RESOLUSI : Rekayasa Teknik Informatika dan Informasi
ISSN 2745-7966 (Media Online)
Vol 2, No 2, November 2021
Hal 58-66
https://djournals.com/resolusi

Pengamanan Data Arsip Pada Balai Desa Sidodadi Menggunakan Kriptografi Modern RC4

Khairul Fahmi[1*]

[1]Fakultas Ilmu Komputer dan Teknologi Informasi, Program Studi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia
Email: [1]khairul.fahmi011@gmail.com

Abstrak−Masalah keamanan dan kerahasiaan data adalah hal yang sangat penting di Balai Desa. Balai Desa memiliki banyak dokumen yang bersifat rahasia seperti data penduduk dan juga data lainnya. Dokumen atau data rahasia tidak boleh diketahui oleh pihak – pihak luar karena dapat menimbulkan kerugian bagi petugas dan pemerintahan desa. Untuk mengamankan data tersebut maka digunakan algoritma RC4. Algoritma RC4 dipilih karena waktu prosesnya lebih cepat dibanding algoritma lain. Algoritma RC4 menggunakan dua buah Substitution Box (S-Box) yaitu array sepanjang 256 yang berisi permutasi dari bilangan 0 sampai 255, dan S-Box kedua yang berisi permutasi merupakan fungsi dari kunci dengan panjang variabel.

Kata Kunci: Kriptografi, Data Arsip, Enkripsi, Dekripsi, RC4

Abstract−The issue of data security and confidentiality is very important at the Village Hall. The Village Hall has many confidential documents such as population data and other data. Confidential documents or data should not be known by outside parties because they can cause harm to village officials and government. To secure the data, the RC4 algorithm is used. The RC4 algorithm was chosen because the processing time is faster than other algorithms. The RC4 algorithm uses two Substitution Boxes (S-Box), namely an array of 256 lengths containing permutations from 0 to 255, and the second S-Box containing permutations is a function of a variable length key.

Keywords: Cryptography, Archived Data, Encryption, Decryption, RC4

1. PENDAHULUAN

Seiring dengan perkembangan teknologi komputer menyebabkan terkaitnya satu komputer dengan komputer lainnya. Hal ini membuka banyak peluang dalam pengembangan aplikasi komputer tetapi juga membuat peluang adanya ancaman terhadap pengubahan dan pencurian data. Dalam hal ini, kearsipan mengubah sistem manual ke sistem komputerisasi dalam era teknologi dan informasi. Karena sering dilakukannya pengiriman atau pertukaran data, maka dilakukan proses penyandian atau enkripsi pada data yang akan dikirimkan. Sebelum dikirim ke penerima melalui media email, penerima harus memiliki aplikasi yang sama untuk mendekripsi data tersebut agar dapat dibaca dan dimengerti kembali. Hal ini dilakukan untuk melindungi data atau informasi tersebut dari segala bentuk ancaman yang tidak diinginkan. Pada Balai Desa Sidodadi terdapat data arsip seperti data penduduk sekitar atau yang lainnya yang bertujuan sebagai sumber informasi dan sumber dokumentasi. Sebagai sumber informasi yang dapat membantu mengingatkan pegawai apabila ada perubahan data sewaktu – waktu.

Informasi yang diarsipkan akan dapat terjaga kerahasiaannya maka perlu dilakukan pengamanan pada data arsip, salah satunya adalah dengan cara melakukan proses penyandian (enkripsi) terhadap informasi yang akan diarsipkan dalam bentuk berbagai file data yang sesuai dengan kebutuhan. Ada beberapa algoritma enkripsi yang telah dipublikasikan, salah satunya adalah algoritma enkripsi *Rivest Code 4 (RC4)* merupakan salah satu algoritma kunci simetris dibuat oleh RSA Data Security Inc (RSADSI) yang berbentuk stream chipper.

Dari pembahasan penelitian ini diharapkan perangkat lunak yang dirancang dapat membantu perusahaan dari segi keamanan data dan informasi, dimana dengan adanya aplikasi ini dapat diketahui cara pengamanan data arsip pada Balai Desa Sidodadi menggunakan kunci Asimetris.

