BAB II
LANDASAN TEORI

2.1. Kriptografi

Kriptografi adalah ilmu yang menjaga keamanan pesan. Sedangkan seseorang dianggap pakar dalam masalah kriptografi sering kali disebut kriptografer. Kata *cryptography* berasal dari kata Yunani *kryptos* (tersembunyi) dan *graphein* (menulis).

Dalam kriptografi terdapat beberapa metode yang cukup penting dalam pengamanan data, untuk menjaga kerahasiaan suatu data salah satunya adalah enkripsi (*encryption*). Enkripsi adalah suatu proses yang dilakukan untuk mengubah pesan asli menjadi *ciphertext*. Sedangkan suatu proses yang dilakukan untuk mengubah pesan tersembunyi menjadi pesan biasa (yang mudah dibaca) disebut deskripsi. Pesan biasa atau pesan asli disebut *plaintext* sedangkan pesan yang telah diubah atau disandikan supaya tidak mudah dibaca disebut dengan *ciphertext*.

Sistem kriptografi kunci simetris biasanya menggunakan kunci yang sama dalam melakukan enkripsi dan dekripsi. Kelemahan yang nampak dari sistem tersebut adalah pengaturan kunci untuk keamanan penggunanya. Intinya tiap pasangan yang berkomunikasi harus memiliki kunci yang berbeda. Jumlah kunci bertambah seiring dengan bertambahnya anggota dalam jaringan sehingga membutuhkan skema pengaturan kunci yang kompleks untuk menjaganya agar tetap teratur dan rahasia.

Algoritma asimetris merupakan algoritma yang didesain agar kunci ynag digunakan pada proses enkripsi dan dekripsi berbeda. Pada algoritma asimetris, kunci publik dapat dikutahui semua orang tetapi pesan hanya dapat didekripsi oleh pengguna yang memiliki pasangan dari kunci publik yaitu kunci privat. Algoritma asimetris menggunakan kunci publik untuk melakukan proses enkripsi dan kunci privat untuk melakukan proses dekripsi. Secara umum, setiap pengguna memiliki sepasang kunci yaitu kunci publik dan kunci privat.

2.2. Algoritma Rivest-Shamir-Adleman (RSA)

2.2.1. Definisi Algoritma RSA

Algoritma RSA adalah metode kriptografi yang ditemukan oleh Ronald L. Rivest, Adi Shamir dan Leonard Adleman pada tahun 1977. Algoritma RSA adalah sebuah blok cipher algorithm (algoritma yang bekerja per blok data) yang mengelompokkan *plaintext* menjadi blok-blok terlebih dahulu sebelum dilakukan enkripsi hingga menjadi *ciphertext*.

RSA adalah salah satu algoritma penyandian yang paling banyak mengundang kontroversi, selain DES (*Data Encryption Standard*). Sejauh ini belum seorang pun yang berhasil menemukan celah keamanan pada DES dan RSA, tetapi tak seorang pun juga yang berhasil memberikan pembuktian ilmiah yang memuaskan dari keamanan kedua teknik sandi ini.

Untuk menyandi informasi dan untuk menerjemahkan pesan tersandi sebuah algoritma penyandian memerlukan sebuah data biner yang disebut kunci. Tanpa kunci yang cocok orang tidak bisa mendapatkan kembali pesan asli dari pesan tersandi. Pada DES digunakan kunci yang sama untuk menyandi (enkripsi) maupun untuk menterjemahan (dekripsi), sedangkan RSA menggunakan dua kunci yang berbeda. Isitilahnya, DES disebut sistem sandi simetris sementara RSA disebut sistem sandi *asimetris*. Kedua sistem ini memiliki keuntungan dan kerugiannya sendiri.

Sistem sandi simetris cenderung jauh lebih cepat sehingga lebih disukai oleh sementara kalangan industri. Kekurangannya, pihak-pihak yang ingin berkomunikasi secara privat harus punya akses ke sebuah kunci DES bersama. Walaupun biasanya pihak-pihak yang terkait sudah saling percaya, skema ini memungkinkan satu pihak untuk memalsukan pernyataan dari pihak lainnya.

