan q.

Langkah 2: Setelah nilai p dan q di pilih, maka langkah selanjutnya alice akan memilih nilai dari pn1 dan bob akan memilih nilai dari pn2, dimana alice dan bob tidak saling memberi tahu nilai dari pilihan mereka, dan pemilihan nilai ini harus mengikuti persyaratan pemilih seperti pada tabel 2. Pada tahap ini alice memilih nilai 6 sebagai pn1 dan bob memilih 5 sebagai pn 2.

Langkah 3: Setelah p, q , pn1, dan pn2 dipilih, maka selanjutnya alice melakukan penghitungan untuk membangkitkan nilai pb1 menggunakan rumus (1), adapun hasil perhitungannya sebagai berikut :

$$pb1 = 7^6 \ mod \ 13$$
$$pb1 = 117649 \ mod \ 13$$
$$pb1 = 12$$

Keterangan :

$pb1$ = nilai *public key client* satu

Dari penghitungan di atas, alice mendapatkan nilai 12 sebagai pb1, selanjutnya alice mengirim nilai pb1 kepada bob.

Langkah 4: Pada tahap ini bob akan melakukan hal yang sama seperti alice, menemukan nilai pb2, dengan menggunakan rumus (2), yaitu sebagai berikut:

$$pb2 = 7^5 \ mod \ 13$$
$$pb2 = 16807 \ mod \ 13$$
$$pb2 = 11$$

Keterangan :

$pb2$ = nilai *public key client* dua

Setelah bob mendapatkan nilai pb2 yaitu sama dengan 11, maka bob mengirimkan nilai tersebut ke alice.

Langkah 5: Setelah alice mendapatkan nilai pb2 dari bob yaitu 11, maka selanjutnya alice melakukan perhitungan untuk menemukan *secret key* yaitu k1, dengan menggunakan rumus (3), yaitu sebagai berikut:

$$k1 = 11^6 \ mod \ 13$$
$$k1 = 1771561 \ mod \ 13$$
$$k1 = 12$$

Keterangan :

$k1$ = nilai *secret key client* satu

Langkah 6: Pada sisi bob juga melakukan hal yang sama, melakukan penghitungan untuk menemukan *secret key* yaitu k2 menggunakan pb1 dari alice yaitu 12, adapun perhitungannya yaitu sebagai berikut menggunakan rumus (4).

$$k2 = 12^5 \ mod \ 13$$
$$k2 = 248831 \ mod \ 13$$
$$k2 = 12$$

Keterangan :

$k2$ = nilai *secret key client* dua

Dapat dilihat dari perhitungan alice dan bob mendapati nilai dari *secret key* dama sama bernilai 12, yang berarti pembangkitan *secret key* k telah berhasil, yaitu:

$$k = 12 = 12$$

$k = 12$

Dari studi kasus algoritma ini, alice dan bob telah mebangkitkan *secret key* yang bernilai 12 dari perhitungan yang telah dilakukan.

2.5.2. Studi Kasus Algoritma *RSA*

Pada studi kasus ini akan dilakukan dua jenis perhitungan yaitu proses enkripsi dan dekripsi, dimana pada kasus ini akan diberikan contoh alice yang akan melakukan pengiriman pesan dan bob yang akan penerimaan pesan, dan pesan yang akan dikirim berupa huruf "B" yang diasumsikan nilai desimal "B = 2", dan dengan kesepakatan diantara keduanya nilai p = 2, q = 7 dan k = 12. Adapun perhitungan yang dilakukan alice dan bobo yaitu sebagai berikut:

A. Proses enkripsi pesan

Pada proses ini dilakukan oleh alice yang berstatus sebagai pengirim pesan, adapun langkah yang dilakukan alice untuk mengenkripsi pesannya yaitu sebagai berikut:

Langkah 1: Pada langkah pertama, alice akan melakukan perhitungan untuk menemukan nilai n1, yaitu dengan menggunakan rumus (6), yaitu sebagai berikut:

$$n1 = 2\ x\ 7$$
$$n1 = 14$$

Keterangan :

