BAB II

LANDASAN TEORI

2.1 Landasan Teori

2.1.1 *Uniform Resource Locator*

Uniform Resource Locator yang disingkat jadi URL adalah alamat yang digunakan untuk mengidentifikasi dan mengakses sumber daya di internet seperti halaman web, gambar, atau dokumen dan sumber daya *online* lainnya. URL sering kali dimulai dengan "http://" atau "https://" dan diikuti dengan nama domain seperti www.example.com. URL memiliki peran yang sangat penting dalam operasional web, yaitu sebagai alamat untuk mengirimkan *request* ke server sehingga terjadi komunikasi *client-server* lalu diperolehlah data dari server (Mukhlis, 2023).

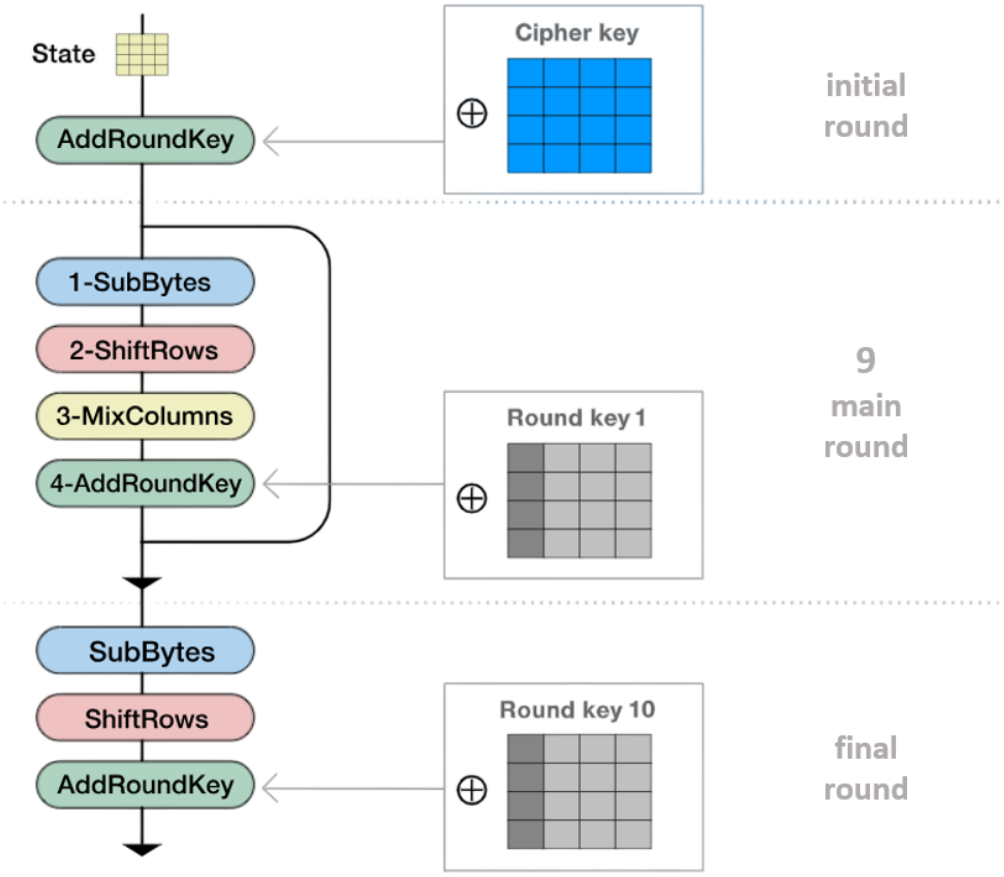
URL merupakan tempat utama untuk *method* GET, sehingga kurang cocok untuk mengirimkan data rahasia karena data yang dikirim melalui GET terlihat langsung di URL. URL memiliki batas panjang maksimum yang bervariasi antar *browser*, sehingga membatasi jumlah data yang dapat dikirim. URL hanya dapat menggunakan tipe data *string*, sehingga tipe data apa pun yang dikirim melalui URL akan dikonversi menjadi *string* (Cholissodin et al., 2023).

2.1.2 *Advanced Encryption Standard*

Advanced Encryption Standard yang sering disingkat jadi AES adalah standar enkripsi baru yang dirancang untuk mengatasi keterbatasan dan kelemahan keamanan pada algoritma *Data Encryption Standar*d (DES). Januari 1997, *National Institute of Standards and Technology* (NIST) mengadakan kompetisi terbuka untuk menciptakan standar algoritma kriptografi baru yang dinamakan AES sebagai

pengganti DES. NIST menerima 15 proposal algoritma dari berbagai negara dan mengadakan konferensi umum pada tahun 1998 dan 1999 untuk mengevaluasi algoritma yang diusulkan, hasilnya NIST memilih 5 finalis terbaik. Konferensi ketiga diadakan pada bulan April 2000, kemudian Oktober 2000 NIST mengumumkan bahwa algoritma *Rijndael* terpilih sebagai pemenang dan diadopsi sebagai standar AES. (Katz & Lindell, 2020).

Rijndael adalah algoritma kriptografi simetris yang diciptakan oleh *Vincent Rijmen* dan *Joan Daemen* di Belgia. Algoritma ini menggunakan metode substitusi dan permutasi, serta sejumlah putaran (*cipher* berulang) di mana setiap putaran menggunakan kunci internal yang berbeda (*round key*). *Rijndael* mendukung panjang kunci 128 bit hingga 256 bit dengan kelipatan 32 bit. Namun panjang kunci yang ditetapkan oleh AES adalah 128, 192, dan 256 bit, sehingga dikenal sebagai AES-128, AES-192, dan AES-256 masing-masing dengan 10, 12, dan 14 putaran. (Daemen & Rijmen, 2020). Proses enkripsi AES dengan kunci 128 bit dapat dilihat pada Gambar 2.1.