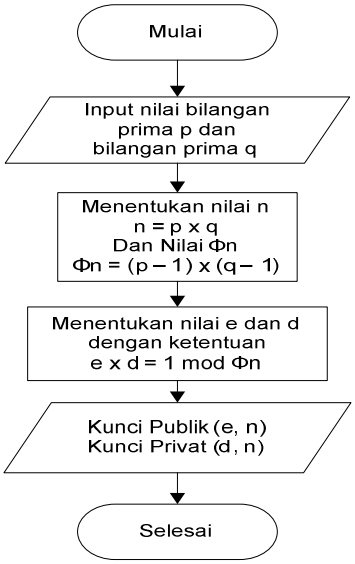
RSA termasuk algoritma asimetris yang berarti memiliki sepasang kunci, yaitu kunci publik dan kunci privat. Dalam RSA hanya digunakan satu algoritma untuk melakukan enkripsi dan deskripsi. Peberdaannya hanya terletak pada eksponen yang digunakan. Kunci public (n, e) sebagai kunci enkripsi dan kunci privat (n, d) sebagai kunci deskripsi dimana d, e dan n adalah bilangan bulat positif.

2.2.2. Kelebihan dan Kekurangan Algoritma RSA

Meskipun menjadi sistem standar dunia, RSA memiliki kelebihan dan kekurangan antara lain (Munir, 2006)

1) Kelebihan RSA
   a. Tingkat keamanan berlapis karena memakai dua kunci yang berbeda pada proses enkripsi dan deskripsinya.
   b. Dapat digunakan sebagai tanda tangan digital, sehingga penyangkalan terhadap sesuatu aksi dapat dicegah.
   c. Distribusi kunci menjadi lebih mudah karena jalur aman untuk distribusi kunci tidak lagi diperlukan.
   d. Manajemen kunci menjadi lebih mudah karena tiap pelaku sistem informasi memiliki sepasang kunci, maka untuk n pelaku dibutuhkan total $2n$ kunci saja.
   e. Sulitnya memfaktorkan bilangan non prima menjadi faktor prima.
2) Kekurangan RSA
   a. Kecepatan operasi yang jauh lebih lambat daripada kriptografi simetrik.
   b. Ukuran cipher menjadi sekitar 2 kali lipat ukuran semula.

2.2.3. Proses Pembangkitan Kunci



Gambar 2.1 Flowchart Langkah Pembangkitan Kunci Algoritma RSA

Adapun penjelasan dari gambar 2.1 adalah sebagai berikut :

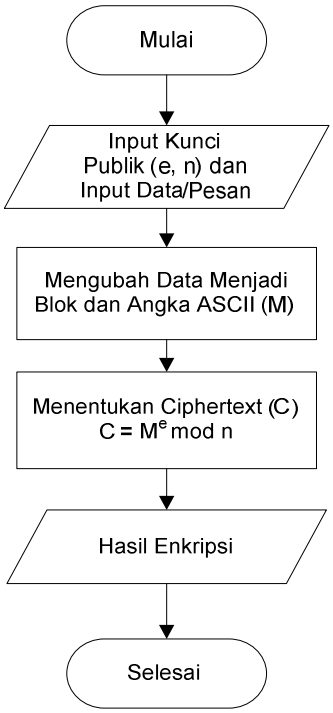
1) Pilih dua bilangan prima p dan q dimana $p \neq q$
2) Hitung $n = p \times q$ dan $\phi(n) = (p-1)(q-1)$
3) Menentukan nilai e secara acak, $1 < e < \phi$ dan prima relatif terhadap ϕ. Prima relatif maksudnya jika faktor persekutuan terbesar (*FPB*) dari e dan ϕ sama dengan satu ($FPB(e, \phi) = 1$). Dan hitung bilangan bulat positif d yang unik, $1 < d < \phi$ dimana $d = e^{-1} \bmod \phi$ sehingga $e.d = 1 \pmod{\phi}$ atau $e.d = k(\phi) + 1$ untuk suatu bilangan bulat k.
4) Didapat kunci publik berupa pasangan (n, e) dan kunci privat (n, d) dimana e adalah *public exponent*, d adalah *private exponent* dan n adalah modulus.
5) Nilai e dan n dipublikasikan tapi nilai d, p, q harus tetap dirahasiakan.

