BAB I

PENDAHULUAN

1.1 Latar Belakang

Pengarsipan merupakan hal yang biasa dilakukan apabila memiliki data dan fakta yang dianggap penting contoh umum benda yang sering diarsipkan adalah surat, dengan ada nya arsip memudahkan dalam menyimpan data data tersebut. Pengolahan arsip di Yayasan Universitas Islam Sumatera Utara masih menggunakan metode konvensional. Sehingga dalam menyimpan data surat masih berupa *filling* cabinet atau penyimpanan *file* di dalam lemari kabinet. Selain memakan *space* ruangan yang besar secara fisik, juga mempersulit apabila data yang ingin diketahui harus dicari terlebih dahulu di lemari kabinet. Belum lagi permasalahan lain seperti keamanan selain rentan terhadap api, lapuk, juga rentan terhadap rayap. Oleh karena itu dibutuhkan solusi untuk mengatasi masalah masalah tersebut.

Teknologi berkembang dan tumbuh begitu pesat, tak heran di banyak tempat sudah di terapkan teknologi baik yang simple maupun teknologi yang kompleks, sebagai contoh teknologi *web app*, *web app* adalah aplikasi yang bisa diakses melalui mesin penjelajah (*search engine*) baik melalui *internet* atau *intranet*. pengembangan nya mudah dan tanpa harus di distribusikan maupun di *install* jika ingin menggunakan nya. Dengan teknologi *web app* permasalahan yang sudah dipaparkan bisa terselesaikan dengan cara membangun sistem yang mampu melakukan pengarsipan secara digital sehingga data tidak lagi disimpan secara fisik

melainkan secara digital di dalam sistem *web app* yang akan di bangun, selain itu arsip dilengkapi dengan enkripsi sehingga data yang disimpan aman.

Berbicara mengenai enkripsi, algoritma *AES* (*Advanced Encryption Standard*) merupakan salah satu jenis algoritma yang digunakan untuk enkripsi, algoritma *AES* merupakan standar kriptografi pengganti *DES* (*Data Encryption Standard*). Algoritma *AES* diresmikan pada 2001 oleh NIST (*National Institute of Standard and Technology*) Amerika. Kelebihan dari algoritma *AES* dilihat dari segi jenis kunci yang simetris maka kecepatan operasi (komputasi) lebih tinggi bila dibandingkan dengan algoritma asimetrik sehingga dapat digunakan pada sistem *real-time*, *AES* mempunyai panjang kunci paling sedikit 128 bit, maka *AES* tahan terhadap serangan *exhaustive key search*. Dengan panjang kunci 128-bit, maka terdapat 2128 ≈ 3,4 x 1038 kemungkinan kunci. Dengan kelebihan yang dimiliki algoritma *AES* maka penulis memilih algoritma *AES* sebagai pengamanan enkripsi yang akan diterapkan pada *web app* E-Arsip Yayasan Universitas Islam Sumatera Utara.

Berdasarkan pemaparan diatas penulis mengangkat judul penelitian dengan judul "**Pengamanan Arsip Dengan Algoritma Enkripsi Aes-256 Untuk Web App E-Arsip Yayasan Universitas Islam Sumatera Utara**".

1.2 Rumusan Masalah

Berdasarkan latar belakang yang mendasari penulis melakukan penelitian ini, penulis merumuskan beberapa rumusan masalah antara lain sebagai berikut :

1. Bagaimana membangun E-Arsip ?
2. Bagaimana mengimplementasikan enkripsi *AES*-256 pada E-Arsip ?

1.3 Batasan Masalah

Agar pembahasan dalam penelitian ini tidak melebar dan memudahkan dalam proses penelitian maupun proses perancangan, maka diperlukan batasan masalah sebagai berikut :

1. E-Arsip menggunakan data pencatatan dan penerimaan surat masuk dan surat keluar.
2. Keamanan arsip pada E-Arsip hanya menggunakan algoritma enkripsi *AES*-256.
3. Penelitian dilakukan di Yayasan Universitas Islam Sumatera Utara.

1.4 Tujuan

Tujuan dari penelitian yang dilakukan oleh penulis antara lain sebagai berikut:

1. Membangun E-Arsip Yayasan Universitas Islam Sumatera Utara.
2. Mengimplementasikan enkripsi *AES*-256 terhadap arsip pada E-Arsip.

1.5 Manfaat

Penulis mengharapkan dari penelitian yang dilakukan dapat memberikan efek yang positif dan memberikan manfaat. Berikut manfaat yang ingin dicapai dalam penelitian ini.

1. Memudahkan dalam mencatat surat masuk dan surat keluar.
2. Mempermudah dalam menyimpan data surat masuk dan surat keluar.
3. Mengamankan data surat Yayasan Universitas Islam Sumatera Utara

1.6 Sistematika Penulisan

Untuk mempermudah dalam penyusunan dan memahami skripsi maka penulis menyajikan sistematika penulisan sebagai berikut :

BAB I : PENDAHULUAN

Pada bab ini akan dijelaskan tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian, sistematika penulisan.

BAB II : TINJAUAN PUSTAKA

Pada bab ini memuat tentang materi-materi pendukung dalam penyusunan skripsi, mulai dari teori-teori yang digunakan, konsep-konsep yang akan diterapkan dalam menyelesaikan permasalahan yang penulis teliti dalam penelitian ini.

BAB III : METODE PENELITIAN

Pada bab ini memuat mengenai metode yang penulis gunakan dalam menyelesaikan rumusan masalah, tahap-tahap mengenai teknik pengolahan data, perancangan aplikasi, dan pembuatan aplikasi.

BAB IV : HASIL DAN PEMBAHASAN

Pada bab ini memuat hasil-hasil yang didapat dari penelitian serta melakukan pembahasan atas hasil yang diperoleh. Kesulitan yang ditemukan saat perancangan dan pembuatan aplikasi.

BAB V : KESIMPULAN DAN SARAN

Pada bab ini memuat kesimpulan dan saran penulis atas penelitian yang dilakukan.

