BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Berikut ini adalah beberapa penelitian terkait yang dapat diambil dari beberapa jurnal penelitian yang berkaitan dengan penelitian ini :

Salah satunya yaitu penelitian yang dilakukan oleh Arina Listyarini Dwiastuti, dengan judul “Menebak Password RAR Menggunakan RAR Password Recovery Magic dengan Algoritma Brute Force.”, yang berkisimpulan:

1. Algoritma brute force memang tidak secerdas dan tidak sebagus algoritma canggih yang ada.
2. algoritma brute force sangatlah membantu untuk memecahkan persoalan yang sepertinya tidak mungkin, seperti menebak password RAR yang jumlah karakternya antara 1 sampai 127 dengan jenis karakter yang berbagai macam (mulai dari huruf, angka, sampai simbol yang jumlahnya sangat banyak).
3. algoritma brute force mencoba satu-satu kemungkinan yang jumlahnya sangat banyak, maka waktu yang dibutuhkan sangatlah lama sehingga dibutuhkan kesabaran.
4. jika kita membutuhkan waktu yang singkat dan algoritma yang sangat bagus, kita tidak dapat menjadikan algoritma brute force sebagai alternatif.

Penelitian yang dilakukan oleh Rauf Riyantono Wahyu Pramusinto (2018), dengan judul "Aplikasi pengamanan surat elektronik (*Email*) menggunakan *Algoritma Advanced Encryption Standart* 128 (*AES*-128) dan *Rivest Cipher Code* 4 (*RC4*) berbasis *web*", yang berkesimpulan:

1. Dengan adanya aplikasi kriptografi menggunakan metode *Advanced Encryption Standard*(*AES*) dan *Rivest Cipher* 4 (*RC4*) ini dapat mengamankan dokumen penting atau informasi yang ada di Klinik Cahaya Madani agar dapat lebih aman kerahasiaannya dari orang-orang yang tidak bertanggung jawab.
2. Teknik kriptografi dengan metode *Advanced Encryption Standard* (*AES*) dan *Rivest Cipher* 4 (*RC4*) pada aplikasi mengamanan dokumen data obat dan data lapora narkotika, psikotropika yang akan dikirim dengan memanfaatkan media surat elektronik (*e-mail*) berbasis *web* berhasil diimplementasikan pada Klinik Cahaya Madani
3. Aplikasi ini juga dapat mengembalikan data yang sudah diamankan kriptografi menggunakan metode *Advanced Encryption Standard* (*AES*) dan *Rivest Cipher* 4 (*RC4*) menjadi data yang orisinil tanpa mengalami perubahan sedikitpun.
4. Waktu untuk mengenkripsi pesan berbanding lurus dengan ukuran *teks* dan ukuran *file* yang akan dienkripsi, semakin kecil ukuran teks dan *file* nya maka semakin cepat waktu pengenkripsiannya.

Penelitian yang di lakukan oleh Faizal Zuli1, Ari Irawan (2016), dengan judul "Jurnal Format Implementasi Kriptografi Dengan Algoritma *Blowfish* dan *Riverst Shamir Adleman* (*RSA*) Untuk Proteksi File" yang berkesimpulan :

1. Enkripsi dengan algoritma *blowfish* terdiri dari dua bagian, yaitu ekspansi kunci yang berfungsi merubah kunci (minimum 32-bit, maksimum 448-bit) menjadi beberapa array subkunci (*subkey*) dengan total 4168 byte(18x32-bit untuk *P-array* dan 4x256x32-bit untuk *S-box* sehingga totalnya 33344 bit atau 4168 *byte*) dan enkripsi data.
2. Enkripsi dengan algoritma *RSA* memiliki dua buah kunci yang berbeda, yaitu kunci *public* dan kunci *private*. Prinsip kerja algoritma *RSA* menggunakan operasi pemangkatan dan operasi mod (*modulus*) yang menghasilkan nilai yang relatif acak.
3. Ukuran *file* sebelum dan sesudah dienkripsi dengan algoritma *Blowfish* dan *RSA* terjadi perubahan karena penambahan ukuran *file* sesuai dengan algoritma yang digunakan.

