BAB II
LANDASAN TEORI

2.1 Studi Literatur

Penelitian yang dilakukan oleh (Bryando et al., 2020) Enkripsi File Teks Menerapkan Algoritma Skipjack. Pada penelitian yang dilakukan Bryado, Hapifa, Rodiana pada tahun 2020 menceritakan adanya pengiriman data yang sangat rentan dari gangguan oknum yang ingin menegetahui secara paksa isi informasi dari orang lain. Masalah ini dapat diatasi dengan melalukan proses enkripsi pada teks atau dokumen yang akan di kirim sehinnga tidak bisa dibaca oleh orang yang tidak punya kepentingan. Proses enkripsi data menggunakan metode skipjack. Metode ini merupakan cara proses enkripsi pada file text pada suatu aplikasi internet dengan contoh email service dan mempunyai 32 tahap dan disertakan rumus permutasi yang merupakan bagian dari metode skipjack, dengan memakai metode skipjack ini data atau teks yang akan dikirim melalui internet sudah dienkripsi oleh pengirim, kemudian sampai kepada penerima yang didekripsikan kembali menjadi data semula atau plaintext oleh penerima.

Harahap (2021) melakukan penelitian dengan judul "Implementasi Algoritma Skipjack Untuk Mengamankan Audio". Penelitian yang dilakukan (Harahap, 2021) yaitu proses enkripsi dan dekripsi algoritma skipjack sebanyak 32 putaran artinya algoritma pertama diputar dan diulang sebanyak 32 kali untuk mendapatkan hasil pengamanan dengan mengubah audio MP3 ke dalam bentuk heksadesimal terlebih dahulu, lalu menyelesaikan dengan rule A dan rule B. bilangan heksadesimal audio dan kunci diperoleh, diubah ke bilangan biner terlebih dahulu lalu melakukan proses XOR. Implementasi algoritma skipjack memberikan pengamanan yang kuat sehingga audio tersebut tidak akan tersebar kepada orang yang tidak memiliki hak.

Gaol (2021) melakukan penelitian dengan judul "Implementasi Algoritma Skipjack Dalam Pengamanan File Video". Pada penelitian yang dilakukan oleh (Gaol, 2021) menerapkan algoritma skipjack pada file video, yang mana video bersifat privasi tersebut dapat ditonton oleh orang yang tidak berhak jika file video tanpa pengamanan.Untuk meminimalisir hal ini maka diperlukan pengamanan tingkat kedua yaitu dengan mengacak video tersebut sehingga informasi visual dari

video tersebut tidak dapat terlihat oleh orang yang tidak memiliki kunci. Maka informasi dalam file video tersebut diubah dalam bentuk kode atau isyarat dimana kode inilah yang akan dimanipulasi. Dengan demikian file video perlu diamankan dengan pengamanan yang baik, sehingga perlu membuat enkripsi dan dekripsi pada file video. Algoritma skipjack salah satu algoritma kriptografi yang dapat digunakan untuk mengamankan file video.

Berikut ini adalah penjelasan hal-hal yang membedakan setiap kajian terkait, seperti terlihat pada Tabel 2.1 berikut ini.

Tabel 2.1 Pembanding Kajian Terkait

No	Penulis	Judul	Keterangan
1	Bryando H, Nurul Hapifa, Rodiana (2020), Universitas Budi Darma	Enkripsi File Teks Menerapkan Algoritma Skipjack	1. Mengenkripsi dan mendekripsikan file berbasis *desktop*. 2. Menerapkan Algoritma Skipjack untuk proses enkripsi dan dekripsi.
2	Rahmadani Harahap (2021), Universitas Budi Darma	Implementasi Algoritma Skipjack Untuk Mengamankan Audio	1. Aplikasi berbasis *desktop*. 2. Menerapkan Algoritma Skipjack untuk pengamanan rekaman file audio suara dengan format MP3.
3	Rina Wanty Lumban Gaol (2021), Universitas Budi Darma	Implementasi Algoritma Skipjack Dalam Pengamanan File Video	1. Aplikasi berbasis *desktop*. 2. Menerapkan Algoritma skipjack untuk mengacak nilai pixel file video berformat MP4.

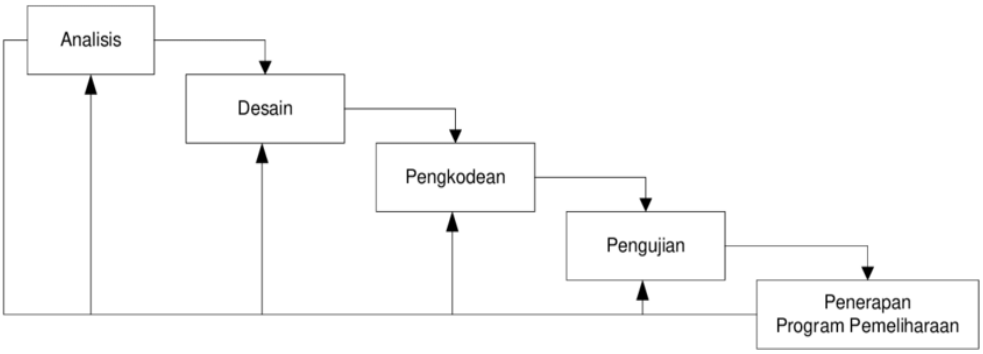
Berikut adalah penelitian yang dilakukan oleh penulis, seperti terlihat pada Tabel 2.2 berikut ini.

Tabel 2.2 Penelitian Yang Dilakukan Penulis

No	Penulis	Judul	Keterangan
1	Hakim Fajar (2022), Universitas Tanjungpura Pontianak	Penerapan Algoritma Skipjack Terhadap Penyandian Data File Berbentuk Teks	1. Aplikasi berbasis *website.* 2. Menerapkan Algoritma skipjack untuk proses enkripsi dan dekripsi. 3. Dapat mengirim pesan teks menjadi pesan enkripsi 4. Dapat mengirim segala jenis file untuk dienkripsikan 5. File yang akan dienkripsikan berukuran dibawah 5MB. 6. Menghasilkan 10 subkunci enkripsi otomatis dengan jumlah kunci berapun menggunakan fungsi md5. 7. Menggunakan mode CBC dengan kunci IV. 8. Dapat mengirim langsung file yang telah dienkripsikan ke penerima melalui aplikasi. 9. Penerima dapat melihat file yang di enkripsi dan dekripsi.

2.2 Model *Waterfall*

Terdapat beberapa metodologi *Systems Development Life Cycle* (SDLC) yang biasa digunakan dalam membangun sebuah sistem, salah satunya adalah model *waterfall*. *Waterfall* merupakan model yang bersifat sistematis dan termasuk dalam model klasik, nama lainnya adalah *Linear Sequential Model* (Pressman, 2012). Tahapan-tahapan model *waterfall* dapat dilihat pada Gambar 2.1.



Gambar 2.1 Model *Waterfall*

Penjelasan tahapan-tahapan *waterfall* tersebut yaitu:

1. Analisis

Fase ini merupakan proses analisis terhadap sistem yang sedang berjalan dengan tujuan untuk mendapatkan jawaban mengenai pengguna sistem, cara kerja sistem dan waktu penggunaan sistem, sehingga kebutuhan yang diperlukan untuk sistem baru akan didapatkan.

2. Desain

Desain merupakan proses penentuan cara kerja sistem dalam hal perancangan antarmuka, *database*, dan perancangan alur program. Perancangan diperlukan untuk menggambarkan sistem baru untuk memenuhi kebutuhan *user*.

3. Pengkodean

Tahapan pengkodean yaitu tahap rancangan sistem yang dibentuk menjadi suatu kode program untuk pembuatan sistem.

4. Pengujian

Pengujian dilakukan untuk mengetahui kesesuaian sistem berjalan sesuai prosedur atau tidak dan memastikan sistem terhindar dari *error* yang terjadi. Pengujian juga dilakukan untuk memastikan kevalidan dalam proses *input* sehingga dapat menghasilkan output yang sesuai.

