BAB II
TINJAUAN PUSTAKA

Pada bab ini, penulis membahas hal-hal yang menjadi referensi dalam upaya merancang sistem pengamanan data hasil registrasi peserta pelatihan dan sertifikasi menggunakan algoritma *advanced encryption standard* dan algoritma *base64*.

2.1. Pencapain Terdahulu

Penulis menggunakan penelitian terdahulu sebagai tolak ukur untuk tidak terjadinya kesamaan terhadap apa yang sedang dirancang oleh penulis dalam penelitian ini. Penulis menggunakan referensi dari beberapa jurnal internasional dengan tujuan untuk mendapatkan sebuah inspirasi baru. Oleh karena itu, penulis perlu mempelajari penelitian terdahulu sehingga penulis dapat merancang sistem pada aplikasi *web* yang sesuai. Berikut ini hasil-hasil penelitian terdahulu.

1. Penelitian Khairul Muttaqin dan Jefril Rahmadoni (2020), berjudul "*Analysis and Design of File Security System AES (Advanced Encryption Standard) Cryptography Based*". Pada penelitian ini penulis menggunakan metode algoritma *advanced encryption standard*, kemudian penelitian yang dilakukan oleh penulis bertujuan untuk menganalisis dan mendesain sistem keamanan untuk suatu *file* menggunakan metode algoritma *advanced encryption standard* untuk *cryptography* sehingga *file* yang berisi informasi sensitif dapat dienkripsi dan didekripsi.

   Berdasarkan penelitian yang dilakukan terdapat 3 hal yang dapat disimpulkan:

   a. Hasil dari penelitian yang telah dilakukan membuktikan bahwa pesan atau *file* yang *digenerated* adalah dalam bentuk *format ciphertext*, sehingga pesan ataupun *file* tersebut tidak dapat dibaca jadi dapat dikatakan bahwa sistem bekerja dengan baik.
   b. Penggunaan *web programming* untuk membangung suatu *file security system* tampaknya cukup dengan mengimplementasikan proses algoritma *advanced encryption standard*.

c. Aplikasi *cryptography* menggunakan algoritma *advanced encryption standard* sensitif terhadap perubahan dari *ciphertext* yang diinput, hal tersebut didasarkan pada fakta bahwa perubahan kata *encryption key* (*ciphertext*) berbeda dengan *decryption key* (*plaintext*) yang mana akan menyebabkan perubahan data ketika akan didekripsi ke bentuk semula, dengan kata lain *file* tidak dapat didekripsi.

2. Penelitian Fathurrahmad dan Ester (2020), berjudul "*Development and Implementation of The Rijndael Algorithm and Base-64 Advanced Encryption Standard (AES) for Website Data Security*". Pada penelitian ini penulis menggunakan metode algoritma *advanced encryption standard*, kemudian penelitian yang dilakukan oleh penulis bertujuan untuk mengembangkan dan mengimplementasikan algoritma *rijndael* dan algoritma *base64* untuk tujuan keamanan data pada aplikasi berbasis *web*. Berdasarkan penelitian yang dilakukan terdapat 2 hal yang dapat disimpulkan:
   a. Berdasarkan hasil yang diteliti bahwa proses enkripsi dan dekripsi, menggunakan algoritma *base64* dan algoritma *rijndael* yang mana telah diintegrasikan dapat meningkatkan keamanan data secara lebih baik.
   b. Algoritma *advanced encryption standard* dapat dipakai untuk menggantikan peran dari algoritma *base64*, di mana pada implementasinya algoritma *advanced encryption standard* memiliki kecepatan yang baik dalam proses enkripsi. Namun, untuk pengembangan lebih lanjut dibutuhkan tambahan algoritma, yaitu *Secure Hash Algorithm-256* (SHA-256) untuk diintegrasikan dengan kedua algoritma tersebut.
3. Penelitian Muhammad Farras Muttaqin, Yaddarabullah, dan Silvester Dian Handy Permana (2020), berjudul "*Implementation of AES-128 and Token-Base64 to Prevent SQL Injection Attacks via HTTP*". Pada penelitian ini penulis menggunakan metode algoritma *base64*, kemudian penelitian yang dilakukan oleh penulis bertujuan untuk memberikan solusi alternatif dalam membuat suatu *security system* yang dapat mencegah *SQL injection attacks* dengan menerapkan algoritma kriptografi.

Berdasarkan penelitian yang dilakukan terdapat 2 hal yang dapat disimpulkan:

a. Kombinasi antara algoritma *advanced encryption standard* 128 *bit* dan algoritma *token base64* dapat mencegah *SQL injection attacks* dengan persentase 100% dari 83 upaya *penetration testing* menggunakan parameter *GET* dan *POST*.
b. Kombinasi dari dua algoritma tersebut dapat mencegah *SQL injection attacks via HTTP*. Namun, dengan penggunaan kombinasi antara kedua algoritma tersebut mengakibatkan *loading time* pada *server web* menjadi lebih lama dengan persentase 31.26% dibanding dengan tidak menggunakan kedua algoritma tersebut pada *security system*.

4. Penelitian Abolore Muhamin Logunleko, Kolawole Bariu Logunleko, dan Olanrewaju Olaide Lawal (2020), berjudul "*An End-to-End Secured Email System using Base64 Algorithm*". Pada penelitian ini penulis menggunakan metode algoritma *base64*, kemudian penelitian yang dilakukan oleh penulis bertujuan untuk mengembangkan suatu sistem yang mana dapat mengamankan informasi sensitif yang dikirim melalui *email* dengan menyediakan enkripsi yang aman, cepat, dan kuat, di mana membuat pesan menjadi sulit untuk ditafsirkan oleh penyerang saat dalam transmisi sehingga membuatnya tahan terhadap berbagai bentuk serangan.

   Berdasarkan penelitian yang dilakukan terdapat 2 hal yang dapat disimpulkan:

   a. Antarmuka penggunaan sistem sederhana dan mudah digunakan. Sistem menyediakan layanan *online*, sehingga untuk melakukan enkripsi dan dekripsi isi pesan *email* dapat digunakan secara langsung dengan memakai *email* pribadi *user* yang telah terdaftar.
   b. Dengan menggunakan algoritma *base64* maka dapat secara signifikan mengurangi *phishing* dan berpotensi mengurangi *spam message*. Solusinya tergantung pada algoritma *base64* sebagai komponen dari arsitektur *cryptography*.

5. Penelitian Bagus Satrio Waluyo Poetro, Sam Farisa Chaerul Haviana, dan Arief Budiman (2021), berjudul "*Web Based Security System Academic Exam Questions Using Advanced Encryption Standard*". Pada penelitian ini penulis menggunakan metode algoritma *advanced encryption standard*, kemudian penelitian yang dilakukan oleh penulis bertujuan untuk merancang suatu sistem keamanan (*security system*) yang mana berfungsi untuk melindungi soal ujian supaya data tidak dapat dibaca oleh siswa sebelum pada waktunya, dengan menggunakan algoritma *advanced encryption standard*.

   Berdasarkan penelitian yang dilakukan terdapat hal yang dapat disimpulkan:

   a. Hasil penelitian ini menunjukkan bahwa *cryptography* menggunakan algoritma *advanced encryption standard* dapat memberikan hasil yang maksimal, di mana metode *advanced encryption standard* dapat diterapkan pada sistem penyimpanan data *virtual* sehingga dapat untuk melindungi data yang akan ditransmisi.

