IMPLEMENTASI ALGORITMA KOMBINASI RC4 DAN BASE64 UNTUK MENGAMANKAN DATABASE KLIEN PT INFOKES

Nurhikmah Taliasih[1], Irawan Afrianto[2]

[1,2] Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur No. 102 – 116 Bandung
E-mail : ntaliasih@gmail.com[1], irawan.afrianto@email.unikom.ac.id[2]

ABSTRAK

PT Infokes Indonesia merupakan sebuah perusahaan yang bergerak dalam pembangunan perangkat lunak yang berfokus pada layanan kesehatan. Saat ini pada PT Infokes transaksi *database* dengan klien dilakukan secara konvensional yaitu dengan mengirimkan *database* yang disimpan dalam media penyimpanan sekunder melalui kurir ekspedisi. Hal tersebut menyebabkan *database* dapat dengan mudah dibaca dan dikhawatirkan menjadi celah kebocoran informasi yang bersifat privasi kepada publik. Dalam penelitian ini menggunakan algoritma kriptografi RC4 yang memiliki kelebihan dalam kecepatan maupun tingkat efesiensi dalam penyimpanan data dari hasil enkripsinya. Kriptografi RC4 yang digunakan, dikombinasikan dengan Base64 untuk menambah penyandian setelah dilakukan enkripsi. Hasil pengujian *black box* dan *white box* pada penelitian ini menunjukan bahwa implementasi algoritma kriptografi RC4 dan algoritma Base64 berjalan dengan baik. Selain itu, hasil pengujian dengan menggunakan Wireshark menunjukkan bahwa jalur transaksi *database* telah diamankan dengan protokol HTTPS dan *database* dalam keadaan terenkripsi. Hasil pengujian dengan melakukan *cryptanalysis* menggunakan CrypTool pada kombinasi algoritma RC4 dan Base64 tidak dapat menghasilkan *database* asli, sehingga pengombinasian algoritma RC4 dan Base64 memiliki tingkat kemanan yang lebih baik untuk mengamankan *database* dibandingkan ketika hanya menggunakan algoritma RC4 saja.

Kata Kunci : Keamanan Data, Kriptografi, RC4, Base64, *Database*

1. PENDAHULUAN

PT Infokes Indonesia yang berfokus pada pengembangan produk dan solusi teknologi informasi kesehatan online dan terpadu di Indonesia telah menghasilkan beberapa produk di antaranya yaitu ePuskesmas, eHospital, eClinic, dan lain-lain. Pada PT Infokes arsip *database* klien disimpan dalam media penyimpanan sekunder. Selain itu transaksi *database* dengan klien dilakukan secara konvensional, yaitu dengan mengirimkan *database* yang sudah tersimpan di dalam media penyimpanan sekunder melalui kurir ekspedisi. Hal tersebut menyebabkan *database* dapat dibaca dengan mudah oleh orang lain dan dikhawatirkan menjadi celah kebocoran informasi yang bersifat privasi dan rahasia kepada publik. Meskipun belum pernah ada kasus di PT Infokes namun kebocoran informasi tersebut pernah terjadi, misalnya pada PT Pindad [1] dan Polsek Mangkubumi [2].

Teknik kriptografi merupakan salah satu alternatif solusi yang dapat digunakan dalam pengamanan informasi [1] [2] [3] [4]. Algoritma RC4 dipilih karena memiliki kelebihan dalam kecepatan pemrosesan data [5] bahkan mencapai 10 kali lebih cepat dari DES [6]. Selain itu, RC4 memiliki tingkat efisiensi yang baik dalam penyimpanan data pada *database*, karena hasil enkripsi yang dihasilkan sama jumlahnya dengan karakter aslinya [7] dan pada RC4, jika masukkan yang akan dienkripsi mengandung kata berulang maka akan menghasilkan ciphertext yang acak [8]. Meskipun demikian, algoritma RC4 memiliki kerentanan terhadap Bit Flipping Attack atau BFA [9] dan *cryptanalysis* yang dapat mengetahui pesan asli dari analisis terhadap kunci yang mungkin digunakan [10]. Sehingga dalam penelitian ini dilakukan upaya untuk meningkatkan kinerja algoritma RC4 dengan menambahkan kombinasi menggunakan algoritma Base64 untuk mengamankan transaksi *database* dengan klien.

1.1 *Database*

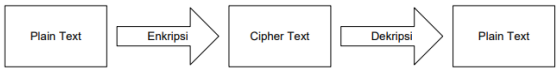
Secara umum, *database* berarti koleksi data yang saling terkait. Secara praktis, *database* dapat dianggap sebagai suatu penyusun data yang terstruktur yang disimpan dalam media pengingat (hard disk) yang tujuannya adalah agar data tersebut dapat diakses dengan mudah dan cepat [11].

1.2 Kriptografi

Teknik kriptografi adalah ilmu yang bersandarkan pada teknik matematika untuk berurusan dengan keamanan informasi seperti kerahasiaan, keutuhan data dan otentikasi entitas [12].

Pesan atau data asli sebelum dienkripsi disebut *plaintext*. Sedangkan pesan yang sudah diacak disebut *ciphertext*. Proses pengubahan *plaintext*

menjadi *ciphertext* disebut dengan enkripsi, sedangkan proses pengubahan *ciphertext* kembali menjadi *plaintext* disebut dengan dekripsi [13].



Gambar 1. Proses Enkripsi – Dekripsi [13]

1.3 RC4

Sistem sandi RC4 dikembangkan oleh Ronald Rivest pada tahun 1987 merupakan algoritma *stream cipher* yang paling banyak digunakan. Misalnya pada protokol SSL/TLS. RC4 merupakan *stream cipher* yang berorientasi *byte*. Masukan algoritma enkripsi RC4 merupakan sebuah *byte*, kemudian dilakukan operasi XOR dengan sebuah *byte* kunci, dan menghasilkan sebuah *byte* sandi [6] [12].