Gambar 2. 1 Tahapan Enkripsi Algoritma AES 128 (Berger, 2019)

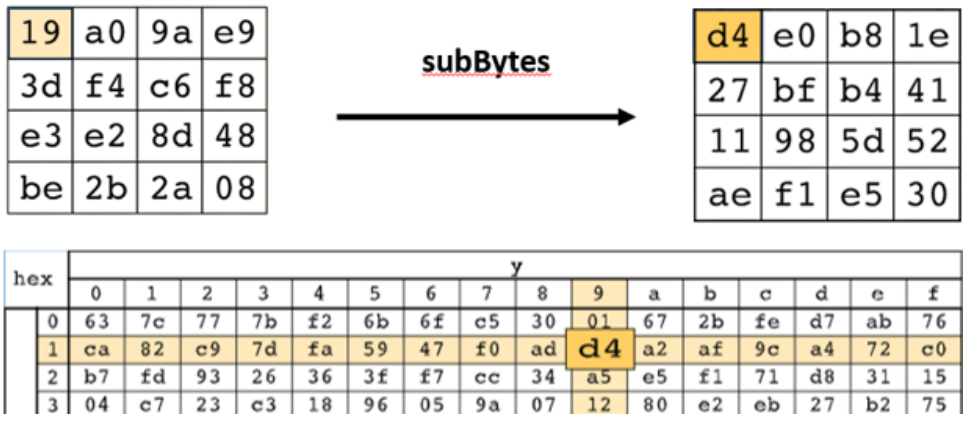
Gambar 2.1 memperlihatkan bahwa *plaintext* direpresentasikan dalam *state* berupa matriks berukuran 16 *byte* (128 bit) dengan format heksadesimal, demikian pula dengan *cipher key*. Terdapat beberapa transformasi penting dalam tahapan enkripsi algoritma AES, yaitu SubBytes, ShiftRows, MixColumns, dan AddRoundKey.

a. *SubBytes*, melakukan substitusi *byte* menggunakan tabel substitusi (S-box) yang ditunjukkan pada Gambar 2.2.

	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

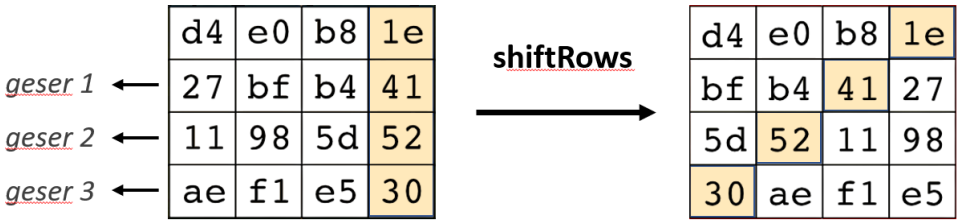
Gambar 2. 2 Tabel S-Box (Andriyanto et al., 2020)

Proses *SubBytes* menggunakan karakter pertama *byte* sebagai indeks baris dan karakter kedua sebagai indeks kolom. Nilai pada pertemuan baris dan kolom inilah yang menjadi hasil substitusi, seperti pada Gambar 2.3.



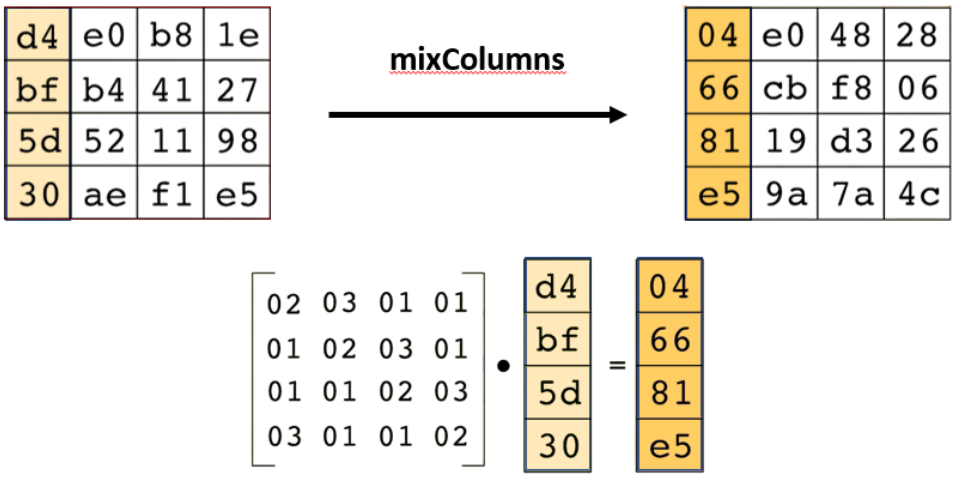
Gambar 2. 3 Transformasi *subBytes* (Berger, 2019)

b. *ShiftRows*, menggeser baris dalam *state* ke kiri dengan jumlah posisi pergeseran yang telah ditentukan. Baris pertama tidak mengalami pergeseran, sementara baris kedua digeser satu posisi ke kiri, baris ketiga digeser dua posisi, dan baris keempat digeser tiga posisi. Operasi *ShiftRows* ditunjukkan pada Gambar 2.4



Gambar 2. 4 Transformasi *shiftRows* (Berger, 2019)

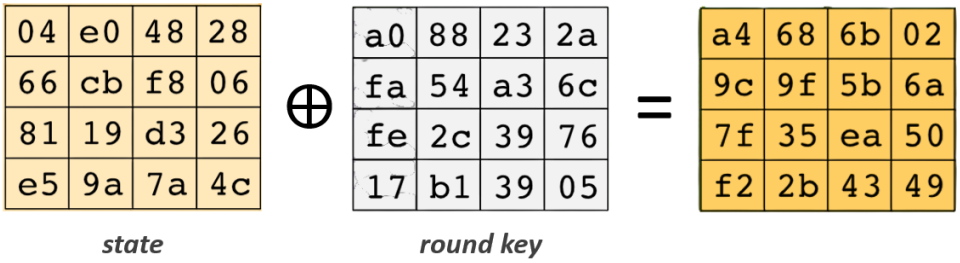
c. *MixColumns*, mengalikan setiap kolom dalam *state* dengan matriks yang telah ditentukan, sebagaimana ditunjukkan pada Gambar 2.5.