5. Penerapan Program Pemeliharaan

Fase ini yaitu penerapan program dan pemeliharaan sistem yang berguna untuk melihat kemampuannya, mengecek jika masih ada ditemukan *error* atau ada penambahan fitur-fitur yang belum ada pada sistem tersebut. Pemeliharaan diperlukan ketika adanya perubahan dari pengguna seperti ketika ada pergantian sistem operasi, atau perangkat lainnya.

2.3 Kriptografi

Kriptografi merupakan ilmu yang mempelajari teknik matematika yang berhubungan dengan aspek keamanan informasi, seperti kerahasiaan data, keabsahan data, dan integritas data serta autentikasi data (Menezes et al., 1996). Kriptografi adalah ilmu mengenai teknik enkripsi dimana data diacak menggunakan suatu kunci enkripsi menjadi sesuatu yang sulit dibaca oleh seseorang yang tidak memiliki kunci dekripsi (Kromodimoeljo, 2010).

Dekripsi menggunakan kunci dekripsi mendapatkan kembali data asli. Proses enkripsi dilakukan menggunakan suatu algoritma dengan beberapa parameter. Biasanya algoritma tidak dirahasiakan, bahkan enkripsi yang mengandalkan kerahasiaan algoritma dianggap sesuatu yang tidak baik. Rahasia terletak di beberapa parameter yang digunakan, jadi kunci ditentukan oleh parameter. Parameter yang menentukan kunci dekripsi itulah yang harus dirahasiakan (parameter menjadi ekuivalen dengan kunci) (Kromodimoeljo, 2010). Dapat dilihat pada gambar 2.2 (Kromodimoeljo, 2010)

Gambar 2.2 Proses Enkripsi dan Dekripsi

Dapat dilihat dari Gambar 2.2 merupakan proses dari enkripsi dan dekripsi suatu file teks. Pesan asli disebut plaintext, dan pesan menyamar disebut ciphertext. Pesan terakhir, dikemas dan dikirim, disebut kriptogram. Proses transformasi plaintext menjadi ciphertext disebut enkripsi atau enciphering. Proses kebalikan dari mengubah ciphertext menjadi plaintext, yang dicapai oleh penerima yang memiliki pengetahuan untuk menghapus menyamar, disebut dekripsi atau mengartikan.

2.3.1 Tujuan Algoritma Kriptografi

Adapun tujuan algoritma kriptografi adalah sebagai berikut (Menezes et al., 1996) :

1. Kerahasiaan, adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka/mengupas informasi yang telah disandi.
2. Integritas data, adalah berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan pensubtitusian data lain ke dalam data yang sebenarnya.
3. Autentikasi, adalah berhubungan dengan identifikasi/pengenalan, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui kanal harus didentifikasi keaslian, isi data, waktu pengiriman, dan lain-lain.
4. Non-rapudiasi, adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman atau terciptanya suatu informasi oleh yang mengirimkan atau membuat

2.4 *National Security Agency* (NSA)

National Security Agency (NSA) atau Badan Keamanan Nasional adalah sebuah agensi kriptografi milik pemerintah Amerika. NSA bertugas untuk mengumpulkan dan menganalisis komunikasi negara lain, serta melindungi informasi milik Amerika Serikat. NSA mengkoordinasi, mengarahkan, serta menjalankan aktivitas-aktivitas amat istimewa mempunyai tujuan sebagai mengumpulkan informasi intelijen dari luar negeri, menggunakan kriptoanalisis. Selain itu, NSA melindungi komunikasi pemerintah dan sistem informasi di Amerika Serikat dari agensi lainnya, yang melibatkan kriptografi tingkat tinggi (Bamford, 2001).

2.5 Algoritma Skipjack

Algoritma skipjack adalah salah satu algoritma yang dimana Algoritma skipjack ini memakai kunci yang sama untuk melakukan proses enkripsi dan

dekripsinya. Algoritma skipjack dikembangkan pada tahun 1987 dan dipublikasikan pada tahun 1998 oleh badan keamanan amerika serikat. Algoritma skipjack memiliki masukan (*Plaintext*) yang berukuran 64 bit yang kemudian data tersebut diubah ke bentuk kumpulan blok-blok data berukuran 64 bit yang selanjutnya diproses dengan menggunakan kunci yang sama untuk menghasilkan keluaran (*Chipertext*) (Suprianto, 2007). Algoritma skipjack merupakan *iterated block* cipher yang dalam proses enkripsi maupun dekripsinya dengan 32 putaran yang memiliki dua tipe, yaitu Rule A dan Rule B. Setiap putaran dibuat dalam bentuk feedback linear shift register dengan tambahan permutasi G non-linear. Rule B pada dasarnya adalah kebalikan dari Rule A dengan beberapa perbedaan pada pengaturan posisi. Skipjack melakukan 8 putaran terhadap Rule A, setelah itu melakukan 8 putaran terhadap Rule B, setelah itu melakukan 8 putaran lain dari Rule A dan 8 putaran lain dari Rule B untuk menghasilkan data keluaran (*Chipertext*). Operasi matematik yang digunakan dalam algoritma skipjack ini adalah XOR dan permutasi . Kombinasi tersebutlah yang membuat algoritma skipjack ini mempunyai tingkat keamanan yang sangat tinggi.

Operasi matematik kriptografi yang digunakan di dalam rule A, rule B, rule A-1, dan rule B-1 tersebut adalah XOR dan permutasi. Operasi permutasi dilakukan dengan menggunakan sebuah tabel substitusi yang disebut dengan F-Table dan kunci rahasia. Nilai – nilai dalam F-Table diberikan dalam nilai Heksadesimal. Baris pertama pada tabel (x0…xF) menunjukkan kolom sedangkan kolom pertama pada tabel (0x…Fx) menunjukkan baris. Nilai – nilai F-Table dapat dilihat pada Gambar 6.2 (Asri, 2009).

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	A3	d7	09	83	f8	48	f6	f4	b3	21	15	78	99	b1	af	f9
1x	e7	2d	4d	8a	ce	4c	ca	2e	52	95	D9	1e	4e	38	44	28
2x	0a	df	02	a0	17	f1	60	68	12	b7	7a	c3	e9	fa	3d	53
3x	96	84	6b	ba	f2	63	9a	19	7c	ae	e5	f5	f7	16	6a	a2
4x	39	b6	7b	0f	c1	93	81	1b	ee	b4	1a	ea	d0	91	2f	b8
5x	55	b9	da	85	3f	41	bf	e0	5a	58	80	5f	66	0b	d8	90
6x	35	d5	c0	a7	33	06	65	69	45	00	94	56	6d	98	9b	76
7x	97	fc	b2	c2	b0	fe	db	20	e1	eb	D6	e4	dd	47	4a	1d
8x	42	ed	9e	6e	49	3c	cd	43	27	d2	07	d4	de	c7	67	18
9x	89	cb	30	1f	8d	c6	8f	aa	c8	74	dc	c9	5d	5c	31	a4
Ax	70	88	61	2c	9f	0d	2b	87	50	82	54	64	26	7d	03	40
Bx	34	4b	1c	73	d1	c4	fd	3b	cc	fb	7f	ab	e6	3e	5b	a5
Cx	Ad	04	23	9c	14	51	22	f0	29	79	71	7e	ff	8c	0e	e2
Dx	0c	ef	bc	72	75	6f	37	a1	ec	d3	8e	62	8b	86	10	e8
Ex	08	77	11	be	92	4f	24	c5	32	36	9d	cf	f3	a6	bb	ac
Fx	5e	6c	a9	13	57	25	b5	e3	bd	a8	3a	01	05	59	2a	46

Gambar 2.3 F-Table

Perhitungan kriptografi dengan metode XOR melakukan penyandian pada file atau ciphertext dengan mengubah setiap karakter pada file atau ciphertext menjadi sebuah nilai berdasarkan tabel ASCII. Menurut ISO 8859-1 dan Microsoft Windows Latin-1 ASCII merupakan kepanjangan dari *(American Standard Code for Information Interchange),* dan pengertian dari ASCII sendiri adalah suatu standar internasional dalam kode huruf dan simbol seperti *Decimal* dan Unicode tetapi ASCII lebih bersifat universal, contohnya 181 adalah untuk karakter "µ". Tabel ASCII dapat dilihat pada gambar 2.4 dibawah ini.