BAB II

TINJAUAN PUSTAKA

2.1 Arsip

Menurut Kamus Besar Bahasa Indonesia, arsip adalah simpanan surat-surat penting. Berdasarkan hal tersebut, dapat diartikan bahwa tidak semua surat dikatakan arsip. Surat dapat dikatakan arsip apabila memenuhi syarat sebagai berikut:

1. Surat tersebut harus masih mempunyai kepentingan (bagi lembaga, organisasi, instansi, perseorangan) baik untuk masa kini maupun masa akan datang.
2. Surat tersebut, karena masih mempunyai nilai kepentingan harus disimpan dengan mempergunakan suatu sistem tertentu sehingga dengan mudah dan cepat ditemukan apabila sewaktu-waktu diperlukan kembali. (Wursanto, 2004: 13)

Di Indonesia sendiri, kearsipan telah diatur dalam undang-undang No.43 tahun 2009 tentang kearsipan. Dalam undang-undang tersebut, dinyatakan bahwa arsip adalah sebagai berikut:

1. Kearsipan adalah hal-hal yang berkenaan dengan arsip
2. Arsip adalah rekaman kegiatan atau peristiwa dalam berbagai bentuk dan media sesuai dengan perkembangan teknologi informasi dan komunikasi yang dibuat dan diterima oleh lembaga negara, pemerintahan daerah, lembaga pendidikan, perusahaan, organisasi politik, organisasi kemasyarakatan, dan perorangan dalam pelaksanaan kehidupan bermasyarakat, berbangsa, dan

bernegara.

Selain dari pengertian diatas, arsip dapat diartikan pula sebagai suatu badan (*agency*) yang melakukan segala kegiatan pencatatan penanganan, penyimpanan, dan pemeliharaan surat-surat yang mempunyai arti penting baik ke dalam maupun ke luar, baik yang menyangkut soal-soal pemerintahan maupun non-pemerintahan dengan menerapkan kebijaksanaan dan sistem tertentu yang dapat dipertanggung jawabkan.

2.2 Algoritma *AES*-256

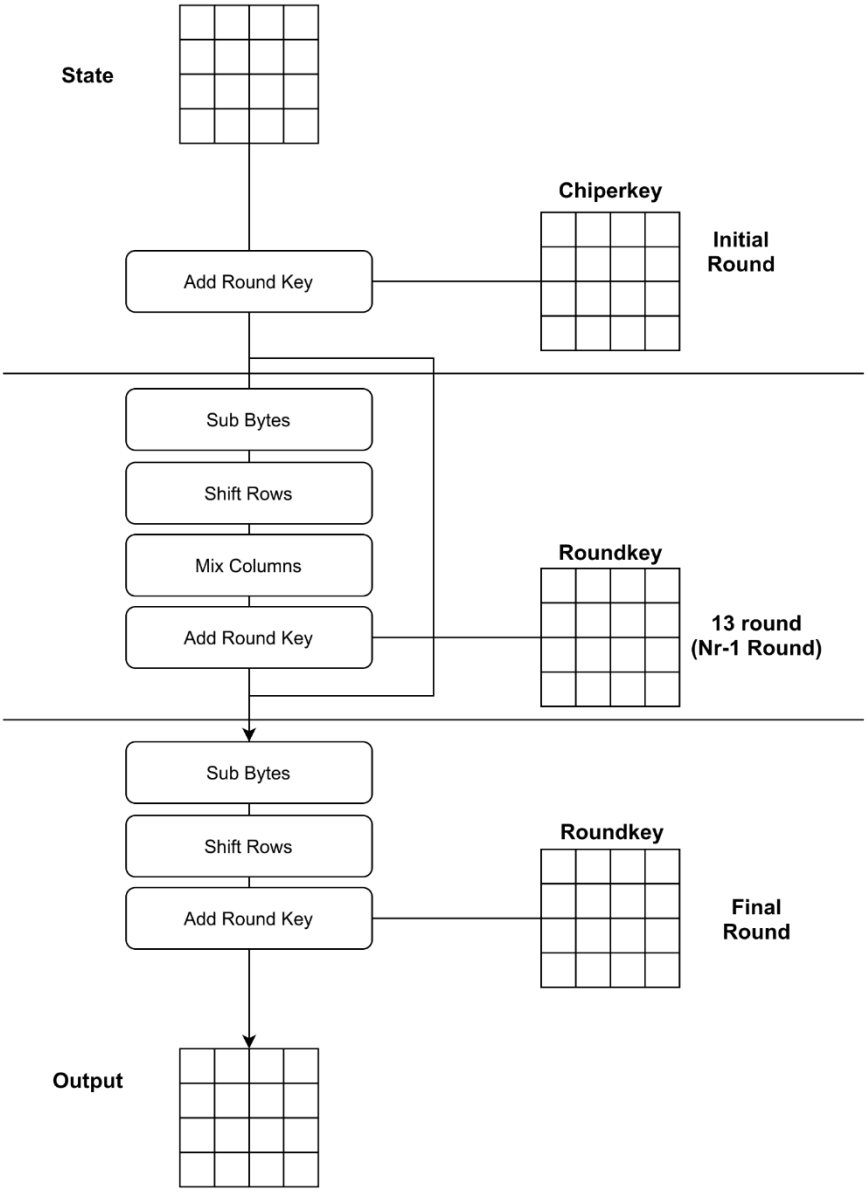
Kriptografi merupakan salah satu solusi atau metode pengamanan data yang tepat untuk menjaga kerahasiaan dan keaslian data, serta dapat meningkatkan aspek keamanan suatu data atau informasi. Metode ini bertujuan agar informasi yang bersifat rahasia dan dikirim melalui suatu jaringan, seperti LAN atau internet, tidak dapat diketahui atau dimanfaatkan oleh orang atau pihak yang tidak berkepentingan. Kriptografi mendukung kebutuhan dua aspek keamanan informasi, yaitu perlindungan terhadap kerahasiaan data informasi dan perlindungan terhadap pemalsuan dan pengubahan informasi yang tidak diinginkan.

Untuk mengetahui apakah suatu algoritma kriptografi dapat mengamankan data dengan baik dapat dilihat dari segi lamanya waktu proses pembobolan untuk memecahkan data yang telah disandikan. Seiring dengan perkembangan teknologi komputer yang semakin canggih, maka dunia teknologi informasi membutuhkan algoritma kriptografi yang lebih kuat dan aman. Saat ini, *AES* (*Advanced Encryption Standard*) merupakan algoritma *cipher* yang cukup aman untuk melindungi data atau informasi yang bersifat rahasia. Pada tahun 2001, *AES*

digunakan sebagai 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 yang bervariasi, yaitu 128 *bit*, 192 *bit*, dan 256 *bit*.