2. METODOLOGI PENELITIAN

2.1 Kriptografi

Dalam teknologi informasi telah dan sedang dikembangkan cara-cara untuk menangkal berbagai bentuk serangan, seperti penyadapan dan pengubahan data yang sedang dikirimkan. Salah satu cara yang ditempuh untuk mengatasi masalah ini adalah dengan menggunakan kriptografi yang menggunakan transformasi data sehingga data yang dihasilkan tidak dapat dimengerti oleh pihak yang tidak berhak mengakses. Transformasi ini memberikan solusi pada dua macam masalah keamanan data, yaitu masalah privasi (*privacy*) dan keotentikan (*authentication*). *Privacy* mengandung arti bahwa data yang dikirimkan hanya dapat dimengerti oleh penerima yang sah atau berhak. Sedangkan keotentikan (*authentication*) mencegah pihak ketiga untuk mengirimkan data atau mengubah data yang diberikan.

2.2 Algoritma RC4

RC4 merupakan jenis aliran kode yang berarti operasi enkripsinya dilakukan per karakter 1 byte untuk sekali operasi. Algoritma kriptografi *Rivest Code* 4 (*RC4*) merupakan salah satu algoritma kunci simetris dibuat oleh *RSA* Data *Security Inc* (*RSADSI*) yang berbentuk stream chipper. Algoritma ini ditemukan pada tahun 1987 oleh Ronald Rivest dan menjadi simbol keamanan *RSA* (*Rivest Shamir Adleman*). *RC4* menggunakan panjang kunci dari 1 sampai 256 byte yang

ISSN 2745-7966 (Media Online)
Vol 2, No 2, November 2021
Hal 58-66
https://djournals.com/resolusi

digunakan untuk menginisialisasikan tabel sepanjang 256 byte. Tabel ini digunakan untuk generasi yang berikut dari pseudo random yang menggunakan XOR dengan plainteks untuk menghasilkan cipherteks. Masing-masing elemen dalam tabel saling ditukarkan minimal sekali. *RC4* merupakan merupakan salah satu jenis *stream cipher*, yaitu memproses unit atau input data pada satu saat. Dengan cara ini enkripsi atau dekripsi dapat dilaksanakan pada panjang yang variabel. Algoritma ini tidak harus menunggu sejumlah input data tertentu sebelum diproses. Metode enkripsi RC4 sangat cepat kurang lebih 10 kali lebih cepat dari DES.

Algoritma RC4 Stream Cipher untuk melakukan enkripsi-dekripsi:

1. Proses inisialisasi S-Box (Array S)
   For i = 0 to 255, S[i] = i
2. Proses inisialisasi S-Box (Array K)
   For i = 0 to 255, S[i] = i
3. Kemudian lakukan langkah pengacakan S-Box. i=0; j=0
   For i= 0 to 255{ j = (j+S [i] + [k] mod 256
   Swap S[i] dan S[j]}
4. Membuat pseudorandom byte, i= (i+1) mod 256
   J = (j+S[i]) mod 256
   Swap S[i] dan S[j]
   T=(S[i] + S[j]) mod 256
   K=S[t]

3. HASIL DAN PEMBAHASAN

3.1 Analisa Masalah

Pada Balai Desa Sidodadi, keamanan data dalam arsip pendataan penduduk maupun arsip lainnya sangat penting untuk menjamin agar data yang diarsipkan tidak jatuh ke pihak ketiga. Namun data-data tersebut memiliki berbagai macam jenis dan jumlah yang tidak sedikit, oleh karena itu Balai Desa Sidodadi membangun sebuah sistem aplikasi kearsipan yang bertujuan untuk memudahkan proses kegiatan pengarsipan dan pencarian kembali data arsip.

Untuk itu perlu dilakukan impelentasi metode-metode pengamanan data pada arsip namun tidak merubah aplikasi-aplikasi yang sudah ada. Aplikasi yang akan dirancang ini merupakan aplikasi yang sudah lama digunakan namun fungsinya belum dimaksimalkan. Oleh karena itu metode kriptografi *Rivest Code 4* ini sangat penting diterapkan agar data yang bersifat rahasia tersebut tidak mudah jatuh ke pihak ketiga.