Gambar 2. 5 Transformasi *mixColumns* (Berger, 2019)

d. *AddRoundKey*, melakukan operasi XOR antara setiap *byte* dalam *state* dengan *byte* yang bersesuaian pada *round key*. *Round key* dihasilkan melalui

proses ekspansi kunci dari *Round key* pada putaran sebelum nya. Operasi *addRoundKey* diperlihatkan pada Gambar 2.6.

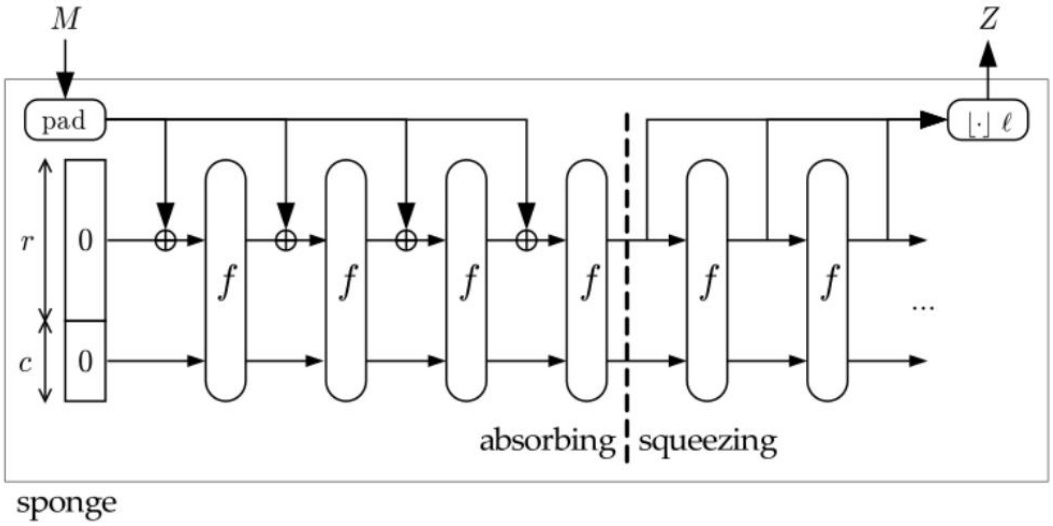


Gambar 2. 6 Transformasi *addRoundKey* (Berger, 2019)

Keuntungan menggunakan AES adalah aman, cepat, dan fleksibel. AES jauh lebih cepat dibandingkan dengan DES. Opsi *multiple key length* adalah keuntungan terbesar karena semakin panjang *key* nya, semakin sulit untuk memecahkannya. (Arizal & Sidabutar, 2022).

2.1.3 *Secure Hash Algorithm 3*

Secure Hash Algorithm 3 sering disingkat jadi SHA-3 merupakan standar terbaru dalam keluarga *Secure Hash Algorithm*. Algoritma ini sering disebut sebagai algoritma Keccak, yang merupakan pemenang dalam kompetisi SHA-3. Keccak diciptakan oleh *Michael Peeters*, *Guido Bertoni*, *Joan Daemen*, *Ronny Van Keer*, dan *Gilles Van Assche*, serta pertama kali diperkenalkan pada tahun 2016. Alih-alih bergantung pada fungsi kompresi seperti finalis SHA-3 lain nya, Keccak menggunakan fungsi non-kompresi yaitu konstruksi *sponge* yang terdiri dari dua fase yakni fase penyerapan (*absorbing*) dan fase pemerasan (*squeezing*) (Raihan Aulia, 2023). Konstruksi *sponge* dapat dilihat pada Gambar 2.7.

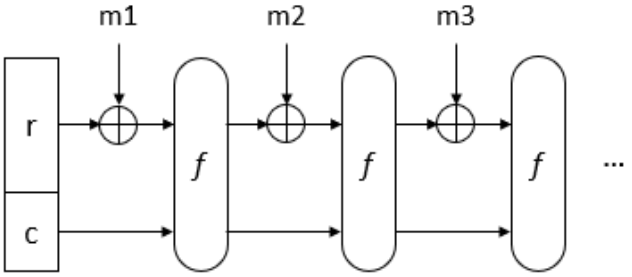


Gambar 2. 7 Konstruksi *sponge* algoritma keccak SHA-3 (Raihan Aulia, 2023)

Berdasarkan Gambar 2.7 proses *hashing* diawali dengan tahap praproses yang disebut *padding*, yaitu penambahan bit penggenap pada pesan *M* sehingga panjangnya habis dibagi dengan *r*. *Rate (r) + capacity (c)* membentuk *state (S)* yang digunakan untuk menangkap pesan serta dipersiapkan untuk diproses pada tahap-tahap selanjutnya.

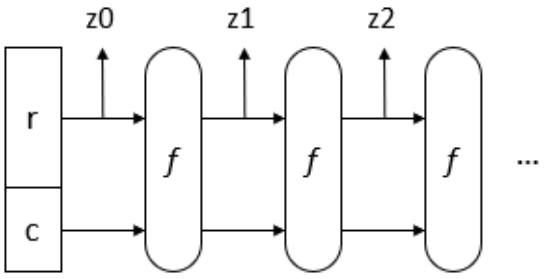
Pesan *M* yang telah melalui proses *padding* kemudian dibagi menjadi beberapa blok, yaitu *m1*, *m2*, *m3*, dan seterusnya hingga pesan M habis, yang mana panjang blok-blok tersebut harus sama besar dengan panjang *r*. Blok pertama (*m1*) dioperasikan dengan XOR terhadap *r* bit pertama dari *state S*. Hasil operasi ini menjadi masukan bagi fungsi permutasi, yang menghasilkan *state* baru (*S'*). Proses serupa diterapkan pada blok berikutnya, di mana *m2* dioperasikan dengan XOR terhadap *r* bit pertama dari state *S'* dan kembali diproses melalui fungsi permutasi untuk membentuk *state* berikutnya. Tahap ini disebut tahap penyerapan (*absorbing*)

yang berlangsung hingga seluruh blok pesan diserap oleh konstruksi *sponge*. Proses penyerapan pada konstruksi *sponge* dapat dilihat pada Gambar 2.8.