2.3 Proses Enkripsi *AES* 256

Proses enkripsi algoritma *AES* terdiri dari 4 jenis transformasi *bytes*, yaitu *SubBytes*, *ShiftRows*, *Mixcolumns*, dan *AddRoundKey*. Pada awal proses enkripsi, *input* yang telah salinkan ke dalam *state* akan mengalami transformasi *byte AddRoundKey*. Setelah itu, *state* akan mengalami transformasi *SubBytes*, *ShiftRows*, *MixColumns* dan *AddRoundKey* secara berulang-ulang sebanyak Nr. Proses ini dalam algoritma *AES* disebut sebagai *round function*. *Round* yang terakhir agak berbeda dengan *round-round* sebelumnya dimana pada *round* terakhir, *state* tidak mengalami transformasi *MixColumns*. Berikut gambaran simulasi dari proses enkripsi pada Algoritma *AES*.



Gambar 2.1 Proses Enkripsi *AES*

Berikut penjelasan mengenai gambar :

2.3.1 AddRoundKey

Dalam *initial round*, transformasi *AddRoundKey*() dilakukan terhadap kunci utama. Sedangkan dalam 10 *round* yang lain, proses *AddRoundKey* dilakukan terhadap kunci putaran (*round key*). Proses *AddRoundKey* didefinisikan sebagai operasi XOR antara *array state* dengan *round key*. Operasi XOR dilakukan pada

masing - masing *byte* dalam *array* sehingga menghasilkan nilai baru pada *array* hasil dengan ukuran *array* hasil sama dengan ukuran *array state* awal dan *array key*, yaitu sebesar 4x4. Hasil untuk masing-masing baris dan kolom pada *array state* hasil diperoleh dari hasil operasi XOR antara *array state* awal dengan *array key* untuk baris dan kolom yang sama.

2.3.2 SubBytes

Transformasi *SubBytes*() memetakan setiap *byte* dari *array state* dengan menggunakan tabel substitusi *S-Box*. Tabel *S-Box* dapat dilihat pada gambar berikut.

		y														
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	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

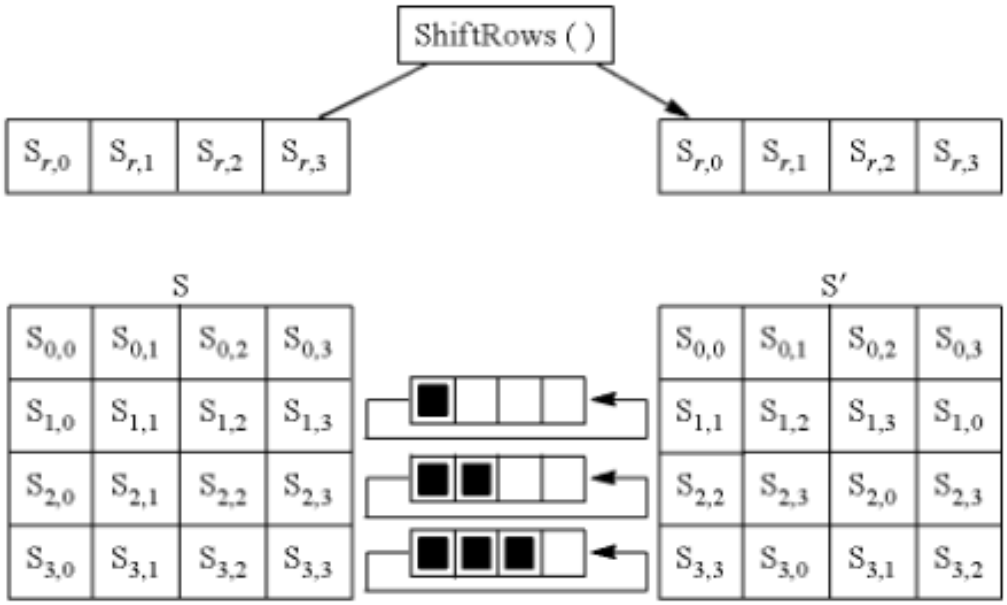
Gambar 2.2 *S-Box*

Cara pensubstitusian adalah sebagai berikut: untuk setiap *byte* pada *array*,

misalkan S[r,c] = xy, yang dalam hal ini xy adalah digit heksadesimal dari nilai S[r,c], maka nilai substitusinya, yang dinyatakan dengan S'[r,c], adalah elemen di dalam *S-Box* yang merupakan perpotongan baris x dengan kolom y.

2.3.3 ShiftRows

Transformasi *ShiftRows*() melakukan pergeseran secara *wrapping* (*siklik*) pada 3 baris terakhir dari *array state*. Jumlah pergeseran bergantung pada nilai baris (r). Baris r = 1 digeser sejauh 1 *byte*, baris r = 2 digeser sejauh 2 *byte* dan baris r = 3 digeser sejauh 3 *byte*. Baris r = 0 tidak digeser.



Gambar 2.3 Ilustrasi *ShiftRows*

2.3.4 MixColumns

Transformasi *MixColumns*() dilakukan setelah transformasi *ShiftRows*, merupakan sumber utama dari difusi pada algoritma *AES*. Difusi merupakan prinsip yang menyebarkan pengaruh satu *bit plaintext* atau kunci ke sebanyak mungkin *ciphertext*. Transformasi *MixColumns*() mengalikan setiap kolom dari *array state*

dengan *polinom* a(x) mod (x4 + 1). Setiap kolom diperlakukan sebagai *polinom* 4 suku pada GF (28). *Polinom* a(x) yang ditetapkan pada persamaan 1

a (x) = {03}x3 + {01} x2 + {01} x + {02} (1)

Transformasi ini dinyatakan sebagai perkalian matriks seperti pada persamaan 2

s'(x) = a(x) ⊗ s (x) (2)

Hasil dari perkalian matriks tersebut, setiap *byte* dalam kolom *array state* akan digantikan dengan nilai baru. Persamaan matematis untuk setiap *byte* tersebut pada persamaan 3