Persamaan proses enkripsi:

$ci = pi \oplus ki$ (1)

Persamaan proses dekripsi:

$pi = ci \oplus ki$ (2)

1.4 Base64

Algoritma Base64 merupakan salah satu algoritma untuk *encoding* dan *decoding*. Tujuan dari *encoding* adalah untuk merubah bentuk atau format data. Algoritma Base64 mengubah suatu data ke dalam format ASCII yang didasarkan pada bilangan dasar 64 atau bisa dikatakan sebagai salah satu metode yang digunakan untuk melakukan *encoding* (penyandian) terhadap data binary. Karakter yang dihasilkan pada transformasi Base64 ini terdiri dari A..Z, a..z dan 0..9, serta ditambah dengan dua karakter terakhir yang bersimbol yaitu + dan / serta satu buah karakter sama dengan (=) yang digunakan untuk penyesuaian dan menggenapkan data binary atau istilahnya disebut sebagai *padding* [4] [14].

Tabel 1. Tabel Indeks Base64 [4]

Index (6 bit data)	Char *Encoding* Base64	Index (6 bit data)	Char *Encoding* Base64	Index (6 bit data)	Char *Encoding* Base64	Index (6 bit data)	Char *Encoding* Base64
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
						pad	=

1.5 Internet

Internet adalah kelompok atau kumpulan dari jutaan komputer. Penggunaan internet memungkinkan untuk mendapatkan informasi dari komputer yang ada di dalam kelompok tersebut dengan asumsi bahwa pemilik komputer memberikan izin akses. Untuk mendapatkan sebuah informasi, sekumpulan protokol harus digunakan, yaitu sekumpulan aturan yang menetapkan bagaimana suatu informasi dapat dikirim dan diterima [15].

1.6 HTTPS *(HyperText Transfer Protocol)*

HTTPS merupakan HTTP yang menggunakan SSL (*Secure Socket Layer*). SSL adalah protokol enkripsi yang dipanggil melalui web server dengan menggunakan HTTPS. Penggunaan protokol SSL pada jaringan sangat penting untuk mengamankan data di dalam sambungan jaringan.. SSL merupakan jenis *sockets communications* yang berada di antara *transmision control protocol/internet protocol* (TCP/IP) dan *application layer* [16].

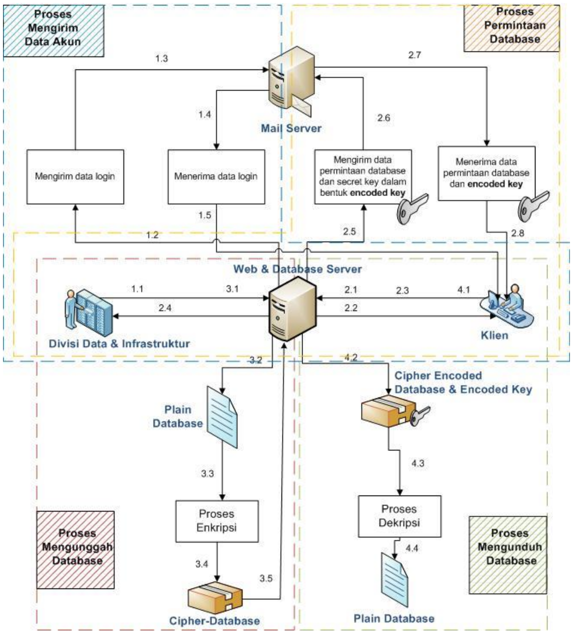
1.7 PHP

PHP secara umum dikenal sebagai bahasa pemrograman *script* yang membuat dokumen HTML secara *on the fly* yang dieksekusi di *server web*, dokumen HTML yang dihasilkan dari suatu aplikasi bukan dokumen HTML yang dibuat dengan menggunakan editor teks atau editor HTML. PHP dikenal juga sebagai bahasa pemrograman *server side* [17].

2. ISI PENELITIAN

2.1 Gambaran Umum Sistem

Sistem yang akan dibangun pada penelitian ini adalah sistem keamanan dengan kombinasi algoritma kriptografi dan *encoding* berbasis web. Di mana divisi Data dan Infrastruktur akan membuatkan akun untuk klien sesuai surat pesanan dan data akun tersebut akan dikirim kan melalui email. Selanjutnya klien dapat melakukan permintaan *database* melalui sistem. Divisi Data dan Infrastruktur akan mengunggah *database* sesuai permintaan dan melakukan enkripsi melalui sistem yang sudah dibangun. Proses enkripsi dilakukan dengan algoritma kriptografi RC4 dan *encoding* Base64. Kemudian *database* yang sudah terenkripsi atau berbentuk *cipher encoded database* akan tersimpan dalam *database server*. *Cipher encoded database* ini selanjutnya dapat diunduh klien untuk didekripsi menjadi file *database* asli dengan memasukkan *secret key* yang dikirim melalui email. *Secret key* yang dikirim dalam bentuk *encoded key* karena sudah dilakukan pengubahan sebelumnya dengan menggunakan algoritma *encoding* Base64.



Gambar 2. Gambaran Umum Sistem

2.2 Analisis Algoritma

Pada penelitian ini algoritma yang digunakan adalah algoritma kriptografi RC4 dan algoritma Base64. Tahap analisis algoritma digunakan untuk mengetahui rangkaian proses dari algoritma yang digunakan sehingga dapat diimplementasikan ke dalam sistem yang dibangun..

2.2.1 Analisis Pembangkitan Kunci RC4

Proses pembangkitan kunci dilakukan terlebih dahulu sebelum melakukan enkripsi maupun dekripsi pada algoritma RC4. RC4 yang merupakan bagian dari tipe *stream cipher* memiliki cara yang sama dengan konsep umum membangkitkan *keystream* pada *stream cipher*. *Keystream* dibangkitkan dengan *keystream generator* lalu dilakukan XOR antara *key* tersebut dengan *plaintext.*

Algoritma RC4 memakai S-box (*Substitution-box*) dengan larik 256 *byte* yang berukuran 16 x 16 sebagai *keystream generator*.

