2. DASAR TEORI

2.1. Rangkaian Bit Biner

Bit adalah suatu satuan informasi yang berupa data biner, 0 dan 1. Pada umumnya, bit digunakan untuk mengintepretasikan informasi dalam media digital atau data digital. Bit juga digunakan dalam skema koding ASCII (*American Standard Code for Information Interchange*), yang mengintepretasikan karakter dalam bentuk biner, decimal, oktadesimal maupun heksadesimal (lihat Tabel 2.1.).

Tabel 2.1. Daftar Kode ASCII

BIN	DEC	OCT	HEX	CHR		BIN	DEC	OCT	HEX	CHR
00000000	000	000	00	NUL	(null)	10000000	128	200	80	Ç
00000001	001	001	01	SOH	(start of heading)	10000001	129	201	81	ü
00000010	002	002	02	STX	(start of text)	10000010	130	202	82	é
00000011	003	003	03	ETX	(end of text)	10000011	131	203	83	â
00000100	004	004	04	EOT	(end of transmission)	10000100	132	204	84	ä
00000101	005	005	05	ENQ	(enquiry)	10000101	133	205	85	à
00000110	006	006	06	ACK	(acknowledge)	10000110	134	206	86	å
00000111	007	007	07	BEL	(bell)	10000111	135	207	87	ç
00001000	008	010	08	BS	(backspace)	10001000	136	210	88	ê
00001001	009	011	09	TAB	(horizontal tab)	10001001	137	211	89	ë
00001010	010	012	0A	LF	(NL line feed, new line)	10001010	138	212	8A	è
00001011	011	013	0B	VF	(vertical tab)	10001011	139	213	8B	ï
00001100	012	014	0C	FF	(NP form feed, new page)	10001100	140	214	8C	î
00001101	013	015	0D	CR	(carriage return)	10001101	141	215	8D	ì
00001110	014	016	0E	SO	(shift out)	10001110	142	216	8E	Ä
00001111	015	017	0F	SI	(shift in)	10001111	143	217	8F	Å
00010000	016	020	10	DLE	(data link escape)	10010000	144	220	90	É
00010001	017	021	11	DC1	(device control 1)	10010001	145	221	91	æ
00010010	018	022	12	DC2	(device control 2)	10010010	146	222	92	Æ
00010011	019	023	13	DC3	(device control 3)	10010011	147	223	93	ô
00010100	020	024	14	DC4	(device control 4)	10010100	148	224	94	ö
00010101	021	025	15	NAK	(negative acknowledge)	10010101	149	225	95	ò
00010110	022	026	16	SYN	(synchronous idle)	10010110	150	226	96	û
00010111	023	027	17	ETB	(end of trans. block)	10010111	151	227	97	ù
00011000	024	030	18	CAN	(cancel)	10011000	152	230	98	ÿ
00011001	025	031	19	EM	(end of medium)	10011001	153	231	99	Ö
00011010	026	032	1A	SUB	(substitute)	10011010	154	232	9A	Ü
00011011	027	033	1B	ESC	(escape)	10011011	155	233	9B	¢

00011100	028	034	1C	FS	(file separator)	10011100	156	234	9C	£
00011101	029	035	1D	GS	(group separator)	10011101	157	235	9D	¥
00011110	030	036	1E	RS	(record separator)	10011110	158	236	9E	Pts
00011111	031	037	1F	US	(unit separator)	10011111	159	237	9F	*f*
00100000	032	040	20	Space		10100000	160	240	A0	á
00100001	033	041	21	!		10100001	161	241	A1	í
00100010	034	042	22	‘’		10100010	162	242	A2	ó
00100011	035	043	23	#		10100011	163	243	A3	ú
00100100	036	044	24	$		10100100	164	244	A4	ñ
00100101	037	045	25	%		10100101	165	245	A5	Ñ
00100110	038	046	26	&		10100110	166	246	A6	ª
00100111	039	047	27	‘		10100111	167	247	A7	º
00101000	040	050	28	(		10101000	168	250	A8	¿
00101001	041	051	29	)		10101001	169	251	A9	⌐
00101010	042	052	2A	*		10101010	170	252	AA	¬
00101011	043	053	2B	+		10101011	171	253	AB	½
00101100	044	054	2C	,		10101100	172	254	AC	¼
00101101	045	055	2D	-		10101101	173	255	AD	¡
00101110	046	056	2E	.		10101110	174	256	AE	«
00101111	047	057	2F	/		10101111	175	257	AF	»
00110000	048	060	30	0		10110000	176	260	B0	░
00110001	049	061	31	1		10110001	177	261	B1	▒
00110010	050	062	32	2		10110010	178	262	B2	▓
00110011	051	063	33	3		10110011	179	263	B3	│
00110100	052	064	34	4		10110100	180	264	B4	┤
00110101	053	065	35	5		10110101	181	265	B5	╡
00110110	054	066	36	6		10110110	182	266	B6	╢
00110111	055	067	37	7		10110111	183	267	B7	╖
00111000	056	070	38	8		10111000	184	270	B8	╕
00111001	057	071	39	9		10111001	185	271	B9	╣
00111010	058	072	3A	:		10111010	186	272	BA	║
00111011	059	073	3B	;		10111011	187	273	BB	╗
00111100	060	074	3C	<		10111100	188	274	BC	╝
00111101	061	075	3D	=		10111101	189	275	BD	╜
00111110	062	076	3E	>		10111110	190	276	BE	╛
00111111	063	077	3F	?		10111111	191	277	BF	┐
01000000	064	100	40	@		11000000	192	300	C0	└
01000001	065	101	41	A		11000001	193	301	C1	┴
01000010	066	102	42	B		11000010	194	302	C2	┬
01000011	067	103	43	C		11000011	195	303	C3	├
01000100	068	104	44	D		11000100	196	304	C4	─
01000101	069	105	45	E		11000101	197	305	C5	┼
01000110	070	106	46	F		11000110	198	306	C6	╞
01000111	071	107	47	G		11000111	199	307	C7	╟
01001000	072	110	48	H		11001000	200	310	C8	╚
01001001	073	111	49	I		11001001	201	311	C9	╔