DC	AC	AS+UC	DC	All	DC	All	DC	All	DC	AC	UC	EA	DC	AC	UC	EA	DC	AC	UC	EA	DC	AC	UC	EA
NUL	NUL	NUL	32	SP	64	@	96	`	128	€	xxx	Ç	160		nbsp	á	192	À	À	└	224	à	à	Ó
1	☺	SOH	33	!	65	A	97	a	129		xxx	ü	161	¡	¡	í	193	Á	Á	┴	225	á	á	ß
2	☻	STX	34	"	66	B	98	b	130	‚	BPH	é	162	¢	¢	ó	194	Â	Â	┬	226	â	â	Ô
3	♥	ETX	35	#	67	C	99	c	131	ƒ	NBH	â	163	£	£	ú	195	Ã	Ã	├	227	ã	ã	Ò
4	♦	EOT	36	$	68	D	100	d	132	„	IND	ä	164	¤	¤	ñ	196	Ä	Ä	─	228	ä	ä	õ
5	♣	ENQ	37	%	69	E	101	e	133	…	NEL	à	165	¥	¥	Ñ	197	Å	Å	┼	229	å	å	Õ
6	♠	ACK	38	&	70	F	102	f	134	†	SSA	å	166	¦	¦	ª	198	Æ	Æ	ã	230	æ	æ	µ
7	•	BEL	39	'	71	G	103	g	135	‡	ESA	ç	167	§	§	º	199	Ç	Ç	Ã	231	ç	ç	þ
8	◘	BS	40	(	72	H	104	h	136	ˆ	HTS	ê	168	¨	¨	¿	200	È	È	╚	232	è	è	Þ
9	○	HT	41	)	73	I	105	i	137	‰	HTJ	ë	169	©	©	®	201	É	É	╔	233	é	é	Ú
10	◙	LF	42	*	74	J	106	j	138	Š	VTS	è	170	ª	ª	¬	202	Ê	Ê	╩	234	ê	ê	Û
11	♂	VT	43	+	75	K	107	k	139	‹	PLD	ï	171	«	«	½	203	Ë	Ë	╦	235	ë	ë	Ù
12	♀	FF	44	,	76	L	108	l	140	Œ	PLU	î	172	¬	¬	¼	204	Ì	Ì	╠	236	ì	ì	ý
13	♪	CR	45	-	77	M	109	m	141		RI	ì	173	-	-	¡	205	Í	Í	═	237	í	í	Ý
14	♫	SO	46	.	78	N	110	n	142	Ž	SS2	Ä	174	®	®	«	206	Î	Î	╬	238	î	î	¯
15	☼	SI	47	/	79	O	111	o	143		SS3	Å	175	¯	¯	»	207	Ï	Ï	¤	239	ï	ï	´
16	►	DLE	48	0	80	P	112	p	144		DSC	É	176	°	°	░	208	Ð	Ð	ð	240	ð	ð	-
17	◄	DC1	49	1	81	Q	113	q	145	‘	PU1	æ	177	±	±	▒	209	Ñ	Ñ	Đ	241	ñ	ñ	±
18	↕	DC2	50	2	82	R	114	r	146	’	PU2	Æ	178	²	²	▓	210	Ò	Ò	Ê	242	ò	ò	‗
19	‼	DC3	51	3	83	S	115	s	147	“	STS	ô	179	³	³	│	211	Ó	Ó	Ë	243	ó	ó	¾
20	¶	DC4	52	4	84	T	116	t	148	”	CCH	ö	180	´	´	┤	212	Ô	Ô	È	244	ô	ô	¶
21	§	NAK	53	5	85	U	117	u	149	•	MW	ò	181	µ	µ	Á	213	Õ	Õ	ı	245	õ	õ	§
22	▬	SYN	54	6	86	V	118	v	150	–	SPA	û	182	¶	¶	Â	214	Ö	Ö	Í	246	ö	ö	÷
23	↨	ETB	55	7	87	W	119	w	151	—	EPA	ù	183	·	·	À	215	×	×	Î	247	÷	÷	¸
24	↑	CAN	56	8	88	X	120	x	152	˜	SOS	ÿ	184	¸	¸	©	216	Ø	Ø	Ï	248	ø	ø	°
25	↓	EM	57	9	89	Y	121	y	153	™	xxx	Ö	185	¹	¹	╣	217	Ù	Ù	┘	249	ù	ù	¨
26	→	SUB	58	:	90	Z	122	z	154	š	SCI	Ü	186	º	º	║	218	Ú	Ú	┌	250	ú	ú	·
27	←	ESC	59	;	91	[	123	{	155	›	CSI	ø	187	»	»	╗	219	Û	Û	█	251	û	û	¹
28	∟	FS	60	<	92	\	124	|	156	œ	ST	£	188	¼	¼	╝	220	Ü	Ü	▄	252	ü	ü	³
29	↔	GS	61	=	93	]	125	}	157		OSC	Ø	189	½	½	¢	221	Ý	Ý	¦	253	ý	ý	²
30	▲	RS	62	>	94	^	126	~	158	ž	PM	×	190	¾	¾	¥	222	Þ	Þ	▌	254	þ	þ	■
31	▼	US	63	?	95	_	127	DEL	159	Ÿ	APC	ƒ	191	¿	¿	┐	223	ß	ß	▀	255	ÿ	ÿ	nbsp

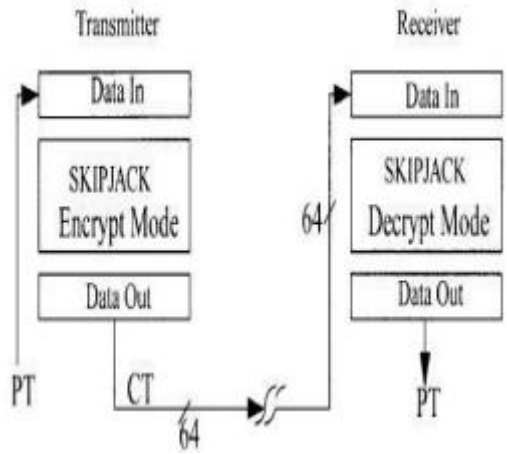
Gambar 2.4 Tabel ASCII

2.5.1 Mode Operasi Algoritma Skipjack

Mode Operasi Skipjack merupakan subset dari dekripsi mode FIPS-81 untuk DES. Mode-mode tersebut terbagi 4 mode yaitu ECB *(Electronic Code Book),* CBC *(Cipher Block Chaining),* CFB *(Cipher Feed Back), OFB (Output Feed Back)* (Munir, 2006):

1. ECB *(Electronic Code Book)*

Pada mode ECB, data dibagi menjadi 64-bit blok dan setiap blok dienkripsi satu per satu. Enkripsi dilakukan terpisah terhadap masing-masing blok Artinya, jika data yang dikirimkan melalui jaringan atau saluran telepon, transmisi kesalahan hanya akan mempengaruhi blok yang bersangkutan. ECB merupakan metode terlemah karena tidak ada langkah-langkah keamanan tambahan yang diimplementasikan selain algoritma dasar Skipjack. Mode ECB dapat dilihat gambar 2.5 (Munir, 2006)

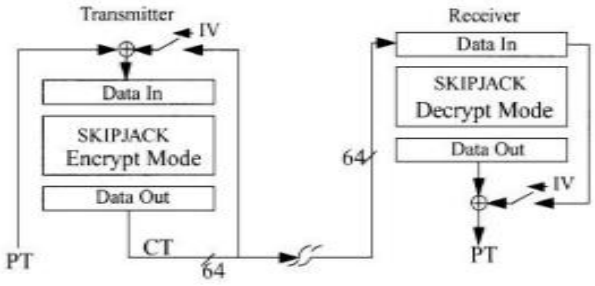


Gambar 2.5 Mode ECB

Namun, ECB adalah yang tercepat dan termudah untuk diimplementasikan. Tidak ada banyak informasi mengenai modus operasi yang banyak digunakan, tetapi untuk kebanyakan jenis ciphers blok, ECB adalah yang paling umum diterapkan. Modus operasi ini digunakan oleh Private Encryptor.