2.2. Tinjauan Teoritis

Pada bagian ini akan dijelaskan beberapa uraian mengenai hal-hal yang dipelajari dan dipakai oleh penulis dalam penelitian ini.

2.2.1. *Cryptography* (Kriptografi)

Pada *cryptography* terdapat sistem enkripsi yang mana merupakan proses *vital* dalam mengamankan informasi dari akses yang ilegal, sehingga informasi tersebut hanya dapat diakses oleh pihak tertentu (Risyan, 2020). *Cryptography* merupakan proses dalam mengubah pesan sehingga isi dari pesan tersebut disembunyikan dari pihak yang tidak memiliki hak untuk membaca isi pesan tersebut. Sederhananya, *cryptography* adalah seni ilmu penulisan pesan rahasia (*encipher*) salah satunya dengan melibatkan teknik penggunaan algoritma. *Cryptography* ditujukan supaya hanya pengirim dan penerima pesan saja yang dapat mengetahui isi dari pesan tersebut dengan melakukan proses enkripsi-dekripsi (Risyan, 2020).

Cryptography dapat dimanfaatkan untuk membuat sistem yang aman pada *electronic transaction*, seni ilmu *cryptography* pada umumnya diterapkan pada sistem pengamanan pesan *email*, *detail* kartu kredit, *media* penyimpanan data, dan informasi yang bersifat sensitif lainnya. Dengan diadopsinya ilmu *cryptography*, maka pengirim pesan dapat mengenkripsi isi dari pesan, sehingga pesan aman pada saat diteruskan pada jaringan komputer, di lain sisi penerima pesan dapat mendekripsi isi dari pesan untuk selanjutnya menerima informasi dari pesan tersebut dalam *format* penulisan yang dapat dibaca oleh manusia (Risyan, 2020).

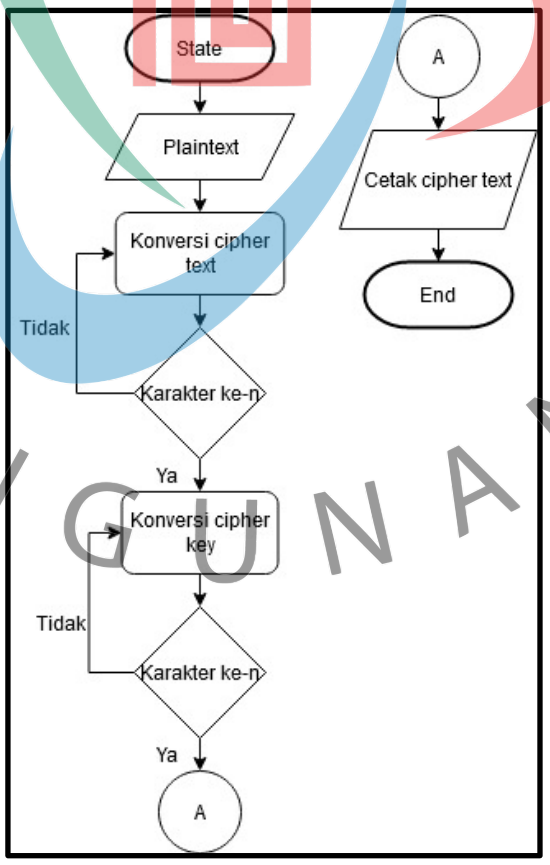
Setiap bisnis terutama yang memiliki keterkaitan dengan menyimpan informasi pribadi pengguna perlu memperhatikan dan memanfaatkan *cryptography* sebagai teknik pengamanan data yang diperlukan pada organisasinya. Salah satu teknik yang dimiliki *cryptographers*, yaitu memastikan sistem keamanan pada *transfer* data pribadi maupun pada saat migrasi data. Secara kolektif, manfaat tersebut membuat perusahaan dapat memiliki kepercayaan penuh dalam melakukan bisnis di *digital era* seperti saat ini (Risyan, 2020). Kesimpulannya bahwa, *cryptography* adalah praktek yang mana untuk mengamankan komunikasi dan data dari hadapan *cybercriminals*, artinya ilmu *cryptography* perlu dikuasai oleh seorang *cyber security*.

2.2.2. Algoritma *Advanced Encryption Standard*

Algoritma *Advanced Encryption Standard* (AES) atau dengan nama lain, yaitu algoritma *rijndael*. Algoritma *AES* sendiri merupakan kebaruan yang telah menggantikan algoritma pendahulunya, yaitu algoritma *Data Encryption Standard* (DES). Algoritma *advanced encryption standard* telah diadopsi, dan dipakai untuk menjadi standar enkripsi di negara *United States of America* (Primartha, 2012).

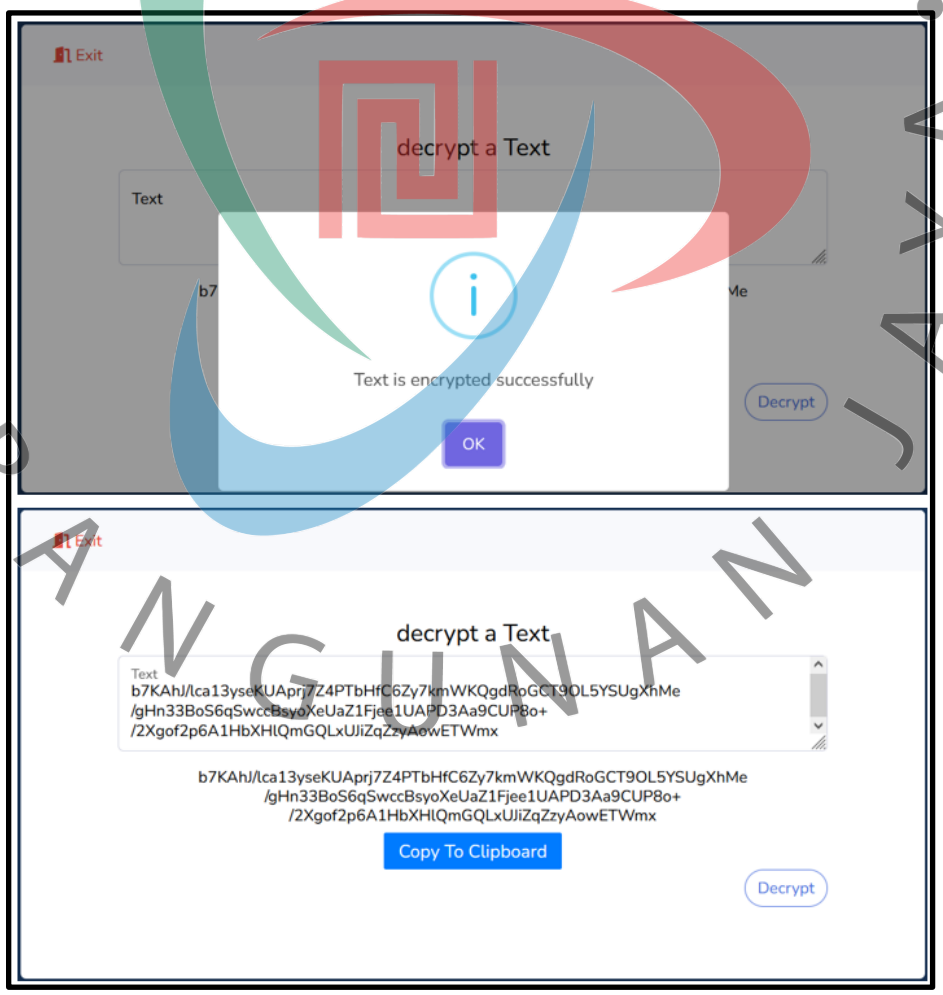
Dengan namanya, yaitu *advanced encryption standard* jika diartikan dalam Bahasa Indonesia berarti algoritma ini merupakan enkripsi mutakhir di mana algoritma ini dianggap susah untuk diretas, karena proses enkripsi (*encipher*) dan dekripsi (*decipher*) membutuhkan yang namanya *cipher key* (Primartha, 2012), berikut merupakan contoh *cipher key* "���a~���ĞF��œ�GVNH�i���>".