Proses pembangkitan kunci pada RC4 adalah sebagai berikut:

1) Inisialiasasi S-Box

Pertama, inisialisasi S-Box dengan panjang 256 *byte*, dengan S[0]=0, S[1]=1, S[2]=2,…, S[255]=255 sehingga *array* S menjadi:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

2) Melakukan padding kunci K sehingga panjang kunci K = 256

Inisialisasi 256 *byte* kunci *array* K. Misalkan kunci (secret key) adalah: **ifk** ulang kunci sampai memenuhi seluruh *array* k dan ubah format ke dalam bentuk ASCII. K[0]=i=105, K[1]=f=102, K[2]=k=107,…,[255]=i=105.

Sehingga *array* K menjadi:

105	102	107	105	102	107	105	102	107	105	102	107	105	102	107	105
102	107	105	102	107	105	102	107	105	102	107	105	102	107	105	102
107	105	102	107	105	102	107	105	102	107	105	102	107	105	102	107
105	102	107	105	102	107	105	102	107	105	102	107	105	102	107	105
102	107	105	102	107	105	102	107	105	102	107	105	102	107	105	102
107	105	102	107	105	102	107	105	102	107	105	102	107	105	102	107
105	102	107	105	102	107	105	102	107	105	102	107	105	102	107	105
102	107	105	102	107	105	102	107	105	102	107	105	102	107	105	102
107	105	102	107	105	102	107	105	102	107	105	102	107	105	102	107
105	102	107	105	102	107	105	102	107	105	102	107	105	102	107	105
102	107	105	102	107	105	102	107	105	102	107	105	102	107	105	102
107	105	102	107	105	102	107	105	102	107	105	102	107	105	102	107
105	102	107	105	102	107	105	102	107	105	102	107	105	102	107	105
102	107	105	102	107	105	102	107	105	102	107	105	102	107	105	102
107	105	102	107	105	102	107	105	102	107	105	102	107	105	102	107
105	102	107	105	102	107	105	102	107	105	102	107	105	102	107	105

3) Permutasi untuk S-Box

Operasi permutasi S-Box akan menghasilkan *array* di dalam S-Box yang berbeda dari urutan sebelumnya. Dalam operasi ini digunakan variabel i dan j ke indeks *array* S[i] dan K[i]. Pada permulaan akan dilakukan inisialisasi i dan j dengan 0. Operasi ini dilakukan sebanyak 256 kali dengan pengulangan rumusan (j + S[i]+K[i]) mod 256 yang diikuti dengan penukaran S[i] dengan S[j].

for i = 0 to 255
j = (j + S[i] + K[i]) mod 256
swap S[i] dan S[j]

Dengan algoritma seperti di atas maka dengan nilai awal i = 0, j=0 dan dilakukan perulangan sampai i=255, akan menghasilkan *array* S seperti di bawah ini:

iterasi ke-1:
i = 0, maka
j = (j + S[i] + K[i]) mod 256
= (j + S[0] + K[0]) mod 256
= (0 + 0 + 105) mod 256
= 105

Swap S[0] dan S[105] sehingga menghasilkan *array*:

105	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
96	97	98	99	100	101	102	103	104	0	106	107	108	109	110	111
112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Iterasi dilanjutkan sampai iterasi ke-256. Akhir dari iterasi dan *swap* yang dilakukan akan menghasilkan *array* sebagai berikut:

105	97	61	102	92	108	242	125	210	165	194	42	0	18	139	176
121	44	226	161	17	8	65	1	215	23	55	135	201	227	162	239
122	112	140	85	12	46	250	83	225	177	231	134	218	211	59	4
53	51	155	131	101	164	116	156	74	224	133	93	86	149	228	5
117	178	234	22	157	75	254	127	90	16	172	111	77	196	64	118
179	203	57	56	251	123	202	252	186	39	245	174	28	50	151	113
34	98	37	153	79	24	119	130	146	76	144	173	212	190	114	10
63	147	115	246	30	168	132	15	232	189	126	214	197	160	206	120
171	222	198	19	248	62	185	217	88	229	240	100	71	60	243	237
29	110	163	109	205	89	26	191	148	27	183	2	45	166	142	167
95	182	106	124	20	58	3	6	223	187	209	181	43	145	141	66
169	67	47	87	72	244	170	193	82	200	73	188	103	68	216	255
192	221	54	154	159	184	249	220	199	235	40	99	152	21	204	38
136	70	158	36	49	78	208	104	81	207	230	96	32	137	52	238
236	94	107	241	219	253	129	233	9	69	143	91	31	128	80	7
41	180	35	175	150	84	33	213	14	25	195	13	48	11	247	138

4) Membangkitkan *random key*

Berikutnya adalah proses pembangkitan kunci yang diperoleh secara random dengan melakukan operasi perhitungan sebanyak jumlah *byte* dari *plaintext*. Perhitungannya adalah sebagai berikut:

Pembangkitan *random key* ke-1
Inisialiasasi terlebih dahulu nilai i = 0; j = 0
i = (i + 1) mod 256
= (0 + 1) mod 256
= 1 j = (j + S[i]) mod 256
= (0 + S[1]) mod 256
= (0 + 97) mod 256
= 97

Sehingga hasilnya:
S[i]=S[1] dan S[j]=S[97]

Kemudian lakukan *swap* terhadap *array* hasil iterasi terakhir :
S[i]=S[97]=98 dan S[j]=S[1]=97

t = (S[i] + S[j]) mod 256
= (98 + 97) mod 256
= 195

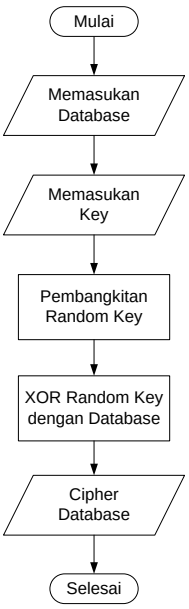
K = S[t] = S[195] = 154 (nilai yang akan di-XOR-kan dengan *byte plaintext* ke-1)

Operasi pembangkitan *random key* dilanjutkan sampai sebanyak *byte plaintext* yang akan dienkripsi dan akan menghasilkan *random key* sebagai berikut:

154	99	54	33	86	200	242	36	0	29	165	238	174	108	144	91
241	6	248	77	197	179	69	196	243	137	109	16	173	8	179	230
238	150	4	202	58	201	209	141	48	221	244	112	157	35	178	39
195	203	237	172	15	231	51	85	37	20	190	11	195	188	9	215
118	46	87	13	46	253	133	170	67	181	209	85	...			