$n1$ = hasil perhitungan nilai masukkan p dan q

Langkah 2: Pada langkah selanjutnya, alice melakukan perhitungan untuk menemukan nilai n2, yaitu dengan menggunakan rumus (6) yang ditambahkan dengan nilai k hasil dari proses algoritma *Diffie-Hellman*, yaitu sebagai berikut:

$$n2 = 2\ x\ 7\ x\ 12$$

$$n2 = 168$$

Keterangan :

$n2$ = hasil perhitungan nilai masukkan p, q dan k

Langkah 3: Langkah selanjutnya, alice melakukan penghitungan untuk mengetahui nilai ϕ(n) dengan menggunakan rumus (7), yaitu sebagai berikut :

$$\phi(n1) = (2-1)x(7-1)$$

$$\phi(n1) = 1\ x\ 6$$

$$\phi(n1) = 6$$

Keterangan :

$\phi(n1)$ = nilai hasil perhitungan p dan q

Langkah 4: Langkah selanjutnya, alice melakukan pembangkitan *public key* (e') yang merupakan nilai yang akan digunakan untuk enkripsi dengan menggunakan rumus (8), dan pada kasus ini alice akan memilih nilai dari e' yang harus bernilai > 1 sampai menghasilkan nilai yang sama dengan 1 yaitu sebagai berikut:

$$GCD(2,6) = 2$$

$$GCD(3,6) = 3$$

$$GCD(4,6) = 2$$

$$GCD(5,6) = 1$$

Dari perhitungan diatas, alice telah menemukan nilai e' yaitu 5 dan pada tahap ini alice telah membangkitkan *public key* dengan nilai (5,168).

Langkah 5: Langkah selanjutnya, alice melakukan enkripsi pada pesan "B" yang diasumsikan bernilai desimal "2" dengan menggunakan rumus (11), adapun perhitungannya yaitu sebagai berikut:

$$c = 2^5 \ mod \ 168$$
$$c = 32 \ mod \ 168$$
$$c = 32$$

Keterangan :

c = nilai *ciphertext* hasil dari proses enkripsi *plaintext*

Lalu alice mengirimkan *chipertext* (c) yang bernilai 32 kepada bob.

B. Proses dekripsi

Pada tahap ini akan dilakukan oleh bob, dimana bob diasumsikan sebagai penerima pesan, dan dalam kasus ini bob menerima pesan dari alice berupa *chipertext* (c) yang bernilai 4, lalu bob melakukan perhitungan untuk mengembalikan nilai tersebut ke aslinya dengan melakukan proses dekripsi dengan menggunakan rumus (10), Adapun perhitungan yang dilakukan bob adalah sebagai berikut:

Langkah 1: Bob melakukan perhitungan untuk membangkitkan nilai *privat key* (d') nya dengan menggunakan rumus (10), dimana pada kasus ini bob akan mencari nilai m yang apabila dilakukan perhitungan kepada nilai m dari rumus (10) akan menghasilkan bilangan bulat, sehingga bob mendapati i = 9 yang akan menghasilkan bilangan bulat, sebagai berikut :

$$d' = \frac{1 + 9\ (6)}{5}$$

$$d' = 11$$

Keterangan :

d' = nilai kunci dekripsi

Langkah 2: Setelah mendapatkan nilai d' = 11, lalu bob melakukan dekripsi pada pesan "B = 2" dengan menggunakan rumus (12), yaitu sebagai berikut :

$$md = 32^{11}\ mod\ 14$$
$$md = 36028797018963968\ mod\ 14$$
$$md = 2$$

Keterangan :

md = nilai *plaintext* hasil akhir (*output*)

Dari penghitungan bob diatas, bob telah mendapati hasil dekripsi (md = 2), dimana itu merupakan nilai yang sama pada pesan yang alice kirim yaitu "B = 2", dari hasil tersebut bob telah menerima pesan dari alice yaitu (B = 2), dan proses enkripsi dan dekripsi oleh alice dan bob telah berhasil dilakukan.