II.2. Studi Literatur

II.2.1. Algoritma RSA

Algoritma *RSA* merupakan salah satu algoritma public key yang populer dipakai dan bahkan masih dipakai hingga saat ini. Kekuatan algoritma ini terletak pada proses *eksponensia*l, dan pemfaktoran bilangan menjadi 2 bilangan prima yang hingga kini perlu waktu yang lama untuk melakukan pemfaktorannya.

Algoritma ini dinamakan sesuai dengan nama penemunya, *Ron Rivest, Adi Shamir* dan *Adleman* (*Rivest-Shamir-Adleman*) yang dipublikasikan pada tahun 1977 di *MIT*, menjawab tantangan yang diberikan algoritma pertukaran kunci *Diffie Hellman.*

Skema *RSA* sendiri mengadopsi dari skema *block cipher*, dimana sebelum dilakukan enkripsi, plainteks yang ada dibagi – bagi menjadi blok – blok dengan panjang yang sama, dimana *plainteks* dan *cipherteksnya* berupa *integer* (bilangan bulat) antara 1 hingga n, dimana n berukuran biasanya sebesar 1024 bit, dan panjang bloknya sendiri berukuran lebih kecil atau sama dengan log(n) +1 dengan basis 2.

Fungsi enkripsi dan dekripsinya dijabarkan dalam fungsi berikut :

$C = M^e$ *mod* n (fungsi enkripsi)

$M = C^d$ *mod* n (fungsi dekripsi)

C = *Cipherteks*

M = *Message* / *Plainteks*

e = kunci publik

d= kunci privat

n = modulo pembagi(akan dijelaskan lebih lanjut)

Kedua pihak harus mengetahui nilai e dan nilai n ini, dan salah satu pihak harus memilki d untuk melakukan dekripsi terhadap hasil enkripsi dengan menggunakan *public key* e. Penggunaan algoritma ini harus memenuhi kriteria berikut :

1. Memungkinkan untuk mencari nilai e, d, n sedemikian rupa sehingga Med mod n = M untuk semua M < n.
2. Relatif mudah untuk menghitung nilai Me mod n dan Cd mod n untuk semua nilai M < n.
3. Tidak memungkinkan mencari nilai d jika diberikan nilai n dan e.

Syarat nilai e dan d ini, gcd(d,e)=1

sebelum memulai penggunaan *RSA* ini, terlebih dahulu kita harus memiliki bahan – bahan dasar sebagai berikut :

1. p, q = 2 bilangan prima yang dirahasiakan

2. n, dari hasil p.q

3. e, dengan ketentuan gcd (Φ(n), e) =1

4. d, e^{-1} (mod Φ(n))

Saya akan berikan satu contoh :

1. Pilih 2 bilangan prima, misalnya p = 17 dan q = 11.
2. Hitung n = pq = 17 × 11 = 187.
3. Hitung Φ(n) = (p – 1)(q – 1) = 16 × 10 = 160.
4. Pilih nilai e sedemikian sehingga relatif prima terhadap Φ(n) = 160 dan kurang dari Φ(n); kita pilih e = 7.
5. Hitung d sedemikian sehingga de ≡ 1 (mod 160) dan d < 160.Nilai yang didapatkan d = 23,karena 23 × 7 = 161 = (1 × 160) + 1; d dapat dihitung dengan Extended Euclidean Algorithm.

Nah, nilai e dan d inilah yang kita sebut sebagai *Public Key* (e) dan *Private Key* (d). Pasangan Kunci Publiknya ={7,187} dan Kunci Privatnya = {23, 187}

Sekarang kita aplikasikan dalam proses enkripsi.

Misalnya kita punya M 88.

Untuk proses enkripsi, kita akan menghitung $C = 88^7 \bmod 187$.

$= 88^7 \bmod 187$.

=894,432 mod 187

=11

Nah, kita mendapatkan nilai C =11.