2.2.2 Analisis Algoritma Enkripsi RC4

Algoritma enkripsi RC4 beroperasi dalam *byte*, berarti XOR dilakukan setiap satu *byte plaintext* dengan satu *byte* key dari keystream. *Flowchart* dari algoritma enkripsi RC4 adalah sebagai berikut:



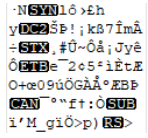
Gambar 3. *Flowchart* Algoritma Enkripsi RC4

Dari proses pembangkitan kunci sebelumnya telah diperoleh *random key*, selanjutnya dilakukan operasi XOR antara *byte* dari *random key* dengan *plaintext* yang ditampilkan pada tabel berikut:

Tabel 2. Operasi XOR Enkripsi RC4

RANDOM KEY			PLAINTEXT			XOR		
Char	Des	Biner	Char	Des	Biner	Char	Des	Biner
š	154	10011010	-	45	00101101	·	183	10110111
c	99	01100011	-	45	00101101	N	78	01001110
6	54	00110110		32	00100000	[SYN]	22	00010110
!	33	00100001	M	77	01001101	l	108	01101100
V	86	01010110	y	121	01111001	ô	244	11110100
È	200	11001000	S	83	01010011	›	155	10011011
ò	242	11110010	Q	81	01010001	£	163	10100011
$	36	00100100	L	76	01001100	h	104	01101000
0	0	00000000		32	00100000		32	00100000
	29	00011101	d	100	01100100	y	121	01111001
¥	165	10100101	u	117	01110101	[DO2]	18	00010010
î	238	11101110	m	109	01101101	Š	138	10001010
®	174	10101110	p	112	01110000	Þ	222	11011110
l	108	01101100		32	00100000	!	33	00100001
□	144	10010000	1	49	00110001	¡	161	10100001
[	91	01011011	0	48	00110000	k	107	01101011
...								

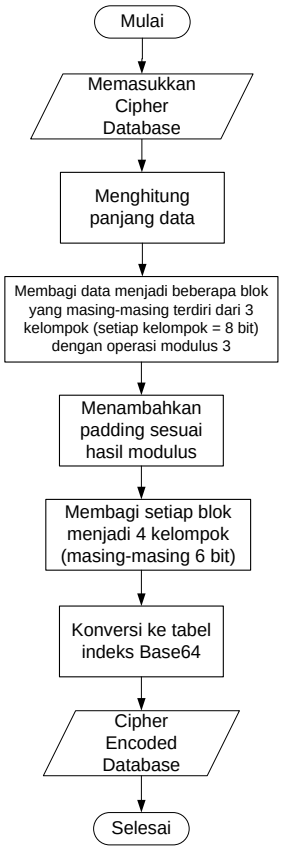
Dari proses XOR antara *plaintext* dengan *random key*, maka hasil enkripsi RC4 selengkapnya seperti yang ditunjukan



Gambar 4. Hasil Enkripsi RC4

2.2.3 Analisis Algoritma *Encoding* Base64

Berdasarkan hasil enkripsi RC4 pada **Gambar 4.** Hasil Enkripsi RC4, selanjutnya dilakukan *encoding* menggunakan algoritma Base64 dengan tahapan proses sebagai berikut:



Gambar 5. *Flowchart* Algoritma *Encoding* Base64

Dalam penelitian ini, algoritma Base64 dioperasikan setelah algoritma RC4 sehingga masukkannya berupa hasil enkripsi RC4 (*cipher database*). Contoh implementasinya adalah sebagai berikut:

Operasi Algoritma *Encoding* Base64 dari kelompok *byte* ke-1:

Cipher	·				N				[SYN]			
ASCII	183				78				22			
Bit	1	0	1	1	0	1	1	0	1	0	0	0	1	1	1	0	0	0	0	1	0	1	1	0
Index	45				52				56				22			
Cipher Encoded	t				0				4				W			

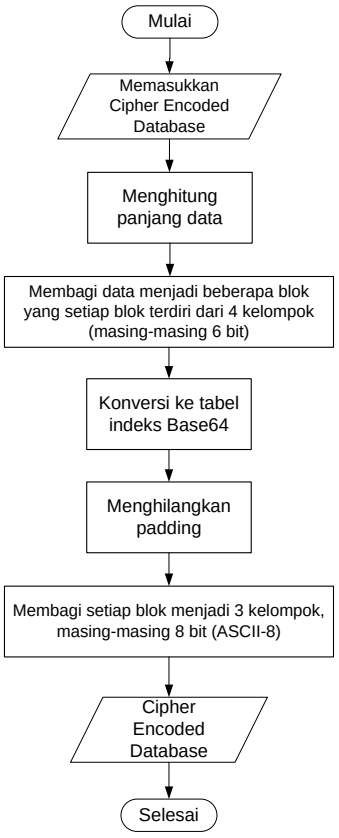
Operasi dilanjutkan sampai akhir hingga menghasilkan *cipher encoded database* seperti **Gambar 6**.

```
t04WbPSbo2ggeR
KK3iGha983zm3C
9wK4I9t+lOWhSn
nqlBdlrzKiNbLs
yHTGTyucMDn6lk
fAxbrGQt4Yr7CT
ZoY60hrvkklfZ+
```

Gambar 6. Hasil *Encoding* Base64

2.2.4 Analisis Algoritma *Decoding* Base64

Jika dalam proses *encoding* data dikelompokkan per 3 *byte* = 24 bit (1 *byte* = 1 karakter), di dalam proses *decoding*nya tetap dikelompokkan dalam blok yang berisi 24 bit data namun, 1 karakter diwakili oleh 6 bit data. Proses *decoding* pada Base64 diilustrasikan seperti gambar berikut ini:



