

**PENCEGAHAN *MAN IN THE MIDDLE ATTACK* PADA  
TRANSMISI DATA *WEB SERVICE* MENGGUNAKAN  
*END TO END ENCRYPTION (E2EE)***



**Skripsi**

Untuk memenuhi syarat memperoleh Derajat  
Sarjana Teknik (S.T.)

**Oleh:**

**O. RIASTANJUNG**

**NIM 2001020039**

**JURUSAN TEKNIK INFORMATIKA  
FAKULTAS TEKNIK DAN TEKNOLOGI KEMARITIMAN  
UNIVERSITAS MARITIM RAJA ALI HAJI  
TANJUNGPINANG  
2024**

**PENCEGAHAN *MAN IN THE MIDDLE* ATTACK PADA  
TRANSMISI DATA *WEB SERVICE* MENGGUNAKAN  
*END TO END ENCRYPTION (E2EE)***



**Skripsi**

Untuk memenuhi syarat memperoleh derajat  
Sarjana Teknik (S.T.)

**Oleh:**

**O. RIASTANJUNG**

**NIM 2001020039**

Telah mengetahui dan disetujui oleh :

**Pembimbing I,**

**Pembimbing II,**

Hendra Kurniawan, S.Kom., M.Sc.Eng., Ph.D.

NIP. 198404022014041001

Nurul Hayaty, S.T., M.Cs.

NIP. 199103272019032019

## HALAMAN PENGESAHAN

Judul Skripsi : Pencegahan *Man In The Middle Attack* Pada Transmisi Data  
*Web Service* Menggunakan *End To End Encryption* (E2EE)  
Nama Mahasiswa : O. Riastanjung  
NIM : 2001020039  
Jurusan : Teknik Informatika

Telah dipertahankan di depan Dewan Penguji dan dinyatakan lulus  
pada tanggal 17 Juli 2024

### *Susunan Tim Pembimbing dan Penguji*

Jabatan	Nama Dosen	Tanda Tangan	Tanggal
Pembimbing I	: Hendra Kurniawan, S.Kom., M.Sc.Eng., Ph.D.	.....	17/07/2024
Pembimbing II	: Nurul Hayaty, S.T., M.Cs.	.....	17/07/2024
Ketua Penguji	: Martaleli Bettiza, S.Si, M.Sc.	.....	17/07/2024
Anggota Pengujij I	: Muhamad Radzi Rathomi, S.Kom., M.Cs.	.....	17/07/2024
Anggota Penguji II	: Nola Ritha, S.T., M.Cs.	.....	17/07/2024

**Tanjungpinang, 17 Juli 2024**

**Universitas Maritim Raja Ali Haji**

**Dekan Fakultas Teknik dan Teknologi Kemaritiman**

**Ir. Sapta Nugraha, S.T., M.Eng.**

**NIP 198904132015041005**



## PERNYATAAN ORISINALITAS

Dengan ini saya menyatakan bahwa skripsi saya yang berjudul Pencegahan *Man In The Middle Attack* Pada Transmisi Data *Web Service* Menggunakan *End To End Encryption* (E2EE) adalah benar karya saya dengan arahan dari komisi pembimbing dan belum diajukan dalam bentuk apa pun kepada perguruan tinggi mana pun. Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan maupun tidak diterbitkan dari penulis lain telah disebutkan dalam teks dan dicantumkan dalam Daftar Pustaka di bagian akhir skripsi ini.

Jika kemudian hari ternyata terbukti pernyataan saya ini tidak benar dan melanggar peraturan yang sah dalam karya tulis dan hak intelektual maka saya bersedia ijazah yang telah saya terima untuk ditarik kembali oleh Universitas Maritim Raja Ali Haji.

Tanjungpinang, \_\_ Juli 2024

Yang menyatakan

Materai Rp. 6000,-

O. Riastanjung

## **HALAMAN PERSEMBAHAN**

*Terpujilah Tuhan Yang Maha Esa yang telah mengizinkan penulis tetap sehat dalam jasmani dan batin.*

*Tidak lupa pula saya panjatkan syukur saya kepada Tuhan Yang Maha Esa berkat rahmat dan karunianya, saya berhasil mencapai tahap penulisan ini.*

*Hasil penulisan karya tulis penelitian ini saya persembahkan untuk keluarga saya, yang telah mensupport saya, kepada teman-teman saya yang telah mendukung baik dari segi moralitas maupun sisi pendidikan untuk terus menopang saya menyelesaikan penulisan karya tulis ini.*

*Semoga seluruh teman-teman saya juga termotivasi dan semangat dalam menyelesaikan pendidikan Sarjana mereka, dan semoga Tuhan Yang Maha Esa selalu memberkati seluruh langkah-langkah yang kita lakukan ke depannya.*

## **HALAMAN MOTO**

*"Natus Vincere"*

"Terlahir Untuk Menang"