2.2.4. Proses Enkripsi

Proses enkripsi dengan algoritma RSA dilakukan dengan menghitung *exponen plaintext* dalam operasi modulo n (modulo = sisa pembagian) untuk setiap blok pesan atau data sehingga dapat menghasilkan ciphertext. Eksponen yang digunakan adalah public exponent e. Operasi ini bisa dituliskan dengan persamaan berikut :

$$C = M^e \bmod n \tag{2.1}$$

C = ciphertext
M = pesan (plaintext)
e = public exponent
n = modulus



Gambar 2.2 Flowchart Langkah Proses Enkripsi Algoritma RSA

Penjelasan dari gambar 2.2 mengenai langkah-langkah dalam proses enkripsi algoritma *RSA* adalah sebagai berikut :

1) Menginputkan kunci publik yang berupa pasangan *(e, n)* beserta pesan atau data yang akan dienkripsi.
2) Representation pesan atau data menjadi blok-blok dan diubah menjadi bilangan bulat positif *M*
3) Hitung $C = M^e \ mod \ n$.
4) Dan dihasilkan ciphertext (C) yang merupakan hasil dari enkripsi.

2.2.5. Proses Dekripsi

Sedangkan pada proses deskripsi, yang dilakukan hampir sama dengan enkripsi tapi eksponen yang digunakan adalah private exponent d untuk mengembalikan pesan seperti semula. Operasi ini bisa dituliskan dengan persamaan berikut :

$$M = C^d \ mod \ n$$

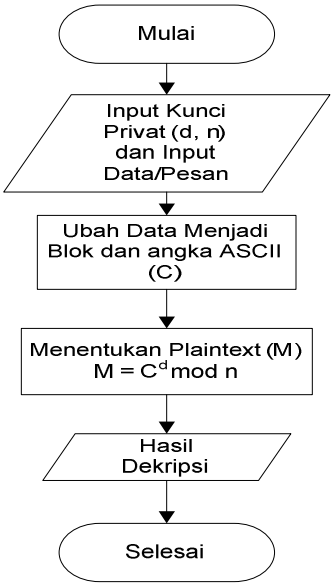
(2.2)

C = ciphertext

d = private exponent

n = modulus

M = pesan (plaintext)



Gambar 2.3 Flowchart Langkah Proses Dekripsi Algoritma *RSA*

Penjelasan dari gambar 2.3 mengenai langkah-langkah dalam proses dekripsi algoritma *RSA* adalah sebagai berikut :

5) Menginputkan kunci privat yang berupa pasangan (d, n) beserta pesan atau data yang akan didekripsi.
6) Representation pesan atau data menjadi blok-blok dan diubah menjadi bilangan bulat positif C
7) Hitung $M = C^e \bmod n$.
8) Dan dihasilkan plaintext (M) yang merupakan hasil dari dekripsi dan merupakan pesan atau data yang sebenarnya.