Selanjutnya, nilai C ini dikirimkan kepada penerima untuk didekripsi dengan kunci privat miliknya.

$M = C^d \bmod n$

$= 11^{23} \bmod 187$

=79,720,245 mod187

= 88

(Sumber : nurezkapahlevi : Tahun 2012)

II.2.2. Algoritma AES

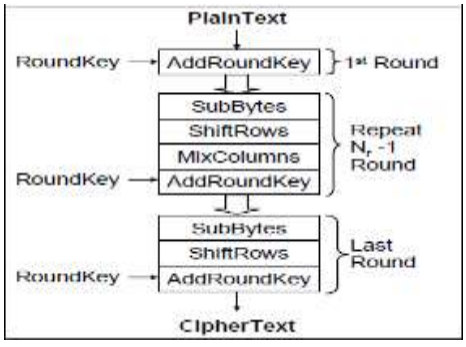
AES merupakan standar algoritma kriptografi terbaru yang dipublikasikan oleh *NIST* (*National Institute of Standard and Technology*) sebagai pengganti algoritma *DES* (*Data Encryption Standard*) yang sudah berakhir masa penggunaannya. Algoritma *AES* adalah algoritma kriptografi yang dapat mengenkripsi dan mendekripsi data dengan panjang kunci 128 bit, 192 bit, dan 256 bit. Pada algoritma *AES* (Yuniati dkk, 2009) perbedaan panjang kunci akan mempengaruhi jumlah *round* yang akan diimplementasikan pada algoritma *AES*.

	Jumlah Key (Nk)	Ukuran Block (Nb)	Jumlah Putaran (Nr)
AES – 128	4	4	10
AES – 192	6	4	12
AES – 256	8	4	14

Gambar II.1 Tabel Jumlah proses berdasarkan bit blok dan kunci

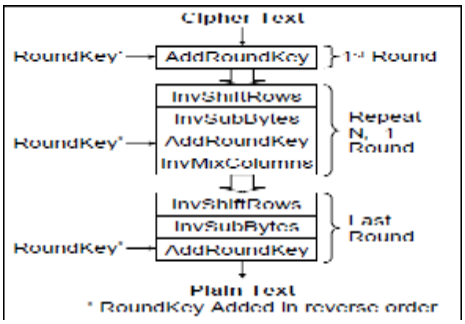
(Sumber : Yuniati dkk, 2009)

Pada table diatas memperlihatkan jumlah *round* / putaran (Nr) yang harus diimplementasikan pada masing-masing panjang kunci. Dibawah ini adalah diagram proses *enkripsi* dan *dekripsi* dari algoritma *AES*.



Gambar II.2 Diagram Proses Enkripsi

(Sumber : Kristoforus, Aditya, 2012)



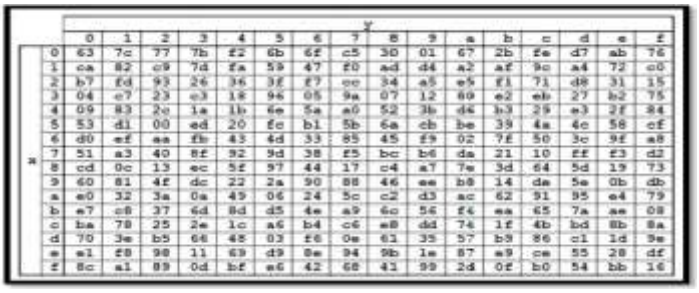
Gambar II.3 Diagram Proses Dekripsi

(Sumber : Kristoforus, Aditya, 2012)

Proses enkripsi algoritma *AES* terdiri dari 4 jenis transformasi *bytes*, yaitu *SubBytes, ShiftRows, Mixcolumns,* dan *AddRoundKey.*

a. Sub bytes

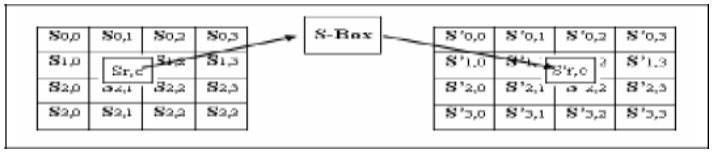
Proses *SubBytes* adalah mengganti setiap *byte state* dengan *byte* pada sebuah tabel yang dinamakan tabel *S-Box.* Sebuah tabel *S-Box* terdiri dari 16x16 baris dan kolom dengan masing-masing berukuran 1 *byte.*



x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Gambar II.4 *S-box.*

(Sumber : Yuniati dkk, 2009)