01001010	074	112	4A	J		11001010	202	312	CA	╩
01001011	075	113	4B	K		11001011	203	313	CB	╦
01001100	076	114	4C	L		11001100	204	314	CC	╠
01001101	077	115	4D	M		11001101	205	315	CD	═
01001110	078	116	4E	N		11001110	206	316	CE	╬
01001111	079	117	4F	O		11001111	207	317	CF	╧
01010000	080	120	50	P		11010000	208	320	D0	╨
01010001	081	121	51	Q		11010001	209	321	D1	╤
01010010	082	122	52	R		11010010	210	322	D2	╥
01010011	083	123	53	S		11010011	211	323	D3	╙
01010100	084	124	54	T		11010100	212	324	D4	╘
01010101	085	125	55	U		11010101	213	325	D5	╒
01010110	086	126	56	V		11010110	214	326	D6	╓
01010111	087	127	57	W		11010111	215	327	D7	╫
01011000	088	130	58	X		11011000	216	330	D8	╪
01011001	089	131	59	Y		11011001	217	331	D9	┘
01011010	090	132	5A	Z		11011010	218	332	DA	┌
01011011	091	133	5B	[		11011011	219	333	DB	█
01011100	092	134	5C	\		11011100	220	334	DC	▄
01011101	093	135	5D	]		11011101	221	335	DD	▌
01011110	094	136	5E	^		11011110	222	336	DE	▐
01011111	095	137	5F	_		11011111	223	337	DF	▀
01100000	096	140	60	`		11100000	224	340	E0	α
01100001	097	141	61	a		11100001	225	341	E1	ß
01100010	098	142	62	b		11100010	226	342	E2	Γ
01100011	099	143	63	c		11100011	227	343	E3	π
01100100	100	144	64	d		11100100	228	344	E4	Σ
01100101	101	145	65	e		11100101	229	345	E5	σ
01100110	102	146	66	f		11100110	230	346	E6	µ
01100111	103	147	67	g		11100111	231	347	E7	τ
01101000	104	150	68	h		11101000	232	350	E8	Φ
01101001	105	151	69	i		11101001	233	351	E9	Θ
01101010	106	152	6A	j		11101010	234	352	EA	Ω
01101011	107	153	6B	k		11101011	235	353	EB	δ
01101100	108	154	6C	l		11101100	236	354	EC	∞
01101101	109	155	6D	m		11101101	237	355	ED	φ
01101110	110	156	6E	n		11101110	238	356	EE	ε
01101111	111	157	6F	o		11101111	239	357	EF	∩
01110000	112	160	70	p		11110000	240	360	F0	≡
01110001	113	161	71	q		11110001	241	361	F1	±
01110010	114	162	72	r		11110010	242	362	F2	≥
01110011	115	163	73	s		11110011	243	363	F3	≤
01110100	116	164	74	t		11110100	244	364	F4	⌠
01110101	117	165	75	u		11110101	245	365	F5	⌡
01110110	118	166	76	v		11110110	246	366	F6	÷
01110111	119	167	77	w		11110111	247	367	F7	≈

01111000	120	170	78	x		11111000	248	370	F8	°
01111001	121	171	79	y		11111001	249	371	F9	·
01111010	122	172	7A	z		11111010	250	372	FA	·
01111011	123	173	7B	{		11111011	251	373	FB	√
01111100	124	174	7C	|		11111100	252	374	FC	ⁿ
01111101	125	175	7D	}		11111101	253	375	FD	²
01111110	126	176	7E	~		11111110	254	376	FE	■
01111111	127	177	7F	DEL		11111111	255	377	FF	

Sumber: http://www.table-ascii.com(2008)

Algoritma dalam kriptografi modern beroperasi dalam mode bit, bukan dalam mode karakter seperti pada kriptografi klasik. Hal ini berarti informasi yang akan diproses dengan algoritma kriptografi modern harus diterjemahkan dalam rangkaian bit-bit. Rangkaian bit-bit tersebut dapat dipecah menjadi sejumlah blok-blok bit, tergantung pada besarnya panjang blok yang dibutuhkan dalam algoritma kriptografi yang digunakan.

Sebagai contoh, misalnya sebuah teks “wisuda” diterjemahkan menjadi rangkaian bit 011101110110100101110011011101010110010001100001, dimana rangkaian bit tersebut diperoleh dari mengintepretasikan karakter-karakter yang ada dengan nilai binernya sebagai berikut

w i s u d a

01110111 01101001 01110011 01110101 01100100 01100001

Bila rangkaian bit ini dibagi menjadi blok-blok bit yang panjangnya 4 bit, maka akan diperoleh 12 blok bit sebagai berikut

0111 0111 0110 1001 0111 0011 0111 0101 0110 0100 0110 0001

7 7 6 9 7 3 7 5 6 4 6 1

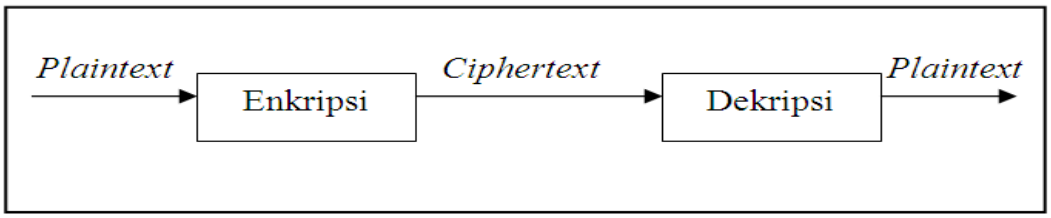
Bila rangkaian bit ini dibagi menjadi blok-blok bit yang panjangnya 3 bit, maka akan diperoleh 16 blok bit sebagai berikut