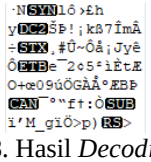
Gambar 7. *Flowchart* Algoritma *Decoding* Base64

Cipher encoded database yang akan diimplementasikan sesuai hasil *encoding* Base64 pada **Gambar 6**. Sehingga prosesnya adalah sebagai berikut:

Operasi Algoritma *decoding* Base64 dari kelompok *byte* ke-1:

Cipher Encoded	t				0				4				W			
Index	45				52				56				22			
Bit	1	0	1	1	0	1	1	1	0	1	0	0	1	1	1	0	0	0	0	1	0	1	1	1
ASCII	183								78								22							
Cipher	·								N								[SYN]							

Operasi dilanjutkan sampai akhir hingga menghasilkan *cipher database* seperti berikut

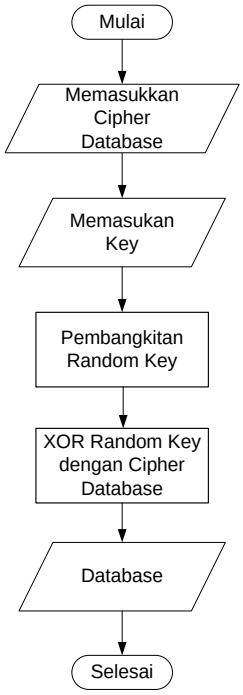


Gambar 8. Hasil *Decoding* Base64

2.2.5 Analisis Algoritma Dekripsi RC4

RC4 menggunakan fungsi dekripsi dan enkripsi yang sama karena operasi yang dilakukan hanyalah XOR antara *random key* yang dibangkitkan dengan *plaintext*. Sehingga pada dekripsinya, proses XOR

dilakukan antara *random key* dengan *cipher database*. *Flowchart* proses dekripsi RC4 adalah sebagai berikut:



Gambar 9. *Flowchart* Algoritma Dekripsi RC4

Pada proses dekripsi menghasilkan pembangkitan kunci yang sama ketika melakukan enkripsi. Operasi XOR yang dilakukan pada tahap dekripsi adalah sebagai berikut:

Tabel 3. Operasi XOR Dekripsi RC4

CIPHER DATABASE			*RANDOM KEY*			HASIL XOR		
Char	**Des**	**Bin**	**Char**	**Des**	**Bin**	**Char**	**Des**	**Bin**
·	183	10110111	š	154	10011010	-	45	00101101
N	78	01001110	c	99	01100011	-	45	00101101
	22	00010110	6	54	00110110		32	00100000
l	108	01101100	!	33	00100001	M	77	01001101
ô	244	11110100	V	86	01010110	y	121	01111001
›	155	10011011	È	200	11001000	S	83	01010011
£	163	10100011	ò	242	11110010	Q	81	01010001
h	104	01101000	$	36	00100100	L	76	01001100
	32	00100000	0	0	00000000		32	00100000
y	121	01111001		29	00011101	d	100	01100100
	18	00010010	¥	165	10100101	u	117	01110101
Š	138	10001010	î	238	11101110	m	109	01101101
Þ	222	11011110	®	174	10101110	p	112	01110000
!	33	00100001	1	108	01101100		32	00100000
¡	161	10100001	□	144	10010000	1	49	00110001
k	107	01101011	[	91	01011011	0	48	00110000
.....								

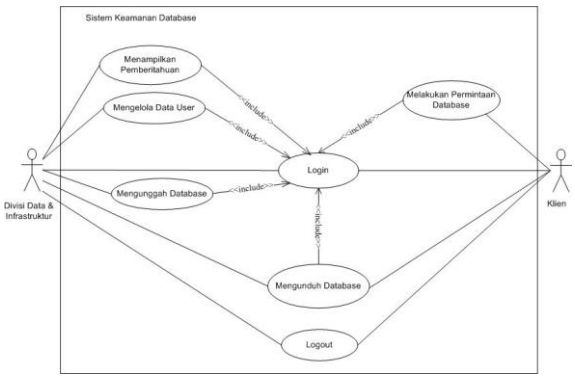
Dari proses XOR antara *cipher database* dengan *random key*, menghasilkan *plain database* sebagai berikut:

```
-- MySQL
dump
10.16
Distrib
10.3.10-Ma
riaDB,
```

Gambar 10. Hasil Dekripsi RC4

2.3 *Use Case* Diagram

Use case diagram merupakan diagram yang menggambarkan interaksi antara sistem yang dibangun dengan pengguna yang disebut aktor maupun dengan sistem lain atau eksternal. Berikut ini **Gambar 11** merupakan use case diagram yang terdiri dari dua actor yaitu Divisi Data & Infrastruktur dan Klien.

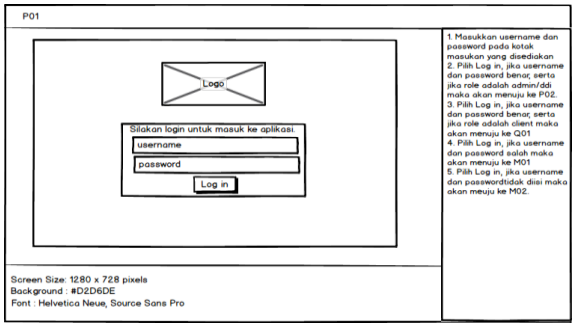


Gambar 11. *Use Case Diagram* Sistem Keamanan *Database*

2.4 Perancangan Antarmuka

Perancangan antarmuka merupakan desain tampilan dari sebuah sistem yang bertujuan untuk menggambarkan sistem yang akan dibangun.