Yang mana jika seorang kriminal siber berhasil mengambil suatu informasi secara ilegal dari database, maka kriminal siber tidak akan dapat mendekripsi informasi tersebut, karena pelaku tidak memiliki *cipher key* yang digunakan pada saat proses enkripsi pesan terjadi. Untuk ilustrasi proses enkripsi dapat diperhatikan pada gambar 2.1. ilustrasi proses enkripsi *AES*.



Gambar 2.1. Ilustrasi Proses Enkripsi *AES*

Algoritma *Advanced Encryption Standard* digunakan oleh penulis untuk seni ilmu kriptografi, yang mana kriptografi sendiri merupakan proses untuk mengamankan suatu informasi dengan cara mengenkripsi informasi tersebut, enkripsi merupakan proses merubah kalimat dari bentuk *plaintext* menjadi *ciphertext*. Berikut penulis melakukan *demo* dengan membuat sendiri aplikasi *cryptool* (*cryptography tool*) menggunakan bahasa pemrograman *PHP* dibantu dengan *framework CodeIgniter* 4 untuk tujuan *pentest* (*penetration testing*), pengujian terjadi pada sisi *client* dan *server* dapat diperhatikan pada gambar 2.2. *AES encryption*. Dalam hal ini penulis menggunakan algoritma *advanced encryption standard* 256 *bit* dan algoritma *base64 encoding*, di mana input awal berupa *plaintext*, dan output berupa hasil enkripsi, yaitu *ciphertext* yang telah *diencode* oleh sistem, hasil enkripsi disimpan oleh sistem ke dalam database, dan sistem menampilkan pesan *alert* bahwa pesan telah berhasil dienkripsi.

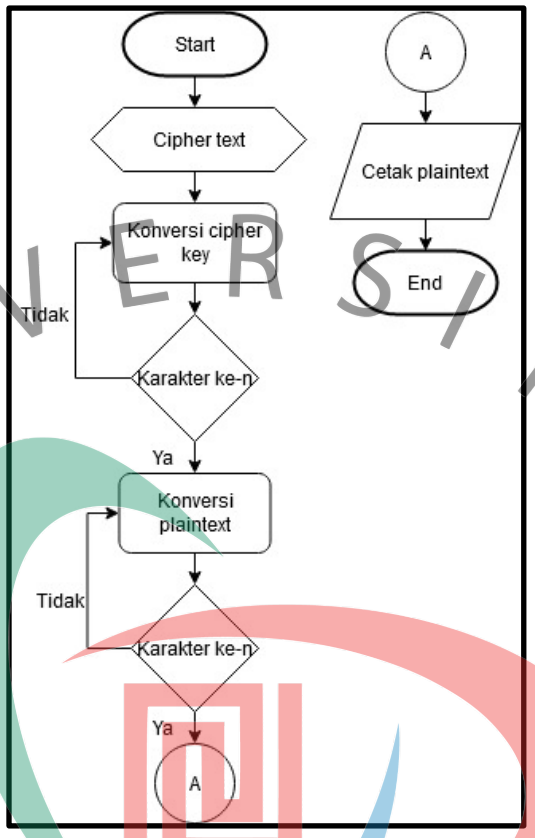


Gambar 2.2. *AES Encryption*

Dekripsi merupakan proses merubah kalimat dari bentuk *ciphertext* menjadi *plaintext*, di mana penulis mencoba melakukan *data breaching* dengan melakukan dekripsi pesan. Namun, hasilnya gagal karena penulis tidak memiliki *cipher key* yang sesuai saat digunakan pada proses enkripsi sebelumnya, dapat diperhatikan pada gambar 2.3. *penetration testing*.

Gambar 2.3. *Penetration Testing*

Untuk ilustrasi proses dekripsi dapat diperhatikan pada gambar 2.4. ilustrasi proses dekripsi *AES*.



Gambar 2.4. Ilustrasi Proses Dekripsi *AES*

Algoritma *advanced encryption standard* memiliki 3 *block cipher*, yaitu *AES-128 bit*, *AES-192 bit*, dan *AES-256 bit* (Primartha, 2012). Pada penelitian ini, penulis memakai *AES-256 bit*. Untuk perbedaan setiap *bit* dapat diperhatikan pada tabel 2.1. *block cipher AES*.

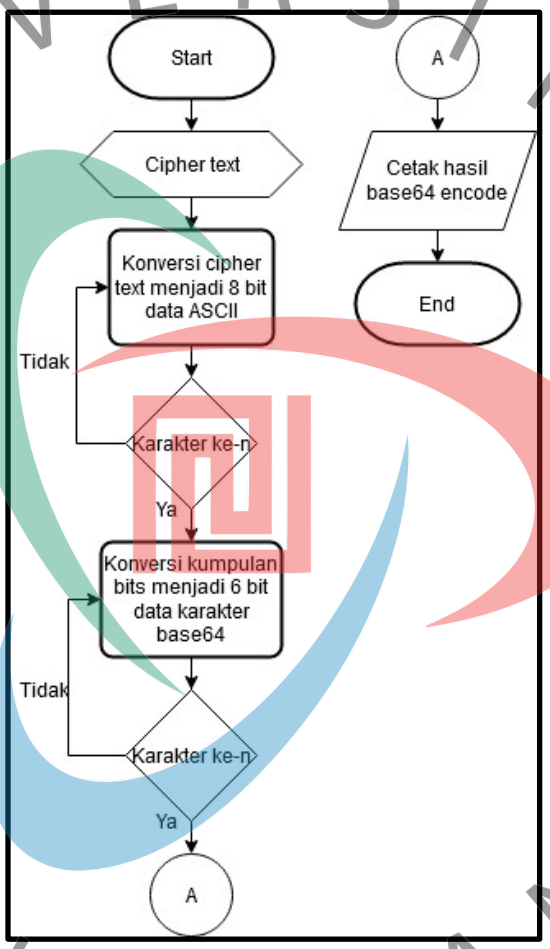
Tabel 2.1. *Block Cipher AES*

	Panjang Kunci	**Ukuran Blok**	**Jumlah Putaran**
AES-128	128 (4)	128 (4)	10
AES-192	192 (6)	128 (4)	12
AES-256	256 (8)	128 (4)	14

2.2.3. Algoritma *Base64*

Algoritma *base64* merupakan algoritma untuk proses *encoding* suatu informasi *plaintext* menjadi *binary text* dalam *format string American standard code for information interchange* (Minarni, 2019), dan untuk proses *decoding* di mana

informasi dalam bentuk *binary text* dikembalikan menjadi *plaintext*. Karakter yang dihasilkan dari transformasi algoritma *base64 encoding* terdiri dari A-Z, a-z, 0-9, dan ditambah dengan dua karakter terakhir yang bersimbol "+" dan "/" serta satu buah simbol "=", berikut merupakan *binary text* dari *base64 encoding* "dQwacZrAuDGW7p/MlMnsbN5SO2BAJXbRtbXv+jkF1Cw=". Untuk ilustrasi proses *encoding* dapat diperhatikan pada gambar 2.5. ilustrasi proses *base64 encoding*.