011 101 110 110 100 101 110 011 011 101 010 110 010 001 100 001

3 5 6 6 4 5 6 3 3 5 2 6 2 1 4 1

2.2. Kriptografi

Kriptografi adalah ilmu yang mempelajari kerahasiaan pesan. Secara terminologi, kriptografi berasal dari dua kata dalam bahasa Yunani, *κρυπτός* (baca: *kryptós*) yang berarti tersembunyi dan *γράφω* (baca: *gráfo*) yang berarti tulisan, yang bila diartikan secara bebas berarti tulisan yang tersembunyi. Bruce Schneiner dalam bukunya Applied Cryptography, 2nd ed. (1996) mendefinisikan kriptografi sebagai ilmu dan seni untuk menjaga keamanan pesan (*cryptography is the art and science of keeping messages secure*). Secara umum, kriptografi merupakan teknik pengamanan informasi yang dilakukan dengan cara mengolah informasi awal (atau diistilahkan dengan *plaintext*) dengan suatu kunci tertentu menggunakan suatu metode enkripsi tertentu sehingga menghasilkan suatu informasi baru (atau diistilahkan dengan *chipertext*) yang tidak dapat dibaca secara langsung. *Chipertext* tersebut dapat dikembalikan menjadi informasi awal (*plaintext*) melalui proses dekripsi. Skema singkat mengenai proses tersebut dapat dilihat pada Gambar 2.1.



Gambar 2.1. Skema singkat proses kriptografi

2.2.1. Kriptografi Modern

Kriptografi modern dipicu oleh perkembangan peralatan komputer. Dengan berkembangnya komputer, proses kriptografi yang lebih kompleks menjadi sangat mungkin untuk dihasilkan. Selama jangka waktu yang relatif lama, kriptografi menjadi bidang khusus yang digeluti secara oleh pihak militer, misalnya National Security Agency (Agen Keamanan Nasional) di Amerika, Uni Soviet, Inggris, Perancis, Israel dan negara-negara maju lainnya. Negara-negara

tersebut membelanjakan miliaran dolar untuk mengamankan komunikasi mereka dari pihak luar, sembari terus-menerus mempelajari kode-kode rahasia negara lain. Dengan adanya persaingan seperti ini maka kriptografi terus berkembang sesuai dengan perkembangan dunia militer.

Dewasa ini, kriptografi tidak hanya dimonopoli oleh pihak militer saja. Hal yang sama juga dilakukan oleh pihak-pihak yang menginginkan komunikasi mereka tidak diketahui oleh pihak lain. Apalagi dalam dunia dengan persaingan yang sedemikian tinggi, banyak pihak yang rela mengeluarkan sejumlah uang hanya untuk menjaga kerahasiaan (*privacy*) mereka. Hal ini juga semakin memicu munculnya teknik-teknik kriptografi yang baru.

Sejarah kriptografi modern dimulai pada awal tahun 1970 dimana Horst Feistel, seorang ilmuwan pada International Business Machines Corporation (perusahaan IBM), membangun Lucifer, sebuah perangkat kriptografi yang terkomputerisasi. Sejak saat itu mulai bermunculan algoritma-algoritma kriptografi modern yang semakin beragam. Algoritma-algoritma kriptografi yang semula mayoritas dihasilkan oleh kalangan militer menjadi semakin banyak ketika kalangan akademis, seperti para peneliti, juga ikut ambil bagian dalam mengembangkan kriptografi.

Setidaknya terdapat dua perbedaan yang signifikan antara kriptografi klasik dengan kriptografi modern. Perbedaan yang pertama adalah kriptografi modern beroperasi dalam mode bit, tidak seperti kriptografi klasik yang beroperasi dalam mode karakter, semua bentuk enkripsi dilakukan pada masing-masing karakter. Operasi dalam mode bit berarti semua informasi (baik plainteks, kunci maupun cipherteks) direpresentasikan dalam rangkaian (*string*) bit biner, 0 dan 1. Algoritma enkripsi dan dekripsi memproses semua informasi dalam bentuk rangkaian bit. Perkembangan kriptografi modern dalam mode bit dipicu oleh penggunaan komputer yang merepresentasikan semua informasi dalam bentuk biner. Tentu saja hal tersebut juga berpengaruh pada media penunjang kriptografi dimana pada kriptografi modern dibutuhkan perangkat komputer untuk melakukan proses kriptografi. Hal ini terjadi karena algoritma yang digunakan

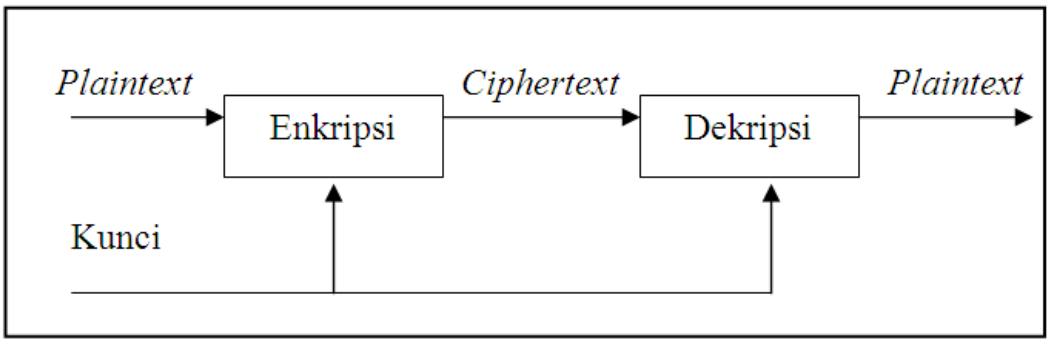
memiliki tingkat kerumitan yang relatif tinggi (tentunya dengan tingkat keamanan yang relatif tinggi juga).