2.5.3. Studi Kasus Dekripsi *RSA-CRT*

Pada studi kasus ini akan dilakukan contoh perhitungan menggunakan algoritma *RSA-CRT*, dimana pada kasus ini akan diberikan contoh alice yang akan melakukan pengiriman pesan dan bob yang akan penerimaan pesan, dan pesan yang akan dikirim berupa huruf "B" yang diasumsikan nilai desimal "B = 2", dan dengan menggunakan nilai dari proses sebelumnya yaitu p = 2, q = 7, k = 12, ϕ(n1) = 6,. Adapun perhitungan yang dilakukan alice dan bob yaitu sebagai berikut

Pada tahap ini akan dilakukan oleh bob, dimana bob diasumsikan sebagai penerima pesan, dan dalam kasus ini bob menerima pesan dari alice berupa *chipertext* (c) dari proses sebelumnya yang bernilai 43, lalu

bob melakukan perhitungan untuk mengembalikan nilai tersebut ke aslinya yaitu “B = 2” dengan melakukan proses dekripsi, adapun perhitungan yang dilakukan bob adalah sebagai berikut:

Langkah 1: Bob melakukan perhitungan untuk membangkitkan nilai *privat key* (d’) nya dengan menggunakan rumus (10), dimana pada kasus ini bob akan mencari nilai m yang apabila dilakukan perhitungan kepada nilai m dari rumus (10) akan menghasilkan bilangan bulat, sehingga bob mendapati i = 9 yang akan menghasilkan bilangan bulat, adapun perhitungannya sebagai berikut :

$$d' = \frac{1 + 9\ (6)}{5}$$

$$d' = 11$$

Keterangan :

d' = nilai kunci dekripsi

Langkah 2: Setelah bob mengetahui kunci dekripsi (d’) yaitu 11, lalu bob melakukan perhitungan untuk mencari nilai dp dan dq dengan menggunakan rumus (13) dan (14), yaitu sebagai berikut:

$$dp = 11\ mod\ (2 - 1)$$

$$dp = 11\ mod\ 1$$

$$dp = 0$$

Keterangan :

dp = nilai hasil perhitungan dari nilai d’ dan p

$$dq = 11\ mod\ (7 - 1)$$

$$dq = 11\ mod\ 6$$

$$dq = 5$$

Keterangan :

dq = nilai hasil perhitungan dari nilai d' dan q

Langkah 3: Setelah bob mengetahui nilai dari dp dan dq yaitu 0 dan 5, maka selanjutnya bob melakukan perhitungan untuk mengetahui nilai dai *qInv* dengan menggunakan rumus (15), yaitu sebagai berikut:

$$qInv = 2^{-1} \ mod \ 7$$

$$qInv = 4$$

Keterangan :

$qInv$ = nilang hasil perhitungan modulus invers q dan p

Langkah 4: Setelah bob mengetahui nilai *qInv* yaitu 4, selanjutnya bob melakukan perhitungan untuk menemukan nilai dari m1 dan m2 dengan menggunakan rumus (16) dan (17), yaitu sebagai berikut:

$$m1 = 43^0 \ mod \ 2$$

$$m1 = 1 \ mod \ 2$$

$$m1 = 1$$

Keterangan :

$m1$ = nilai hasil perhitungan *chipertext*

$$m2 = 43^5 \ mod \ 7$$

$$m2 = 147008443 \ mod \ 7$$

$$m2 = 1$$

Keterangan :