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

Gambar 2.4 Perkalian *Matrix*

s'0,c = ({02}• s0,c) ⊕ ({03}• s1,c) ⊕ s2,c ⊕ s3,c

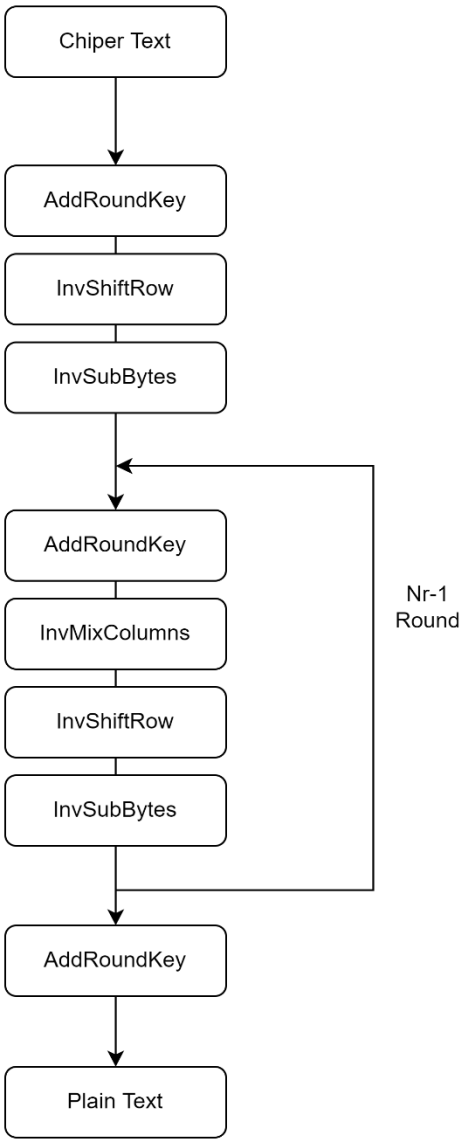
s'1,c = s0,c ⊕ ({02} • s1,c) ⊕ ({03} • s2,c) ⊕ s3,c

s'2,c = s0,c ⊕ s1,c ⊕ ({02}• s2,c)⊕ ({03}• s3,c)

s'3,c = ({03} • s0,c) ⊕ s1,c ⊕ s2,c ⊕({02} • s3,c) (3)

2.4 Proses Dekripsi *AES* 256

Transformasi *cipher* dapat dibalikkan dan diimplementasikan dalam arah yang berlawanan untuk menghasilkan *inverse cipher* yang mudah dipahami untuk algoritma *AES*. Transformasi *byte* yang digunakan pada *invers cipher* adalah *InvShiftRows*, *InvSubBytes*, *InvMixColumns*, dan *AddRoundKey*. Algoritma dekripsi dapat dilihat pada skema berikut ini :

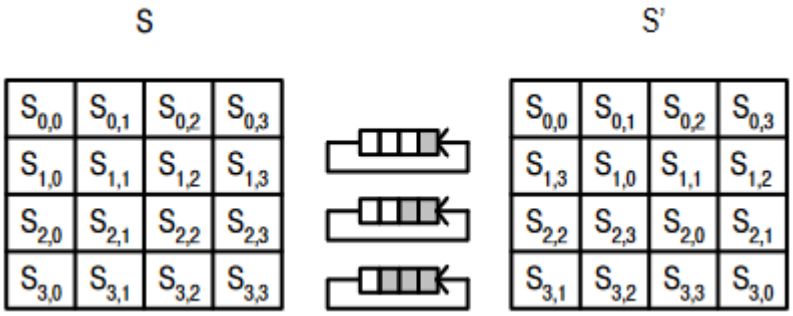


Gambar 2.5 Proses Dekripsi *AES*

Berikut penjelasan mengenai gambar :

2.4.1 InvShiftRows

InvShiftRows adalah transformasi *byte* yang berkebalikan dengan transformasi *ShiftRows*. Pada transformasi *InvShiftRows*, dilakukan pergeseran *bit* ke kanan sedangkan pada *ShiftRows* dilakukan pergeseran *bit* ke kiri. Ilustrasi transformasi InvShiftRows terdapat pada gambar berikut.



Gambar 2.6 Ilustrasi *InvShiftRows*

2.4.2 InvSubBytes

InvSubBytes juga merupakan transformasi *bytes* yang berkebalikan dengan transformasi *SubBytes*. Pada *InvSubBytes*, tiap elemen pada *state* dipetakan dengan menggunakan tabel *Inverse S-Box*. Tabel *Inverse S-Box* akan ditunjukkan dalam gambar berikut.

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

Gambar 2.7 *Inverse S-Box*

2.4.3 InvMixColumns

Setiap kolom dalam state dikalikan dengan matriks perkalian dalam *AES*. Perkalian dalam matriks dapat dituliskan :

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

Gambar 2.8 Perkalian *Matrix Invercolum*

Hasil dari perkalian matriks tersebut, setiap *byte* dalam kolom *array state* akan digantikan dengan nilai baru. Persamaan matematis untuk setiap *byte* tersebut pada persamaan 4

s'0,c = ({0E}•s0,c) ⊕({0B}• s1,c)⊕({0D}•s2,c) ⊕({09}• s3,c)

s'1,c = ({09}•s0,c) ⊕ ({0E} • s1,c) ⊕ ({0B} • s2,c) ⊕ ({0D}•s3,c)

s'2,c = ({0D}•s0,c)⊕ ({09}•s1,c)⊕ ({0E}• s2,c)⊕ ({0B}• s3,c)

s'3,c = ({0B}•s0,c) ⊕ ({0D}•s1,c)⊕({09}•s2,c) ⊕({0E} • s3,c) (4)

2.4.4 AddRoundKey

Proses transformasi *AddRoundKey* pada proses dekripsi merupakan transformasi yang bersifat *self-invers* dengan syarat menggunakan kunci yang sama.

2.5 Tools Perancangan

Dalam merancang suatu sistem dan diharapkan sistem tersebut berjalan dengan semestinya dan sesuai dengan tujuan dari sistem tersebut, maka proses perancangan sistem sangat diperlukan. Layak nya membangun sebuah rumah harus memiliki *blueprint* dari rumah tersebut, maka sama hal nya dengan merancang sebuah sistem diperlukan gambaran sistem yang dibutuhkan dan diharapkan.

Dalam perancangan sistem, para pengembang sering menggunakan *UML* (*Unified Modelling Language*). *UML* adalah sekumpulan alat yang biasanya berupa diagram untuk merancang dan memodelkan bagaimana sistem tersebut bekerja, bagaimana pengguna dapat berinteraksi dengan sistem, bagaimana tata cara kerja dari sistem dan fitur-fitur yang terdapat di sebuah sistem yang nantinya akan diimplementasikan.

Penggunaan *UML* bermanfaat memanajemen kompleksitas dari sistem, mendeteksi kesalahan yang mungkin terjadi ketika diimplementasikan, menjelaskan tata kerja dari sistem kepada para pihak yang berkepentingan.