Gambar 2. 8 Tahap penyerapan pada konstruksi *sponge* (Raihan Aulia, 2023)

Setelah seluruh blok pesan telah diserap dan melalui proses permutasi, *r* bit pertama dari *state* hasil permutasi diambil sehingga menghasilkan blok *hash z0*. Blok *z0* kemudian ditambahkan ke dalam hasil *hash* (*Z*). Jika Z belum mencapai panjang yang diinginkan, maka *state* diproses kembali menggunakan fungsi permutasi menghasilkan *state* baru dan *r* bit pertama kembali diekstraksi sebagai *z1* lalu ditambahkan ke dalam *Z*. Proses ini berulang hingga panjang *hash* mencapai ukuran yang diinginkan. Tahap ini lah yang disebut tahap pemerasan (*squeezing*) yang ditunjukkan pada Gambar 2.9.



Gambar 2. 9 Tahap pemerasan pada konstruksi *sponge* (Raihan Aulia, 2023)

Varian SHA-3 meliputi SHA-3:224, SHA-3:256, SHA-3:348, dan SHA-3:512, yang menghasilkan *message digest* masing-masing sebesar 224, 256, 348,

dan 512 bit, dengan menggunakan ukuran blok masing-masing sebesar 1152, 1088, 832, dan 576 bit (Khan et al., 2022).

2.1.4 Base64

Base64 adalah salah satu algoritma yang digunakan untuk melakukan *encoding* data ke dalam format ASCII. Base64 sering digunakan di internet karena hasil dari *encode* Base64 berupa *plaintext* yang memungkinkan pengiriman data, sehingaa data jauh lebih mudah dikirim dibandingkan dengan mengirimkan data dalam format biner (Y. P. Putra et al., 2021).

Algoritma Base64 bekerja dengan membagi data biner menjadi blok-blok 6 bit, lalu setiap blok tersebut dikonversi menjadi karakter Base64 menggunakan tabel *encoding* yang terdiri dari 64 karakter yaitu huruf besar (A-Z), huruf kecil (a-z), angka (0-9), serta dua simbol tambahan (+ dan /). Jika data asli bukan kelipatan 3 karakter (*byte*), maka ditambahkan karakter *padding* (=) sebagai penggenap pada hasil *encoding* (Y. P. Putra et al., 2021).

Berbeda dengan ASCII yang merepresentasikan suatu karakter dari data biner berukuran 1 *byte* (8 bit), Base64 hanya menggunakan data biner 6 bit untuk merepresentasikan 1 karakter, seperti yang terdapat pada Gambar 2.10.

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

Gambar 2. 10 *Index* Base64 (Bhushan, 2020)

Proses *encoding* menggunakan algoritma Base64 dimulai dengan mengonversi data ke dalam bentuk representasi biner. Misalnya, jika data yang akan di-*encode* berasal dari karakter ASCII seperti *string* “Man”, maka setiap karakter dikonversi ke bentuk biner berdasarkan nilai desimalnya dalam tabel ASCII. Hasil konversinya adalah sebagai berikut.

M = 01001101

a = 01100001

n = 01101110

Seluruh bit hasil konversi kemudian digabungkan menjadi satu rangkaian data biner berikut.

010011010110000101101110.

Setelah proses penggabungan, data biner tersebut dibagi menjadi blok-blok yang masing-masing terdiri dari 6 bit. Hasil pembagi blok dalam contoh ini adalah

010011 010110 000101 101110

Setiap blok 6 bit ini kemudian dipetakan ke bentuk karakter yang sesuai dalam tabel Base64.

010011 → T

010110 → W

000101 → F

101110 → u

Berdasarkan pemetaan tersebut, keempat blok tersebut menghasilkan karakter ‘T’, ‘W’, ‘F’, dan ‘u’. Demikian, hasil akhir dari *encoding string* “Man” menggunakan algoritma Base64 adalah “TWFu” (Supiyandi et al., 2020).

2.1.5 *SQL Injection*

SQL Injection adalah suatu serangan yang mengeksploitasi kerentanan pada lapisan *database* sebuah aplikasi web. Cara kerja serangan ini adalah dengan menginjeksi perintah SQL ke dalam form input atau URL. Sehingga penyerang dapat mengirimkan perintah ke *database* aplikasi web tersebut dan pada akhirnya mengambil alih kontrol atas *database* tersebut. Apabila penyerang berhasil menguasai *database*, penyerang dapat mencuri data yang bersifat pribadi dan rahasia, seperti nama pengguna, kata sandi, tanggal lahir, dan informasi lainnya (Putranto et al., 2022).

SQL Injection terjadi ketika aplikasi tidak dapat memfilter data yang tidak dipercaya dalam *query database*. Sehingga penyerang bisa menggunakan perintah

SQL yang dirancang khusus untuk mengelabui sistem agar *database* menjalankan perintah yang tidak terduga. Dampak dari teknik serangan Injeksi SQL cukup beragam diantaranya sebagai berikut (Nugraha, 2019).

a. Melakukan *Bypass* terhadap Mekanisme Otentikasi

   SQL Injection memungkinkan penyerang untuk masuk ke dalam aplikasi dengan hak akses administratif, tanpa memerlukan nama pengguna dan kata sandi yang valid.

b. Melakukan Modifikasi Data

   SQL Injection juga memungkinkan penyerang bisa mengubah data yang tersimpan di *database* atau menyisipkan konten berbahaya ke dalam halaman web. Jika yang diserang adalah *database* perbankan, penyerang mungkin saja dapat memanipulasi transaksi nasabah atau memindahkan dana dari rekening nasabah ke rekening yang dikendalikannya, sehingga menyebabkan kerugian finansial bagi korban.

c. Melakukan *Compromised* terhadap Ketersediaan Data

   Dengan menghapus seluruh data yang tersimpan di *database*, *SQL Injection* juga dapat mengancam aspek ketersediaan (*availability*) dari sistem tersebut.

d. Melakukan Pencurian Informasi

   SQL Injection juga bisa digunakan oleh penyerang untuk mencuri informasi sensitif yang tersimpan di dalam *database*.

e. Melakukan Impersonasi Pengguna

   Penyerang juga dapat memanfaatkan *SQL Injection* untuk melakukan impersonasi dengan menggunakan akun pengguna aktif di dalam *database*.