2. CBC *(Cipher Block Chaining)*

Dalam modus operasi ini, setiap *block ciphertext* yang dienkripsi dengan modus ECB di-XOR dengan plaintext berikutnya yang akan dienkripsi, sehingga seluruh blok bergantung pada blok sebelumnya. Untuk mencari *plaintext* dari blok tertentu, diperlukan ciphertext, kunci, dan ciphertext untuk blok sebelumnya. Blok pertama yang akan dienkripsi tidak memiliki ciphertext sebelumnya, sehingga *plaintext* untuk blok pertama adalah hasil XOR dengan angka 8-bit yang disebut *initialization Vecto*r, atau IV. Mode CBC dapat dilihat gambar 2.6 (Munir, 2006).

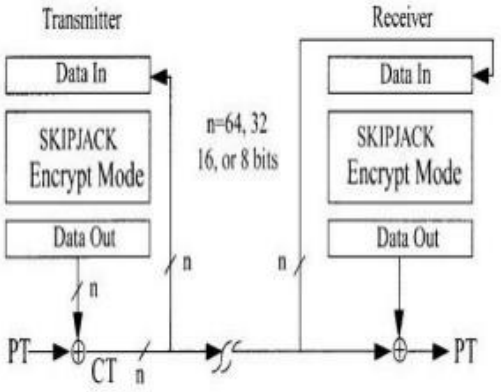


Gambar 2.6 Mode CBC

Jadi jika terdapat kesalahan transmisi dalam data yang dikirimkan melalui jaringan atau saluran telepon, kesalahan akan dibawa ke semua blok sampai blok terakhir. Modus operasi ini lebih aman daripada ECB karena langkah XOR tambahan menambahkan satu lapisan lagi ke proses enkripsi.

3. CFB *(Cipher Feed Back)*

Dalam mode ini, blok *plaintext* yang kurang dari 64 bit dapat dienkripsi. Biasanya, proses yang khusus digunakan untuk menangani file yang ukurannya tidak utuh 8 bit. *Private Encryptor* menangani kasus ini dengan menambahkan beberapa dummy byte ke akhir file sebelum melakukan enkripsi. Plaintext sebenarnya tidak melalui proses Skipjack, tetapi hanya di-XOR dengan *output* blok. Proses ini dilakukan dengan cara 64-bit yang disebut blok *Shift Register* digunakan sebagai masukan *plaintext* pada Skipjack. Blok ini dibentuk untuk beberapa nilai yang berubah-ubah dan dienkripsi dengan algoritma Skipjack. Mode CFB dapat dilihat gambar 2.7 (Munir, 2006).



Gambar 2.7 Mode CFB

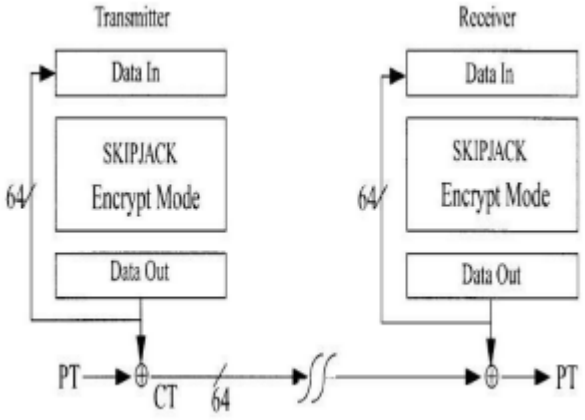
Kemudian *ciphertext* di-*pass* melalui sebuah komponen tambahan yang disebut kotak-M yang hanya memilih *left-most* M *bits* dari *ciphertext*, M adalah

jumlah bit dalam yang ingin dienkripsi. Nilai ini di-XOR dengan plaintext dan hasilnya adalah final ciphertext. Akhirnya *ciphertext* di-*feedback* pada *Shift Register*, dan digunakan sebagai seed untuk blok berikutnya yang akan dienkripsi. Seperti pada mode CBC, kesalahan dalam satu blok akan mempengaruhi semua blok selama pengiriman data. Modus operasi ini sama dengan CBC dan sangat aman, tetapi lebih lambat dari ECB karena kompleksitas yang ditambahkan.

4. OFB *(Output Feed Back)*

Mode ini mirip dengan mode CFB. Namun pada OFB, hasil *ciphertext* dari proses enkripsi di*feedback* ke dalam *Shift Register*. *Shift Register* tersebut di-*assign* pada sebuah nilai awal yang dapat berubah-ubah dan dijadikan sebagai parameter dalam algoritma Skipjack. Hasil dari proses Skipjack di-*pass* melalui M-Box, kemudian di-*feedback* ke dalam *Shift Register* untuk mempersiapkan blok berikutnya. Nilai ini kemudian di-XOR dengan *plaintext* yang panjangnya kurang dari 64 bit dan hasilnya adalah *ciphertext* akhir.

Tidak seperti CFB dan CBC, kesalahan dalam satu blok dalam mode OFB tidak akan mempengaruhi blok yang lain karena setelah blok penerima memiliki Shift Register awal, Shift Register yang baru akan terus di-generate tanpa masukan input data lagi. Namun, modus ini kurang aman daripada mode CFB karena hanya ciphertext dan hasil ciphertext Skipjack yang diperlukan untuk menemukan plaintext dari blok terakhir. Informasi tentang kunci pun tidak dibutuhkan. Mode OFB dapat dilihat gambar 2.8 (Munir, 2006).



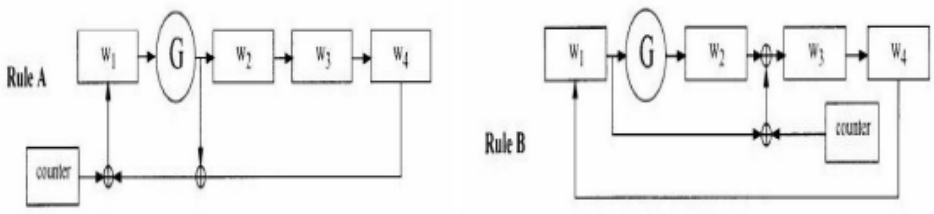
Gambar 2.8 Mode OFB

2.5.2 Manfaat Penerapan Algoritma Skipjack

Penerapan Algoritma Skipjack digunakan sebagai Fortezza. Fortezza adalah sistem keamanan informasi yang berbasis pada PC Card. Setiap individu yang berhak untuk melihat informasi yang dilindunginya diberikan Fortezza Card yang menyimpan private key dan data lainnya yang dibutuhkan untuk mendapatkan akses. Fortezza berisi sebuah keamanan *microprocessor* yang disebut Capstone (MYK-80). Capstone menggunakan penerapan algoritma Skipjack. Fortezza Card telah digunakan di pemerintahan, militer, dan aplikasi perbankan untuk melindungi data yang sensitif. Card dapat dipertukarkan dengan berbagai jenis peralatan yang mendukung Fortezza. Kunci dan program dapat dibuat ulang Fortezza dikembangkan untuk proyek Clipper Chip Pemerintah AS dan telah digunakan oleh Pemerintah AS di berbagai aplikasi. Fortezza Card (KOV-8) yang asli adalah produk Type II yang berarti tidak dapat digunakan untuk informasi yang rahasia.