Semakin banyaknya algoritma-algoritma kriptografi juga disertai dengan keberagaman tingkat kompleksitas yang berbeda-beda. Mulai dari yang sangat sederhana sampai yang sangat kompleks, Berdasarkan hal tersebut, maka muncul klasifikasi-klasifikasi tertentu terhadap kriptografi. Selain klasifikasi-klasifikasi, keberagaman algoritma kriptografi ternyata juga memunculkan adanya kebutuhan akan suatu standar yang dapat diakui dan dimanfaatkan secara global. Ini adalah perbedaan yang kedua, yaitu diterapkannya standar-standar tertentu dalam penggunaan kriptografi.

Publikasi pertama mengenai kriptografi terjadi di *United States Federal Register* pada tanggal 17 Maret 1975 dimana pada kesempatan itu dipublikasikan algoritma *Data Encryption Standard* (DES). Publikasi tersebut dilakukan dalam upaya untuk mengembangkan fasilitas komunikasi elektronik yang aman untuk kalangan bisnis, seperti perbankan maupun organisasi keuangan yang lain. Setelah mengalami beberapa perbaikan, pada tahun 1977, *United States' National Bureau of Standard* [NBS] (Biro Standar Nasional Amerika) bersama *National Security Agency* (Agen Keamanan Nasional) mengadopsi *Data Encryption Standard* (DES) dan mempublikasikannya dalam dokumen *Federal Information Processing Standard Publication* dengan kode *FIPS PUB 46-3*. Adapun pada Januari 1989, United States' National Bureau of Standard Biro telah berganti nama menjadi National Institute of Standard and Technology [NIST] (Institut Standar dan Teknologi). Publikasi algoritma dan spesifikasi oleh Biro Standar Nasional Amerika ini menstimulasi ketertarikan masyarakat umum dan kalangan akademis untuk semakin mengembangkan kriptografi.

Peristiwa selanjutnya yang signifikan terhadap perkembangan kriptografi, bahkan secara fundamental mempengaruhi cara sistem kriptografi bekerja terjadi pada tahun 1976. Peristiwa itu adalah munculnya publikasi sebuah makalah dengan judul "*New Direction of Cryptography*" yang diajukan oleh Whitfield Diffie dan Martin Hellman. Makalah tersebut memperkenalkan suatu metode baru yang radikal untuk masalah pendistribusian kunci dalam kriptografi,

yang merupakan salah satu masalah besar dalam kriptografi. Metode baru yang ditawarkan dalam makalah tersebut dikenal dengan nama metode pertukaran kunci Diffie-Hellman (*Diffie-Hellman key exchange*). Metode ini menawarkan penggunaan dua buah kunci dalam proses kriptografi, yaitu kunci publik dan kunci privat, dimana kedua buah kunci tersebut mempunyai relasi matematis satu dengan yang lain. Metode ini memungkinkan pihak pengirim pesan dan pihak penerima pesan saling memperoleh informasi yang sama melalui saluran publik, yang secara umum, tingkat keamanannya kurang terjamin. Munculnya makalah tersebut juga menstimulasi perkembangan satu algoritma baru pada penggunaan kriptografi, yaitu algoritma kunci nirsimetri.



Gambar 2.2. Skema kriptografi modern dengan kunci

2.2.2. Tujuan Kriptografi

Pada dasarnya, perkembangan kriptografi bertujuan untuk memberikan layanan keamanan, yang juga dinamakan sebagai aspek-aspek keamanan(Munir,2006) dalam pertukaran informasi. Adapun aspek-aspek keamanan tersebut antara lain[1]:

- Kerahasiaan data (*data confidentiality*), adalah layanan yang ditujukan untuk menjaga agar informasi tidak dapat dibaca oleh pihak-pihak yang tidak memiliki hak untuk itu. Di dalam kriptografi, biasanya hal ini diterapkan dengan mengenkripsi informasi menjadi *cipher* teks.

- Integritas data (*data integrity*), adalah layanan yang menjamin bahwa informasi tersebut adalah benar (asli dan utuh) dan belum mengalami manipulasi selama pengiriman informasi. Untuk menjaga integritas data, dalam kriptografi harus memiliki kemampuan untuk mengenali manipulasi (modifikasi) informasi oleh pihak-pihak yang tidak berhak.
- Otentikasi data (*data authentication*), adalah layanan untuk mengidentifikasi kebenaran pihak-pihak yang berkomunikasi (*user authentication*) ataupun kebenaran sumber informasi (*data origin authentication*). Dua pihak yang saling berkomunikasi harus dapat saling mengindentifikasi satu sama lain sehingga dapat memastikan sumber informasi. Otentikasi sumber informasi secara tidak langsung juga memberikan kepastian mengenai integritas data.
- Nirpenyangkalan (*non repudiation*), adalah layanan untuk mencegah salah satu pihak melakukan pengelakan (penyangkalan) informasi. Baik pihak pemberi informasi mengelak telah memberikan informasi maupun pihak penerima informasi mengelak telah menerima informasi. Dengan adanya layanan ini, pihak-pihak yang melakukan komunikasi tidak dapat menyangkali bahwa mereka telah mengirim atau menerima informasi.

2.2.3. Algoritma Kriptografi

Algoritma dalam kriptografi merupakan sekumpulan aturan (fungsi matematis yang digunakan) untuk proses enkripsi dan proses dekripsi. Dalam beberapa metode kriptografi terdapat perbedaan antara fungsi enkripsi dan fungsi dekripsi.