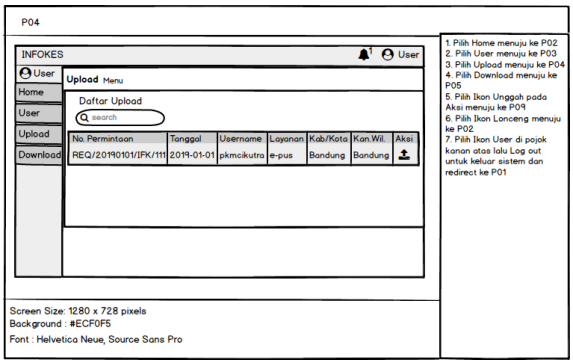
Perancangan antarmuka sistem keamanan *database* PT Infokes untuk login adalah sebagai berikut:



Gambar 12. Perancangan Antarmuka Login

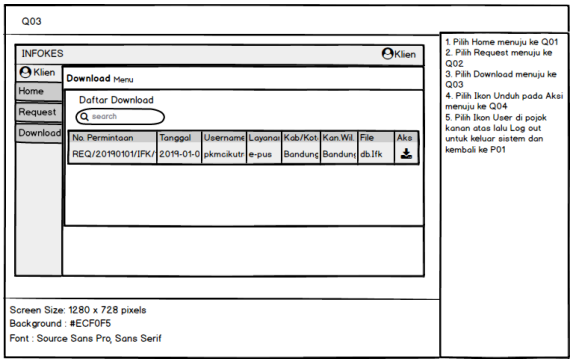
Untuk perancangan antarmuka setelah login akan dibagi menjadi dua, sesuai hak aksesnya. Yaitu hak akses untuk admin yang digunakan oleh Divisi Data & Infrastruktur, dan hak akses klien yang digunakan oleh klien PT Infokes.

Perancangan antarmuka unggah *database* pada halaman Divisi Data & Infrastruktur adalah sebagai berikut:



Gambar 13. Perancangan Antarmuka Unggah *Database*

Sedangkan pada sisi klien, terdapat perancangan antarmuka unduh *database* seperti ditampilkan gambar berikut:



Gambar 14. Perancangan Antarmuka Unduh *Database*

2.5 Pengujian

Pengujian yang dilakukan ada beberapa proses yaitu pengujian *black box*, pengujian *white box* dan pengujian keamanan *database* dengan menggunakan *CrypTool* versi 1.4.41 dan *Wireshark* versi 3.0.1..

2.5.1 Pengujian *Black Box*

Pengujian *black box* merupakan tahap uji secara fungsional yang dilakukan untuk mengetahui kesesuaian fungsi-fungsi di dalam sistem beserta masukan dan keluaran dengan spesifikasi yang dibutuhkan. Berikut ini merupakan pengujian *black box* pada proses melakukan permintaan *database* yang dilakukan oleh klien:

Tabel 4. Pengujian *Black Box* Melakukan Permintaan *Database*

Kasus dan Hasil Uji (data normal)			
Data Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Secret Key: ifk	Setelah klik 'Proses' data tersebut berhasil disimpan dan tampil notifikasi permintaan *database* pada halaman Divisi Data & Infrastruktur serta terkirim *encoded key* ke *email* klien yang melakukan permintaan.	Setelah klik 'Proses' data tersebut berhasil disimpan dan tampil notifikasi permintaan *database* pada halaman Divisi Data & Infrastruktur serta terkirim *encoded key* ke *email* klien yang melakukan permintaan.	Diterima
Kasus dan Hasil Uji (data kosong)			
Data Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Secret Key : (tidak diisi)	Tidak bisa melakukan permintaan dan muncul pesan "Data tidak boleh kosong."	Tidak bisa melakukan permintaan dan muncul pesan "Data tidak boleh kosong."	Diterima

2.5.2 Pengujian *White box*

Pada tahap ini, pengujian *white box* yang akan digunakan merupakan bentuk Basis Path Testing. Pengujian dilakukan pada proses pembangkitan kunci dan enkripsi RC4.

Pengujian pembangkitan kunci dan enkripsi RC4 diawali dengan melakukan pemeriksaan pada *source code* kemudian mengubahnya ke dalam bentuk *flowgraph*. Tahap pengujian *white box* adalah sebagai berikut:

1. Memeriksa *Source Code* Pembangkitan Kunci dan Enkripsi RC4.

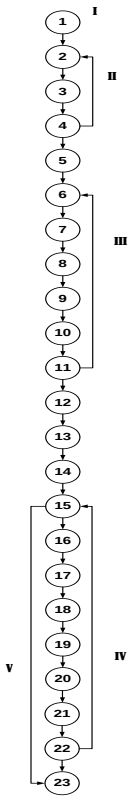
Berikut adalah *source code* atau kode program dari proses pembangkitan kunci dan enkripsi RC4 yang dijalankan sistem.

Tabel 5. *Source Code* Pembangkitan Kunci dan Enkripsi RC4

Line	*Source Code*
1	$s = array();
2	for ($i = 0; $i < 256; $i++) {
3	$s[$i] = $i;
4	}
5	$j = 0;
6	for ($i = 0; $i < 256; $i++) {
7	$j = ($j + $s[$i] + ord($key[$i % strlen($key)])) % 256;
8	$x = $s[$i];
9	$s[$i] = $s[$j];
10	$s[$j] = $x;
11	}
12	$i = 0;
13	$j = 0;
14	$res = '';
15	for ($y = 0; $y < strlen($str); $y++) {
16	$i = ($i + 1) % 256;
17	$j = ($j + $s[$i]) % 256;
18	$x = $s[$i];
19	$s[$i] = $s[$j];
20	$s[$j] = $x;
21	$res .= $str[$y] ^ chr($s[($s[$i] + $s[$j]) % 256]);
22	}
23	return $res;

2. Membuat *Flowgraph* Pembangkitan Kunci dan Enkripsi RC4

Flowgraph pembangkitan kunci dan enkripsi RC4 seperti ditampilkan pada **Gambar 15**.