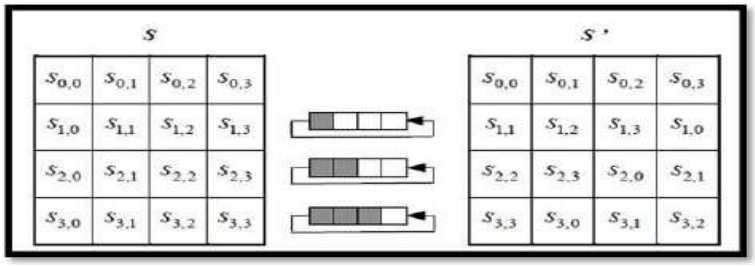


Gambar II.5 Proses *sub bytes*

(Sumber : Yuniati dkk, 2009)

b. Shift Rows

Shift rows adalah proses pergeseran bit, bit paling kiri akan dipindahkan menjadi bit paling kanan (rotasi bit).



Gambar II.6 Proses *shift row*

(Sumber : Yuniati dkk, 2009)

c. Mix Columns

MixColumns mengoperasikan setiap elemen yang berada dalam satu kolom pada state. lebih jelasnya bisa dilihat pada perkalian matriks dibawah ini :

$$\begin{bmatrix} S'_{0,C} \\ S'_{1,C} \\ S'_{2,C} \\ S'_{3,C} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,C} \\ S_{1,C} \\ S_{2,C} \\ S_{3,C} \end{bmatrix}$$

Gambar II.7 Proses perkalian matriks

(Sumber : Yuniati dkk, 2009)

Hasil dari perkalian matriks diatas dapat dilihat dibawah ini :

$$s'_{0,c} = (\{02\} \bullet s_{o,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$
$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$
$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$
$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c})$$

Gambar II.8 Hasil perkalian matriks

(Sumber : Yuniati dkk, 2009)

Berikut adalah contoh perhitungan *MixColumns* :

Input = D4 BF 5D 30

Output (0) = (D4*2) XOR (BF*3) XOR (5D*1) XOR (30*1)

= E(L(D4)+L(02)) XOR E(L(BF)+L(03)) XOR 5D XOR 30

= E(41+19) XOR E(9D+1) XOR 5D XOR 30

= E(5A) XOR E(9E) XOR 5D XOR 30

= B3 XOR DA XOR 5D XOR 30

= 04

Output (1) = (D4*1) XOR (BF*2) XOR (5D*3) XOR (30*1)

= D4 XOR E(L(BF)+L(02)) XOR E(L(5D)+L(03)) XOR 30

= D4 XOR E(9D+19) XOR E(88+01) XOR 30

= D4 XOR E(B6) XOR E(89) XOR 30

= D4 XOR 65 XOR E7 XOR 30

= 66

Output (2) = (D4*1) XOR (BF*1) XOR (5D*2) XOR (30*3)

= D4 XOR BF XOR E(L(5D)+L(02)) XOR E(L(30)+L(03))

= D4 XOR BF XOR E(88+19) XOR E(65+01)

= D4 XOR BF XOR E(A1) XOR E(66)

= D4 XOR BF XOR BA XOR 5D

= 81

Output (3) = (D4*3) XOR (BF*1) XOR (5D*1) XOR (30*2)

= E(L(D4)+L(03)) XOR BF XOR 5D XOR E(L(30)+L(02))

= E(41+01) XOR BF XOR 5D XOR E(65+19)

= E(42) XOR BF XOR (5D) XOR (E(7E)

= 67 XOR BF XOR 5D XOR 60

= E5

Hasil dari *MixColumns* dicari dengan menggunakan tabel *Galois Field Multiplication*. Cara yang dilakukan adalah mirip dengan mencari nilai *SubByte*, tetapi tabel yang digunakan bukan tabel *S-Box*, melainkan *E-Table* dan *L-Table* dari *Galois Field Multiplication*.

E Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	01	03	05	0F	11	33	55	FF	1A	2E	72	96	A1	F8	13	35
1	5F	E1	38	48	D8	73	95	A4	F7	02	06	0A	1E	22	66	AA
2	E5	34	5C	E4	37	59	EB	26	6A	BE	D9	70	90	AB	E6	31
3	53	F5	04	0C	14	3C	44	CC	4F	D1	68	B8	D3	6E	B2	CD
4	4C	D4	67	A9	E0	3B	4D	D7	62	A6	F1	08	18	28	78	88
5	83	9E	B9	D0	6B	BD	DC	7F	81	98	B3	CE	49	DB	76	9A
6	B5	C4	57	F9	10	30	50	F0	0B	1D	27	69	BB	D6	61	A3
7	FE	19	2B	7D	87	92	AD	EC	2F	71	93	AE	E9	20	60	A0
8	FB	16	3A	4E	D2	6D	B7	C2	5D	E7	32	56	FA	15	3F	41
9	C3	5E	E2	3D	47	C9	40	C0	5B	ED	2C	74	9C	BF	DA	75
A	9F	BA	D5	64	AC	EF	2A	7E	82	9D	BC	DF	7A	8E	89	80
B	9B	B6	C1	58	E8	23	65	AF	EA	25	6F	B1	C8	43	C5	54
C	FC	1F	21	63	A5	F4	07	09	1B	2D	77	99	B0	CB	46	CA
D	45	CF	4A	DE	79	8B	86	91	A8	E3	3E	42	C6	51	F3	0E
E	12	36	5A	EE	29	7B	8D	8C	8F	8A	85	94	A7	F2	0D	17
F	39	4B	DD	7C	84	97	A2	FD	1C	24	6C	B4	C7	52	F6	01

Gambar II.9 *E-Table* dari *Galois Field Multiplication*

(Sumber : Barent, 2014)

L Table

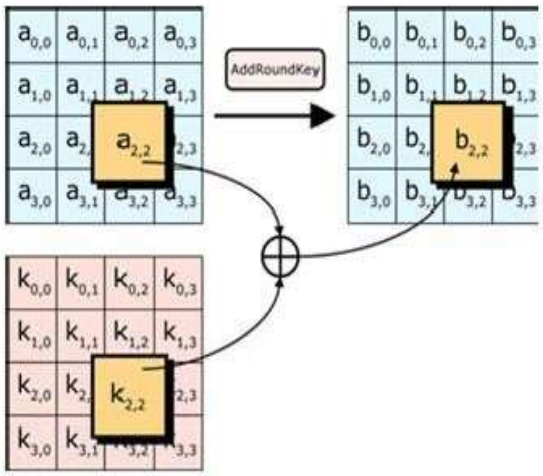
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		00	19	01	32	02	1A	C6	4B	C7	1B	68	33	EE	DF	03
1	64	04	E0	0E	34	8D	81	EF	4C	71	08	C8	F8	69	1C	C1
2	7D	C2	1D	B5	F9	B9	27	6A	4D	E4	A6	72	9A	C9	09	78
3	65	2F	8A	05	21	0F	E1	24	12	F0	82	45	35	93	DA	8E
4	96	8F	DB	BD	36	D0	CE	94	13	5C	D2	F1	40	46	83	38
5	66	DD	FD	30	BF	06	8B	62	B3	25	E2	98	22	88	91	10
6	7E	6E	48	C3	A3	B6	1E	42	3A	6B	28	54	FA	85	3D	BA
7	2B	79	0A	15	9B	9F	5E	CA	4E	D4	AC	E5	F3	73	A7	57
8	AF	58	A8	50	F4	EA	D6	74	4F	AE	E9	D5	E7	E6	AD	E8
9	2C	D7	75	7A	EB	16	0B	F5	59	CB	5F	B0	9C	A9	51	A0
A	7F	0C	F6	6F	17	C4	49	EC	D8	43	1F	2D	A4	76	7B	B7
B	CC	BB	3E	5A	FB	60	B1	86	3B	52	A1	6C	AA	55	29	9D
C	97	B2	87	90	61	BE	DC	FC	BC	95	CF	CD	37	3F	5B	D1
D	53	39	84	3C	41	A2	6D	47	14	2A	9E	5D	56	F2	D3	AB
E	44	11	92	D9	23	20	2E	89	B4	7C	B8	26	77	99	E3	A5
F	67	4A	ED	DE	C5	31	FE	18	0D	63	8C	80	C0	F7	70	07