2.5.3 Struktur Algoritma Skipjack

Skipjack merupakan iterated blok cipher dengan 32 putaran yang memiliki dua tipe, yaitu Rule A dan Rule B. Setiap putaran dibuat dalam bentuk feedback linear shift register dengan tambahan permutasi G non-linear. Rule B pada dasarnya adalah kebalikan dari Rule A dengan beberapa perbedaan pada pengaturan posisi. Skipjack melakukan delapan putaran terhadap Rule A, setelah itu melakukan delapan putaran Rule B, setelah itu melakukan delapan putaran lain dari Rule A dan delapan putran lain dari Rule B. Definisi dari Rule A dan Rule B diberikan pada Gambar. Counter merupakan angka yang berurut dari 1 sampai 32. G adalah permutasi Feistel 4 putaran dimana fungsi F ditetapkan sebagai bit 8x8 S-box (disebut sebagai tabel F). Setiap putaran dari G dienkripsi dengan kunci 8-bit, dapat dilihat gambar 2.9 (Munir, 2006).

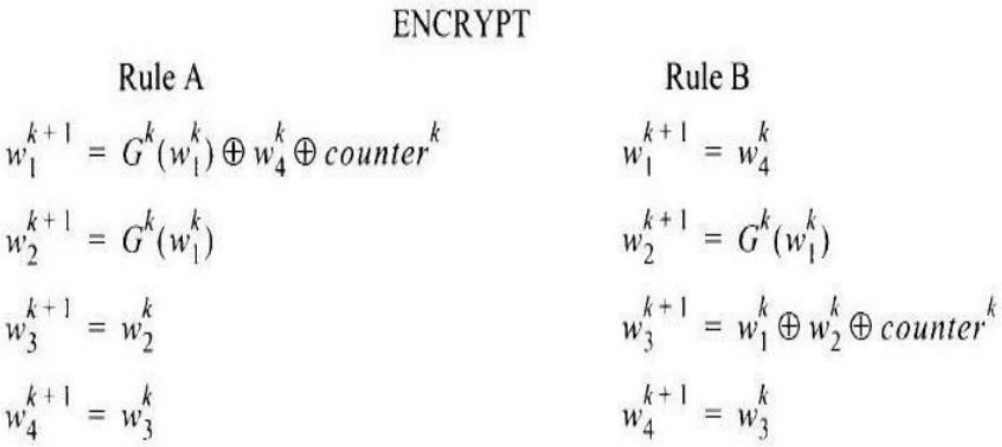


Gambar 2.9 Rule A dan Rule B

2.5.4 Proses Enkripsi dan Dekripsi Algoritma Skipjack

1. Proses Enkripsi

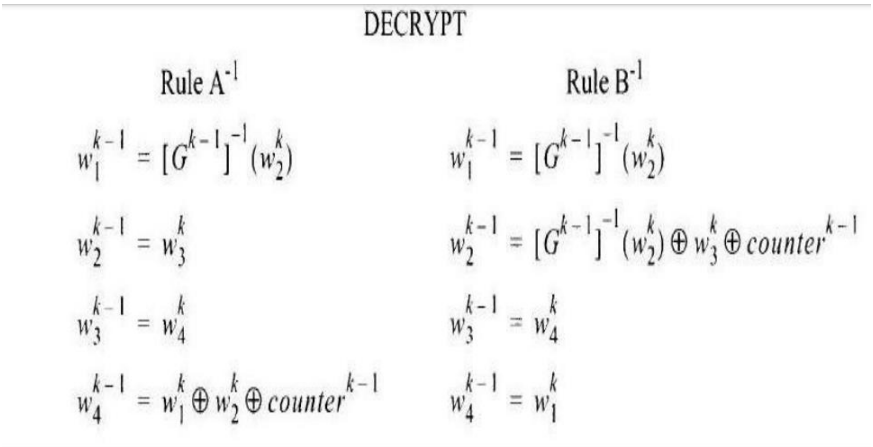
Saat proses dimulai, counter bernilai 1. Algoritma melakukan Rule A sebanyak 8 kali, lalu berganti melakukan Rule B sebanyak 8 kali. Berganti melakukan Rule A sebanyak 8 kali lagi dan menyelesaikan enkripsi dengan 8 putaran Rule B. Counter bertambah setiap menyelesaikan satu langkah. Proses enkripsi dapat dilihat gambar 2.10 (Munir, 2006).



Gambar 2.10 Proses Enkripsi

2. Proses Dekripsi

Algoritma dimulai dengan nilai counter 32. Algoritma melakukan Rule B ^ -1 untuk 8 langkah, Rule A ^ -1 untuk 8 langkah, Rule b ^ -1 selama 8 langkah lagi dan akhirnya Rule ^ -1 untuk 8 langkah lain. Counter berkurang satu setiap langkahnya. Proses dekripsi dapat dilihat gambar 2.11 (Munir, 2006).



Gambar 2.11 Proses Dekripsi

2.6 *Unified Modelling Language* (UML)

Unified Modeling Language (UML) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak.

UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek.

Unified Modelling Language (UML) adalah bahasa pemodelan untuk sistem atau perangkat lunak. Pemodelan (*modeling)* sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami (Nugroho, 2010).

Sedangkan menurut (Shalahuddin & Sukamto, 2013) UML (*Unified Modeling Language*) adalah salah satu standar bahasa visual yang banyak digunakan di dunia industri untuk mengidentifikasi *requirement*, membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML hanya berfungsi untuk melakukan pemodelan, jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek.

UML disebut sebagai bahasa pemodelan bukan metode. Kebanyakan metode terdiri paling sedikit prinsip, bahasa pemodelan dan proses. Bahasa pemodelan (sebagian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secara cepat.

Di dalam UML ada 13 buah diagram yang dikelompokkan ke dalam tiga kategori yaitu:

1. *Structure Diagrams,* yaitu kumpulan diagram-diagram yang menggambarkan struktur statis dari sistem yang dimodelkan.
2. *Behavior Diagrams,* yaitu kumpulan diagram-diagram yang menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi di dalam sistem.
3. *Interaction Diagram,* yaitu kumpulan diagram-diagram yang menggambarkan interaksi sistem dengan sistem lain ataupun interaksi antar subsistem dalam sebuah sistem.

Secara garis besar, beberapa diagram utama sudah dapat menggambarkan keseluruhan sistem. Diagram tersebut antara lain *use case diagram, class diagram,* dan *activity diagram*.

2.6.1 *Use Case Diagram*

Use Case Diagram menurut (Widodo, 2011) Diagram *use case* bersifat statis, yang memperlihatkan himpunan *Use Case* dan aktor-aktor (suatu jenis khusus dari kelas) dan menggambarkan apa saja aktivitas yang dilakukan oleh suatu sistem dari sudut pandang pengamatan luar. Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna yang menjadi persoalan itu apa yang dilakukan bukan bagaimana melakukannya. *Use Case Diagram* menggambarkan fungsionalitas yang diharapkan dari sistem.

Deskripsi simbol-simbol yang digunakan pada *Use Case Diagram* dapat dilihat pada Tabel 2.3 (Nugroho, 2010).

Tabel 2.3 Deskripsi Notasi pada Use Case Diagram

No	Notasi	Nama	Deskripsi
1		*Use Case*	Menggambarkan fungsionalitas yang dimiliki sistem.
2		*Actor*	Menggambarkan semua objek di luar sistem (bukan hanya pengguna sistem/perangkat lunak) yang berinteraksi dengan sistem yang dikembangkan.

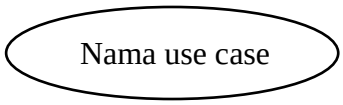
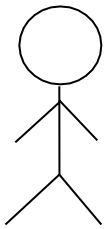
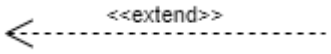
No	Notasi	Nama	Deskripsi
3		*Association*	Lintasan komunikasi antara *actor* dengan *use case.*
4	<<extend>> - - - - - - - ->	*Extended* (Ekstensi)	Penambahan perilaku ke suatu *use case* dasar.
5	<<include>> < - - - - - - -	*Include* (Menggunakan)	Penambahan perilaku ke suatu *use case* dasar yang secara *explicit* mendeskripsikan penambahan tersebut.
6		*Generalization* (Generalisasi)	Hubungan generalisasi dan spesialisasi (umum - khusus) antara dua buah *use case* dimana fungsi yang satu merupakan fungsi yang lebih umum dari lainnya.