Gambar 2.5. Ilustrasi Proses *Base64 Encoding*

Berikut penulis akan menunjukkan secara *manual* perhitungan *decimal* dan *bit* pada proses *encoding* dan *decoding*, karena *base64* menggunakan bilangan *decimal* untuk mengubah *plaintext* ke dalam bentuk *binary text* dengan *format American Standard Code for Information Interchange* (ASCII), maka jika penulis akan *mengencode* kalimat "Ardean Reza Angkasa" artinya penulis harus mengikuti *format ASCII*, dapat diperhatikan pada tabel 2.2. *ASCII*.

Tabel 2.2. *ASCII*

Decimal	*Char*	*Decimal*	*Char*	*Decimal*	*Char*	*Decimal*	*Char*
0	[*Null*]	32	[*Space*]	64	@	96	`
1	[*Start of heading*]	33	!	65	A	97	a
2	[*Start of text*]	34	“	66	B	98	b
3	[*End of text*]	35	#	67	C	99	c
4	[*End of transmission*]	36	$	68	D	100	d
5	[*Enquiry*]	37	%	69	E	101	e
6	[*Acknowledge*]	38	&	70	F	102	f
7	[*Bell*]	39	‘	71	G	103	g
8	[*Backspace*]	40	(	72	H	104	h
9	[*Horizontal tab*]	41	)	73	I	105	i
10	[*Line feed*]	42	*	74	J	106	j
11	[*Vertical tab*]	43	+	75	K	107	k
12	[*Form feed*]	44	,	76	L	108	l
13	[*Carriage return*]	45	-	77	M	109	m
14	[*Shift out*]	46	.	78	N	110	n
15	[*Shift in*]	47	/	79	O	111	o
16	[*Data link escape*]	48	0	80	P	112	p
17	[*Device control 1*]	49	1	81	Q	113	q
18	[*Device control 2*]	50	2	82	R	114	r
19	[*Device control 3*]	51	3	83	S	115	s
20	[*Device control 4*]	52	4	84	T	116	t
21	[*Negative acknowledge*]	53	5	85	U	117	u
22	[*Synchronous idle*]	54	6	86	V	118	v
23	[*Eng of trans. block*]	55	7	87	W	119	w
24	[*Cancel*]	56	8	88	X	120	x
25	[*End of medium*]	57	9	89	Y	121	y
26	[*Substitute*]	58	:	90	Z	122	z
27	[*Escape*]	59	;	91	{	123	{
28	[*File separator*]	60	<	92	\	124	|
29	[*Group separator*]	61	=	93	]	125	}
30	[*Record separator*]	62	>	94	^	126	~
31	[*Unit separator*]	63	?	95	_	127	[*Del*]

Setelah menentukan input yang akan *diencode* dengan *format ASCII*, maka selanjutnya penulis menggunakan perhitungan *binary*, dapat diperhatikan pada tabel 2.3. *binary conversion*, di mana dalam perhitungan *binary* terdapat nilai maksimal, yaitu 255 sehingga jika nilai angka yang akan dihitung melebihi 255 maka perhitungan *binary* tidak dapat dilakukan. Pada bilangan *binary* setiap *value* diwakilkan dengan nilai 1 dan nilai 0 jika *value* tidak digunakan.

Tabel 2.3. *Binary Conversion*

128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
1	1	1	1	1	1	1	1

Berikutnya adalah mengubah nilai pada perhitungan bilangan *binary* tersebut ke dalam beberapa blok dengan ketentuan transformasi algoritma *base64*, yaitu 6 *bit*. Dapat diperhatikan pada tabel 2.4. algoritma *base64*. Tabel algoritma *base64* digunakan sebagai dasar pemahaman untuk perubahan nilai *binary*.

Tabel 2.4. Algoritma *Base64*

Value	***Binary***	***Char***	***Value***	***Binary***	***Char***	***Value***	***Binary***	***Char***
0	000000	A	22	010110	W	44	101100	s
1	000001	B	23	010111	X	45	101101	t
2	000010	C	24	011000	Y	46	101110	u
3	000011	D	25	011001	Z	47	101111	v
4	000100	E	26	011010	a	48	110000	w
5	000101	F	27	011011	b	49	110001	x
6	000110	G	28	011100	c	50	110010	y
7	000111	H	29	011101	d	51	110011	z
8	001000	I	30	011110	e	52	110100	0
9	001001	J	31	011111	f	53	110101	1
10	001010	K	32	100000	g	54	110110	2
11	001011	L	33	100001	h	55	110111	3
12	001100	M	34	100010	i	56	111000	4
13	001101	N	35	100011	j	57	111001	5
14	001110	O	36	100100	k	58	111010	6
15	001111	P	37	100101	l	59	111011	7
16	010000	Q	38	100110	m	60	111100	8
17	010001	R	39	100111	n	61	111101	9
18	010010	S	40	101000	o	62	111110	+
19	010011	T	41	101001	p	63	111111	/
20	010100	U	42	101010	q		(*Pad*)	=
21	010101	V	43	101011	r			

Berikut adalah langkah proses *base64 encoding* yang dilakukan oleh penulis secara *manual*:

1. Input

   Pesan: Ardean Reza Angkasa

   ASCII: 65 114 100 101 97 110 32 82 101 122 97 32 65 110 103 107 97 115 97

   Binary 8 *bit*: 01000001 01110010 01100100 01100101 01100001 01101110 00100000 01010010 01100101 01111010 01100001 00100000 01000001 01101110 01100111 01101011 01100001 01110011 01100001

Value base64: 16 23 9 36 25 22 5 46 8 5 9 37 30 38 4 32 16 22 57 39 26 54 5 51 24 16

2. Proses

Binary 6 *bit*: 010000 010111 001001 100100 011001 010110 000101 101110 001000 000101 001001 100101 011110 100110 000100 100000 010000 010110 111001 100111 011010 110110 000101 110011 011000 010000

Hasil *bit* tersebut diubah ke dalam beberapa blok dengan ketentuan transformasi algoritma *Base64*, yaitu 6 *bit*. Dapat diperhatikan pada tabel 2.5. *base64 encoding*.