*"Not Reaching This Far Just To Be Average"*

*"A Man Will Die, But Not His Ideas"*

*"Take The Risk or Lose The Chance"*

*"I don't stop when tired, I stop when i'm done"*

*"Trying can result in failure, but not trying is an absolute failure"*

## KATA PENGANTAR

Puji syukur atas kehadiran Tuhan Yang Maha Esa yang telah melimpahkan rahmat, hidayah dan rezekinya sehingga penulis dapat menyelesaikan penyusunan skripsi yang berjudul "*Pencegahan Man In The Middle Attack Pada Transmisi Data Web Service Menggunakan End To End Encryption (E2EE)*" ini hingga selesai sebagai salah satu persyaratan dalam memenuhi syarat memperoleh derajat sarjana teknik (S.T.) pada Fakultas Teknik dan Teknologi Kemaritiman Jurusan Teknik Informatika Universitas Maritim Raja Ali Haji.

Pada kesempatan ini penulis ingin mengucapkan terimakasih kepada seluruh pihak yang sudah membantu penulis dalam menyelesaikan tugas akhir ini. Penulis tidak dapat membalas semua kebaikan yang telah diterima, semoga Tuhan Yang Maha Esa senantiasa memberikan kebahagiaan dan keberkahan kepada kita semua. Penulis ingin mengucapkan terimakasih banyak kepada orang-orang yang berperan penting dalam kehidupan penulis, yakni :

1. Ibu Ceng Miao Fung dan Bapak Tji Tjin Liong yang telah menjadi sosok orang tua yang berperan penting dalam mendidik saya dari kecil hingga saat penulis menulis penelitian skripsi ini. Terimakasih atas dukungan dan doa orang tua penulis sehingga penulisan dilaksanakan dengan lancar dan minim hambatan.
2. Seluruh kakak perempuan dan kakak laki-laki saya yaitu, Livyna Valerie Lay, Nurhayati, Susan, Sutrisno Gunawan, dan O. Midiyanto yang selalu menanyakan kapan selesai pendidikan dan selalu menopang dan memberikan dukungan setiap harinya agar saya selalu fokus dalam menyelesaikan pendidikan sarjana saya.
3. Bapak Sapta Nugraha, S.T., M.Eng., selaku Dekan Fakultas Teknik dan Teknologi Kemaritiman Universitas Maritim Raja Ali Haji.
4. Bapak Muhamad Radzi Rathomi, S. Kom., M.Cs., selaku Ketua Program Studi Teknik Informatika.
5. Bapak Hendra Kurniawan, S.Kom., M.Sc.Eng., Ph.D. selaku Dosen Pembimbing I yang telah memberikan semangat, motivasi, meluangkan



waktu, tenaga dan pemikirannya untuk membimbing saya menyusun skripsi ini.

6. Ibu Nurul Hayaty, S.T., M.Cs., selaku Dosen Pembimbing II yang telah memberikan motivasi, semangat, meluangkan waktu dalam membantu saya menghadapi kesulitan dalam penyusunan skripsi ini.
7. Bapak dan Ibu dosen serta staf tata usaha Fakultas Teknik dan Teknologi Kemaritiman Universitas Maritim Raja Ali Haji yang telah banyak memberikan pengetahuan selama masa perkuliahan saya sehingga menambah wawasan, ilmu pengetahuan, pengalaman dan bantuannya yang membantu saya sehingga berhasil sampai ke tahap ini untuk menyelesaikan pendidikan sarjana teknik saya.
8. Keluarga besar penulis yang tak lupa memberikan dan dukungan kepada juga kepada penulis.
9. Sahabat penulis ketika berkuliah, yaitu Eryan Kurniawan, Wahyu Seto Pranata, Rezi Afrialdi, Leonardo Tegarsuan, Alramadan Oloansyah, Alwan Hidayat, Arifian Syahputra, Agnes Gabriella Manik, Anindya Sekar Paramitha, Arya Rahmansyah, Aznul Khairi, Boyke, Dela Nifari, Erlina Dwi Pratiwi, Ejika Oktaviani, Ezy, Fariz Rahmat Firdaus, Ferya, Ghora Laziola, Irvantoni Ilham, Jupri, Kasirajil Masyukur, Kurrata Aini, M Irfan Raif, Muhammad Fadli, Muhammad Fadhly Azzuhri, Mia Al Fiani, Musliha, Nifia Syufriana, Nur Alifa, Raka, Rama Setiawan, Richard Robinson Sandy, Riswan Arta Kurnia, Samuel Miskan Hanock, Siska Anggraeni, Syahri Ramadhan, Trinanda, Yudha Edy Payo Gurusinga, Wan Alfi Gustiardi, dan Wan Fariz Dewantara.
10. Terakhir, untuk O. Riastanjung yaitu penulis sendiri yang telah tetap semangat dan terus maju hingga menyelesaikan seluruh yang telah dimulai.