Gambar 15. *Flowgraph* Pembangitan Kunci dan Enkripsi RC4

3. Menghitung *Cyclomatic complexity*

a. Menghitung *cyclomatic complexity* dari jumlah region
   V(G)= Region = 5

b. Menghitung *cyclomatic complexity* dari Edge dan Node
   V(G)= Edge – Node + 2
   = 26 – 23 + 2 = 5

c. Menghitung *cyclomatic complexity* dari Predicate Code (P)
   V(G)= P + 1
   = 4 + 1 = 5

4. Menentukan *Independet Path*
   Path 1 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23
   Path 2 = 1-2-3-4-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23
   Path 3 = 1-2-3-4-5-6-7-8-9-10-11-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23
   Path 4 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-15-16-17-18-19-20-21-22-23
   Path 5 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-23

5. Membuat *Graph* Matriks Pembangkitan Kunci dan Enkripsi RC4

Dalam pengujian *basis path*, digunakan *graph* matriks yang merupakan matrik berukuran sejumlah *node* pada *flowgraph*. Adapun *graph* matriks pembangkitan kunci dan enkripsi RC4 adalah sebagai berikut.

Tabel 6. *Graph* Matriks Pembangkitan Kunci dan Enkripsi RC4

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	n(E)-1
1		1																						0
2			1																					0
3				1																				0
4		1			1																			1
5						1																		0
6							1																	0
7								1																0
8									1															0
9										1														0
10											1													0
11						1						1												1
12													1											0
13														1										0
14															1									0
15																1							1	1
16																	1							0
17																		1						0
18																			1					0
19																				1				0
20																					1			0
21																						1		0
22															1								1	1
23																								0
Total																								4

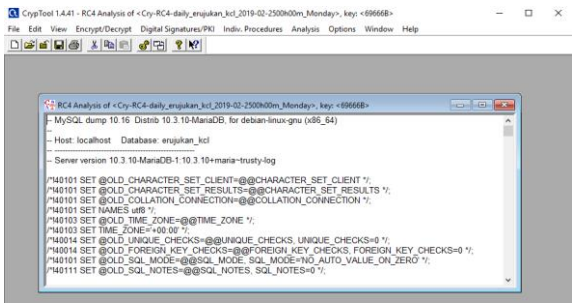
6. Kesimpulan Pengujian Pembangkitan Kunci dan Enkripsi RC4

Berdasarkan pengujian *white box* pada pembangkitan kunci dan enkripsi RC4 dihasilkan nilai *cyclomatic complexity* yang sama yaitu 5. Sehingga dapat disimpulkan bahwa sistem berjalan dengan baik, karena setiap pengujian menghasilkan nilai yang sama.

2.5.3 Pengujian Menggunakan *CrypTool*

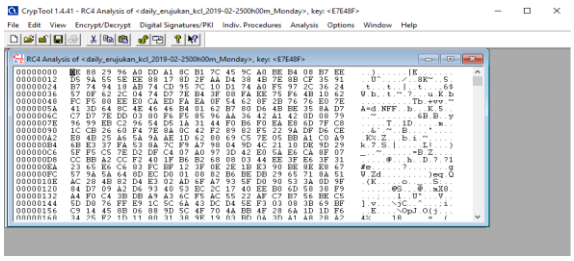
Pengujian ini dilakukan untuk membuka *database* menggunakan *CrypTool*. Untuk pengujian keamanan enkripsi sebagai perbandingan dilakukan pengujian terhadap hasil enkripsi RC4 dan pengujian terhadap hasil pengombinasian enkripsi RC4 dan *encoding* Base64**.**

Pengujian terhadap hasil enkripsi RC4 dengan menggunakan *CrypTool*, hasil *cryptanalysis* menunjukkan bahwa data sesuai dengan *database* asli atau *plain database* seperti ditampilkan pada **Gambar 16**



Gambar 16. Hasil *Cryptanalysis* Pada RC4

Sementara pengujian terhadap hasil enkripsi RC4 yang dikombinasi *encoding* Base64 dengan menggunakan *CrypTool*, hasil *cryptanalysis* menunjukkan bahwa data tidak sesuai dengan *database* asli atau *plain database* seperti ditampilkan pada **Gambar 17**.

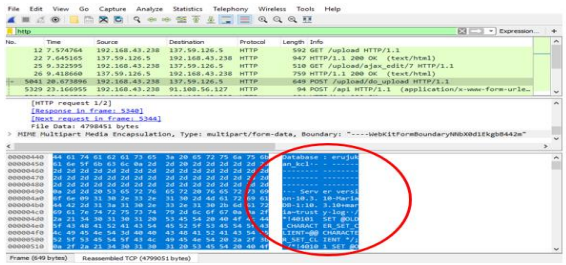


Gambar 17. Hasil *Cryptanalysis* Pada Kombinasi RC4 dan Base64

2.5.4 Pengujian Menggunakan *Wireshark*

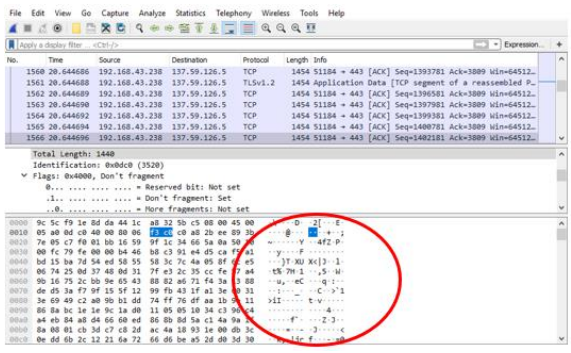
Pengujian keamanan *database* menggunakan *Wireshark* dilakukan untuk mengetahui apakah *database* yang dikirimkan dalam keadaan aman atau tidak. Pengujian dilakukan pada dua tahap yaitu pengujian pada saat sistem web yang dibangun masih menggunakan protokol HTTP dan pengujian setelah menggunakan protokol HTTPS. Maisng-masing pengujian dilakukan proses mengunggah *database*.

Pada pengujian dengan protokol HTTP, saat dilakukan proses mengunggah *database*, aplikasi Wireshark dapat dengan mudah memonitor lalu lintas pengiriman *database.* Pada pengujian ini dapat diketahui bahwa *database* yang diunggah dapat dibaca dengan mudah seperti ditampilkan pada **Gambar 18**



Gambar 18. Hasil Pengujian Keamanan *Database* pada protokol HTTP

Sementara hasil pemonitoran Wireshark pada sistem web yang sudah menggunakan protokol HTTPS dapat dilihat bahwa jalur dari pengguna ke server sudah diamankan dengan enkripsi. Sehingga *database* asli yang diunggah tidak dapat dibaca seperti ditampilkan pada **Gambar 19**.