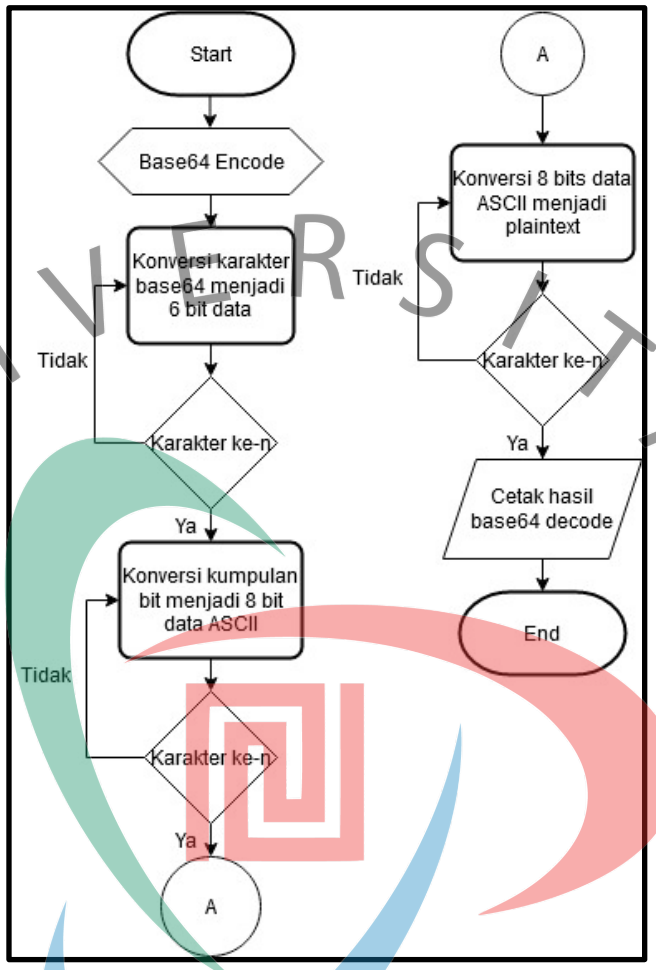
Tabel 2.5. *Base64 Encoding*

Value	***Binary***	***Character***
16	010000	Q
23	010111	X
9	001001	J
36	100100	k
25	011001	Z
22	010110	W
5	000101	F
46	101110	u
8	001000	I
5	000101	F
9	001001	J
37	100101	l
30	011110	e
38	100110	m
4	000100	E
32	100000	g
16	010000	Q
22	010110	W
57	111001	5
39	100111	n
26	011010	a
54	110110	2
5	000101	F
51	110011	z
24	011000	Y
16	010000	Q

1. Output

Dari tabel 2.5. *base64 encoding*, didapatkan hasil jika pesan “Ardean Reza Angkasa” *diencoding* dengan algoritma *base64* maka menjadi “QXJkZWFuIFJlemEgQW5na2FzYQ”.

Untuk ilustrasi proses *decoding* dapat diperhatikan pada gambar 2.6. ilustrasi proses *base64 decoding*.



Gambar 2.6. Ilustrasi Proses *Base64 Decoding*

Berikut adalah langkah proses *base64 decoding* yang dilakukan oleh penulis secara *manual*, untuk proses *decoding* maka hasil dari *encoding* diubah ke dalam nilai *value* berdasarkan tabel 2.4. algoritma *base64*, dan dilanjutkan dengan mengubah ke bentuk *binary* 6 *bit* yang mana kemudian diubah menjadi *binary* 8 *bit*.

1. Input

   Pesan *encoding*: QXJkZWFuIFJlemEgQW5na2FzYQ

   Value base64: 16 23 9 36 25 22 5 46 8 5 9 37 30 38 4 32 16 22 57 39 26 54 5 51 24 16

2. Proses

   Binary 6 *bit*: 010000 010111 001001 100100 011001 010110 000101 101110 001000 000101 001001 100101 011110 100110 000100 100000

010000 010110 111001 100111 011010 110110 000101 110011 011000 010000

Binary 6 *bit*, diubah menjadi *binary* 8 *bit*

Binary 8 *bit*: 01000001 01110010 01100100 01100101 01100001 01101110 00100000 01010010 01100101 01111010 01100001 00100000 01000001 01101110 01100111 01101011 01100001 01110011 01100001

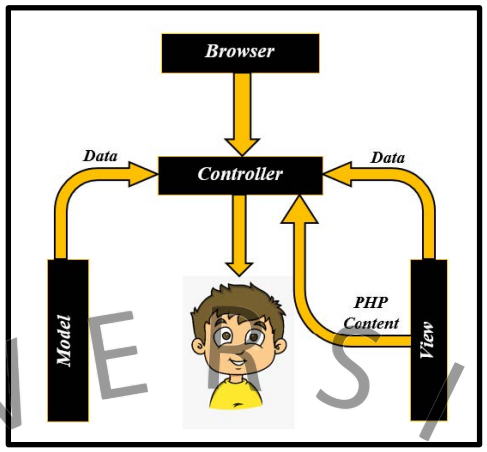
3. Output

   Dari proses *decoding* didapatkan hasil jika pesan *encoding* "QXJkZWFuIFJlemEgQW5na2FzYQ" *didecoding* dengan algoritma *base64* maka menjadi "Ardean Reza Angkasa".

2.2.4. *Framework Codeigniter*

Codeigniter merupakan suatu *framework* yang pemrogramannya menggunakan bahasa pemrograman *PHP* dan *object-oriented programming system*, *codeigniter* bersifat *open source* artinya gratis, dapat dimodifikasi isi dari programnya oleh *developer* sesuka hati, dan *easy to use*. *Developer* memakai *codeigniter* untuk membuat *web* yang dinamis, dan *codeigniter* sendiri mengikuti pola *MVC*, artinya *codeigniter* terstruktur karena kode-kodenya terbagi ketiga bagian, yaitu: (Babastudio, 2020)

1. M => *Model*, berguna untuk menyimpan *file PHP* yang telah Anda buat untuk mengkoneksikan *class* ke tabel yang Anda inginkan pada suatu database, sehingga *class* pada *file* tersebut dapat Anda pakai untuk mengambil data dari tabel yang terdapat pada database.
2. V => *View*, merupakan tempat di mana Anda menyimpan *file PHP* yang bertujuan untuk menampilkan tampilan *web* Anda, jadi untuk *file* pemrograman dari desain *webnya*, Anda simpan dalam *folder view*.
3. C => *Controller*, berguna untuk menampilkan hasil pemrograman yang anda buat pada *view* dan *model*, juga berguna untuk berpindah halaman pada suatu *website*, jadi *file* berisi kode program yang anda buat dan simpan di *View* dan *Model* akan dihubungkan dan dioperasikan di *Controller*.



Gambar 2.7. Skema Tahapan *MVC*

Framework memudahkan *programmer* dalam membuat suatu *web* yang dinamis karena didalamnya terdapat banyak *libraries* yang dapat digunakan dan dimanfaatkan oleh *programmer* untuk membuat halaman *website*. Oleh karena itu, suatu *framework* digemari karena *easy to use* sehingga *programmers* dapat mengoperasikan program-program secara lebih baik dan sesuai apa yang dikehendaki.

CodeIgniter ditemukan dan dikembangkan pertama kali oleh Rick Ellis di tahun 2006 dan api sebagai logo memaknai *codeigniter* dengan cepat dapat membakar dan menyalakan api semangat dari *web developers* dalam merancang program *PHP* untuk membangun aplikasi *web* yang responsif dan dinamis (Babastudio, 2020).