Tanjungpinang, \_\_ Juli 2024

O. Riastanjung

## DAFTAR ISI

HALAMAN JUDUL .....	i
HALAMAN PERSETUJUAN .....	ii
HALAMAN PENGESAHAN .....	iii
PERNYATAAN ORISINALITAS .....	v
HALAMAN PERSEMBAHAN .....	vi
HALAMAN MOTO .....	vii
KATA PENGANTAR .....	viii
DAFTAR ISI .....	x
DAFTAR TABEL .....	xii
DAFTAR GAMBAR .....	xiii
GLOSARIUM .....	xv
ABSTRAK .....	xv
ABSTRACT .....	xvii
BAB I PENDAHULUAN .....	1
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	4
1.3 Batasan Masalah .....	4
1.4 Tujuan Penelitian .....	4
1.5 Manfaat Penelitian .....	5
1.6 Sistematika Penulisan .....	6
BAB II KAJIAN LITERATUR .....	7
2.1 Tinjauan Pustaka .....	7
2.2 Landasan Teori .....	9
2.2.1 Sistem Keamanan .....	9
2.2.2 <i>Man In The Middle Attack</i> .....	10
2.2.3 <i>End To End Encryption</i> .....	12
2.2.4 <i>Advanced Encryption Standard (AES)</i> .....	13
2.2.5 Enkripsi AES 128 .....	14
2.2.6 Dekripsi AES 128 .....	16
2.2.7 <i>Web Service</i> .....	18
2.2.8 Burp Suite .....	20
BAB III METODE PENELITIAN .....	21
3.1 Waktu Penelitian .....	21
3.2 Tahapan Penelitian .....	21

3.3	Jenis Penelitian .....	22
3.4	Alat atau Instrumen Penelitian .....	22
3.5	Analisis dan Perancangan.....	23
3.5.1	Analisis Data .....	23
3.5.2	Perancangan Algoritma Enkripsi AES 128 .....	24
3.5.3	Perancangan Algoritma Dekripsi AES 128.....	35
3.5.4	Perancangan Skenario Serangan MITM .....	41
3.5.5	Pengujian .....	43
3.6	Perancangan <i>Data Flow Diagram</i> (DFD) .....	46
3.6.1	DFD Tingkat 0 .....	46
3.6.2	DFD Tingkat 1 .....	46
3.7	Tampilan <i>User Interface</i> (UI) .....	47
3.8	Implementasi Sistem <i>End To End Encryption</i> .....	49
3.8.1	Ekspansi Kunci AES 128 .....	49
3.8.2	<i>Web Server</i> (Sisi <i>Backend</i> ) .....	58
3.8.3	<i>Web Client</i> (Sisi <i>Frontend</i> ) .....	67
3.9	Implementasi Serangan MITM <i>Sniffing</i> .....	74
3.9.1	Menjalankan <i>Web Client</i> dan <i>Web Server</i> .....	75
3.9.2	Melakukan <i>Forward Port</i> Pada <i>Web Client</i> dan <i>Web Server</i> .....	76
3.9.3	Melakukan <i>Sniffing</i> Dengan Burp Suite .....	77
BAB IV HASIL DAN PEMBAHASAN .....		81
4.1	Analisis Sistem <i>End To End Encryption</i> .....	81
4.2	Hasil Serangan Pada <i>Web Service</i> .....	82
BAB V PENUTUP .....		90
5.1	Kesimpulan.....	90
5.2	Saran.....	90
DAFTAR PUSTAKA .....		92
LAMPIRAN .....		98

## DAFTAR TABEL

Tabel 1. Perbedaan Jenis AES (Yaomulfurqqan dan Pramusinto, 2023) .....	13
Tabel 2. Instrumen Penelitian .....	23
Tabel 3. Tabel Pengujian Enkripsi dan Dekripsi .....	44
Tabel 4. Tabel Pengujian Skenario <i>Man In The Middle Attack</i> .....	44
Tabel 5. Perbandingan Hasil Serangan Pada Sistem.....	82