Gambar II.10 *L-Table* dari *Galois Field Multiplication*

(Sumber : Barent, 2014)

d. Add Round Key

Pada proses ini *subkey* digabungkan dengan *state*. Proses penggabungan ini menggunakan operasi *XOR* untuk setiap *byte* dari s*ubkey* dengan *byte* yang bersangkutan dari *state*. Untuk setiap tahap, *subkey* dibangkitkan dari kunci utama dengan menggunakan proses *key schedule*. Setiap *subkey* berukuran sama dengan *state* yang bersangkutan.

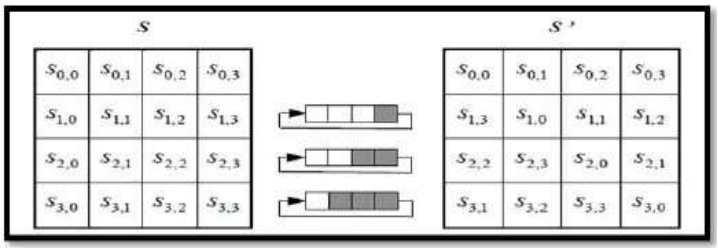


Gambar II.11 Proses *Add Round*
(Sumber : Surian, 2009)

Proses *dekripsi* algoritma *AES* berbeda dengan proses *enkripsi*, transformasi *cipher* dapat dibalikkan dan diimplementasikan dalam arah yang berlawanan. Transformasi *byte* yang digunakan pada *invers cipher* adalah *InvShiftRows*, *InvSubBytes*, *InvMixColumns*, dan *AddRoundKey*.

a. InvShiftRows

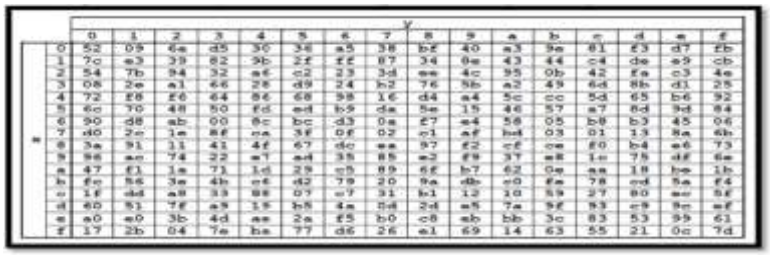
InvShiftRows adalah berkebalikan dengan transformasi *shiftrows* pada proses enkripsi. Pada transformasi *InvShiftRows*, dilakukan pergeseran bit ke kanan.



Gambar II.12 Transformasi *InvShiftRows*
(Sumber : Yuniati dkk, 2009)

b. InvSubBytes

InvSubBytes adalah berkebalikan dengan transformasi *SubBytes* pada proses *enkripsi*. Pada *InvSubBytes*, tiap elemen pada *state* dipetakan dengan menggunakan table *Inverse S-Box.*



Gambar II.13 *Inverse S-Box*
(Sumber : Yuniati dkk, 2009)

c. InvMixColumns

Setiap kolom dalam *state* dikalikan dengan matriks perkalian dalam *AES*. Perkalian dalam matriks dapat dilihat dibawah ini :