2.3. Algoritma ElGamal

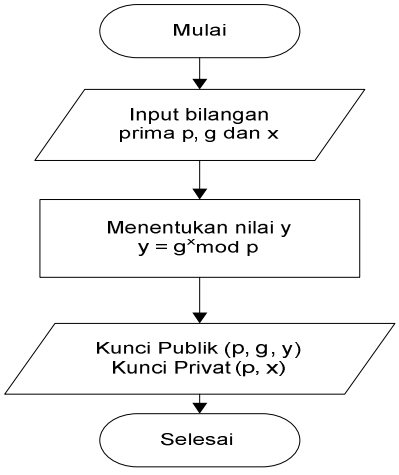
2.3.1. Definisi Algoritma ElGamal

Algoritma Elgamal diciptakan oleh ilmuan mesir, yaitu Taher ElGamal pada tahun 1984. Algoritma ElGamal merupakan algoritma dalam kriptografi yang termasuk dalam kategori algoritma asimetris. Keamanan algoritma ElGamal terletak pada kesulitan perhitungan logaritma diskret pada bilangan modulo prima yang besar sehingga upaya untuk menyelesaikan masalah logaritma ini menjadi sangat sukar.

Algoritma ElGamal mempunyai kunci publik yang berupa tiga pasang bilangan dan kunci rahasia berupa satu bilangan. Algoritma ini mempunyai kerugian pada cipherteksnya yang mempunyai panjang dua kali lipat dari plainteksnya. Akan tetapi, algoritma ini mempunyai kelebihan pada enkripsi. Untuk plainteks yang sama, algoritma ini memberikan cipherteks yang berbeda (dengan kepastian yang dekat) setiap kali plainteks di enkripsi.

Algoritma ElGamal terdiri dari tiga proses, yaitu proses pembentukan kunci, proses enkripsi dan proses dekripsi. Sama halnya dengan algoritma RSA, algoritma ElGamal juga merupakan cipher blok, yaitu melakukan proses enkripsi pada blok-blok plainteks dan menghasilkan blok-blok cipherteks yang kemudian dilakukan proses dekripsi dan hasilnya digabungkan.

2.3.2. Proses Pembentukan Kunci



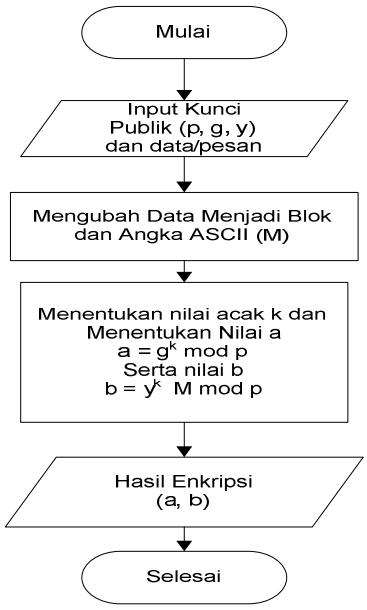
Gambar 2.4 Flowchart Langkah Pembangkitan Kunci ElGamal

Pembentukan kunci terdiri atas pembentukan kunci publik dan kunci privat. Dalam gambar 2.4, pada proses ini dibutuhkan sebuah bilangan prima *p* yang digunakan untuk membentuk dua bilangan acak *g* dan *x* dimana $g < p$ dan $1 \leq x \leq p - 2$. Kunci publik algoritma ElGamal terdiri atas pasangan 3 bilangan *{p, g, y}* dimana

$$y = g^x \ mod \ p \tag{2.3}$$

sedangkan untuk kunci privatnya terdiri atas dua bilangan *{p, x}.*

2.3.3. Proses Enkripsi



Gambar 2.5 Flowchart Langkah Enkripsi Algoritma ElGamal

Pada gambar 2.5 diatas Proses enkripsi membutuhkan kunci publik (*p, g, y*) dan sebuah bilangan integer acak *k* dengan ketentuan $1 \leq k \leq p - 1$ yang dijaga kerahasiaannya oleh yang mengenkripsi pesan atau teks. Untuk setiap karakter dalam pesan dienkripsi menggunakan bilangan *k* yang berbeda-beda. Satu karakter yang direpresentasikan dengan menggunakan bilangan bulat ASCII akan menghasilkan kode dalam bentuk blok yang terdiri atas dua nilai *(a, b)*.