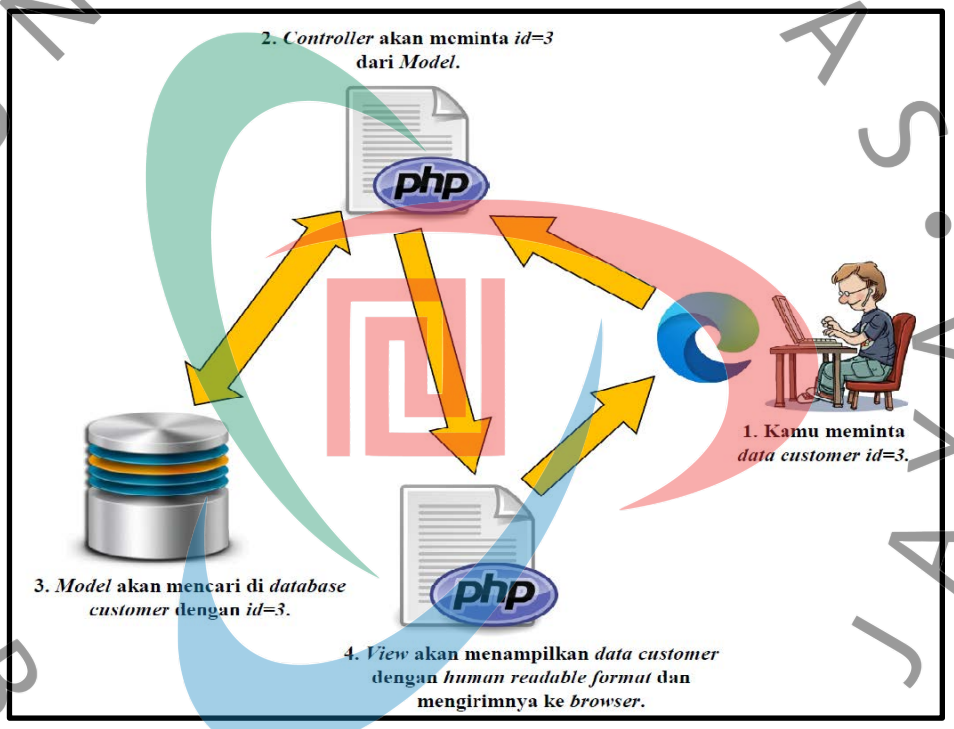
Fitur-fitur *codeigniter*: (Babastudio, 2020)

1. *Framework* berbasis *Model-View-Controller* (MVC)
2. *Support* terhadap mayoritas jenis database
3. Mendukung *query builder*
4. Bersifat *independent*
5. Validasi *form* dan data
6. *Session management*
7. Mengamankan *website* dari *SQL injection attack*, dan *cross site scripting*
8. *File uploading class*

Codeigniter adalah salah satu *framework* yang menerapkan pola *Model-View-Controller* (MVC). Dalam hal ini pengguna dapat mencari data pada suatu tabel yang tersedia dalam database, *Controller* akan merespon permintaan tersebut,

berikutnya *Controller* akan meminta *Model* untuk mengambil informasi yang tersedia pada tabel dan menampilkan (*View*) data tersebut apabila tersedia (Babastudio, 2020).

Sebagai contoh, ketika Anda mau mencari data *id=3* pada suatu tabel, maka *Controller* akan merespon permintaan Anda, kemudian *Controller* akan meminta *Model* mengakses tabel tersebut untuk mencari *id=3*, apabila tersedia maka *Model* akan mengirimkan data *id=3* ke *Controller*, berikutnya *Controller* akan memproses data *id=3* pada *View*, dalam *View* data akan diolah ke dalam bentuk *human-readable*, dan hasilnya ditampilkan pada *web browser* Anda (Babastudio, 2020).



Gambar 2.8. Cara Kerja *MVC* pada *CodeIgniter*

2.2.5. Database

Database merupakan layanan yang dipakai untuk menyimpan data berupa *plaintext* pada sistem. Data yang disimpan pada database akan membentuk suatu kumpulan informasi, yang mana informasi ini dapat diakses, dan diolah menggunakan suatu program komputer untuk memperoleh informasi data dari database tersebut. Database menjadi representasi mengenai kumpulan informasi yang saling berhubungan dan dikelola menjadi suatu data yang disimpan pada

komputer dengan sedemikian rupa sehingga data tersebut dapat memenuhi berbagai kebutuhan dalam menyimpan dan menyajikan informasi (Setiawan, 2021).

Perangkat lunak yang dipakai dalam database untuk mengelola data dan memanggil *query* dari kumpulan data yang terdapat pada database disebut dengan *Database Management System* (DBMS). Istilah database sendiri berawal dari ilmu komputer, walaupun sebenarnya memiliki arti yang lebih luas lagi di mana database memiliki persamaan dengan bidang-bidang di luar ilmu komputer, kesamaan pada database terdapat pada masa sebelum revolusi industri, berupa buku kas (*cash book*), kuitansi atau bukti penerimaan pembayaran, dan kumpulan data yang memiliki hubungan dengan akuntansi (Setiawan, 2021).

Pada database *system* terdapat beberapa hal untuk dibahas, yaitu: (Setiawan, 2021)

1. Perangkat Keras (*Hardware*)

   Perangkat keras pada umumnya dalam database *system* adalah memori sekunder, seperti *hardisk* eksternal.
2. Sistem Operasi (*Operating System*)

   Sistem operasi pada database *system* adalah perangkat lunak yang dapat dipakai dalam mengoperasikan program komputer untuk mengolah database, seperti *MySQL* dengan perangkat lunak adalah *HeidiSQL*.
3. Basis data (Database)

   Dalam basis data terdapat kumpulan tabel, yang mana pada setiap tabel memiliki data dan informasi yang pada umumnya bersifat rahasia (*confidential*).
4. *Database Management System* (DBMS)

   Sistem manajemen pada basis data merupakan pengolahan yang dikerjakan oleh sistem pada perangkat lunak database *system*, sehingga dalam hal ini pengolahan basis data tidak diolah oleh pemakai secara langsung.
5. Pemakai (*User*)

   Pemakai melakukan interaksi pada basis data dengan memanipulasi data yang terdapat pada kumpulan tabel dalam basis data, hal ini dapat diwujudkan dengan menggunakan suatu bahasa pemrograman, seperti *MySQL*.

2.2.6. Bahasa Pemrograman *PHP*

PHP merupakan bahasa *server-side scripting*, bahasa pemrograman *PHP* digunakan untuk merancang baik *web* statis maupun *web* dinamis dengan tujuan untuk membangun suatu aplikasi berbasis *web*. *PHP* adalah singkatan dari *hypertext preprocesso*r, sebelum itu disebut dengan *personal home page* (Beon Intermedia, 2019).

Script sendiri merupakan sekumpulan intruksi pemrograman yang diuraikan saat terjadinya *runtime*, sedangkan bahasa *scripting* merupakan bahasa yang dapat menguraikan atau mengolah suatu skrip ketika terjadinya *runtime* di mana pada umumnya hal ini dilakukan dalam lingkungan perangkat lunak. Karena *PHP* merupakan *server-side scripting* maka programnya (*script*) akan dijalankan dan diproses oleh *server* (Beon Intermedia, 2019).

Terdapat bahasa pemrograman lainnya yang dapat digunakan untuk membangun dan merancang suatu *web* bukan cuman bahasa pemrograman *PHP*. Namun, *PHP* digunakan supaya dapat membuat *website* yang dinamis karena bahasa pemrograman *PHP* dapat digunakan untuk menyimpan *data* ke dalam database, membuat grafik, sistem *e-voting*, memproses *online form*, dan masih banyak lagi.