2.6 Langkah-langkah Pembuatan *UML*

Dalam membuat *UML* tidak terdapat aturan yang baku dalam penggunaannya, akan tetapi para pengembang di seluruh dunia sering menggunakan langkah langkah sebagai berikut dalam membuat *UML*. Berikut adalah langkah langkah yang paling umum untuk membuat *UML* :

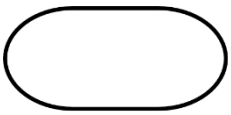
1. Membuat *Functional requirement*

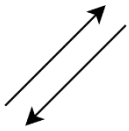
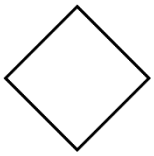
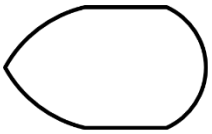
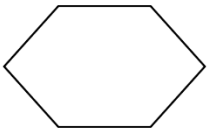
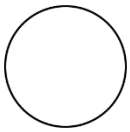
Pada tahap ini pengembang biasanya memuat hal-hal yang dibutuhkan oleh sistem, bercerita sistem tersebut digunakan untuk keperluan apa, hal apa saja yang dapat dilakukan oleh sistem.

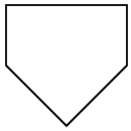
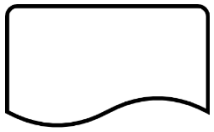
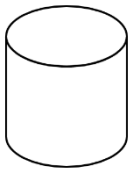
2. Membuat *Flowchart* Diagram

Flowchart atau bagan alur adalah diagram yang menampilkan langkah-langkah dan keputusan untuk melakukan sebuah proses dari suatu program. Setiap langkah digambarkan dalam bentuk diagram dan dihubungkan dengan garis atau arah panah. *Flowchart* berperan penting dalam memutuskan sebuah langkah atau fungsionalitas dari sebuah proyek pembuatan program yang melibatkan banyak orang sekaligus. Selain itu dengan menggunakan bagan alur proses dari sebuah program akan lebih jelas, ringkas, dan mengurangi kemungkinan untuk salah penafsiran. Penggunaan *flowchart* dalam dunia pemrograman juga merupakan cara yang bagus untuk menghubungkan antara kebutuhan teknis dan non-teknis. Adapun simbol-simbol yang digunakan dalam pembuatan *flowchart* diagram dapat dilihat pada tabel berikut:

Table 2.1 Simbol *Flowchart* Diagram

Simbol	Nama Simbol	Penjelasan Simbol
	*Input/Output*	Simbol yang menyatakan proses *input* dan *output* tanpa tergantung dengan jenis peralatannya.
	*Terminator*	Simbol untuk permulaan (*start*) atau akhir (*stop*) dari suatu kegiatan.

	*Process*	Simbol yang menunjukkan pengolahan yang dilakukan oleh komputer.
	*Flow*	Simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga *connecting line*.
	*Decision*	Simbol pemilihan proses berdasarkan kondisi yang ada.
	*Predifine Process*	Simbol untuk pelaksanaan suatu bagian (*sub-program*)/prosedure.
	*Display*	Simbol yang menyatakan peralatan output yang digunakan.
	*Preparation*	Simbol yang menyatakan penyediaan tempat penyimpanan suatu pengolahan untuk memberikan nilai awal.
	*On-page Reference*	Simbol untuk keluar-masuk atau penyambungan proses dalam lembar kerja yang sama.

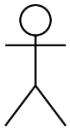
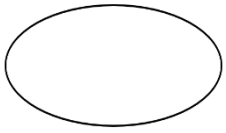
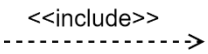
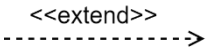
	*Off-page Reference*	Simbol untuk keluar-masuk atau penyambungan proses pada lembar kerja yang berbeda.
	Document	Simbol yang menyatakan input berasal dari dokumen dalam bentuk kertas atau *output* dicetak dalam bentuk kertas.
	*Database*	Tempat penyimpanan data

Sumber : https://www.dicoding.com/blog/flowchart-adalah/

3. Membuat *Use Case* Diagram

Pada tahap ini membuat aktor atau *user* tipe apa saja yang terlibat dalam sistem dan menentukan hal yang bisa dilakukan *user* terhadap sistem. Adapun simbol-simbol yang digunakan dalam pembuatan *use case* diagram dapat dilihat pada tabel berikut:

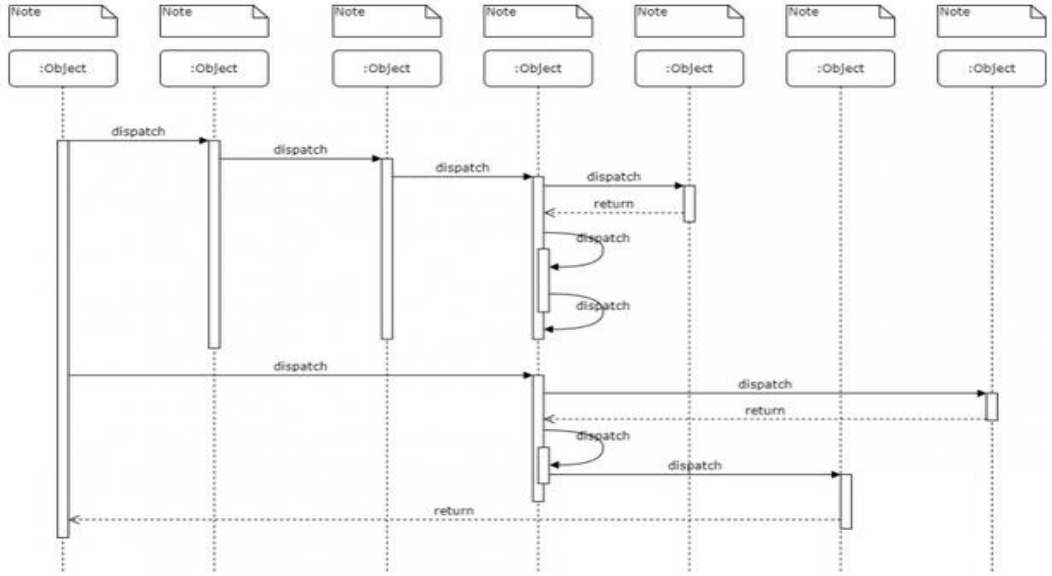
Table 2.2 Simbol *Use Case* Diagram

Simbol	Nama Simbol	Penjelasan Simbol
	*Actor*	Menggambarkan tokoh atau seseorang yang berinteraksi dengan sistem. Dan dapat menerima dan memberi informasi pada sistem.
	*Use Case*	Menjelaskan fungsi dari kegunaan sistem yang di rancang.
	Association	Menghubungkan antara *use case* dengan aktor tertentu.
	*Include*	Menunjukkan bahwa *use case* satu merupakan bagian dari *use case* lainnya.
	*Extend*	Menunjukkan arah panah secara putus-putus dari *use case* ke *base use case*.