2.1.6 *Cipher Block Chaining*

Cipher Block Chaining (CBC) adalah suatu metode yang menerapkan mekanisme umpan balik (*feedback*) pada setiap blok bit, di mana hasil enkripsi dari blok sebelumnya digunakan sebagai umpan balik dalam proses enkripsi *block current*. Proses ini dimulai dengan melakukan XOR terhadap blok *plaintext* yang akan dienkripsi dengan blok *ciphertext* hasil enkripsi sebelumnya, kemudian hasil operasi XOR tersebut dimasukkan ke dalam fungsi enkripsi. Demikian, setiap blok *ciphertext* yang dihasilkan tidak hanya bergantung pada blok *plaintext* yang bersangkutan, tetapi juga pada seluruh blok *plaintext* sebelumnya. Proses dekripsi dilakukan dengan cara memasukkan blok *ciphertext* yang akan didekripsi ke dalam fungsi dekripsi, kemudian melakukan XOR terhadap hasil dekripsi tersebut dengan blok *ciphertext* sebelumnya. Blok *ciphertext* sebelumnya berfungsi sebagai umpan maju (*feedforward*) yang berperan pada tahap akhir proses dekripsi (Siregar, 2023).

Salah satu konsep penting dalam mode CBC adalah inisialisasi vektor (IV). IV merupakan nilai acak yang digunakan untuk memulai proses enkripsi pada blok pertama. Penggunaan IV yang berbeda akan menghasilkan *ciphertext* yang berbeda meskipun *plaintext* yang digunakan sama. Hal ini sangat penting untuk mencegah serangan yang dikenal dengan istilah *bit flipping attack*, di mana penyerang berusaha memodifikasi blok *ciphertext* dengan cara mengubah blok *plaintext* (Yeni et al., 2023).

2.1.7 Acunetix

Acunetix adalah alat pengujian keamanan aplikasi web otomatis yang mengaudit aplikasi web dengan memeriksa kerentanan seperti *SQL Injection* dan

kerentanan lain yang dapat dieksploitasi. Acunetix memindai situs web atau aplikasi web apa pun yang dapat diakses melalui *browser* dan menggunakan protokol HTTP/HTTPS. Acunetix menawarkan solusi yang kuat dan unik untuk menganalisis aplikasi web siap pakai dan khusus termasuk yang menggunakan JavaScript, AJAX, dan aplikasi web Web 2.0. Acunetix memiliki *crawler* canggih yang dapat menemukan hampir semua berkas (Inviciti, 2025).

Acunetix bekerja dengan menganalisis situs web melalui proses *crawling* yang menelusuri semua tautan yang tersedia, termasuk yang dibuat secara dinamis menggunakan JavaScript serta yang ditemukan dalam robots.txt dan sitemap.xml. Proses ini menghasilkan peta situs yang digunakan untuk meluncurkan pemeriksaan terarah terhadap setiap bagian situs. Setelah *crawling* selesai, Acunetix secara otomatis melakukan serangkaian pemeriksaan kerentanan pada setiap halaman yang ditemukan dengan mengidentifikasi titik-titik di mana data dapat dimasukkan, lalu menguji berbagai kombinasi masukan untuk mendeteksi potensi celah keamanan. Hasil pemindaian akan menampilkan daftar kerentanan yang ditemukan beserta informasi detail seperti data POST yang digunakan, item yang terpengaruh, serta *response* HTTP dari server. Selain itu, Acunetix menyediakan berbagai jenis laporan pemindaian, termasuk laporan Ringkasan Eksekutif, laporan Pengembang, serta laporan seperti PCI DSS atau ISO 27001. Acunetix versi 13 mengklasifikasikan *Threat Level* menjadi empat level sebagai berikut.

a. Level 0 - *Informational*

Informational tidak secara langsung menjadi ancaman keamanan, tetapi dapat memberikan wawasan atau petunjuk bagi pengembang.

b. Level 1 – *Low*

Kerentanan tingkat rendah umumnya berkaitan dengan kurangnya enkripsi pada lalu lintas data atau pengungkapan jalur direktori. Dampak dari kerentanan ini harus dievaluasi berdasarkan konteks penggunaannya dalam aplikasi serta dampak bisnis yang mungkin ditimbulkan.

c. level 2 – *Medium*

Kerentanan tingkat sedang biasanya disebabkan oleh kesalahan konfigurasi server atau kesalahan dalam penulisan kode situs web. Meskipun tidak secara langsung memengaruhi aplikasi atau sistem, kelemahan ini dapat memudahkan gangguan pada server dan memungkinkan penyusupan. Oleh karena itu, tetap penting untuk segera diperbaiki.

d. Level 3 – *High*

Kerentanan tingkat tinggi menempatkan situs web dalam risiko besar untuk diretas dan dapat memungkinkan peretas menemukan celah keamanan lainnya. Kerentanan jenis ini harus segera diperbaiki agar tidak dimanfaatkan oleh pihak yang tidak bertanggung jawab.

2.1.8 SQLmap

SQLmap adalah alat pengujian penetrasi *open-source* yang dirancang untuk mengotomatiskan proses pendeteksian dan eksploitasi kerentanan *SQL Injection*, serta memungkinkan pengambilalihan server *database*. SQLmap dilengkapi dengan mesin deteksi yang kuat serta berbagai fitur khusus yang mendukung kebutuhan penguji penetrasi tingkat lanjut (Guimaraes & Stampar, 2024).