Langkah proses enkripsi :

a. Ambil sebuah karakter dalam pesan yang akan dienkripsi dan transformasi karakter tersebut kedalam kode ASCII sehingga diperoleh bilangan bulat *M*. Adapun daftar kode ASCII masing-masing karakter dapat dilihat pada gambar 2.6 dan gambar 2.7 berikut

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	⌂

Gambar 2.6 Kode-kode ASCII Bagian 1

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	ø
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	⌂

Gambar 2.7 Kode-kode ASCII Bagian 2

b. Hitung nilai *a* dengan rumus

$$a = g^k \bmod p \tag{2.4}$$

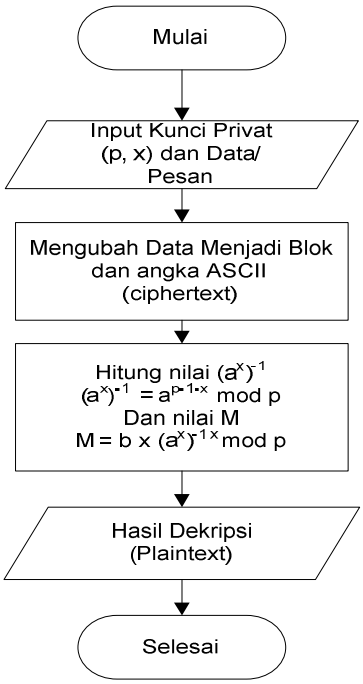
dan hitung nilai *b* dengan rumus $b = y^k M \bmod p$

$$b = y^k M \bmod p \tag{2.5}$$

c. Diperoleh cipherteks untuk karakter *M* tersebut dalam blok *(a, b)*.

d. Lakukan proses diatas untuk seluruh karakter dalam pesan termasuk karakter spasi.

2.3.4. Proses Dekripsi



Gambar 2.8 Flowchar Langkah Dekripsi Algoritma *ElGamal*

Pada gambar 2.8 diatas, dekripsi dari cipherteks menggunkan kunci privat *(p, x)* yang disimpan kerahasiaannya.

Langkah proses dekripsi :

a. Ambil sebuah blok cipherteks dari pesan yang telah dienkripsikan.

b. Dengan menggunakan x dan hitung nilai plainteks dengan menggunakan rumus

$$(a^x)^{-1} = a^{p-1-x} \, mod \, p \tag{2.6}$$

Kemudian hitung dengan rumus

$$M = b \times (a^x)^{-1} \, mod \, p \tag{2.7}$$

2.4. Penelitian Sebelumnya

Sebagai bahan pertimbangan dalam penelitian ini akan dicantumkan hasil penelitian terdahulu oleh peneliti yang pernah penulis baca diantaranya penelitian yang dilakukan oleh M. Taufiq Tamam, Wahyu Dwiono dan Tri Hartanto pada tahun 2010 dengan judul Penerapan Algoritma Kriptografi ElGamal untuk Pengamanan File Citra, yaitu dengan melakukan proses enkripsi dan dekripsi suatu file citra tipe bitmap dengan format piksel 24 bit. File citra tersebut dienkripsi menggunakan algoritma ElGamal dan citra yang dihasilkan menggunakan ekstensi “Este”.

Adapun penelitian lainnya adalah penelitian yang dilakukan oleh Duwi Mujianto pada tahun 2012 dengan judul Aplikasi Pengamanan Data Menggunakan Algoritma RSA (Rivest-Shamir-Adleman) yaitu dengan melakukan enkripsi data pada file teks (.txt) menggunakan algoritma kriptografi asimetris RSA sehingga menghasilkan data berupa bilangan desimal yang kemudian disimpan di dalam file yang mempunyai ekstensi sck (*.sck). Untuk proses dekripsinya, data yang terdapat pada file berekstensi sck tersebut diubah kembali menjadi data yang sebenarnya dan kembali disimpan di file teks (file *.txt).