Pada permasalahan ini, data yang akan di *enkripsi* dan *dekripsi* yaitu data berupa kode kontrak, uraian masalah kegiatan, tanggal dan nilai kontrak dan data lainnya yang ada di arsip Balai Desa Sidodadi dalam format file untuk dijadikan bukti apabila data tersebut dibutuhkan kembali. Pada proses *enkripsi* dan *dekripsi* guna membuat data atau informasi agar tidak dapat dibaca atau dimengerti oleh sembarang orang, kecuali untuk pihak yang berhak. Sedangkan media untuk melakukan pengamanan data dalam sistem komunikasi dalam jaringan komputer disebut *cryptography*.

3.1.1 Penerapan Algoritma RC4

Algoritma kriptografi *Rivest Code* 4 (*RC4*) merupakan salah satu algoritma kunci simetris dibuat oleh *RSA* Data Security Inc (*RSADSI*) yang berbentuk stream chipper. Algoritma ini ditemukan pada tahun 1987 oleh Ronald Rivest dan menjadi simbol keamanan *RSA* (merupakan singkatan dari tiganama penemu: *Rivest Shamir Adleman*). *RC4* menggunakan panjang kunci dari 1 sampai 256 byte yang digunakan untuk menginisialisasikan tabel sepanjang 256 byte.Tabel ini digunakan untuk generasi yang berikut dari pseudo random yang menggunakan XOR dengan plainteks untuk menghasilkan cipherteks. Masing-masing elemen dalam tabel saling ditukarkan minimal sekali.

1. Proses inisialisasi S-Box (Array S)
   for i = 0 to 255, S[i] = i
2. Proses inisialisasi S-Box (Array K)
   for i = 0 to 255, S[i] = i
3. Kemudian lakukan langkah pengacakan S-Box
   i=0; j=0
   for i= 0 to 255{
   j=(j+S[i]+[k] mod 256
   Swap S[i] dan S[j]}
4. Membuat pseudorandom byte
   i= (i+1) mod 256
   j= (j + S[i]) mod 256
   Swap S[i] dan S[j]
   t=(S[i] + S[j]) mod 256
   K=S[t]

Byte K di-XOR-kan dengan plainteks untuk menghasilkan cipherteks atau di-XOR-kan dengan cipherteks untuk menghasilkan plainteks.

ISSN 2745-7966 (Media Online)
Vol 2, No 2, November 2021
Hal 58-66
https://djournals.com/resolusi

Berikut adalah implementasi algoritma RC4 dengan mode 256 byte.

1. Inisialisasi S-Box dengan panjang 256 byte, dengan S[0]=0, S[1]=1, S[2]=2, S[3]=3,…..,S[255]=255 sehingga array S menjadi :

Tabel 1. Tabel Inisialisasi S-Box Dengan Panjang 256 byte

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	221
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	1221	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. Inisialisasi 11 byte kunci array Ki, Misalkan kunci terdiri dari 11 byte yaitu “PEMBANGUNAN” maka kalimat akan diubah kedalam bentuk Desimal “112 101 109 98 97 110 103 117 110 97 110”. Ulangi kunci sampai memenuhi seluruh array K sehingga array K menjadi :

Tabel 2. Tabel Inisialisasi 11 byte Kunci Array Ki

Iter-i	Key-char	Key[i]	Sbox[i]
1	P	112	0
2	E	101	1
3	M	109	2
4	B	98	3
5	A	97	4
6	N	110	5
7	G	103	6
8	U	117	7
9	N	110	8
10	A	97	9
11	N	110	10
12	P	112	11
13	E	101	12
….	....	....	....
….	....	....	....
255	E	101	254
256	M	109	255