$m2$ = nilai hasil perhitungan *chipertext*

Langkah 5: Setelah bob mengetahui nilai dari m1 dan m2 yaitu 1 dan 2, selanjutnya bob melakukan perhitungan untuk menemukan nilai h menggunakan rumus (18), yaitu sebagai berikut:

$$h = 4\ x\ (1 - 1)\ mod\ 2$$

$$h = 4\ x\ (0) mod\ 2$$

$$h = 0\ mod\ 2$$

$$h = 0$$

Keterangan:

h = nilai hasil perhitungan nilai *qInv*, m1, m2 dan p

Langkah 6: Setelah bob mengetahui nilai h yaitu 0, maka selanjutnya bob melakukan perhitungan untuk mengetahui nilai dari plaintext yang sebenarnya dengan menggunakan rumus (19), yaitu sebagai berikut:

$$md = 2 + (0\ x\ 4)$$

$$md = 2 + 0$$

$$md = 2$$

Keterangan:

md = nilai *plaintext* hasil dekripsi

Dari hasil perhitungan diatas, dapat dilihat, bob mendapati hasil dari deskripsi menggunakan algoritma *RSA-CRT* ini dengan nilai $md = 2$, dimana nilai tersebut sama dengan nilai dari plaintext (m) atau pesan yang dikirimkan alice yaitu “B = 2”.

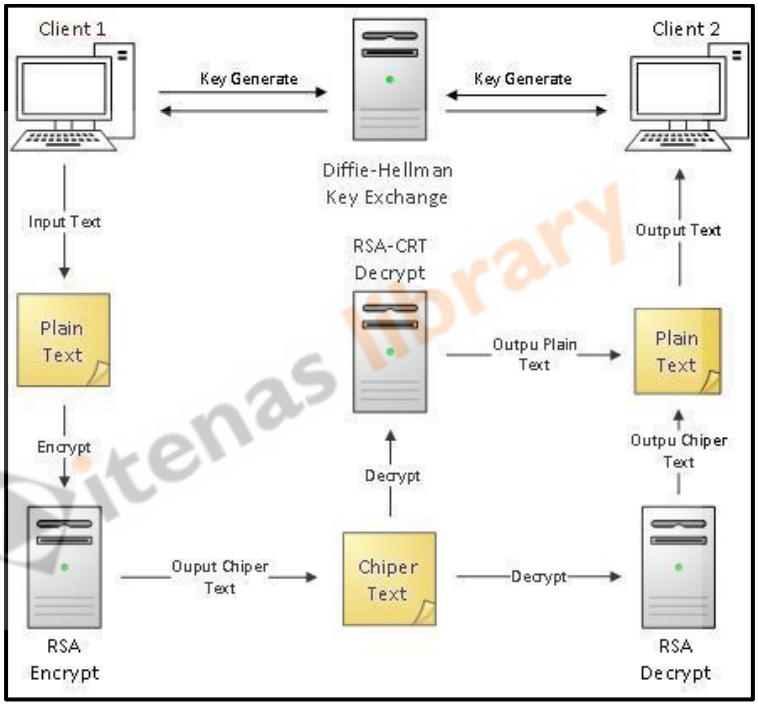
BAB III

METODE PENELITIAN

Pada bab ini dijelaskan metode yang dipakai dalam penelitian, uraian perancangan dan penelitian meliputi perancangan untuk tahap input hingga output yang dihasilkan.

3.1. Alur Kerja Sistem

Pada bagian ini, akan dijelaskan alur kerja sistem dari input pengguna hingga dapat melakukan enkripsi dan dekripsi pesan, pada gambar 3 akan ditampilkan cara kerja sistem secara keseluruhan.



Gambar 3 Alur kerja sistem

Pada gambar 3, terdapat dua *client* dan empat langkah proses yang dilakukan sistem yang dibangun untuk melakukan enkripsi dan dekripsi pada pesan, yaitu sebagai berikut:

1. Langkah pertama yang dilakukan adalah kedua pengguna yaitu *client* 1 dan *client* 2, melakukan pembangkitan *secret key* dengan melakukan proses pertukaran kunci dengan menggunakan algoritma *Diffie-Hellman.* Dimana nilai dari *secret key* tersebut akan digunakan untuk proses selanjutnya dan hanya akan dibangkitkan satu kali pada proses *Diffie-Hellman*.