Sumber : https://www.dicoding.com/blog/apa-itu-uml/

4. Membuat *Sequence* Diagram.

   Sequence diagram merupakan diagram yang menjelaskan interaksi objek berdasarkan urutan waktu. *Sequence* dapat menggambarkan urutan atau tahapan yang harus dilakukan untuk dapat menghasilkan sesuatu, seperti yang tertera pada *Use Case* diagram. Adapun contoh dari *sequence* diagram dapat dilihat pada gambar berikut.

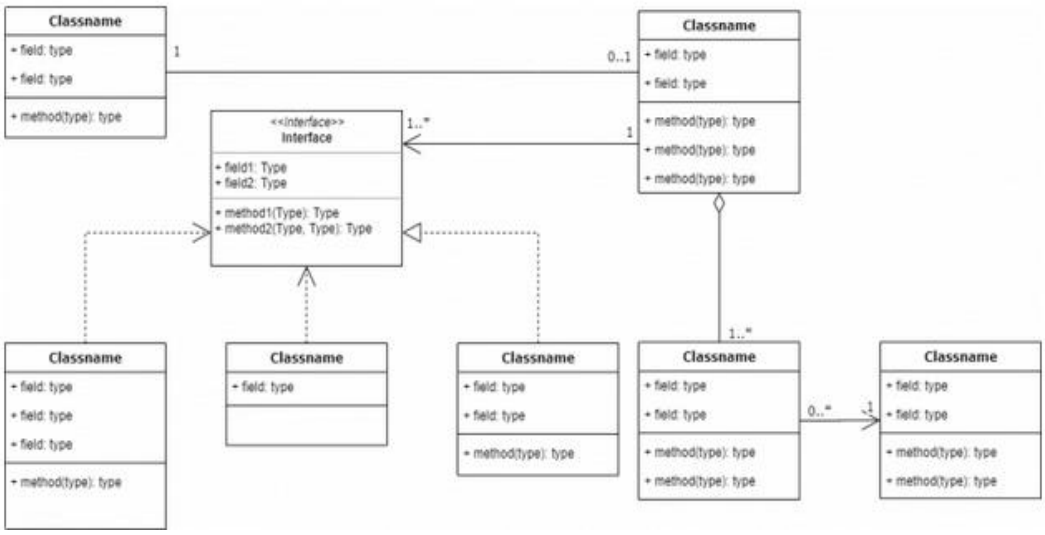


Gambar 2.9 Contoh Dari *Sequence* Diagram

Sumber : https://www.dicoding.com/blog/apa-itu-uml/

5. Membuat *Class* Diagram.

   Pada tahap ini menentukan *class class* apa saja yang digunakan. beserta *method* apa saja yang terdapat didalam *class* tersebut berdasarkan *sequence* diagram yang sudah dibuat. pembuatan *class* diagram dapat mempermudah dalam pengimplementasian sistem yang dirancang ke dalam baris baris kode. Adapun contoh dari *sequence* diagram dapat dilihat pada gambar berikut.



Gambar 2.10 Contoh *Class* Diagram

Sumber : https://www.dicoding.com/blog/apa-itu-uml/

6. Membuat *Activity* Diagram.

   Langkah terakhir adalah membuat *activity* diagram. *Activity* diagram mirip dengan *flowchart*. Setelah 5 langkah sebelum nya dilewati , pada tahap ini menggambarkan bagaimana sistem bekerja secara keseluruhan. Adapun simbol-simbol yang digunakan dalam pembuatan use case diagram dapat dilihat pada tabel berikut:

Table 2.3 Simbol *Activity* Diagram

Simbol	Nama Simbol	Penjelasan Simbol
	*Initial*	Titik awal untuk memulai suatu aktivitas.
	*Final*	Titik akhir untuk mengakhiri aktivitas.

	*Activity*	Menandakan sebuah aktivitas
	*Decision*	Pilihan untuk mengambil keputusan
	Fork Atau *Join*	Digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.
	*Flow final*	Untuk mengakhiri suatu aliran.
	*Swimeline*	Untuk mengelompokkan activity berdasarkan aktor.

Sumber : https://www.dicoding.com/blog/apa-itu-uml/

2.7 Teknologi Yang Digunakan

Dalam penelitian ini penulis menggunakan beberapa teknologi dan bahasa pemrograman yang digunakan dalam membangun aplikasi e-arsip antara lain sebagai berikut :

2.7.1 *HTML*

Hypertext Markup Language (*HTML*) adalah bahasa markah standar untuk dokumen yang dirancang untuk ditampilkan di peramban *internet* dan dibantu oleh

teknologi seperti *Cascading Style Sheets* (*CSS*) dan bahasa *scripting* seperti *JavaScript* dan *VBScript*.

Peramban internet menerima dokumen *HTML* dari *server web* atau dari penyimpanan lokal dan membuat dokumen menjadi halaman *web* multimedia. *HTML* menggambarkan struktur halaman *web* secara semantik dan isyarat awal yang disertakan untuk penampilan dokumen.

Pada tahun 1980 seorang ahli fisika, Tim Berners-Lee, dan juga seorang kontraktor di *CERN* (Organisasi Eropa untuk Riset Nuklir) mengusulkan dan menyusun *ENQUIRE*, sebuah sistem untuk ilmuwan *CERN* dalam membagi dokumen. Sembilan tahun kemudian, Berners-Lee mengusulkan adanya sistem markah berbasis *internet*. Berners-Lee menspesifikasikan *HTML* dan menulis jaringan beserta perangkat lunaknya di akhir 1990.