3. Berikutnya mencampur operasi dimana akan menggunakan variabel i dan j ke index array S[i] dan K[i]. Pertama kita beri nilai inisial untuk i dan j dengan 0. Operasi Pencampuran adalah pengulangan rumusan (j + S[i]+K[i]) mod 256 yang diikuti dengan penukaran S[i] dengan S[j]. Untuk contoh ini, karena kita menggunakan array dengan panjang 256 byte maka algoritma menjadi:

   For i = 0 to 256
   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 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+112) mod 256
   = 112
   Swap S[0] dan S[112]

ISSN 2745-7966 (Media Online)
Vol 2, No 2, November 2021
Hal 58-66
https://djournals.com/resolusi

Iterasi ke-2:
i = 1, maka
j = (j + S[i] + K[i] mod 256
= (j + S[1] + K[1] mod 256
= (112+1+101) mod 256
= 214
Swap S[1] dan S[214]
Iterasi ke-3:
i = 2, maka
j = (j + S[i] + K[i] mod 256
= (j + S[2] + K[2] mod 256
= (214+2+109) mod 256
= 69
Swap S[2] dan S[69]
Iterasi ke-4:
i = 3, maka
j = (j + S[i] + K[i] mod 256
= (j + S[3] + K[3] mod 256
= (69+3+98) mod 256
= 170
Swap S[3] dan S[170]
Iterasi ke-5:
i = 4, maka
j = (j + S[i] + K[i] mod 256
= (j + S[4] + K[4] mod 256
= (170+4+97) mod 256
= 15
Swap S[4] dan S[15]
Iterasi ke-6:
i = 5, maka
j = (j + S[i] + K[i] mod 256
= (j + S[5] + K[5] mod 256
= (15+5+110) mod 256
= 130
Swap S[5] dan S[130]
….
….
….
Iterasi ke-254:
i = 253, maka
j = (j + S[i] + K[i] mod 256
= (j + S[253] + K[253] mod 256
= (57+9+112) mod 256
= 178
Swap S[253] dan S[178]
Iterasi ke-255:
i = 254, maka
j = (j + S[i] + K[i] mod 256
= (j + S[254] + K[254] mod 256
= (178+4+101) mod 256
= 27
Swap S[254] dan S[27]
Iterasi ke-256:
i = 255, maka
j = (j + S[i] + K[i] mod 256
= (j + S[255] + K[255] mod 256
= (27+38+109) mod 256
= 174
Swap S[255] dan S[174]

Hasil yang didapat setelah melakukan seluruh iterasi dari 0 s/d 255 kali iterasi dan melakukan pertukaran *s-box (swap)* adalah sebagai berikut:

ISSN 2745-7966 (Media Online)
Vol 2, No 2, November 2021
Hal 58-66
https://djournals.com/resolusi

Tabel 3. Tabel Hasil Pertukaran *s-box (swap)*

149	165	181	161	148	103	46	99	225	55	195	150	175	41	73	61
124	224	43	137	44	60	122	246	206	226	20	4	82	254	3	109
143	101	176	216	215	114	241	127	52	151	173	192	198	39	132	21
71	2	222	59	45	190	16	75	177	189	207	7	47	32	76	230
217	68	83	144	163	212	201	30	223	157	85	183	167	81	96	146
156	200	26	232	62	10	121	113	220	154	97	155	205	89	134	51
213	242	125	95	211	162	185	130	131	119	139	133	100	120	0	80
78	227	93	239	8	203	12	247	104	22	244	196	65	179	105	152
234	86	64	92	115	6	245	171	40	252	228	106	50	49	35	199
107	110	153	53	15	108	84	140	233	251	66	166	126	14	221	172
237	170	37	194	236	214	117	178	188	57	174	159	36	231	38	1477
229	48	9	202	34	164	250	191	136	29	168	197	160	19	69	253
187	72	235	28	145	25	87	169	31	24	218	27	70	88	18	141
116	186	193	158	204	1	248	63	249	135	111	182	219	56	90	209
238	112	23	184	138	128	98	243	33	54	42	74	118	210	129	240
94	255	67	180	17	13	123	5	58	77	208	79	11	102	142	91

4. Berikutnya adalah proses enkripsi yaitu meng-XOR-kan *pseudo random byte* dengan *plainteks*, misalnya *plainteks* "BDS SUMUT", ubah karakter menjadi bilangan biner.