Konsep matematis yang mendasari algoritma kriptografi adalah relasi antara himpunan, yaitu relasi antara himpunan yang berisi elemen-elemen plainteks dengan himpunan yang berisi elemen-elemen cipherteks. Enkripsi dan dekripsi merupakan fungsi yang memetakan elemen-elemen antara kedua himpunan tersebut. Misalkan himpunan elemen plainteks dinotasikan *P* dan himpunan elemen cipherteks dinotasikan *C*, maka fungsi enkripsi *E* memetakan himpunan *P* ke himpunan *C*,

E (2.1)

dan fungsi dekripsi *D* memetakan himpunan *C* ke himpunan *P*,

D (2.2)

Karena fungsi dekripsi *D* mengembalikan himpunan *C* menjadi himpunan *P* asal, maka algoritma kriptografi harus memenuhi persamaan

$D\ E$ (2.3)

Tingkat keamanan suatu algoritma dalam kriptografi seringkali diukur dari kuantitas proses yang dilakukan dalam suatu fungsi, baik itu fungsi enkripsi maupun fungsi dekripsi. Proses tersebut juga dapat dihubungkan dengan sumber daya yang dibutuhkan untuk mengerjakan fungsi tersebut, misalnya dengan waktu, uang dan lain-lain. Semakin besar proses yang harus dikerjakan, yang berarti juga semakin lama waktu yang dibutuhkan, menunjukkan semakin kuat algoritma kriptografi tersebut.

Pada kriptografi klasik, keamanan kriptografi terletak pada kerahasiaan algoritma kriptografinya. Salah satu contohnya adalah mesin Enigma yang dikeluarkan oleh pemerintah Jerman pada masa Perang Dunia ke-2. Namun, hal ini dapat menjadi titik lemah ketika algoritma ini bocor ke pihak yang seharusnya tidak berwenang sehingga mengharuskan untuk menyusun suatu algoritma baru.

Kriptografi modern mengatasi masalah tersebut dengan penggunaan kunci (atau sandi). Dalam hal ini, algoritma kriptografi sudah tidak dirahasiakan lagi tetapi kunci yang digunakan yang harus dijaga kerahasiaannya. Kunci ini biasanya berupa deretan karakter tertentu. Kunci merupakan parameter tambahan yang digunakan dalam fungsi enkripsi dan fungsi dekripsi. Dengan menggunakan kunci *K*, fungsi enkripsi *E* dan dekripsi *D* dapat ditulis sebagai

E (2.4)

D (2.5)

dan kedua fungsi ini harus memenuhi persamaan

$D \quad E$ (2.6)

Berdasarkan kunci yang digunakan dalam proses enkripsi dan dekripsi pada kriptografi, algoritma kriptografi dapat diklasifikasikan menjadi dua jenis

algoritma, algoritma kunci simetri dan algoritma kunci nirsimetri. Perbedaan mendasar dari kedua algoritma ini terletak pada kunci yang digunakan untuk masing-masing proses enkripsi dan proses dekripsi.

Pada algoritma kunci simetri, kunci yang digunakan dalam proses enkripsi sama dengan kunci yang digunakan dalam proses dekripsi. Sedangkan pada algoritma kunci nirsimetri, kunci yang digunakan dalam proses enkripsi berbeda dengan kunci yang digunakan dalam proses dekripsi. Dengan demikian, jika kunci dalam proses enkripsi dinotasikan *K1* dan kunci dalam proses dekripsi dinotasikan *K2*, maka fungsi enkripsi *E* dan dekripsi *D* dituliskan sebagai

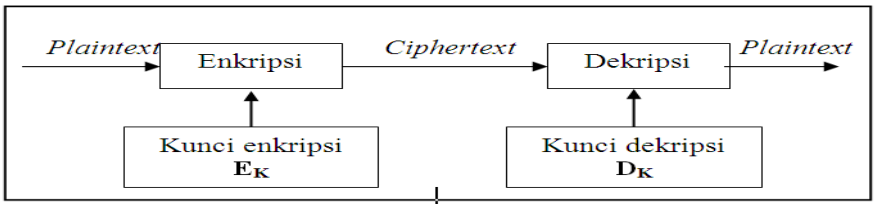
$$E \tag{2.7}$$

$$D \tag{2.8}$$

dan kedua fungsi ini juga harus memenuhi persamaan

$$D \quad E \tag{2.9}$$

Algoritma kunci nirsimetri, sering juga disebut algoritma kunci publik. Hal ini mengandung pengertian bahwa ada satu kunci yang dipublikasikan, yang dapat diketahui oleh semua orang. Kunci publik digunakan dalam proses enkripsi dan kunci privat digunakan dalam proses dekripsi. Dengan adanya kunci publik ini, salah satu pihak dapat mengenkripsi suatu informasi tetapi tidak dapat mendekripsikannya. Jadi informasi tersebut dapat ditransmisikan dengan aman tanpa rasa kuatir akan kebocoran informasi tersebut, sebab informasi tersebut hanya dapat didekripsikan oleh pihak yang memang berhak untuk mendekripsikan informasi tersebut, yaitu pihak yang memang mempunyai kunci privat.



Gambar 2.3. Ilustrasi kriptografi dengan kunci publik

2.3. Secure Hash Algorithm 1 (SHA-1)

Algoritma kriptografi kunci nirsimetri, atau yang sering juga disebut sebagai algoritma kriptografi kunci publik muncul karena adanya keterbatasan media perantara yang terjamin keamanannya untuk mendistribusikan kunci dalam kriptografi. Algoritma kriptografi kunci simetri menggunakan kunci yang sama untuk proses enkripsi maupun proses dekripsi, sehingga kedua belah pihak yang saling bertukar informasi harus sama-sama mempunyai kunci yang sama. Dari proses tersebut muncul permasalahan dalam pendistribusian kunci yang akan digunakan karena baik pihak pengirim informasi maupun pihak penerima informasi harus berbagi kunci yang sama. Saluran publik (seperti pos, telepon, internet, dan sebagainya) juga tidak memungkinkan karena tingkat keamanannya yang tidak jelas. Oleh karena itu, kunci harus dikirimkan melalui pihak ketiga yang terpercaya atau dengan pertemuan langsung. Namun, alternatif ini cenderung lebih lambat dan lebih mahal.