## DAFTAR GAMBAR

<b>Gambar 1.</b> Konsep <i>End To End Encryption</i> (Lestari, 2022) .....	13
<b>Gambar 2.</b> <i>End To End Encryption</i> pada web (Liander, 2022) .....	13
<b>Gambar 3.</b> Tabel S-Box (Selimis dkk., 2007) .....	14
<b>Gambar 4.</b> Alur Enkripsi AES 128 (Yaomulfurqqan dan Pramusinto, 2023).....	16
<b>Gambar 5.</b> Tabel <i>Inverse</i> S-Box (Selimis dkk., 2007) .....	17
<b>Gambar 6.</b> Alur Dekripsi AES 128 (Yaomulfurqqan dan Pramusinto, 2023).....	18
<b>Gambar 7.</b> <i>Web Service</i> REST (Simbulan dan Aryanto, 2024).....	19
<b>Gambar 8.</b> Tahapan Penelitian .....	21
<b>Gambar 9.</b> Data sampel yang digunakan .....	24
<b>Gambar 10.</b> ASCII Heksadesimal (Hamzah, 2014) .....	25
<b>Gambar 11.</b> Substitusi S-BOX .....	26
<b>Gambar 12.</b> MITM - 1 .....	41
<b>Gambar 13.</b> MITM - 2 .....	42
<b>Gambar 14.</b> MITM - 3 .....	42
<b>Gambar 15.</b> MITM – 4 .....	43
<b>Gambar 16.</b> DFD Tingkat 0 .....	46
<b>Gambar 17.</b> DFD Tingkat 1 .....	47
<b>Gambar 18.</b> UI <i>website</i> - 1 .....	48
<b>Gambar 19.</b> UI <i>website</i> - 2 .....	48
<b>Gambar 20.</b> UI <i>website</i> - 3 .....	48
<b>Gambar 21.</b> Kode Pembuatan <i>Roundkey</i> .....	50
<b>Gambar 22.</b> Kode operasi XOR 1 baris .....	51
<b>Gambar 23.</b> Kode pembantu dalam <i>RoundKey</i> .....	52
<b>Gambar 24.</b> Kode konversi heksadesimal ke string dan sebaliknya .....	54
<b>Gambar 25.</b> Kode RCON .....	54
<b>Gambar 26.</b> Kode <i>SubBytes</i> .....	55
<b>Gambar 27.</b> Kode RotWord di ekspansi kunci .....	56
<b>Gambar 28.</b> Kode pembuatan <i>Block Cipher</i> .....	58
<b>Gambar 29.</b> Struktur folder proyek sisi <i>backend</i> .....	58

<b>Gambar 30.</b> Kode utama <i>web server</i> .....	59
<b>Gambar 31.</b> <i>Routes endpoint</i> data pasien .....	60
<b>Gambar 32.</b> <i>Controller</i> data pasien .....	61
<b>Gambar 33.</b> Kode <i>AddRoundKey</i> .....	62
<b>Gambar 34.</b> Kode <i>ShiftRows</i> .....	62
<b>Gambar 35.</b> Kode <i>MixColumns</i> .....	64
<b>Gambar 36.</b> Kode Enkripsi per <i>block</i> AES 128.....	66
<b>Gambar 37.</b> Kode enkripsi AES 128 .....	66
<b>Gambar 38.</b> Struktur proyek sisi <i>web client</i> .....	68
<b>Gambar 39.</b> Kode <i>web service method</i> GET sisi <i>web client</i> dalam request data..	69
<b>Gambar 40.</b> Kode <i>website</i> dan penerapan dekripsi AES 128 .....	70
<b>Gambar 41.</b> Kode <i>InverseShiftRows</i> .....	71
<b>Gambar 42.</b> Kode <i>InverseMixColumns</i> .....	72
<b>Gambar 43.</b> Kode dekripsi per <i>block</i> AES 128 .....	73
<b>Gambar 44.</b> Kode dekripsi AES 128 .....	74
<b>Gambar 45.</b> Proses menjalankan <i>web server</i> pada sisi lokal .....	75
<b>Gambar 46.</b> Proses menjalankan <i>web client</i> pada sisi lokal .....	76
<b>Gambar 47.</b> Proses <i>forward port</i> pada <i>web server</i> .....	76
<b>Gambar 48.</b> Konfigurasi <i>web client</i> setelah <i>web server</i> di <i>forward</i> .....	77
<b>Gambar 49.</b> Proses <i>forward port</i> pada <i>web client</i> .....	77
<b>Gambar 50.</b> Riwayat proses transmisi data <i>web service</i> .....	78
<b>Gambar 51.</b> Proses melakukan <i>sniffing</i> .....	79
<b>Gambar 52.</b> Hasil <i>sniffing</i> pada <i>web service</i> dengan E2EE.....	80
<b>Gambar 53.</b> Proses <i>sniffing</i> pada <i>web service</i> tanpa E2EE.....	80

## GLOSARIUM

MITM	Serangan di mana seorang penyerang memposisikan dirinya di antara dua pihak yang berkomunikasi secara langsung. Tujuannya adalah untuk memata-matai atau bahkan memanipulasi komunikasi tersebut tanpa pengetahuan kedua pihak yang berkomunikasi.
<i>Sniffing</i>	Proses penyadapan atau perekaman lalu lintas data dalam sebuah jaringan. Penyadapan ini dapat dilakukan oleh pihak yang tidak berwenang untuk mendapatkan informasi yang seharusnya bersifat pribadi atau rahasia.
E2EE	Teknik enkripsi yang memastikan bahwa data yang dikirimkan dari satu pihak ke pihak lain hanya bisa dibaca oleh kedua pihak tersebut. Data dienkripsi di sisi pengirim dan hanya bisa didekripsi oleh penerima, sehingga bahkan penyedia layanan tidak bisa membaca isi pesan.
<i>Web Service</i>	Layanan yang disediakan oleh sebuah sistem untuk berinteraksi dengan aplikasi atau sistem lain melalui protokol yang ditentukan (biasanya menggunakan HTTP atau HTTPS). Web service memungkinkan berbagai aplikasi untuk saling berkomunikasi dan bertukar data.
<i>Web Security</i>	Praktik dan teknologi yang digunakan untuk melindungi situs web dan aplikasi web dari ancaman keamanan seperti serangan peretasan, pencurian data, atau penipuan online.