Pada bahasa pemrograman *PHP* terdapat *PHP syntax*, yang merupakan aturan penulisan dalam membuat suatu program *PHP* supaya *compiler* mampu mengerti dengan benar saat mencoba membaca bahasa pemrograman tersebut. Dalam penulisan *PHP* yang benar maka harus diawali dengan *"<? php"* dan diakhiri dengan *"?>"* (Beon Intermedia, 2019).

Contoh penulisan *PHP syntax*:

```
<?php
        echo "Angkasa's Pride";
?>
```

Dari kode program diatas maka akan menghasilkan output, *Angkasa's Pride*.

2.2.7. Aplikasi Berbasis *Web*

Aplikasi berbasis *web* merupakan perangkat lunak yang dapat diakses melalui *web browser* dengan menggunakan koneksi *internet* yang aktif (Elisa, 2016). Pada perkembangan awal, aplikasi *web* yang dirancang dan dibangun sama sekali tidak

menggunakan bahasa pemrograman komputer melainkan hanya bahasa markah, yaitu *Hyper Text Markup Language* (HTML). Pada perkembangannya, sejumlah skrip telah dikembangkan untuk dapat memperluas kemampuan dari bahasa pemrograman *HTML,* seperti *PHP*, *CSS*, dan *JavaScript* (Elisa, 2016).

Beberapa keunggulan yang diperoleh oleh pengguna aplikasi berbasis *web,* yaitu: (Elisa, 2016)

1. Aplikasi berbasis *web* dapat diakses dari mana saja dan kapan saja tanpa harus melakukan *penginstalan* aplikasi pada *device*, hal ini karena aplikasi *web* telah terpasang di *web server*.
2. *Multiplatform*, artinya aplikasi berbasis *web* dapat digunakan pada berbagai macam sistem operasi baik *Linux*, *Windows*, ataupun *Mac OS*, hal ini karena yang dibutuhkan suatu *device* dalam mengakses aplikasi berbasis *web* merupakan *web browser* dan koneksi *internet* yang aktif.
3. Terkait isu lisensi (hak cipta), dalam hal ini menjadi tanggung jawab dari penyedia aplikasi *web* sehingga pengguna tidak harus terbebani dengan isu lisensi.
4. Dapat diakses melalui banyak *device* seperti: komputer, *tab*, dan *handphone* dengan standar *Wireless Application Protocol* (WAP).

Aplikasi *web* terbagi ke dalam dua jenis model, berupa *web* statis dan *web* dinamis. Aplikasi *web* statis dirancang dan dibangun menggunakan *HTML* dalam hal ini kekurangan dari *web* statis adalah keharusan dari *front end developers* sebagai pengembang aplikasi dalam melakukan pemeliharaan (*maintenance*) aplikasi *web* secara terus menerus (*continuously*), sehingga program markah yang dirancang dapat ditujukan untuk mengikuti perkembangan yang sedang terjadi. Namun, dalam hal ini kelemahan tersebut dapat diatasi dengan membangun aplikasi dengan model *web* dinamis (Elisa, 2016).

Perubahan informasi pada halaman dari aplikasi *web* dinamis dapat dilakukan tanpa harus mengubah program komputer atau membuat program komputer pada skrip, dalam hal ini perubahan informasi dapat dilakukan secara langsung melalui aplikasi *web*. Sebagai penerapannya, aplikasi *web* dikoneksikan ke database terlebih dahulu sehingga perubahan informasi dapat dilakukan langsung oleh *operator*

maupun *user*, dalam hal ini maka perubahan tersebut tidak lagi menjadi tanggung jawab oleh *webmaster* (Elisa, 2016).

Arsitektur yang terbagi pada aplikasi *web* dinamis, berupa *client*, *web server*, *middleware*, dan database. *Client* melakukan interaksi terhadap *web server* dengan melalui perangkat komputer. Dalam lingkungan perangkat lunak maka *web server* berkomunikasi dengan *middleware*, di mana selanjutnya *middleware* berkomunikasi dengan database *system*, sehingga data yang tersimpan dalam basis data dapat ditampilkan di *web* dinamis (Elisa, 2016).

2.2.8. *Networking Protocol*

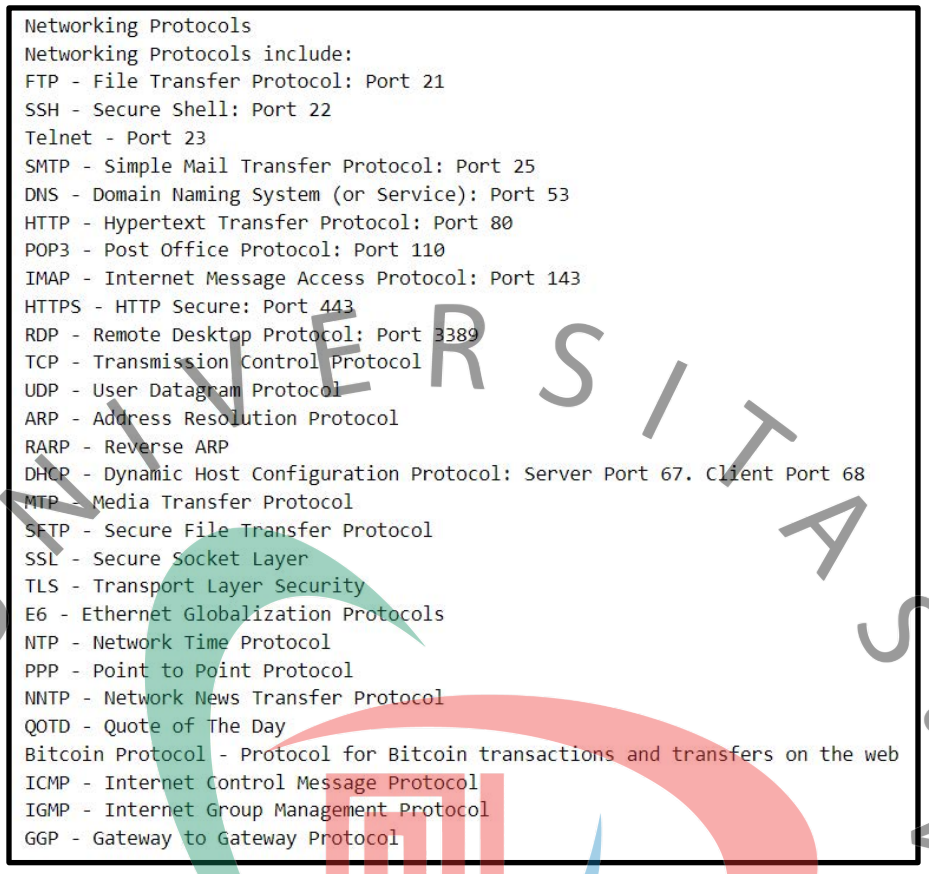
Networking protocol merupakan bagian terpenting dalam proses pertukaran informasi (*information interchange*) antara komputer, teknologi dari *networking protocol* dapat diterapkan pada perangkat lunak dan perangkat keras. *Networking protocol* memiliki peran penting pada aplikasi *web*, hal ini karena berhubungan secara langsung terhadap perkembangan Teknologi dan Informasi (TI), dalam hal ini maka tidak terlepas dari aktivitas yang terjadi di *internet* (Yasin, 2018), maka dengan hal tersebut penulis dapat memanfaatkan *networking protocol* untuk tujuan *web security* dan *network security*.