Dari berbagai macam algoritma kriptografi kunci nirsimetri yang ada, salah satu algoritma kriptografi kunci nirsimetri yaitu algoritma kriptografi SHA. SHA adalah fungsi *hash* satu-arah yang dibuat oleh NIST dan digunakan bersama DSS (*Digital Signature Standard*). Fungsi *hash* satu arah merupakan fungsi pembangun dasar yang dipakai dalam banyak aplikasi, digunakan bersama dengan algoritma kunci public dalam enkripsi dan tanda tangan digital.Oleh NSA, SHA dinyatakan sebagai standar fungsi *hash* satu arah. SHA didasarkan pada MD4 yang dibuat oleh Ronald L. Rivest dari MIT. SHA disebut aman (*secure*) karena dirancang sedemikian rupa sehingga secara komputasi tidak mungkin menemukan pesan yang berkoresponden dengan *message digest* yang diberikan.

Fungsi *hash* satu arah memiliki dua prinsip, yaitu hanya dapat dilakukan satu-arah, berarti dapat melakukan komputasi untuk mendapatkan nilai *hash* dari pesan asli, tetapi tidak mungkin mendapatkan pesan asli dari nilai *hash* tersebut. Kedua, bebas *collision*, berarti sangat tidak mungkin untuk mendapatkan nilai *hash* yang sama dari dua pesan yang berbeda.

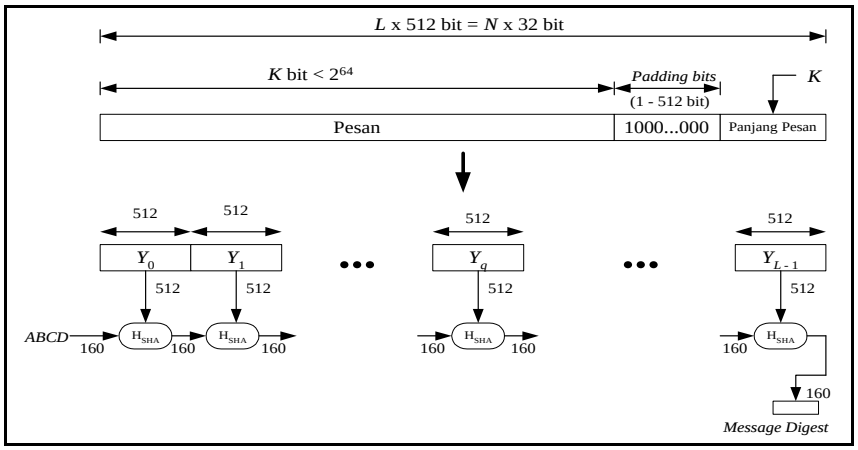
2.3.1. Sejarah Singkat *Secure Hash Algorithm* 1 (SHA-1)

Penelitian untuk suatu standar dalam kriptografi dimulai pada tahun 1973, dimana pada saat itu, *United States National Bureau of Standard* [NBS] (Biro Standar Nasional Amerika) menyelenggarakan kompetisi untuk menciptakan suatu standar kriptografi yang dapat digunakan secara umum. Kompetisi ini disambut baik oleh perusahaan *International Business Machines* (IBM) yang sejak tahun 1970 sudah melakukan penelitian kriptografi melalui proyek Lucifer.

Pada tahun 1990, Ron Rivest menemukan fungsi *hash* MD4. Pada tahun 1992, ia mengembangkan MD4 dan menghasilkan MD5. Pada tahun 1993, NIST mempublikasikan suatu fungsi *hash* baru yang menyerupai MD5 dan dinamakan SHA. Kemudian pada tahun 1995, ditemukan kelemahan dari algoritma tersebut sehingga NSA melakukan pengubahan, menghasilkan fungsi SHA-1.

2.3.2. Algoritma Kriptografi SHA-1

Algoritma SHA menerima inputan maksimal 2^{64} bit dan menghasilkan *message digest* yang panjangnya 160 bit, lebih panjang dari yang dihasilkan oleh MD5.



Gambar 2.4. Skema Algoritma SHA-1

Sumber: *Federal Informaiton Processing Standard Publication* 180-2

Proses Inisialisasi Internal

Penambahan bit pengganjal (*padding bit*), pesan ditambah dengan sejumlah bit pengganjal hingga mencapai 485 bit. Kemudian pesan tersebut ditambah lagi dengan 64 bit yang menyatakan panjang pesan sehingga mencapai kelipatan 512 bit. Hal ini bertujuan agar pesan dapat di proses oleh algoritma SHA1 yang memproses data dengan panjang 512 bit.

Inisialisasi bit penyangga (*buffer*), SHA1 membutuhkan 5 *buffer* sebesar 32 bit (MD5 hanya memakai 4 buah saja). Ke 5 penyangga ini akan dipakai untuk menyimpan hasil antara dan hasil akhir. Inisialisasinya sebagai berikut:

A: 67452301

B: EFCDAB89

C: 98BADCFE

D: 10325476

E: C3D2E1F0

Proses Enkripsi

Pada tahap awal, pesan dibagi menjadi L buah blok yang panjangnya masing-masing 512 bit, kemudian setiap blok diproses dengan penyangga menjadi keluaran 128 bit, dan ini disebut proses H_{SHA}. Gambaran skema proses H_{SHA} ini terlihat pada gambar 2.7.