## ABSTRAK

Riastanjung, O. 2024. *Pencegahan Man In The Middle Attack Pada Transmisi Data Web Service Menggunakan End To End Encryption (E2EE)*, Skripsi. Tanjungpinang: Jurusan Teknik Informatika, Fakultas Teknik dan Teknologi Kemaritiman, Universitas Maritim Raja Ali Haji. Pembimbing I: Hendra Kurniawan, S.Kom., M.Sc.Eng., Ph.D. Pembimbing II: Nurul Hayaty, S.T., M.Cs.

---

Tujuan dari penelitian ini adalah untuk memberikan gambaran penerapan sistem mengamankan transmisi data yang dilakukan pada suatu *web service* agar tercegah dari serangan pembacaan, pencurian, dan pengambilan data dari pihak yang tidak berwenang atau disebut sebagai serangan *man in the middle* jenis *sniffing* yang mana penerapan sistem enkripsi secara dua arah digunakan dalam mengatasi serangan ini yang mana enkripsi ini menggunakan algoritma *AES 128*. Alasan mengambil judul tersebut karena pentingnya pengamanan data yang menjadi prioritas dan salah satu topik permasalahan yang sering terjadi saat ini seperti kebocoran data dan lain sebagainya. Oleh karena itu peneliti mengajukan penelitian ini untuk menganalisis seberapa efektif penerapan sistem enkripsi dua arah bekerja dalam pencegahan serangan ini di *web service*. Hasil penerapan menunjukkan bahwa penerapan sistem yang diajukan berhasil diterapkan sehingga proses transmisi data berhasil diamankan dengan sistem enkripsinya. Percobaan serangan *man in the middle* juga dilakukan menggunakan tools Burp Suite, dan hasil percobaan juga menunjukkan keberhasilan dalam mencegah serangan *man in the middle* ini. Terdapat beberapa hal yang tentunya bisa ditingkatkan, seperti penerapan *pagination* pada *web service* dan pemilihan algoritma enkripsi yang lebih ringan sehingga meningkatkan waktu yang diperlukan agar lebih efisien dan efektif lagi.

**Kata kunci:** *Man In The Middle Attack, Sniffing, End To End Encryption, Web Service, Web Security, AES 128 bit*



## ABSTRACT

Riastanjung, O. 2024. Prevention of Man In The Middle Attacks on Web Service Data Transmission Using End To End Encryption (E2EE), Thesis. Tanjungpinang: Department of Informatics Engineering, Faculty of Maritime Engineering and Technology, University of Maritim Raja Ali Haji. Advisor I: Hendra Kurniawan, S.Kom., M.Sc.Eng., Ph.D. Advisor II: Nurul Hayaty, S.T., M.Cs.

---

The objective of this research is to provide an overview of the implementation of a data transmission security system on a web service to prevent unauthorized access, theft, and data retrieval, commonly known as man-in-the-middle attacks, specifically sniffing attacks. In this context, a two-way encryption system is employed to combat such attacks, using the AES 128 algorithm. The reason for choosing this topic is the importance of data security, which has become a priority and one of the frequently occurring issues today, such as data breaches and similar incidents. Therefore, the researcher proposes this study to analyze the effectiveness of the two-way encryption system in preventing these attacks on web services. The implementation results show that the proposed system was successfully applied, securing the data transmission process with its encryption system. Man-in-the-middle attack experiments were also conducted using the Burp Suite tool, and the results demonstrated success in preventing these attacks. There are several aspects that can be improved, such as the implementation of pagination on the web service and the selection of a lighter encryption algorithm to enhance the time efficiency and effectiveness further.

**Keywords:** *Man In The Middle Attack, Sniffing, End To End Encryption, Web Service, Web Security, AES 128 bit*

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Pada masa modern sekarang, teknologi informasi juga mengalami kemajuan yang signifikan, dan salah satu bentuk kemajuan teknologi informasi ialah *website* yang terhubung oleh *web service*. *Web service* atau layanan *web* merupakan sistem yang terdiri dari berbagai fungsi dan metode yang tersimpan pada suatu *server* dan dapat diakses oleh pengguna dengan sistem arsitektur aplikasi yang berbeda secara *remote* (Ardiansyah dkk., 2023).