Gambar 19. Hasil Pengujian Keamanan *Database* pada protokol HTTPS

3. PENUTUP

3.1 Kesimpulan

Berdasarkan uraian analisis, perancangan, implementasi sampai pengujian, sehingga dapat ditarik kesimpulan sebagai berikut:

1. Sistem dapat mengamankan *database* dengan kriptografi. Berdasarkan pengujian *black box* dan *white box* menunjukan bahwa proses unggah dengan enkripsi RC4 serta *encoding* Base64 dan unduh dengan *decoding* Base64 serta dekripsi RC4 pada *database* berjalan dengan baik.
2. Berdasarkan pengujian menggunakan Wireshark, menujukkan bahwa hasil pemonitoran pada lalu lintas data tidak dapat membaca *database* asli sehingga sistem dapat mengamankan proses transaksi *database* dari Divisi Data dan Infrastruktur kepada klien.
3. Berdasarkan pengujian menggunakan CrypTool, kombinasi algoritma dengan menambahkan Base64 pada algoritma kriptografi RC4 dapat meningkatkan kinerja RC4 dari serangan cryptanalysis yang dilakukan.

3.2 Saran

Sementara saran untuk pengembangan selanjutnya supaya sistem berjalan dengan optimal adalah sebagai berikut :

1. Dapat melakukan proses unggah yang termasuk di dalamnya proses enkripsi lebih dari satu permintaan *database* yang dilakukan oleh klien.
2. Diharapkan pada pengembangan selanjutnya algoritma yang digunakan tidak hanya RC4 dan Base64 namun dikembangkan juga dengan menggunakan algoritma lainnya.

DAFTAR PUSTAKA

[1] A. D. Hidayat dan I. Afrianto, “Sistem Kriptografi Citra Digital Pada Jaringan Intranet Menggunakan Metode Kombinasi Chaos Map dan Teknik Selektif,” *Ultimatics*, vol. 9, pp. 59-66, 2017.

[2] M. Septian, “Penerapan Enkripsi Rabbit Stream Cipher Algorithm untuk Mengamankan File Bersifat Rahasia Di Polsek Mangkubmi Tasikmalaya,” dalam *Skripsi*, Bandung, Universitas Komputer Indonesia, 2018.

[3] R. Aulia, A. Zakir dan A. D. Purwanto, “Penerapan Kombinasi Algoritma Base64 dan ROT47 untuk Enkripsi Database Pasien Rumah Sakit Jiwa Prof. Dr. Muhammad Ildrem,” *Jurnal Nasional Informatika dan Teknologi Jaringan*, vol. 2, pp. 146-151, 2018.

[4] F. W. Christanto, A. P. Rahangiar dan F. d. Fretes, “Penerapan Algoritma Gabungan RC4 dan Base64 pada Sistem Keamanan ECommerce,” dalam *Seminar Nasional*

Aplikasi Teknologi Informasi 2012 (SNATI 2012), Yogyakarta, 2012.

[5] A. A. Okedola dan Y. N. Asafe, "RSA and RC4 Cryptosystem Performance Evaluation Using Image and Text File," *International Journal of Scientific & Engineering Research*, vol. 6, no. 5, pp. 289-294, 2015.

[6] W. K. Semarang, Memahami Model Enkripsi dan Security Data, Yogyakarta: Andi, 2013.

[7] S. A. Agusta dan Triyani A. F. A, "Implementasi Algorithma Stream Cipher RC4 dalam Aplikasi Pendataan Alumni STMIK Amik Riau," *Inovtek Polbeng - Seri Informatika*, vol. 1, pp. 1-8, 2016.

[8] Y. Prayudi dan I. Halik, "Studi dan Analisis ALgoritma Rivest Code 6 (RC6) dalam Enkripsi/Dekripsi Data," dalam *Seminar Nasional Aplikasi Teknologi Informasi 2005 (SNATI 2005)*, Yogyakarta, 2005.

[9] V. S. Arintamy, N. D. W. Cahyani dan A. Mulyana, "Analisis Algorirtma RC4 Sebagai Metode Enkripsi WPA-PSK Pada Sistem Keamanan Jaringan Wireless LAN," *e-Proceeding of Engineering*, vol. 1, pp. 28-35, 2014.

[10] P. Xue , T. Li , H. Dong , C. Liu , W. Ma dan S. Pei , "GB-RC4: Effective brute force attacks on RC4 algorithm using GPU," *Seventh International Green and Sustainable Computing Conference (IGSC)*, pp. 1-6, 2016.

[11] A. Kadir, Tuntutan Praktis Belajar Database Menggunakan MySQL, Yogyakarta: Andi, 2008.

[12] R. Sadikin, "Kriptografi untuk Keamanan Jaringan dan Implementasinya dalam Bahasa Java", Yogyakarta: Andi, 2012.

[13] D. Ariyus, "Pengantar Ilmu Kriptografi: Teori Analisis & Implementasi", Yogyakarta: Andi, 2008.

[14] Y. Adriansyah, "Enkripsi Sederhana dengan Base64 dan Substitusi Monoalfabetik ke Huruf Non-Latin," *Makalah Mahasiswa Teknologi Bandung*, 2010.

[15] J. Simarmata, Rekayasa Web, Yogyakarta: Andi, 2010.

[16] H. Pranata, L. A. Abdillah dan U. Ependi, "Analisis Keamanan Protokol Secure Socket Layer (SSL) Terhadap Proses Sniffing di Jaringan," dalam *Student Colloquium Sistem Informasi & Teknik Informatika (SC-SITI)*, Palembang, 2015.

[17] B. Sidik, Pemrograman Web dengan PHP Revisi Kedua, Bandung: Informatika Bandung, 2014.