Pada tahun yang sama, Berners-Lee dan Robert Cailliau, insinyur sistem data *CERN* berkolaborasi dalam sebuah permintaan untuk pendanaan, namun tidak diterima secara resmi oleh *CERN*. Penjelasan pertama yang dibagi untuk umum dari *HTML* adalah sebuah dokumen yang disebut "Tanda *HTML*", pertama kali disebutkan di *Internet* oleh Tim Berners-Lee pada akhir 1991, tanda ini menggambarkan 18 elemen awal mula versi sederhana dari *HTML*. Kecuali untuk *tag hyperlink*, yang sangat dipengaruhi oleh SGMLguid, *in-house Standard Generalized Markup Language (SGML)* berbasis format dokumen di *CERN*.

HTML adalah bahasa markah yang digunakan peramban untuk menafsirkan dan menulis teks, gambar dan bahan lainnya ke dalam halaman web secara visual maupun suara. Karakteristik dasar untuk setiap item dari markah *HTML* didefinisikan di dalam peramban, dan karakteristik ini dapat diubah atau

ditingkatkan dengan menggunakan tambahan halaman *web* desainer *CSS*. Banyak elemen teks ditemukan di laporan teknis *ISO* pada tahun 1988 TR 9537 Teknik untuk menggunakan *SGML*, yang pada gilirannya meliputi fitur bahasa format teks awal seperti yang digunakan oleh komandan *RUNOFF* dikembangkan pada awal 1960-an untuk sistem operasi, perintah-perintah format ini berasal dari perintah yang digunakan oleh pengetik untuk memformat dokumen *CTSS* secara manual. Namun, konsep *SGML* dari markah umum didasarkan pada unsur-unsur daripada hanya efek cetak, dengan pemisahan struktur dan markah. *HTML* semakin bergerak ke arah yang lebih baik dengan *CSS*.

2.7.2 *CSS*

Cascading Style Sheet (CSS) merupakan aturan untuk mengatur beberapa komponen dalam sebuah *web* sehingga akan lebih terstruktur dan seragam. *CSS* bukan merupakan bahasa pemrograman.

Sama halnya *styles* dalam aplikasi pengolahan kata seperti *Microsoft Word* yang dapat mengatur beberapa *style*, misalnya *heading*, *subbab*, *body text*, *footer*, *images*, dan *style* lainnya untuk dapat digunakan bersama-sama dalam beberapa berkas (*file*). Pada umumnya *CSS* dipakai untuk memformat tampilan halaman *web* yang dibuat dengan bahasa *HTML* dan *XHTML*.

CSS dapat mengendalikan ukuran gambar, warna bagian tubuh pada teks, warna tabel, ukuran *border*, warna *border*, warna *hyperlink*, warna *mouse over*, spasi antar paragraf, spasi antar teks, margin kiri, kanan, atas, bawah, dan parameter lainnya. *CSS* adalah bahasa *style sheet* yang digunakan untuk mengatur tampilan

dokumen. Dengan adanya *CSS* memungkinkan kita untuk menampilkan halaman yang sama dengan format yang berbeda.

Nama *CSS* didapat dari fakta bahwa setiap deklarasi *style* yang berbeda dapat diletakkan secara berurutan, yang kemudian membentuk hubungan ayah-anak (*parent-child*) pada setiap *style*. *CSS* sendiri merupakan sebuah teknologi *internet* yang direkomendasikan oleh *World Wide Web Consortium* atau *W3C* pada tahun 1996. Setelah *CSS* distandarisasikan, *Internet Explorer* dan *Netscape* melepas *browser* terbaru mereka yang telah sesuai atau paling tidak hampir mendekati dengan standar *CSS*.

Untuk saat ini terdapat tiga versi *CSS*, yaitu *CSS1*, *CSS2* dan *CSS3*. *CSS1* dikembangkan berpusat pada pemformatan dokumen *HTML*, *CSS2* dikembangkan untuk memenuhi kebutuhan terhadap format dokumen agar bisa ditampilkan di printer, sedangkan *CSS3* adalah versi terbaru dari *CSS* yang mampu melakukan banyak hal dalam desain *website*. *CSS3* mendukung penentuan posisi konten, *downloadable*, huruf *font*, tampilan pada tabel / table *layout* dan media tipe untuk printer. Kehadiran versi *CSS* yang ketiga diharapkan lebih baik dari versi pertama dan kedua.

CSS3 juga dapat melakukan animasi pada halaman *website*, di antaranya animasi warna hingga animasi 3D. Dengan *CSS3* desainer lebih dimudahkan dalam hal kompatibilitas *websitenya* pada *smartphone* dengan dukungan fitur baru yakni media *query*. Selain itu banyak fitur baru pada *CSS3* seperti: *multiple background, border-radius, drop-shadow, border-image, CSS Math* dan *CSS Object Model*.

2.7.3 *XAMPP*

Betha Sidik (2018:6) *XAMPP* adalah singkatan yang setiap huruf adalah:

1. X: Program ini dapat dijalankan di banyak sistem operasi, seperti Windows, Linux, Mac OS, dan Solaris.
2. A: *Apache*, *server* aplikasi *Web*. *Apache* memilik tugas utama untuk menghasilkan halaman *web* yang benar kepada pengguna terhadap kode *PHP* yang sudah dituliskan oleh pembuat halaman *web*. Jika perlu kode *PHP* juga berdasarkan yang tertulis, dapat *database* diakses dulu (misalnya *MySQL*) untuk mendukung halaman *web* yang dihasilkan.
3. M: *MySQL*, *server* aplikasi *database*. Pertumbuhannya disebut *SQL* singkatan dari *Structured Query Language*. *SQL* merupakan bahasa terstruktur yang difungsikan untuk mengolah *database*. *MySQL* dapat digunakan untuk membuat dan mengelola *database* dan isinya. Bisa juga memanfaatkan *MySQL* guna untuk menambahkan, mengubah, dan menghapus data dalam *database*.
4. P: PHP, bahasa pemrograman *web*. Bahasa pemrograman *PHP* adalah bahasa pemrograman yang digunakan untuk membuat *web* yang *server-side scripting*. *PHP* digunakan untuk membuat halaman *web* dinamis. Sistem manajemen *database* yang sering digunakan dengan *PHP* adalah *MySQL*. Namun *PHP* juga mendukung pengelolaan sistem *database Oracle*, *Microsoft Access*, *Interbase*, d-*base*, *PostgreSQL*, dan sebagainya.
5. P: *Perl*, bahasa pemrograman untuk semua tujuan, pertama kali dikembangkan oleh Larry Wall, mesin UNIX. Perl dirilis pertama kali tanggal 18 Desember 1987 yang ditandai dengan keluarnya *Perl* 1. Pada

versi-versi selanjutnya, *Perl* juga tersedia untuk berbagai sistem operasi *UNIX* (*SunOS, Linux, BSD, HP-UX*), juga tersedia untuk sistem operasi seperti *DOS, Windows, PowerPC, BeOS, VMS, EBCDIC,* dan *PocketPC*.