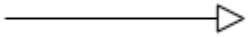
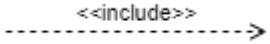
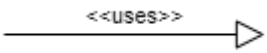
2.6.2 *Activity Diagram*

Activity Diagram menggambarkan *workflow* (aliran kerja) proses bisnis dan urutan aktivitas dalam sebuah proses. *Activity diagram* sangat mirip dengan *flowchart* karena memodelkan *workflow* dari satu aktivitas ke aktivitas lainnya atau dari aktivitas ke status. Membuat *activity diagram* pada awal pemodelan proses cukup menguntungkan untuk membantu memahami keseluruhan proses. *Activity diagram* juga bermanfaat untuk menggambarkan *parallel behavior* atau menggambarkan interaksi antara beberapa *use case.*

Berikut merupakan simbol notasi *Activity Diagram* pada Tabel 2.4 (Shalahuddin & Sukamto, 2013).

Tabel 2.4 Deskripsi Notasi pada Activity Diagram

No.	Simbol	Deskripsi
1.	*Use case* 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja diawal frase nama *use case*.
2.	Aktor/*actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
3.	Assosiasi/*association* 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
4.	Exstensi/*extend* 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu, mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek, biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan.

No.	Simbol	Deskripsi
5.	Generalisasi/*generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua *buah use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya
6.	Menggunakan / *include* / *uses*   	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini Ada dua sudut pandang mengenai *include* di *use case* : - *Include* berarti *use case* yang ditambahkan akan selalu di panggil saat *use case* tambahan dijalankan. - *Include* berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang di tambahkan telah dijalankan sebelum *use case* tambahan di jalankan. Kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

2.6.3 *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas – kelas yang akan dibuat untuk membangun system (Shalahuddin & Sukamto, 2013). Sedangkan menurut (Fowler, 2005) yang

berpendapat bahwa *Class diagram* mendeskripsikan jenis – jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat diantara mereka. *Class diagram* digunakan untuk menggambarkan hubungan kelas – kelas antara satu dengan yang lain seta memiliki atribut dan operasi yang terdapat dalam sistem yang akan dibuat. Atribut merupakan variabel – variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsi – fungsi yang dimiliki oleh suatu kelas. Atribut dan metode dapat memiliki salah satu sifat sebagai berikut:

1. *Private* (-), hanya dapat digunakan oleh *class* yang memilikinya
2. *Public* (+), dapat digunakan oleh *class* lain.
3. *Protected* (#), hanya dapat dipanggil oleh *class* yang bersangkutan jumlah suatu anak yang mewarisinya.

Nilai kardinalitas atau *multiplicity* sebuah *class* menunjukkan jumlah suatu objek yang dapat berhubungan dengan objek lain. Berikut nilai kardinalitas atau *multiplicity* pada Tabel 2.5 dan notasi *class diagram* pada Tabel 2.6(Shalahuddin & Sukamto, 2013).

Tabel 2.5 Jenis-jenis *Multiplicity*

No	Indikator	Keterangan
1	0 .. 1	Nol atau satu
2	1	Hanya satu
3	0 .. *	Nol atau lebih
4	1 .. *	Satu atau lebih

Tabel 2.6 Deskripsi Notasi pada Class Diagram

No	Notasi	Nama	Deskripsi
1	Kelas +atribut +metode()	*Class*	Menggambarkan konsep dasar pemodelan sistem.
2	————	Asosiasi (*Association*)	Sebuah garis solid antara dua *class*, ditarik dari *class* sumber ke *class* target lebih spesifik, digunakan

No	Notasi	Nama	Deskripsi
			dalam struktur pewarisan.
3	- - - - - - - - ->	Ketergantungan (*Dependency)*	Relasi antara dua elemen jika perubahan definisi sebuah elemen (*supplier* atau sumber) dapat menyebabkan perubahan pada elemen lainnya (*Client* atau target).

2.7 Teknologi Pendukung

2.7.1 Web

Menurut (Kadir, 2014), *World Wide Web* (WWW) atau biasa disebut dengan web merupakan salah satu sumber daya Internet yang berkembang pesat. Pertama kali aplikasi web dibangun hanya dengan menggunakan bahasa yang disebut HTML (*HyperText Markup Language*) dan protokol yang digunakan dinamakan HTTP (*HyperText Transfer Protocol*). Pada perkembangan berikutnya, sejumlah skrip dan objek dikembangkan untuk memperluas kemampuan HTML yang sekarang ini terdapat banyak skrip seperti: PHP dan ASP, sedangkan contoh yang berupa objek antara lain adalah applet *(java*) (Kadir, 2014). Jadi aplikasi web atau aplikasi berbasis web (*Web-based application*) adalah aplikasi untuk menyampaikan informasi kepada pengguna yang menggunakan layanan Internet berbasis web.

Dalam aplikasi tersebut, terjadi pertukaran antara *klien* (komputer yang meminta informasi) dengan *server* (komputer yang memasok atau menanggapi informasi). *Web* memberikan informasi secara *online* melalui internet langsung. *Klien* melakukan permintaan informasi dengan menggunakan *browser*. *Server* menerima informasi dan melayani permintaan dari *client*. Hal ini biasa disebut dengan web *server* (contoh *web server: Apache, IIS, Xitami*, dan sebagainya). Setelah itu, web *server* akan berkomunikasi dengan *middleware* (contoh *middleware: ASP, JSP, PHP*, dan sebagainya) untuk bisa berhubungan dengan basis data atau *database* (contoh *database: access, oracle, sql*, dan sebagainya). Setelah berinteraksi dengan *database, server* yang telah mendapatkan informasi akan memberikan tanggapan terhadap *klien* yang meminta informasi tadi.

2.7.2 *HyperText Markup Language* (HTML)

Menurut (Solihin, 2016), HTML merupakan singkatan dari *Hyper Text Markup Language*. HTML dikembangkan pertama kali oleh Tim Berners-Lee bersamaan dengan protokol HTTP (*Hypertext Transfer Protocol*) pada tahun 1989. Tujuan utama pengembangan HTML adalah untuk menghubungkan suatu halaman web dengan halaman web lainnya.

Menurut (Kuswayatno, 2006), HTML merupakan halaman yang berada pada suatu situs internet atau web. HTML merupakan metode yang menautkan (*link*) satu dokumen ke dokumen lain melalui teks. Menurut Deris Setiawan, HTML merupakan framework internet, hampir semua situs web yang ada menggunakan HTML untuk menampilkan teks, grafik, suara, dan animasinya.

HTML adalah sebuah bahasa markah yang digunakan untuk membuat sebuah halaman web, menampilkan berbagai informasi di dalam sebuah penjelajah web Internet dan pemformatan hiperteks sederhana yang ditulis dalam berkas format ASCII agar dapat menghasilkan tampilan wujud yang terintegerasi. Dengan kata lain, berkas yang dibuat dalam perangkat lunak pengolah kata dan disimpan dalam format ASCII normal sehingga menjadi halaman web dengan perintah-perintah HTML.

Bermula dari sebuah bahasa yang sebelumnya banyak digunakan di dunia penerbitan dan percetakan yang disebut dengan SGML (*Standard Generalized Markup Language*), HTML adalah sebuah standar yang digunakan secara luas untuk menampilkan halaman web. HTML saat ini merupakan standar Internet yang didefinisikan dan dikendalikan penggunaannya oleh *World Wide Web Consortium* (W3C). HTML dibuat oleh kolaborasi Caillau TIM dengan Berners-lee Robert ketika mereka bekerja di CERN pada tahun 1989 (CERN adalah lembaga penelitian fisika energi tinggi di Jenewa).