Alat ini menyediakan berbagai opsi yang mencakup identifikasi karakteristik *database (database fingerprinting*), pengambilan data dari *database*, hingga akses ke sistem *file* serta eksekusi perintah pada sistem operasi melalui koneksi *out-of-band* (OOB). Fitur-fitur tersebut menjadikan SQLmap sebagai salah satu alat yang sangat berguna dalam menguji dan meningkatkan keamanan sistem *database* terhadap serangan *SQL Injection*.

SQLmap mendukung berbagai *Database Management Sistem* (DBMS) populer seperti *MySQL, Oracle, PostgreSQL, Microsoft SQL Server*, hingga *database* modern seperti *CockroachDB, ClickHouse* dan masih banyak lainnya. Alat ini mampu mendeteksi dan mengeksploitasi enam teknik utama *SQL Injection*, yaitu *boolean-based blind, time-based blind, error-based, UNION query-based, stacked queries*, dan *out-of-band*. Selain itu, SQLmap juga mendukung koneksi langsung ke *database* dengan kredensial yang tersedia, tanpa melalui *SQL Injection*. SQLmap menyediakan fitur enumerasi yang memungkinkan pengguna memperoleh informasi sensitif seperti nama *database*, tabel, kolom, akun pengguna, hak akses, serta *hash password* yang dapat dikenali secara otomatis dan dicoba dipecahkan melalui metode kamus. SQLmap memungkinkan *dumping* data secara fleksibel, baik seluruh tabel, kolom tertentu, hingga sebagian karakter dari entri data. Fitur pencarian terfokus juga disediakan untuk menemukan informasi spesifik di seluruh struktur *database*. SQLmap bahkan mampu mengakses sistem *file* server (mengunduh/mengunggah *file*) untuk *database* tertentu seperti MySQL, PostgreSQL, dan SQL Server. SQLmap mendukung eskalasi hak akses pengguna

database melalui perintah *getsystem* dari *Metasploit's Meterpreter* untuk mendapatkan hak istimewa lebih tinggi.

2.1.9 *Google Lighthouse*

Google Lighthouse merupakan sebuah alat berbasis *browser* yang terintegrasi dalam *Chrome DevTools* dan mampu memberikan analisis komprehensif secara otomatis dan konsisten terhadap performa halaman web. *Google Lighthouse* mengukur performa halaman web berdasarkan beberapa parameter utama berikut (Google Developers, 2025).

a. *First Contentful Paint* (FCP), mengukur waktu (dalam detik) sejak halaman mulai dimuat hingga konten pertama seperti teks atau gambar ditampilkan pada layar. Semakin rendah nilai FCP, semakin cepat pengguna menerima *response* visual awal dari halaman.
b. *Largest Contentful Paint* (LCP), Menunjukkan waktu yang dibutuhkan untuk merender elemen konten terbesar yang terlihat di layar. Parameter ini digunakan untuk memperkirakan kapan konten utama halaman selesai dimuat.
c. *Total Blocking Time* (TBT), menggambarkan total waktu (dalam milidetik) antara FCP dan saat halaman siap merespons interaksi pengguna. Nilai ini menunjukkan seberapa lama *thread* utama *browser* terhambat oleh eksekusi skrip JavaScript yang berat.
d. *Cumulative Layout Shift* (CLS), menilai stabilitas visual halaman dengan mengukur frekuensi dan besarnya pergeseran elemen secara tak terduga

selama proses pemuatan. Skor CLS idealnya mendekati nol, yang berarti tata letak halaman tetap stabil.

e. *Speed Index* (SI), mengukur kecepatan tampilan konten halaman secara visual selama proses pemuatan. Nilai SI diperoleh dari analisis perbedaan visual antar *frame* saat loading. Semakin kecil nilai SI, semakin cepat halaman terlihat secara utuh.

2.1.10 Korelasi *Pearson*

Uji korelasi adalah suatu metode dalam statistika yang berfungsi untuk menentukan besaran atau mengukur seberapa kuat hubungan antara satu variabel dengan variabel yang lain, dengan tidak mempertimbangkan apakah salah satu variabel bergantung pada variabel lainnya. Semakin jelas hubungan linier antara kedua variabel tersebut, maka semakin tinggi tingkat hubungan dalam garis lurus yang terbentuk (Miftahuddin et al., 2021).

Korelasi *Pearson* adalah jenis korelasi sederhana yang melibatkan satu variabel bebas (*independent*) dan satu variabel terikat (*dependent*). Variabel bebas adalah variabel yang mempengaruhi atau yang menjadi sebab perubahannya atau timbulnya variabel terikat. Variabel terikat merupakan varibel yang dipengaruhi atau yang menjadi akibat, karena adanya varibel bebas. Korelasi *Pearson* menghasilkan koefisien korelasi yang digunakan untuk mengukur seberapa kuat hubungan linier antara kedua variabel. Korelasi *Pearson* diterapkan untuk menentukan tingkat keeratan hubungan antara dua variabel (Miftahuddin et al., 2021). Korelasi *Pearson* dihitung menggunakan persamaan (1) (Sari et al., 2023).

$$r = \frac{n \sum XY - \sum X \sum Y}{\sqrt{[n \sum X^2 - (\sum X)^2]\,[n \sum Y^2 - (\sum Y)^2]}} \quad (1)$$

Keterangan:

r = Koefisien korelasi *Pearson*

n = Jumlah data

$\sum X$ = Jumlah semua nilai X

$\sum Y$ = Jumlah semua nilai Y

$\sum XY$ = Jumlah hasil perkalian nilai X dan Y

$\sum X^2$ = Jumlah semua kuadrat nilai X

$\sum Y^2$ = Jumlah semua kuadrat nilai Y

Korelasi *Pearson* dinotasikan dengan simbol r dan memiliki nilai yang berkisar antara -1 hingga 1. Jika bernilai 1, maka terdapat hubungan linear positif sempurna. jika bernilai -1, maka terdapat hubungan linear negatif sempurna. sedangkan jika bernilai 0, maka tidak ada hubungan linear antara kedua variabel (Sanny & Dewi, 2020).