Secara sederhana, *networking protocol* berupa prosedur yang dipakai untuk menghubungkan aktivitas antara pengirim dengan penerima pesan yang terjadi di jaringan *internet*. *Networking protocol* sendiri diterapkan ke dalam dua jenis perangkat, yaitu perangkat keras dan lunak. Komunikasi yang terjadi di dunia *computer network* hampir semua melibatkan *networking protocol* (Yasin, 2018). Untuk contoh seperti saat mengirimkan *email*, dapat diperhatikan kode program pada gambar 2.9. penggunaan *networking protocol* untuk mengirim *email*. *Email* dapat disebut sebagai sebuah informasi yang berisikan informasi sensitif di mana isi dari pesan hanya boleh diketahui oleh pihak tertentu, sehingga *email* yang dikirimkan kepada penerima dari komputer satu ke komputer lain sebenarnya adalah pengiriman data yang telah melalui beberapa *networking protocol*, seperti enkripsi *Transport Layer Security* (TLS). Proses pada *networking protocol* harus dilalui supaya *email* dapat dikirim dan diterima oleh komputer lain pada *networking protocol* yang sama maupun berbeda (Yasin, 2018).

```php
$developmentMode=true;
$mailer = new PHPMailer($developmentMode);
try {
    $mailer->SMTPDebug = 2;
    $mailer->isSMTP();
    if ($developmentMode){
        $mailer->SMTPOptions=[
            'ssl'=>[
                'verify_peer' => false,
                'verify_peer_name' => false,
                'allow_self_signed' => true
            ]
        ];
    }
$dear= $this->request->getPost('username');
$to = $this->request->getPost('email');
$subject = "Message Subject";
$message = 'Information';
$mailer->SMTPAuth = true;
$mailer->Username = '[redacted]';
$mailer->Password   = '[redacted]';
$mailer->SMTPSecure = 'tls';
$mailer->Port       = 587;
$mailer->setFrom('rezaangkasa@outlook.com', 'SIPS');
$mailer->addAddress($to);
$mailer->isHTML(true);
$mailer->Subject = $subject;
$mailer->Body = $message;
$mailer->send();
$mailer->ClearAllRecipients();
return true;
```

Gambar 2.9. Penggunaan *Networking Protocol* untuk Mengirim *Email*

Pada jaringan komputer di seluruh dunia, terdapat beberapa jenis *networking protocol* yang dapat digunakan untuk tujuannya masing-masig. Beberapa jenis *networking protocol* dapat diperhatikan pada gambar 2.10. *networking protocols*. Karena proses pertukaran informasi antara komputer melibatkan *networking protocol*, maka hal tersebut membuat protokol jaringan sebagai bagian terpenting yang mana perlu diketahui dan dipelajari untuk dapat mengikuti perkembangan sistem informasi dan komunikasi maupun teknologi dan informasi, hal-hal tersebut tidak lepas dari peran perkembangan standar protokol jaringan (Yasin, 2018).



Gambar 2.10. *Networking Protocols*

Networking protocol merupakan suatu aturan dalam memudahkan proses interaksi antara sistem dengan sistem lainnya, di mana *networking protocol* sendiri memiliki kegunaan dalam membantu proses pada pengiriman dan penerimaan data dalam aktivitas yang terjadi di *internet*. Apabila *networking protocols standard* tidak diterapkan dalam dunia jaringan, hal ini akan mengakibatkan proses pada pengiriman dan penerimaan data menjadi tidak mungkin dan sulit untuk dilakukan (Yasin, 2018).

2.2.9. *Laragon*

Laragon dipakai sebagai *universal development environment* oleh *programmers* untuk *PHP*, *Python*, *Java*, dan masih banyak lagi. *Laragon* bersifat *portable* dapat dipakai di mana saja, terisolasi, *easy to use*, fleksibel, sistem operasi pada laragon bekerja cepat, *software* termasuk aplikasi ringan tidak membutuhkan spesifikasi tinggi untuk dapat menjalankan aplikasi *laragon* (Fujimaru, 2018). *Laragon*

memiliki *pretty URLs* sehingga *developer* dapat mengakses *projectnya* degan menggunakan *pretty URLs* seperti "*app.test*" tanpa perlu menggunakan lagi perintah "*localhost/app*", aplikasi *laragon* memiliki *portable folder* sehingga dapat dipindah-pindah tanpa perlu merusak sistemnya, *laragon* memiliki sistem operasi *isolated* artinya terisolasi dari sistem operasi yang terdapat pada komputer lokal, sehingga apapun yang dilakukan dengan *laragon* tidak akan mempengaruhi komputer lokal (Fujimaru, 2018). *Laragon* memiliki sistem pengoperasian yang mudah karena *laragon* sudah mengotomatiskan banyak konfigurasi yang kompleks sehingga *developer* hanya tinggal pakai saja tanpa perlu melakukan konfigurasi yang kompleks, seperti *developer* dapat dengan mudah menambah versi baru dari *PHP*, *Apache*, *MySQL*, *XAMPP*, dan masih banyak lagi, tanpa mengalami kesulitan dan tanpa merusak sistem operasi yang sudah ada (Fujimaru, 2018).

Laragon memiliki arsitektur *modern* dan *powerful*, di mana *laragon* sendiri patut untuk dipakai dalam mengembangkan aplikasi *web* yang *modern*, karena *developer* dapat memakai *Apache* dan *Nginx* sekaligus (Fujimaru, 2018). Penulis menggunakan *Laragon* v5.0.0 *full edition* ditujukan untuk *localhost* pada aplikasi *web* dinamis yang mana dalam *computer networking* suatu *localhost* digunakan untuk mengakses *network services*. Berikut merupakan data dari *localhost* yang dimiliki oleh penulis, dapat diperhatikan pada gambar 2.11. data *localhost* aplikasi *laragon*. Dalam hal ini penulis mengandalkan *Apache* v2.4.53 sebagai *HTTP server* yang dibutuhkan oleh penulis dalam menjalankan aplikasi *web* dinamis pada *web server*. Hal ini karena bahasa pemrograman *PHP* merupakan *server-side scripting*, artinya membutuhkan *server* untuk memproses dan menjalankan skrip program (Beon Intermedia, 2019).

Gambar 2.11. Data *Localhost* Aplikasi *Laragon*

2.2.10. *Microsoft Visual Studi Code*

Microsoft visual studio code adalah aplikasi *cross platform* artinya dapat digunakan di berbagai sitem operasi, seperti *Windows*, *Linux*, dan *Mac OS* (Tasari, 2021). *Microsoft visual studio code* merupakan *software* untuk *code editor* di mana *programmer* dapat menggunakan *microsoft visual studio code* untuk berbagai macam bahasa pemrograman, seperti *Java*, *JavaScript*, *Go*, *Python*, *PHP*, dan lain-lainnya. *Microsoft visual studio code* memiliki lintas *platform code editor* yang ringan, mudah untuk mengelola *extention*, memiliki banyak *extention* yang dapat langsung *diinstal*, mudah digunakan oleh *programmer* (*user friendly*) dalam menulis skrip terutama dalam meracang aplikasi *web* baik statis maupun dinamis (Tasari, 2021).