Proses H_{SHA} sendiri terdiri dari 80 buah putaran (dimana MD5 hanya menggunakan 4 putaran), dan masing-masing putaran menggunakan bilangan penambah K, yaitu:

Putaran $0 \leq t \leq 19$ K_t = 5A827999 (2.10)

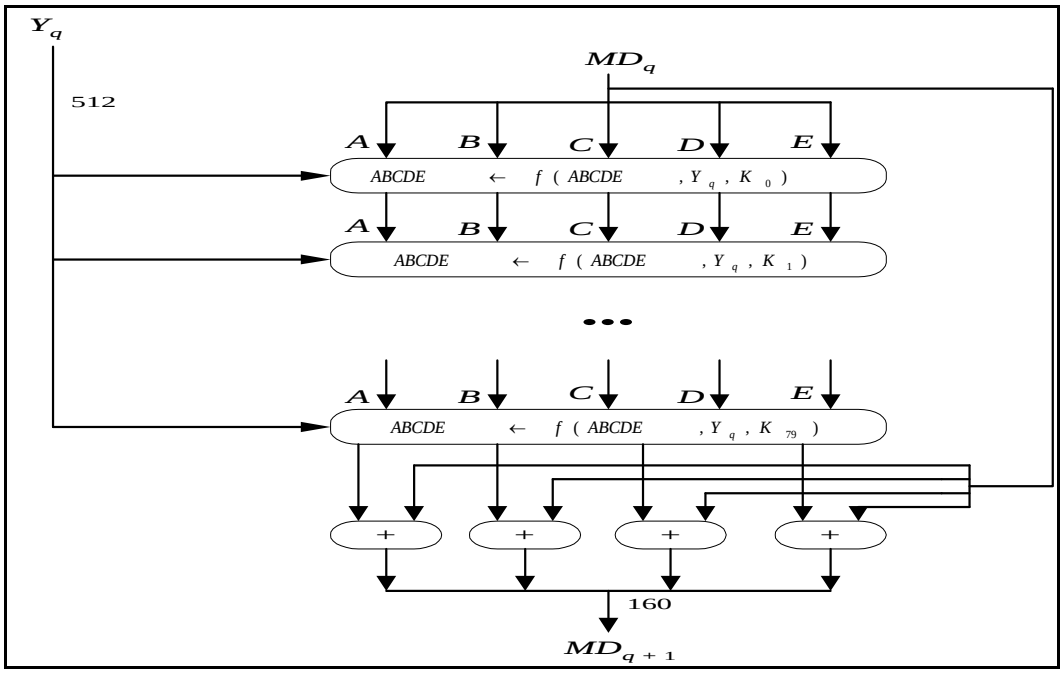
Putaran $20 \leq t \leq 39$ K_t = 6ED9EBA1 (2.11)

Putaran $40 \leq t \leq 59$ K_t = 8F1BBCDC (2.12)

Putaran $60 \leq t \leq 79$ K_t = CA62C1D6 (2.13)

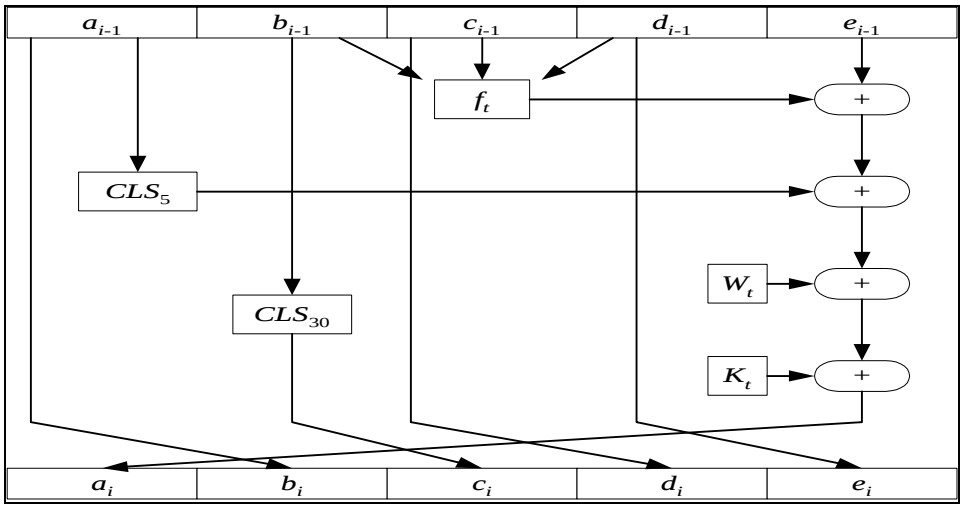
Pada gambar 2.7, Y_q menyatakan blok 512 bit ke q dari pesan yang telah ditambah pengganjal dan tambahan 64 bit nilai panjang pesan semula. MD_q merupakan nilai *Message Diggest* 160 bit dari proses H_{SHA} ke q.

Setiap putaran menggunakan operasi dasar yang sama (dinyatakan sebagai fungsi *f*). Operasi dasar SHA terlihat pada gambar 2.8.



Gambar 2.5. Skema Pengolahan blok 512 bit

Sumber: *Federal Information Processing Standard Publication* 180-2.



Gambar 2.6. Skema Operasi Algoritma Kriptografi SHA-1 dalam satu putaran.
Sumber: *Federal Information Processing Standard Publication* 180-2.

$$a, b, c, d, e \leftarrow (\mathrm{CLS}_5(a) + f_t(b, c, d) + e + W_t \\ a, \mathrm{CLS}_{30}(b), c, d$$

Gambar 2.7. Operasi Dasar Algoritma Kriptografi SHA-1 dalam satu putaran.
Sumber: *Federal Information Processing Standard Publication* 180-2.

dimana :

a, *b*, *c*, *d*, *e* = lima buah penyangga 32-bit (berisi nilai penyangga *A*, *B*, *C*, *D*, *E*)

t = putaran, $0 \le t \le 79$

f_t = fungsi logika

CLS_s = *circular left shift* sebanyak *s* bit

W_t = word 32-bit yang diturunkan dari blok 512 bit yang sedang diproses

K_t = konstanta penambah

Fungsi f_t adalah fungsi logika yang melakukan operasi logika bitwise. Operasi logika yang dapat dilihat pada tabel berikut:

Tabel 2.2. Tabel fungsi logika f_t pada setiap putaran

Putaran	$f_t(b, c, d)$
0..19	$(b \wedge c) \vee (\sim b \wedge d)$
20 .. 39	$b \oplus c \oplus d$
40 .. 59	$(b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$
60 .. 79	$b \oplus c \oplus d$

Sumber: *Federal Information Processing Standard Publication* 180-2.