2.1.11 Entropi *Shannon*

Entropi *Shannon* merupakan ukuran ketidakpastian atau tingkat kerandoman dalam suatu sumber informasi yang diperkenalkan oleh *Claude E. Shannon* pada tahun 1948 dalam makalah berjudul "*A Mathematical Theory of Communication*". Sejak saat itu konsep ini digunakan secara luas dan berkembang menjadi dasar dalam studi termodinamika, teori informasi, analisis data, dan komunikasi. Entropi Shnnon mengukur ketidakpastian dan informasi secara efektif menggunakan istilah matematika (Huang, 2024).

Entropi *Shannon* $H(X)$ untuk suatu variabel acak diskrit *X* dengan distribusi probabilitas $P = \{p_1, p_1, ..., p_n\}$ didefinisikan sebagai persamaan (2).

$$H(X) = -\sum_{i=1}^{n} p_i \ log_2 \ p_i \quad (2)$$

di mana p_i adalah probabilitas terjadinya nilai ke-i dari X, dan logaritma biasanya diambil dalam basis 2, sehingga entropi diukur dalam satuan bit.

Informasi yang diharapkan dari suatu sumber data stokastik diukur menggunakan entropi *Shannon*. Jika semua kemungkinan hasil dari sebuah variabel acak memiliki peluang yang sama, maka entropinya akan bernilai maksimum yang menandakan tingkat ketidakpastian yang paling tinggi. Sebaliknya, ketika hasil sudah pasti, maka tidak ada ketidakpastian karena nilai entropi menjadi nol (Huang, 2024). Nilai maksimum dari entropi *Shannon* yang dapat dicapai secara teoritis dihitung menggunakan rumus $log_2(n)$, di mana *n* menyatakan total simbol (ELLERMAN, 2013).

2.2 Tinjauan Pustaka

2.2.1 Penelitian Terkait

Terdapat beberapa penelitian yang signifikan menggunakan berbagai pendekatan untuk menerapkan enkripsi dalam pengamanan URL. Kontribusi penting dari penelitian-penelitian tersebut terletak pada pemahaman dan penerapan teknik-teknik yang meningkatkan keamanan URL. Temuan-temuan ini memiliki potensi untuk menjadi dasar penting dalam pengembangan metode enkripsi yang lebih baik dalam melindungi informasi yang terkandung dalam URL. Tabel 2.1 menyajikan penelitian terkait yang menjadi landasan dalam penelitian ini.

Tabel 2. 1 Penelitian terkait

Penulis	**Algoritma**	**Objek Enkripsi**	***Ciphertext***	**Hasil Penelitian**
(Iswara et al., 2023)	AES, RC4	*Value* GET	Tetap	Penelitian ini membandingkan algoritma AES dan RC4 untuk enkripsi *value* GET. Percobaan dilakukan terhadap sejumlah URL yang berisikan parameter data dengan ukuran panjang berbeda-beda. Hasil nya AES mendapatkan *ciphertext* yang lebih panjang dan waktu eksekusi yang lebih lama daripada RC4
(Hidayatullah, 2022)	Base64	*Path*	Tetap	Penelitian ini menggunakan algoritma Base64 untuk enkripsi *path* URL guna mencegah pencurian data. Pengujian dilakukan menggunakan *WebCruiser WebVulnerability Scanner*, hasil nya URL terenkripsi dengan dengan aman, sehingga *website* terlindungi dari pencurian data.

Tabel 2. 1 Penelitian terkait (Lanjutan 1)

Penulis	**Algoritma**	**Objek Enkripsi**	***Ciphertext***	**Hasil Penelitian**
(Zebua et al., 2022)	MD5	Parameter	Tetap	Penelitian ini mengenkripsi *query string* dengan algoritma MD5 untuk mencegah *SQL Injection*. Berdasarkan pengujian *black box*, SQLmap, dan *Unit Test* Otomatis, hasil menunjukkan bahwa MD5 efektif mencegah akses pengguna tak terverifikasi dan serangan dengan alat seperti SQLmap.
(Witriyono & Fernandez, 2021)	Base64, SHA, MD5	Parameter	Tetap	Penelitian ini mengenkripsi URL menggunakan algoritma Base64, MD5 dan SHA-1 untuk mengamankan URL presensi berbasis *QR code*. Hasil penelitian menunjukkan bahwa percobaan peretasan dengan injeksi SQL pada parameter URL dari *QR code* terbukti tidak dapat dilakukan, sehingga keamanan pemrosesan presensi berlangsung dengan baik.

Tabel 2. 1 Penelitian terkait (Lanjutan 2)

Penulis	**Algoritma**	**Objek Enkripsi**	***Ciphertext***	**Hasil Penelitian**
(Rusman et al., 2021)	AES	Parameter	Tetap	Penelitian ini menggunakan algoritma AES untuk enkripsi parameter *id* pada URL. Pengujian *SQL Injection* dilakaukan menggunskakan *webCruiser*, hasil nya menunjukkan bahwa tidak terdeteksi *vulnerability* sehingga data-data yang terdapat di dalam nya berhasil disamarkan dan terlindungi.
(T. Putra & Andrian, 2020)	Base64	Parameter	Tetap	Tujuan dari penelitian ini adalah untuk menganalisis tingkat keamanan parameter pada URL yang terenkripsi base64. Hasil penelitian menunjukkan bahwa keamanan URL dari akses ilegal jadi lebih baik namun masih memiliki sedikit celah saat serangan dilakukan menggunakan metode *Brute Force* dengan tingkat keberhasilan 0,00892%.