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Gambar II.14 Proses Perkalian Matriks
(Sumber : Yuniati dkk, 2009)

Hasil dari perkalian dalam matrik adalah :

$$s'_{0,c} = (\{0E\} \bullet s_{o,c}) \oplus (\{0B\} \bullet s_{1,c}) \oplus (\{0D\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c})$$
$$s'_{1,c} = (\{09\} \bullet s_{o,c}) \oplus (\{0E\} \bullet s_{1,c}) \oplus (\{0B\} \bullet s_{2,c}) \oplus (\{0D\} \bullet s_{3,c})$$
$$s'_{2,c} = (\{0D\} \bullet s_{o,c}) \oplus (\{09\} \bullet s_{1,c}) \oplus (\{0E\} \bullet s_{2,c}) \oplus (\{0B\} \bullet s_{3,c})$$
$$s'_{3,c} = (\{0B\} \bullet s_{o,c}) \oplus (\{0D\} \bullet s_{1,c}) \oplus (\{09\} \bullet s_{2,c}) \oplus (\{0E\} \bullet s_{3,c})$$

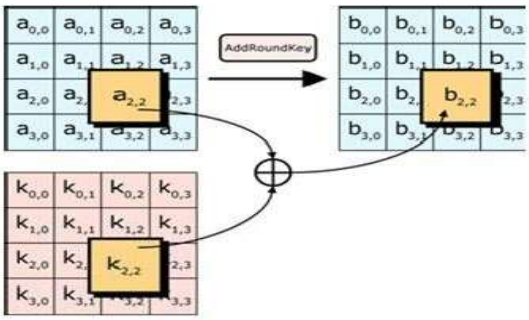
Gambar II.15 Hasil Perkalian Matriks
(Sumber : Yuniati dkk, 2009)

Untuk mencari hasil dari *Invers Mixcolumn* hampir sama dengan perhitungan pada *Mixcolumn* dengan menggunakan tabel E dan tabel L.

d. Add Round Key

Pada proses ini *subkey* digabungkan dengan state. Proses penggabungan ini menggunakan operasi *XOR* untuk setiap *byte* dari *subkey* dengan *byte* yang bersangkutan dari *state*. Untuk setiap tahap, *subkey* dibangkitkan dari kunci

utama dengan menggunakan proses *key schedule*. Setiap *subkey* berukuran sama dengan *state* yang bersangkutan.



Gambar II.16 Proses *Add Round Key*

(Sumber : Surian, 2009)

II.2.3. Microsoft Visual Studio 2017

Microsoft Visual Studio 2017 merupakan salah satu bahasa pemrograman yang andal dan banyak digunakan oleh pengembang untuk membangun berbagai macam aplikasi *Windows*.*Visual Basic* 2017 adalah versi terbaru yang telah diluncurkan oleh *Microsoft* bersama C#, Visual C++, dan *Visual Web Developer* dalam satu paket *Visual Studio 2017*. *Visual Basic* 2017 merupakan aplikasi pemrograman yang menggunakan teknologi. *NET Framework*. Teknologi .*Net Framework* merupakan komponen *Windows* yang terintegritasi serta mendukung pembuatan, penggunaan aplikasi, dan halaman *web*. Teknologi. *NET Framework* mempunyai 2 komponen utama, yaitu *CLR* (*Common Language Runtime*) dan *Class Library*. *CLR* digunakan untuk menjalankan aplikasi yang berbasis.*NET*, sedangkan *Library* adalah kelas pustaka atau perintah yang digunakan untuk membangun aplikasi.

II.2.4. *C-Sharp* (C#)

C# atau yang dibaca *C-sharp* adalah bahasa pemrograman sederhana yang digunakan untuk tujuan umum, dalam artian bahasa pemrograman ini dapat digunakan untuk berbagai fungsi misalnya untuk pemrograman server-side pada website, membangun aplikasi desktop ataupun mobile, pemrograman game dan sebagainya. Selain itu *C#* juga bahasa pemrograman yang berorientasi objek.