Nilai W_1 sampai W_{16} berasal dari 16 word blok yang sedang diproses, sedangkan nilai W_t yang berikutnya diperoleh dari persamaan:

$$W_t = W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3} \quad (2.14)$$

Setelah putaran ke 79, nilai a,b,c,d dan e ditambahkan pada A,B,C,D dan E, dan selanjutnya algortima memproses untuk blok data berikutnya (Y_{q+1}). Keluaran akhir dari SHA adalah hasil penyambungna bit-bit di A,B,C,D dan E.

2.4. Tanda Tangan Digital (*Digital Signature*)

Tanda tangan digital atau skema tanda tangan digital, adalah salah satu tipe dari nirsimetris kriptografi (*asymmetric cryptography*). Untuk pesan yang dikirim melalui jalur yang tidak aman, skema yang diterapkan dengan benar dapat memberikan alasan bagi penerima pesan, bahwa pesan tersebut benar-benar dikirim (asli) oleh si pengirim. Tanda tangan digital setara dengan tanda tangan tradisional dalam berbagai hal, dan lebih sulit dipalsukan daripada tanda tangan tradisional. Tanda tangan digital berbasis kriptografi dan harus diimplementasikan dengan benar. Tanda tangan digital dapat menyediakan akses *non-repudiation*, yaitu berarti si pengirim pesan tidak dapat mengelak bahwa dia bukan si pengirim pesan. Skema tanda tangan digital biasanya terdiri dari tiga algoritma, yaitu:

- Pembangkitan kunci, dimana diminta kunci privat dan akan menghasilkan kunci publik,
- Algoritma pembuatan tanda tangan digital, dengan inputan pesan dan kunci privat,
- Algoritma verifikasi tanda tangan, dengan inputan pesan, kunci publik dan tanda tangan digital.

2.4.1. Skema Schnorr

Di dalam kriptografi, terdapat skema tanda tangan digital Schnorr, dikembangkan oleh Claus-Peter Schnorr. Skema ini diterapkan pertama kali dalam penggunaan *smart-card*.

Sejarah Singkat skema Schnorr

Sejak penemuan algoritma RSA oleh Rivest, Shamir dan Adleman pada tahun 1978, riset telah berfokus pada efisiensi dari skema ini. Skema ini bergantung pada protocol interaktif Chaum, Evertse dan Graaf (1998) yang membuktikan kemampuan logaritma diskrit. Kemudian algoritma ini menggabungkan ide dari skema milik ElGamal (1985) dan Fiat Shamir(1986).

Algoritma tanda tangan digital Schnorr

Algoritma tanda tangan digital Schnorr terdiri dari tiga bagian, yakni pembangkitan tanda tangan, dan verifikasi tanda tangan digital (Kostro).

Pembangkitan kunci publik

Proses pembangkitan kunci ini diawali dengan meminta memasukkan sepasang bilang prima, *p* dan *q*, dengan ketentuan *q* dapat membagi *p*-1.

Kemudian pemilihan bilangan pembangkit *g* dimana *g* merupakan bilangan diantara 1 dan *q*, dengan ketentuan *g* dapat membagi *q*-1, kemudian dengan menginputkan kunci privat *s*, dimana *s* tidak lebih besar dari *q*, dilakukan perhitungan:

$$v = x = g^s \tag{2.15}$$

Dari perhitungan tersebut akan menghasilkan kunci publik yaitu *p,q, g* dan *v* serta bilangan pembangkit *x* yang dipakai dalam pembuatan tanda tangan digital.

Proses pembangkitan tanda tangan

Dengan menggunakan nilai *x*, kemudian dengan pesan asli *M*, maka dilakukan perhitungan dengan menggunakan rumus:

$$e = H(M||x) \tag{2.16}$$

dimana:

- *H* merupakan fungsi *hash* SHA1,
- *M*||*x* merupakan penggabungan pesan dengan nilai x.

Kemudian dari hasil *e*, dilakukan perhitungan dengan rumus:

$$y = s - s .e \tag{2.17}$$

Kemudian tanda tangan digital dari pesan M tersebut yaitu satu set nilai *e* dan *y*.

Proses Verifikasi

Karena menggunakan *hash* SHA1, maka proses verifikasi dari pesan dan tanda tangan bukan merupakan pembalikan dari hasil tanda tangan digital tersebut.

Dengan menggunakan kunci publik yang diterima, maka dilakukan perhitungan dengan menggunakan rumus:

$$x' = g^y \cdot v^e \quad (2.19)$$

Kemudian dengan menggunakan *x'* yang didapat dari perhitungan di atas, maka dibuat tanda tangan digital baru dengan rumusan:

$$e' = H(M||x') \quad (2.20)$$

Kemudian untuk memverifikasi, dilakukan perbandingan dengan tanda tangan digital asli dan hasil penggabungan *e'* dan *y* yang didapat dari perhitungan tersebut. Hasil e akan bernilai sama dengan *e'* apabila nilai *x'* sama dengan *x* atau *v*. Dari nilai $x'=g^y \cdot v^e$, maka

$$V = x' = g^y \cdot v^e \quad (2.21)$$

$$g^s = g^y.(g^s)^e \quad (2.22)$$

$$g^s = g^y \cdot g^{se} \quad (2.23)$$

Karena bilangan basis yang dipangkatkan sama, maka dapat dilakukan perhitungan bilangan eksponen saja menjadi:

$$s = y + se \quad (2.24)$$

sehingga apabila dibalik, menjadi:

$$y = s - se \quad (2.25)$$