2.7.3 *Hypertext Preprocessing* (PHP)

Menurut (Saputra, 2011) yang mengemukakan bahwa PHP atau yang memiliki kepanjangan PHP *Hypertext Preprocessor* merupakan suatu bahasa pemrograman yang difungsikan untuk membangun suatu *website* dinamis. PHP menyatu dengan kode HTML, maksudnya adalah beda kondisi. HTML digunakan sebagai pembangun atau pondasi dari kerangka *layout web*, sedangkan PHP

difungsikan sebagai prosesnya sehingga dengan adanya PHP tersebut, *web* akan sangat mudah di *maintenance*. PHP berjalan pada sisi *server* sehingga PHP disebut juga sebagai bahasa *Server Side Scripting*. Artinya bahwa dalam setiap menjalankan PHP, wajib adanya *web server*.

PHP ini bersifat *open source* sehingga dapat dipakai secara cuma-cuma dan mampu lintas *platform*, yaitu dapat berjalan pada sistem operasi *Windows* maupun *Linux*. PHP juga dibangun sebagai modul pada *web server apache* dan sebagai *binary* yang dapat berjalan sebagai CGI.

2.7.4 *JavaScript*

Javascript menurut (Sunyoto, 2007) adalah bahasa *scripting* yang populer di internet dan dapat bekerja di sebagian besar *browser* populer seperti *Internet Explorer* (IE), *Mozilla Firefox*, *Netscape* dan *Opera*. Kode *Javascript* dapat disisipkan dalam halaman web menggunakan tag *SCRIPT*.

Menurut (Utomo, 2007), *Javascript* merupakan bahasa scripting yang pertama kali dikembangkan oleh Netscape pada tahun 1955. Penulisan *Javascript* berada di dalam dokumen HTML dan pemanggilan program tersebut tergantung pada browser (navigator) yang digunakan dalam memanggil halaman yang terdapat pada script tersebut.

Selanjutnya menurut (Bride, 2007), *Javascript* adalah bahasa pemrograman berbasis browser. Kode-kodenya ditulis langsung ke dalam HTML dari halaman-halaman web dan diterjemahkan serta dieksekusi sebagai respon terhadap aktivitas-aktivitas pada halaman web.

Dengan adanya *javascript*, maka teknik penulisan HTML dapat dilaksanakan dan dijalankan dengan dua cara, yakni dengan membuat program *javascript* untuk menghasilkan dokumen HTML atau dengan membuat dokumen HTML layaknya seperti biasa, sesudah itu jika tersedia program *javascript*, maka menambahkan program *javascript* tersebut sebagai sisipan saja. Beberapa hal tentang *Javascript*:

1. *Javascript* didesain untuk menambah interaktif suatu web.
2. *Javascript* merupakan sebuah bahasa *scripting*.
3. Bahasa *scripting* merupakan bahasa pemrograman yang ringan.
4. *Javascript* berisi baris kode yang dijalankan di komputer (*web browser*).

5. *Javascript* biasanya disisipkan (*embedded*) dalam halaman HTML.
6. *Javascript* adalah bahasa *interpreter* (yang berarti skrip dieksekusi tanpa proses kompilasi).

2.7.5 *Framework Bootstrap*

Bootstrap merupakan sebuah *framework CSS* yang memudahkan pengembang untuk membangun website yang menarik dan responsif. *Bootstrap* adalah *CSS* tetapi dibentuk dengan *LESS*, sebuah *pre-prosessor* yang memberi fleksibilitas dari penggunaan *CSS* biasa. *Bootstrap* dapat dikembangkan dengan tambahan lainnya karena ini cukup fleksibel terhadap pekerjaan *web* yang mengutamakan desain (Otto, 2011).

Bootstrap adalah sebuah *library framework* CSS yang dibuat khusus untuk bagian pengembangan *front-end website*. bootstrap merupakan salah satu *framework* HTML, CSS dan *javascript* yang paling populer di kalangan *web developer*. pada saat ini hampir semua *web developer* telah menggunakan *bootstrap* untuk membuat tampilan *front-end* menjadi lebih mudah dan sangat cepat. karena anda hanya perlu menambahkan *class-class* tertentu untuk misalnya membuat tombol, *grid*, navigasi dan lainnya.

Bootstrap telah menyediakan kumpulan komponen *class interface* dasar yang telah dirancang sedemikian rupa untuk menciptakan tampilan yang menarik, bersih dan ringan. Selain komponen *class interface*, *bootstrap* juga memiliki fitur *grid* yang berfungsi untuk mengatur *layout* pada halaman *website* yang bisa digunakan dengan sangat mudah dan cepat. dengan menggunakan *bootstrap* kita juga diberi keleluasaan dalam mengembangkan tampilan *website* yang menggunakan *bootstrap* yaitu dengan cara mengubah tampilan *bootstrap* dengan menambahkan *class* dan CSS sendiri.

2.7.6 XAMPP

Menurut (MADCOM, 2016) “Xampp adalah sebuah paket kumpulan software yang terdiri dari Apache, MySQL, PhpMyAdmin, PHP, Perl, Filezilla, dan lain.” Xampp berfungsi untuk memudahkan instalasi lingkungan PHP, di mana biasanya lingkungan pengembangan *aplikasi* memerlukan PHP, Apache, MySQL dan PhpMyAdmin.

2.7.7 *MySQL*

Menurut (Winarno et al., 2014) yang mengemukakan bahwa *MySQL* adalah sebuah *software database*. MySQL merupakan tipe data relasional yang artinya MySQL menyimpan datanya dalam bentuk table-tabel yang saling berhubungan. Keuntungan menyimpan data di *database* adalah kemudahannya dalam penyimpanan dan menampilkan data karena dalam bentuk tabel. Sedangkan Menurut (Kustiyaningsih, 2011), MySQL adalah sebuah basis data yang mengandung satu atau jumlah tabel. Tabel terdiri atas sejumlah baris dan setiap baris mengandung satu atau sejumlah tabel. MySQL merupakan sistem manajemen hubungan antara basis data yang sangat cepat dan sempurna. MySql berupa alat bantu untuk memanipulasi basis data, sehingga bisa data dapat dengan mudah diisi, diambil, disusun dan diubah datanya. *Server* MySQL pun dapat mengatur kontrol akses dari data sehingga beberapa pengguna dapat sekaligus bekerja pada waktu yang bersamaan.

MySQL adalah *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi yang menggunakan *database* sebagai sumber dan pengolahan datanya, kepopuleran *MySQL* antara lain karena *MySQL* menggunakan *SQL* sebagai bahasa dasar untuk mengakses *database*-nya sehingga mudah untuk digunakan, kinerja *query* sangat cepat dan mencukupi untuk kebutuhan *database* perusahaan-perusahaan yang berskala kecil sampai menengah, *MySQL* juga bersifat tidak berbayar (Arief, 2011). Berikut beberapa kelebihan MySQL sebagai berikut yaitu:

1. Kemampuan yang tinggi.
2. Tidak dibutuhkan biaya untuk mendapatkan MySQL
3. Mudah untuk konfigurasi dan dipelajari
4. Dapat dijalankan pada beberapa sistem operasi seperti sistem *Unix* dan *Microsoft Windows*.

2.8 Pengujian Perangkat Lunak

Pengujian perangkat lunak merupakan suatu teknik yang digunakan menguji apakah sebuah perangkat lunak yang dihasilkan telah sesuai dengan yang diharapkan atau belum. Menurut (Pressman, 2012), pengujian adalah proses

eksekusi suatu program untuk menemukan kesalahan sebelum digunakan oleh pengguna akhir (*end-user*).

2.8.1 Pengujian Perhitungan Manual

Perhitungan manual yang dilakukan pada tahap proses melakukan enkripsi dan dekripsi dengan rumus sebagai berikut (Harahap, 2021):

1. Pertama mengubah kunci menjadi 10 sub kunci seperti CV[0], CV[1], CV[2], CV[3], CV[4], CV[5], CV[6], CV[7], CV[8], CV[9].
2. Mengubah *plaintext* menjadi 4 bagian (W0,W1,W2,W3). W0 pertama akan menjadi nilai G1 dan G2 pada proses enkripsi dan pada proses dekripsi W0 menjadi G5 dan G6.
3. Menghitung permutasi G untuk mendapatkan nilai W0 pada setiap awal putaran rule A-1 dan rule B2.