Dengan terus majunya perkembangan teknologi, tidak selalu menghasilkan sesuatu yang positif. Dengan pesatnya perluasan pada bidang teknologi informasi di seluruh dunia juga diiringi oleh tingginya faktor risiko penyalahgunaan dan serangan terkait keamanan informasi yang dapat dilakukan oleh berbagai cara dan dilakukan oleh peretas (Amalia dan Nasution, 2024). Keamanan informasi di bidang informasi digital yang terus berkembang memerlukan sistem keamanan yang kuat dan tidak bisa diabaikan, dan kerentanannya memerlukan perhatian kritis (Setyowardhani dkk., 2024). Tujuan dari dibentuknya proses keamanan data adalah untuk melindungi suatu informasi yang telah disimpan, diterima, dikirim, atau dikumpulkan yang mana setiap proses tersebut suatu data atau informasi itu harus dilindungi (Wulandari dan Hwihanus, 2023).

Urgensi keamanan data atau informasi pada suatu *web service* menjadi hal yang perlu diberikan perhatian khusus. Faktor keamanan dalam suatu hubungan antara *client* atau pengguna dengan *server web service* itu belum tentu tejamin keamanannya, yang mana aspek keamanan ini sangat penting agar informasi data yang tersimpan tidak disalahgunakan atau diakses oleh sembarang pihak (Muzakir, 2013). Salah satu urgensi keamanan yang perlu diperhatikan pada *web service* yaitu serangan XSS, *SQL Injection*, *Brute Force Attack*, dan *Man In The Middle* jenis *Sniffing*. Beberapa serangan tersebut memberikan serangan dengan tingkat bahaya yang berbeda dimana *Man In The Middle* memiliki tingkat persentase serangan

tertinggi sebesar 85%, sedangkan XSS 45%, SQL Injection 75% dan Brute Force Attack 80% (Arnaldy dan Perdana, 2019). Hal inilah yang membuat penelitian ini akan berfokus pada serangan *web service* yaitu *Man In The Middle*.

Serangan *Man In The Middle Attack* (MITM) ini adalah kegiatan atau aktivitas dengan teknik melakukan penyerangan pada sisi keamanan, yang mana penyerang ini merupakan orang yang berada di tengah-tengah komunikasi yang dapat dengan bebas mendengarkan bahkan mengubah suatu pesan yang disampaikan (Zulkarnain, 2020). Ada beberapa teknik MITM salah satunya yaitu *Sniffing*. *Sniffing* merupakan suatu teknik yang digunakan oleh pihak tak berwenang, dimana mereka melakukan tindakan kejahatan berupa pencurian dan pengambilan suatu data sensitif dengan pemantauan pertukaran *packets* dalam suatu jaringan (Manguling dan Parenreng, 2023).

Terdapat beberapa cara pencegahan yang dapat dilakukan terkait penyerangan pada *web service Man In The Middle Attack* (MITM). Proses pencegahan yang disarankan diantara lain dengan proses enkripsi. Enkripsi adalah kegiatan yang merubah bentuk informasi yang dapat dipahami dengan mudah oleh seseorang ke bentuk kode yang akan sulit dipahami. Terdapat beberapa penerapan enkripsi yang dapat digunakan antara lain penggunaan SSL, penggunaan SSH, dan *End To End Encryption* (E2EE) untuk mencegah serangan MITM (Wulandari dan Hwihanus, 2023).

Dengan menerapkan *End To End Encryption* merupakan cara yang sangat efektif pada kasus melindungi transmisi pesan digital, kerahasiaan informasi, perlindungan dan privasi suatu informasi. Dari beberapa penerapan yang dapat digunakan, penelitian ini akan menitikberatkan pada *End To End Encryption* (E2EE) yang merupakan proses pertukaran informasi yang dilakukan dengan aman dari pengirim ke penerima dengan melakukan proses enkripsi selama proses pertukarannya yang menyebabkan informasi tidak dapat diakses dan diubah oleh pihak ketiga yang tidak berwenang (Maharani dkk., 2023).

Karena penelitian ini ingin menggunakan E2EE, maka perlu menetapkan suatu algoritma untuk enkripsinya. Terdapat beberapa algoritma enkripsi yang dikenal antara lain, DES (*Data Encryption Standard*), *Blowfish*, IDEA

(*International Data Encryption Algorithm*), dan AES (*Advanced Encryption Standard*). Dimana dari berbagai algoritma enkripsi tersebut menunjukkan bahwa AES merupakan algoritma terbaik dari kecepatan melakukan proses enkripsi dan proses dekripsinya dimana urutan kecepatan algoritma tersebut dari tercepat yaitu AES, *Blowfish*, DES, dan IDEA (Meko, 2018). Oleh karenanya, penelitian ini akan menggunakan algoritma AES sebagai algoritma enkripsi pada sistem *End To End Encryption* yang ingin diterapkan.