Menurut Pratama, I Putu Agus Eka (2014: 440) "*XAMPP* adalah aplikasi *web server* bersifat instan (siap saji) yang dapat digunakan baik di sistem operasi *Linux* maupun di sistem operasi *Windows*.

Menurut Hidayatullah (2015:127), "*XAMPP* merupakan *web server* yang mudah digunakan yang dapat melayani tampilan halaman *web* yang dinamis dan dapat diakses secara lokal menggunakan *web server local* (*localhost*)".

Purbadian (2016:1), menjelaskan bahwa "*XAMPP* merupakan suatu *software* yang bersifat *open source* yang merupakan pengembangan dari *LAMP* (*Linux, Apache, MySQL, PHP* dan *Perl*)".

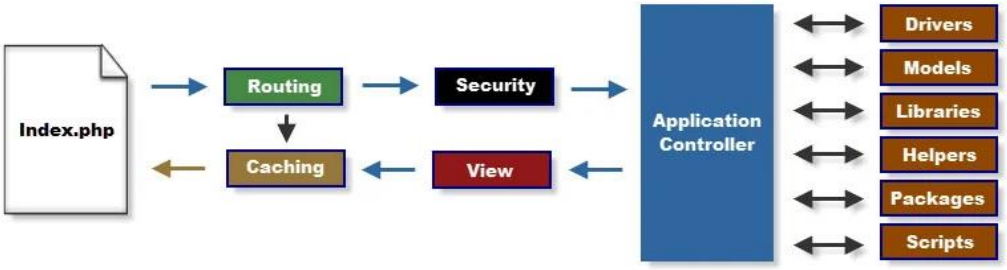
Menarik kesimpulan dari beberapa pendapat para ahli bahwa *XAMPP* adalah perangkat pembantu yang menyediakan alat untuk sebagai jembatan pembuatan sebuah program.

2.7.4 *Codeigniter (CI)*

Framework Codeigniter (*CI*) Salah satu *framework* yang digunakan dalam pembuatan tugas akhir ini adalah *CodeIgniter*. Penulis menggunakan *Framework CodeIgniter* karena untuk melakukan pengembangan program tidak perlu membuat kode dari awal sehingga dalam proses kerja nya pun terasa lebih cepat. Menurut Betha Sidik (2018:2) *Codeigniter* (*CI*) adalah *framework* pengembangan aplikasi (*application development framework*) dengan menggunakan *PHP*, suatu kerangka pembuatan program dengan menggunakan *PHP*.

Pengembangan dapat langsung menghasilkan program dengan cepat, dengan mengikuti kerangka kerja untuk membuat yang telah disiapkan oleh *framework CI* ini. *Codeigniter* adalah *framework* untuk megembangkan sebuah aplikasi dengan menggunakan Bahasa pemrograman *PHP*.

Sedangkan, menurut Raharjo (2015:3) *CodeIgniter* adalah *framework web* untuk bahasa pemrograman *PHP*, yang dibuat oleh Rick Ellis pada tahun 2006, penemu dan pendiri EllisLab. Kesimpulan dari pengertian di atas bahwa *CodeIgniter* adalah *Framework PHP* yang di dalamnya terdapat fitur lengkap aplikasi *web* yang sudah dikemas menjadi satu.



Gambar 2.11 Alur Kerja Codeigniter

Sumber : https://codeigniter.com/userguide3/overview/appflow.html

Agar mudah dalam memahami gambar maka akan dijelasakan secara singkat sebagai berikut mengenai gambar :

1. *File* index.php berfungsi sebagai *front controller*. Menginisialisasi *resource* utama yang dibutuhkan untuk menjalankan *CodeIgniter*.
2. *Router*, memeriksa *HTTP request* untuk menentukan apa yang harus dilakukan.
3. Jika *file cache* ada, dikirim langsung ke *browser*, melewati eksekusi sistem normal.

4. Keamanan, sebelum *controller* aplikasi dimuat, *HTTP request* dan setiap data pengguna yang *disubmit* disaring terlebih dahulu untuk keamanan.
5. *Controller*, memuat *model*, *library* utama, *helper*, dan setiap *resource* lainnya yang diperlukan untuk memproses permintaan khusus.
6. *View*, proses *render* kemudian dikirim ke *web browser* agar dapat dilihat. Jika *caching* diaktifkan, *view* di *cache* terlebih dahulu sehingga pada permintaan berikutnya dapat dilayani.

2.7.5 *Bootstrap*

Bootstrap merupakan sebuah *Framework CSS* untuk membangun *website* yang menarik agar memudahkan pengembang disebut *Bootstrap*. Sulit untuk mengembangkan dan pemeliharaannya jika tidak ada konsistensi terhadap aplikasi *individual*. *Bootstrap* memberikan solusi rapi dan seragam terhadap solusi yang umum, tugas *interface* yang setiap pengembang hadapi.

Menurut Nahado (2014:14), "*Bootstrap* adalah *platform* untuk membuat *interface website* dan *aplikasi* berbasis *web*". *Bootstrap* berisi kode *HTML* dan *CSS* yang telah dilengkapi desain untuk *tipografi*, bentuk, tombol, navigasi dan sebagainya. *Bootstrap* bertujuan untuk meringankan pembuatan dan pengembangan *web*.

Menurut Alatas (2013:2) dalam bukunya yang berjudul *Responsive Web Design* dengan *PHP* dan *Bootstrap*, mengemukakan bahwa "*Bootstrap* merupakan *Framework* ataupun *Tools* untuk membuat aplikasi *web* ataupun situs *web responsive* secara cepat, mudah dan gratis".

Kumpulan alat gratis untuk membuat *website* dan aplikasi *web* atau *modular* yang pada dasarnya terdiri dari serangkaian *stylesheet LESS* yang menerapkan berbagai komponen *toolkit* adalah *Bootstrap* (Riyanto, 2014:18).

Berdasarkan kutipan para ahli, dapat disimpulkan bahwa *bootstrap* merupakan sebuah aplikasi yang dijadikan sebagai alat bantu untuk mendesain halaman *web* yang menarik.