   Rumus perhitungan pada proses enkripsi:

   gi = F(gi+1^ cv[(4*k) mod 10]) ^ gi+1 maka:

   g3= F(g2 ^ cv[(4*k) mod 10]) ^ g1

   g4 = F(g3 ^ cv[(4*k) mod 10]) ^ g2

   g5 = F(g4 ^ cv[(4*k) mod 10]) ^ g3

   g6 = F(g5 ^ cv[(4*k) mod 10]) ^ g4

   W0 = W1 ^ W4 ^ i

   Rumus Perhitungan pada proses dekripsi:

   gi = F(gi-1 ^ cv[(4*k-1)+3 mod 10]) ^ gi-1

   g4 = F(g5 ^ cv[(4*k-1)+3 mod 10]) ^ g6

   g3 = F(g4 ^ cv[(4*k-1)+3 mod 10]) ^ g5

   g2 = F(g3 ^ cv[(4*k-1)+3 mod 10]) ^ g4

   g1 = F(g2 ^ cv[(4*k-1) +3 mod 10]) ^ g3

   W1 = W0 ^ W2 ^ i

2.8.2 Pengujian *Equivalance Partitions*

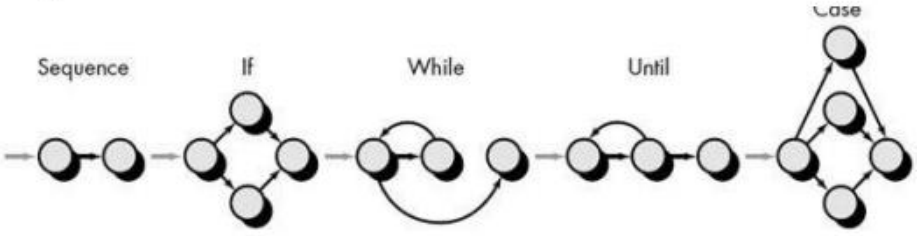
Equivalence Partitioning merupakan sebuah pengujian berdasarkan masukkan data pada setiap form dengan *Test Case*. *Test Case Equivalence* untuk kondisi masukan yang menggambarkan kumpulan keadaan yang valid atau tidak. *Equivalence Partitions* berdasarkan pada premis masukan dan keluaran dari suatu

komponen yang dipartisi ke dalam kelas kelas, menurut spesifikasi dari komponen tersebut, yang akan diperlakukan sama (ekivalen) oleh komponen tersebut (Utami & Asnawati, 2015). Pada proses pengujian terdapat tabel *Test Case* yang berfungsi untuk menyimpulkan apakah sistem berhasil dalam pengujian tipe tersebut atau tidak.

2.8.3 Pengujian *White Box*

Pengujian White Box atau yang disebut juga Glass Box testing adalah pengujian yang menggunakan struktur komponen internal suatu program untuk bahan uji. Pengujian White Box menjamin bahwa jalur independen dalam struktur komponen internal diperiksa setidaknya satu kali. Kemudian memeriksa pemilihan logika berdasarkan dari sisi benar dan salah. Yang terakhir adalah memeriksa perulangan sesuai dalam batasan operasional serta struktur data internal untuk menjamin validitasnya (Pressman, 2012). Teknik yang digunakan dalam pengujian White-Box salah satunya yaitu *Cyclomatic Complexity*.

Cyclomatic Complexity adalah sebuah software metric yang menyediakan ukuran kuantitatif dari kompleksitas logika dari sebuah program. Dengan menggunakan hasil pengukuran atau perhitungan dari metric *cyclomatic complexity*, kita dapat menentukan apakah sebuah program merupakan program yang sederhana atau kompleks berdasarkan logika yang diterapkan pada program tersebut. Apabila dikaitkan dengan pengujian perangkat lunak *(software testing), cyclomatic complexity* dapat digunakan untuk menentukan berapa minimal uji kasus yang harus dijalankan untuk menguji sebuah program dengan menggunakan teknik *basis path testing*. Pada pengujian *basis path*, aliran kontrol logika digambarkan dengan menggunakan *flow graph*. Berikut ini Gambar 2.12 adalah notasi struktur kontrol pada *flow graph* untuk menggambarkan sekuensial, seleksi, maupun perulangan (Pressman, 2014).

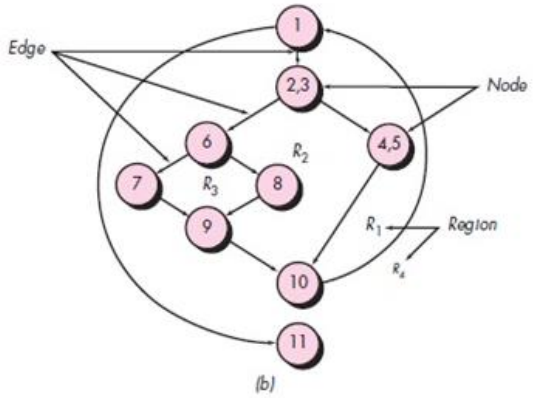


Gambar 2.12 Notasi *Flow Graph*

Notasi lingkaran disebut sebagai *flow graph node* yang digunakan untuk menggambarkan statement-statement berikut:

1. Satu atau lebih statement secara sekuensial yang dikelompokkan.
2. Percabangan seleksi dari satu statement kedua pilihan statement (seleksi)
3. Penggabungan dua statement yang dilanjutkan pada satu statement yang sama (merge)

Sedangkan notasi garis panah disebut sebagai *edge* atau *link*, menggambarkan aliran kontrol. Setiap *edge* harus dihubungkan dari/ke sebuah *node*, meskipun *node* tersebut tidak mewakili sebuah statement khusus. Berikut Contoh sebuah flow graph



Gambar 2.13 Contoh *flow graph*

Dari *flow graph* yang sudah tersedia, *cyclomatic complexity* dari sebuah program dapat dibuat dengan menggunakan rumus dibawah ini:

V(G) = E – N + 2

V(G) = *Cyclomatic Complexity*

E = Total jumlah *Edge*

N = Total jumlah *Node*

Pada contoh *flow graph* Gambar 2.13, dapat dihitung *cyclomatic complexity* nya sebagai berikut: V(G) = 11 edges – 9 nodes + 2 = 4.

Angka 4 dari hasil perhitungan *cyclomatic complexity* menunjukan jumlah *independent path* dari *basis path testing*, atau dengan kata lain menunjukkan jumlah pengujian yang harus dijalankan untuk memastikan semua statement pada program dijalankan minimal sekali (semua statement telah diuji).

Hasil *independent path* pada contoh diatas dapat dijabarkan sebagai berikut:

a. Path 1 = 1-11
b. Path 2 = 1-2-3-4-5-10-1-11
c. Path 3 = 1-2-3-6-8-9-10-1-11
d. Path 4 = 1-2-3-6-7-9-10-1-11

Catatan:

- *Independent path* adalah setiap path yang dilalui program yang menunjukkan satu set baru dari pemrosesan statement atau dari sebuah kondisi baru.
- *Independent path* pada *flow graph* harus melewati sedikitnya satu *edge* yang belum pernah dilewati oleh *path* sebelumnya.
- *Independent path* selalu dimulai dari node awal hingga ke node terakhir.
- *Independent path* yang dibuat pertama kali adalah *independent path* terpendek.

Untuk menetapkan standar nilai maksimum tipe prosedur dan tingkat resiko pada *cyclomatic complexity* dapat dilihat pada Tabel 2.7 sebagai berikut.

Tabel 2.7 Nilai *Cyclomatic Complexity*

Nilai CC	Tipe Prosedur	Tingkat Resiko
1-4	Prosedur sederhana	Rendah
5-10	Prosedur yang terstruktur dengan baik dan stabil	Rendah
11-20	Prosedur yang lebih kompleks	Menengah
21-50	Prosedur yang kompleks dan kritis	Tinggi
> 50	Rentan kesalahan, sangat mengganggu, prosedur tidak dapat diuji	Sangat Tinggi