Tabel 4. Tabel Biner *Plainteks*

Karakter	Desimal	Hexa	Biner
B	66	42	01000010
D	68	44	01000100
S	83	53	01010011
[spasi]	32	20	00100000
S	83	53	01010011
U	85	55	01010101
M	77	4d	01001101
U	85	55	01010101
T	84	54	01010100

Berikut iterasi 1: Inisialisasi i dan j dengan i = 0; j = 0;

i = (i+1) mod 256
= (0+1) mod 256 = 1
j = (j+S[i]) mod 256
= (j+S[1]) mod 256
= (0+165) mod 256
= 165
Swap S[1] dan S[165]
Swap S[165] dan S[214]
t = (S[i] + S[j] mod 256
= (S[1] + S[165] mod 256
= (214+165) mod 256
= 123
K = S[t] =S[123] = 196 = 11000100

Tabel 5. Enkripsi Plainteks "B"

Plainteks	B 66 42 01000010
Key (K)	11000100
XOR	10000110
Chiperteks	134 86 ┼

Iterasi ke 2:

i = (i+1) mod 256
= (1+1) mod 256

= 2
j = (j+S[i]) mod 256
= (j+S[2]) mod 256
= (165+181) mod 256
= 90
Swap S[2] dan S[90]
Swap S[181] dan S[97]
t = (S[i] + S[j] mod 256
= (S[2] + S[90] mod 256
= (97+181) mod 256
= 22
K = S[t] =S[22] = 122 = 01111010

Tabel 6. Enkripsi Plainteks "D"

	D
Plainteks	68
	44
	01000100
Key (K)	01111010
XOR	00111110
	62
Chiperteks	3E
	>

Iterasi ke 3:
i = (i+1) mod 256
= (2+1) mod 256
= 3
j = (j+S[i]) mod 256
= (j+S[3]) mod 256
= (90+161) mod 256
= 251
Swap S[3] dan S[251]
Swap S[161] dan S[79]
t = (S[i] + S[j] mod 256
= (S[3] + S[251] mod 256
= (79+161) mod 256
= 240
K = S[t] =S[240] = 94 = 01011110

Tabel 7. Enkripsi Plainteks "S"

	S
Plainteks	83
	53
	01010011
Key (K)	01011110
XOR	00001101
	13
Chiperteks	d

….
….
….
….

Iterasi ke 8:
i = (i+1) mod 256
= (7+1) mod 256
= 8
j = (j+S[i]) mod 256
= (j+S[8]) mod 256
= (135+225) mod 256

= 104
Swap S[8] dan S[104]
Swap S[225] dan S[131]
t = (S[i] + S[j] mod 256
= (S[8] + S[104] mod 256
= (131+225) mod 256
= 100
K = S[t] =S[100] = 211 = 11010011

Tabel 8. Enkripsi Plainteks "U"

Plainteks	U 85 55 01010101
Key (K)	11010011
XOR	10000110
Chiperteks	134 86 ┼

Iterasi ke 9:
i = (i+1) mod 256
= (8+1) mod 256
= 9
j = (j+S[i]) mod 256
= (j+S[9]) mod 256
= (104+55) mod 256
= 159
Swap S[9] dan S[159]
Swap S[55] dan S[172]
t = (S[i] + S[j] mod 256
= (S[9] + S[159] mod 256
= (172+55) mod 256
= 227
K = S[t] =S[227] = 184 = 10111000

Tabel 9. Enkripsi Plainteks "T"

Plainteks	T 84 54 01010100
Key (K)	10111000
XOR	11101100
Chiperteks	236 ec ì

5. Berikutnya adalah proses dekripsi yaitu meng-XOR-kan *pseudo random byte* dengan *Chiperteks*, dan *Chiperteks* nya adalah "**86, 3E, d, bb, 44, 2e, 4, 86, ec**". *Chiperteks* terdiri dari 9 karakter maka terjadi 9 iterasi. Sebelum di-iterasi ubah karakter menjadi bentuk bilangan biner.