II.2.5. *RAR* dan *ZIP*

a. *Rar*

Format *RAR* menciptakan kompres/tekanan yang mantap/padat jika dibandingkan dengan format *ZIP*. Kelebihan *RAR* yang lain adalah mendukung multi volume. Jika Anda harus menciptakan suatu arsip multi-volume, maka menggunakan *RAR* adalah pilihahnnya. Format *RAR* juga mempunyai beberapa hal penting lain yang tidak dijumpai dalam format *ZIP*. Misalnya, kemampuan *RAR* untuk merekontruksi arsip yang rusak, perlindungan/kunci arsip, dan melindungi arsip penting dari modifikasi yang tidak disengaja. Format *RAR* bisa menangani file yang bisa dikatakan ukuran tak terbatas (sampai kepada 8.589.934.591 GB). Sedangkan ukuran maksimum dari satu file ZIP hanya 2 GB. Sistem file yang mendukung file berukuran lebih dari 4 GB adalah NTFS atau yang lebih terbaru dari itu.

b. *Zip*

Format *ZIP* adalah yang paling populer. Sebagai contoh, kebanyakan arsip di Internet adalah arsip berformat *ZIP*. Jika Anda akan mengirimkan suatu arsip RAR ke seseorang, mungkin saja teman Anda tidak mempunyai *WinRAR* untuk

mengekstrak muatan yang ada dalam arsip *RAR*, hal ini bisa saja membuat Anda untuk memilih format *ZIP*. Pada sisi lain, Anda dapat mengirimkan suatu arsip yang berjenis self-extracting (dapat mengekstrak dirinya sendiri). Arsip self-extracting memiliki sedikit ukuran yang lebih besar, tetapi dapat mengekstrak dirinya tanpa bantuan program apa pun. Keuntungan *ZIP* yang lain adalah kecepatan. Proses penciptaan arsip *ZIP* lebih cepat dari arsip *RAR.*

II.2.6. Unified Modeling Language (UML)

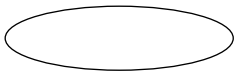
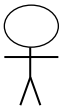
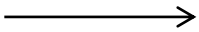
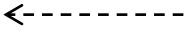
Unified Modelling Language (*UML*) adalah sebuah "bahasa" yang telah menjadi standar dalam industry untuk visualisasi, merancang dan mendokumentasi sistem piranti lunak. *UML* menawarkan sebuah standar untuk merancang model sebuah sistem.

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan *UML* adalah sebagai berikut :

1. *Use Case* Diagram

Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram dapat dilihat pada tabel II.1.

Tabel II.1. Simbol *Use Case*

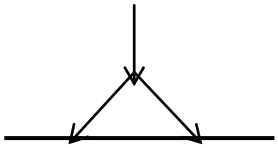
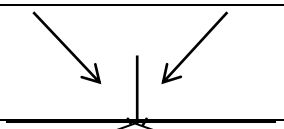
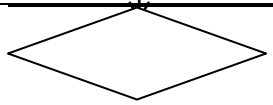
Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, dan dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki *control* terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	Include, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Urva dan Siregar, 2015 : 94)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel II.2.

Tabel II.2. Simbol *Activity Diagram*

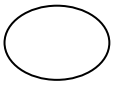
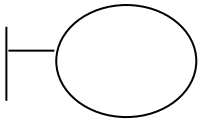
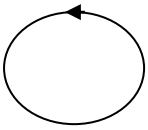
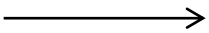
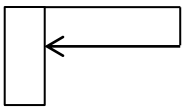
Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false*.
	*Swimlane*, untuk menunjukkan siapa melakukan apa.

(Sumber : Urva dan Siregar, 2015 : 94)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram* dapat dilihat pada tabel II.3.

Tabel II.3. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation*, *activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Urva dan Siregar, 2015 : 95)

4. *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. *Class diagram* juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti yang dapat dilihat pada tabel II.4.

Tabel II.4. *Multiplicity Class Diagram*

Multiplicity	**Penjelasan**
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Urva dan Siregar, 2015 : 95)