Berdasarkan pemilihan algoritma yang telah dipilih, *Advanced Encryption Standard* (AES) terdapat beberapa jenis yang dibedakan berdasarkan jumlah bit yang digunakan yaitu 128 bit, 192 bit dan 256 bit. AES adalah suatu algoritma dalam Kriptografi yang dikerjakan pada *blok cipher* yang menggunakan berbagai teknik seperti substitusi, permutasi dan rotasi pada setiap putaran di setiap blok yang akan dilakukan enkripsi dan dekripsi. AES 128 bit ini akan memiliki panjang kunci sebanyak 128 bit dengan jumlah proses sebanyak 10 putaran (Luqman dkk., 2022). AES 128 merupakan salah satu algoritma untuk mengamankan data yang teraman dengan percobaan serangan secara *Brute Force Attack* maka memerlukan sekitar 1 miliar tahun untuk memecahkan kode enkripsi nya (Basatwar, 2023).

Berdasarkan beberapa fenomena yang telah dijelaskan pada paragraph-paragraf sebelumnya, maka diperlukan suatu sistem keamanan yang mendukung dalam mengatasi urgensi keamanan data dari serangan MITM pada *web service*. Menurut Yaomulfurqqan dan Pramusinto (2023) juga menyatakan jika penerapan AES pada keamanan data suatu *website* sangat tepat karena penggunaannya yang fleksibel dalam pengaksesan dan penggunaannya.

Oleh karena itu, maka penulis penelitian ingin melakukan penelitian pada pencegahan serangan keamanan yang dapat terjadi pada *website* terutama pada *service* mendapatkan seluruh data pasien untuk diterapkan enkripsi dengan *End To End Encryption* pada sisi *Web Service (Server Backend)* dan *Web Client (Server Frontend)*.

## 1.2 Rumusan Masalah

Setelah memuat beberapa fenomena pada latar belakang yang telah dijelaskan pada subbab sebelumnya, maka pada penelitian ini merumuskan dan memfokuskan pokok masalah yang akan dikerjakan selama kegiatan penelitian ini berlangsung, yaitu bagaimana efektivitas penerapan *End To End Encryption* dengan AES 128 bit dalam mengatasi serangan keamanan *Man In The Middle Attack* dengan serangan *Sniffing*?

## 1.3 Batasan Masalah

Agar penelitian lebih terfokus sesuai dengan apa yang ingin dicapai, maka perlu ditetapkan batasan masalah pada penelitian ini yang mana batasan masalah bertujuan untuk membatasi bidang dan cakupan penelitian yang akan diteliti. Berikut beberapa batasan masalah pada penelitian ini :

1. Kegiatan penelitian ini hanya memfokuskan pada salah satu jenis serangan *Man In The Middle Attack* yaitu *Sniffing*.
2. Penelitian ini membatasi *service* dari *web* yang ingin diterapkan yaitu pada *service* pengambilan data.
3. Penelitian ini akan menggunakan *website dummy* yang dibuat untuk melakukan simulasi serangan.
4. Penelitian ini tidak mencakup evaluasi performa sistem secara umum.
5. Implementasi atau penerapan sistem akan dibuat dengan bahasa pemrograman Javascript dan beberapa library seperti Burp Suite, ExpressJS, CryptoJS, Prisma.io, dan PostgreSQL.

## 1.4 Tujuan Penelitian

Dilakukannya penelitian ini bertujuan untuk menganalisis, meneliti dan mengimplementasikan mekanisme atau sistem dalam pencegahan serangan keamanan pada *web service* yaitu *Man In The Middle Attack* dengan jenis *Sniffing* menggunakan *End To End Encryption* yang menerapkan Algoritma *Advanced Encryption Standard* (AES) dengan 128 bit dalam proses pertukaran data atau

informasi dari *Web Service (Server Backend)* dan *Web Client (Server Frontend)* yang ada pada *website*.

### 1.5 Manfaat Penelitian

Mengenai manfaat penelitian yang menjadi nilai dalam penelitian yang dilakukan ini baik untuk peneliti, dan pembaca baik itu dalam segi teoritis ataupun praktis, berikut kajian manfaat penelitian yang dihasilkan dari penelitian ini :

1. Manfaat secara Teoritis
  - a. Berkontribusi terhadap ilmu pengetahuan dalam penerapan sistem keamanan untuk pencegahan serangan yang dapat terjadi pada pertukaran informasi atau data.
  - b. Menambah wawasan literatur bagi peneliti maupun pembaca penelitian ini terkait pentingnya keamanan data dan bagaimana cara mengamankan data informasi tersebut.
  - c. Membantu mendalami kajian ilmu terkait pencegahan serangan pada *web service* dengan penerapan sistem keamanan dan metode enkripsi.
2. Manfaat secara Praktis
  - a. Memberikan wawasan cara mengimplementasikan keamanan informasi atau data pada saat pertukaran informasi antara *web service* dan *client*.
  - b. Menambah pengetahuan terkait kegiatan penyerangan yang dilakukan dengan metode MITM dengan jenis *Sniffing* untuk mendapatkan suatu informasi atau data dari *web service*.
  - c. Mengetahui cara penerapan AES 128 sebagai metode enkripsi suatu informasi atau data.