Tabel 2. 1 Penelitian terkait (Lanjutan 3)

Penulis	**Algoritma**	**Objek Enkripsi**	***Ciphertext***	**Hasil Penelitian**
(Bhaskara et al., 2020)	AES	*Path*	Tetap	Penelitian ini fokus pada enkripsi direktori *file* menggunakan algoritma AES (*Rijndael*) untuk menyediakan media berbagi *file* yang aman. Hasil pengujian menunjukkan bahwa *path* direktori pada URL terenkripsi dengan aman, sehingga hanya pengguna tervalidasi yang dapat mengaksesnya.
(Arif, 2020)	Base64, Rotation13	Parameter	Tetap	Penelitian ini melakukan *encoding* berlapis pada parameter *id* menggunakan algoritma Base64 dan Rotation13, di mana hasil *encoding* Base64 di-*encode* lagi dengan Rotation13. Hasil penelitian menunjukkan bahwa algoritma Base64 dan Rotation13 dapat dikombinasikan dengan baik dan menutup celah keamanan pada URL.

Tabel 2. 1 Penelitian terkait (Lanjutan 4)

Penulis	**Algoritma**	**Objek Enkripsi**	***Ciphertext***	**Hasil Penelitian**
(Wijaya, 2020)	AES	*Variable* & *value* GET	Tetap	Fokus penelitian ini adalah mengenkripsi *variable* dan *value* GET menggunakan AES-128 untuk mencegah *SQL Injection*. URL terenkripsi diuji menggunakan SQLmap, hasil nya menunjukkan bahwa integritas dari URL yang terenkripsi lebih terjaga dan metode *SQL Injection* tidak dapat dilakukan.
(Sipayung & Ginting, 2019)	3DES	*Value* GET	Tetap	Fokus penelitian ini adalah menggunakan algoritma 3DES untuk mengenkripsi *value* GET. URL terenkripsi 3DES diuji serangan *SQL Injection* menggunakan *Web Cruiser Web Vulnerability Scanner*. Hasil nya algoritma 3DES terbukti dapat mengamankan *resource website* dari serangan *SQL Injection.*

Tabel 2.1 menyajikan beberapa penelitian terkait yang membahas berbagai metode enkripsi yang diterapkan pada URL. Setiap penelitian memiliki fokus yang berbeda, baik dari segi algoritma yang digunakan, objek enkripsi, maupun evaluasi *ciphertext* yang dihasilkan. Selanjutnya matriks penelitian akan disajikan untuk memperjelas kesenjangan penelitian dan menunjukkan posisi penelitian yang diusulkan dalam konteks studi sebelumnya.

2.2.2 Matriks Penelitian

Matriks penelitian disajikan untuk memberikan gambaran yang lebih jelas mengenai cakupan penelitian terdahulu serta membandingkannya dengan penelitian yang diusulkan. Matriks ini mengklasifikasikan penelitian-penelitian sebelumnya berdasarkan algoritma yang digunakan, metode yang diterapkan, serta objek enkripsi yang diteliti. Adanya matriks penelitian ini memungkinkan kesenjangan penelitian dapat dilihat secara jelas, serta bagaimana penelitian yang diusulkan memiliki keunikan tersendiri dan berkontribusi dalam pengembangan metode enkripsi URL. Matriks penelitian tersebut ditampilkan pada Tabel 2.2.

Tabel 2. 2 Matriks penelitian

No	Peneliti	Ruang Lingkup															
		Algoritma							Metode			Objek enkripsi				Ciphertext	
		AES	3DES	RC4	SHA	MD5	Base64	Rot13	Enkripsi	Hashing	Encoding	Variable GET	Value GET	Parameter	Path	Tetap	Dinamis
1	(Iswara et al., 2023)	√	-	√	-	-	-	-	√	-	-	-	√	-	-	√	-
2	(Hidayatullah, 2022)	-	-	-	-	-	√	-	-	-	√	-	-	-	√	√	-
3	(Zebua et al., 2022)	-	-	-	-	√	-	-	-	√	-	-	-	√	-	√	-
4	(Witriyono & Fernandez, 2021)	-	-	-	√	√	√	-	-	√	√	-	-	√	-	√	-
5	(Rusman et al., 2021)	√	-	-	-	-	-	-	√	-	-	-	-	√	-	√	-
6	(T. Putra & Andrian, 2020)	-	-	-	-	-	√	-	-	-	√	-	-	√	-	√	-
7	(Bhaskara et al., 2020)	√	-	-	-	-	-	-	√	-	-	-	-	-	√	√	-
8	(Arif, 2020)	-	-	-	-	-	√	√	-	-	√	-	-	√	-	√	-
9	(Wijaya, 2020)	√	-	-	-	-	-	-	√	-	-	√	√	-	-	√	-
10	(Sipayung & Ginting, 2019)	-	√	-	-	-	-	-	√	-	-	-	√	-	-	√	-
11	Usulan Penelitian	√	-	-	√	-	√	-	√	√	√	√	√	√	√	-	√

Berdasarkan Tabel 2.2, beberapa penelitian terkait penerapan enkripsi URL telah mengusulkan berbagai teknik, termasuk enkripsi *variable* serta *value* GET, parameter *id,* dan *path* URL. Hasil evaluasi menunjukkan bahwa teknik-teknik tersebut memiliki performa yang baik dalam mengamankan data pada URL dengan tingkat keamanan yang signifikan. Namun, belum terdapat penelitian yang secara khusus mengenkripsi seluruh *path* URL secara dinamis di mana enkripsi URL yang sama akan menghasilkan *ciphertext* yang berubah-ubah. Selain itu, belum terdapat penelitian yang secara khusus mengintegrasikan tenkik enkripsi, *hashing* dan *encoding* secara bersamaan dalam pengamanan URL, sehingga terdapat kesempatan yang signifikan untuk mengisi kesenjangan ini. Oleh karena itu, penelitian ini difokuskan pada pengembangan teknik enkripsi URL yang dinamis dan menyeluruh mengenkripsi seluruh *path* URL, dengan menggabungkan keunggulan AES dalam enkripsi data, kemampuan SHA dalam *hashing* dan Base64 dalam *encoding* untuk menciptakan metode enkripsi yang lebih optimal dalam mengamankan URL. Motode ini dapat memberikan kontribusi signifikan dan pemahaman yang lebih komprehensif terhadap keamanan URL.