Tabel 10. Tabel Biner *Chiperteks*

Karakter	Decimal	Hexa	Biner
┼	134	86	10000110
>	62	3E	00111110
	13	d	00001101
»	187	bb	10111011
D	68	44	01000100
.	46	2e	00101110
┘	4	4	00000100
.	46	2e	00101110
Ì	236	ec	11101100

ISSN 2745-7966 (Media Online)
Vol 2, No 2, November 2021
Hal 58-66
https://djournals.com/resolusi

Data dalam bentuk Chiperteks sehingga setelah disimpan dapat kembali diubah menjadi plainteks dengan meng-XOR-kan dengan kunci yang sama.
Berikut Iterasi 1:

Tabel 11. Dekripsi *Chiperteks* "86"

Chiperteks	┼ 134 86
Key (K)	10000110 11000100
XOR	01000010
Plainteks	66 42 B

Berikut Iterasi 2:

Tabel 12. Dekripsi *Chiperteks* "3E"

Chiperteks	> 3E 62
Key (K)	00111110 01111010
XOR	01000100
Plainteks	44 68 D

Berikut Iterasi 3:

Tabel 13. Dekripsi *Chiperteks* "d"

Chiperteks	13 d 00001101
Key (K)	01011110
XOR	01010011
Plainteks	83 53 S

Berikut Iterasi 4:

Tabel 14. Dekripsi *Chiperteks* "bb"

Chiperteks	» 187 bb
Key (K)	10111011 10011011
XOR	00100000
Plainteks	32 20 [spasi]

….
….
….
….

ISSN 2745-7966 (Media Online)
Vol 2, No 2, November 2021
Hal 58-66
https://djournals.com/resolusi

Berikut Iterasi 7:

Tabel 15. Dekripsi *Chiperteks* "4"

Chiperteks	┘ 4 4
Key (K)	00000100 01001001
XOR	01001101
Plainteks	77 4d M

Berikut Iterasi 8:

Tabel 16. Dekripsi *Chiperteks* "86"

Chiperteks	┼ 134 86
Key (K)	10000110 11010011
XOR	01010101
Plainteks	85 55 U

Berikut Iterasi 9:

Tabel 17. Dekripsi *Chiperteks* "ec"

Chiperteks	Ì 236 ec
Key (K)	11101100 10111000
XOR	01010100
Plainteks	84 54 T

4. KESIMPULAN

Adapun kesimpulan yang dapat diambil dari penelitian yang telah dilakukan Penerapan kriptografi RC4 dalam mengamankan data arsip untuk mengubah isi pesan menjadi bentuk simbol-simbol yang tidak dapat dimengerti maknanya dengan menggunakan operasi *XOR* antara *plainteks* dengan kunci yang dimasukkan, hasil dari enkripsi ini akan disisipkan atau disembunyikan sehingga keamanan data arsip lebih terjaga kerahasiaan nya. Metode RC4 menggunakan panjang kunci dari 1 sampai 256 *byte* yang digunakan untuk menginisialisasi tabel dengan panjang 256 *byte*.

REFERENCES

[1] D. Wirdasari, Prinsip Kerja Kriptografi dalam Mengamankan Informasi, vol. 5, no. 2, 2008.
[2] Janner Simarmata, (2006). Pengamanan Sistem Komputer.
[3] Karina Novita Suryani, (2009). Algoritma RC4 Sebagai Metode Enkripsi. Makalah IF2091 STRUKTUR DISKRIT.
[4] Sulindawati, & Fathoni, M. (2010). Pengantar Analisa Perancangan Sistem. Jurnal SAINTIKOM Vol.9 No.2.
[5] Rossa A.S, M.Shalahuddin, (2014). Rekayasa Perangkat Lunak dan Berorientasi Objek. Bandung Informatika.
[6] Deliarnoor Nandang Alamsyah, (2014). Aspek Hukum Dalam Kearsipan.
[7] Arianto, (2016). Pengertian File Extension disetiap Nama File.