## **1.6 Sistematika Penulisan**

Proses penulisan tugas akhir ini dilakukan secara sistematis. Berikut adalah langkah-langkah yang diikuti dalam menyusun skripsi ini:

### **BAB I PENDAHULUAN**

Pada bab ini menjelaskan tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

### **BAB II KAJIAN LITERATUR**

Pada bab ini menjelaskan tentang penelitian-penelitian terdahulu, konsep, dan teori yang pernah digunakan dalam studi kasus dan metode yang sama.

### **BAB III METODE PENELITIAN**

Pada bab ini menjelaskan tentang fokus dan lama penelitian bahan atau materi penelitian, jenis data yang digunakan, alat pengumpulan data, alat atau instrumen penelitian, kerangka penelitian, pengumpulan data, serta analisa dan perancangan.

### **BAB IV HASIL DAN PEMBAHASAN**

Pada bab ini membahas tentang pengujian dan pembahasan dari aplikasi yang akan dibangun.

### **BAB V PENUTUP**

Pada bab ini membahas mengenai kesimpulan dan saran dari hasil penelitian.

### **DAFTAR PUSTAKA**

Pada bagian ini berisikan sumber-sumber yang digunakan untuk pendukung pada kajian literatur.

## **BAB II**

### **KAJIAN LITERATUR**

#### **2.1 Tinjauan Pustaka**

Khasanah dan Sutabri (2023) melakukan penelitian dengan judul “Analisis Pencegahan Pencurian Data Melalui Aplikasi Whatsapp Menggunakan Metode Kriptografi”. Penelitian ini membahas terkait penerapan fitur *End To End Encryption* yang ada dalam aplikasi Whatsapp yang menggunakan AES-256 dalam enkripsi pesan dari satu pengguna ke pengguna lain dengan tujuan penelitian ini untuk menyampaikan hasil fitur ini apakah bekerja dengan baik atau tidak. Hasil penelitian ini menyatakan penerapan *End To End Encryption* telah bekerja dengan sangat baik dalam pencegahan terjadinya pencurian informasi oleh pihak tidak berwenang.

Muzakir (2013) melakukan penelitiannya yang berjudul “Sistem Keamanan Data Pada *Web Service* Menggunakan *XML Encryption*”. Penelitian ini bertujuan untuk melakukan antisipasi serangan keamanan data terutama terkait kebocoran informasi yang dikirimkan dari *web service* ke aplikasi yang menggunakan layanannya yang dilakukan pada *web service* SOAP dengan penerapan *XML Encryption* pada request pesannya. Hasil yang ditunjukkan oleh penelitian ini juga menyebutkan jika penerapan *XML Encryption* bekerja dengan baik karena pesan *request* telah terenkripsi dengan baik, dan dapat didekripsi dengan baik pula.

Mouli dan Jevitha (2016) dengan penelitian mereka yang berjudul “*Web Services Attacks and Security- A Systematic Literature Review*”. Penelitian ini bertujuan untuk mengumpulkan data dari berbagai kasus yang terjadi terkait penyerangan yang dapat dilakukan pada *web service* dengan hasil penelitian menyebutkan penyerangan yang dapat dilakukan yaitu *XML Injection*, *XPath Injection*, *SQL Injection*, *Spoofing*, *Denial Of Service*, dan *Man In The Middle Attack*.

Amalia dan Nasution (2024) dengan penelitian mereka yang berjudul "Tantangan Terkini Dalam Keamanan Informasi Analisis Resiko Dan Upaya Perlindungan" yang menjelaskan terkait pentingnya keamanan di masa kini dengan



beberapa aspek yang perlu diperhatikan terutama pada suatu sistem informasi dan teknologi yang terdiri atas aspek kerahasiaan, integritas dan, ketersediaan suatu informasi. Penelitian ini juga memberikan beberapa contoh serangan yang dapat dilakukan pada aspek tersebut, contohnya serangan keamanan kerahasiaan informasi dengan *Man In The Middle Attack* dan upaya pencegahan yang dapat dilakukan yaitu dengan penerapan kriptografi.

Wulandari dan Hwihanus (2023) yang menulis penelitian dengan judul "Peran Sistem Informasi Akuntansi Dalam Pengaplikasian Enkripsi Terhadap Peningkatan Keamanan Perusahaan". Paper ini menjelaskan pentingnya sistem keamanan dalam suatu sistem informasi akuntansi atau SIA terutama pada keamanan informasi dan pentingnya keamanan data tersebut. Pada paper ini menerapkan SSL, SSH, dan *End To End Encryption* dalam mengatasi upaya pembobolan keamanan sistem informasi akuntansi. Hasil paper ini menyatakan penerapan beberapa metode enkripsi ini sangat membantu dalam mengamankan data informasi untuk SIA tersebut.

Awalsyah dkk, (2023) dengan penelitiannya yang berjudul "Implementasi *Caesar Cipher* Dalam Mengenkripsikan Pesan Pada Serangan *Man In The Middle Attack*" yang memaparkan tentang penerapan metode enkripsi dengan substitusi *Caesar Cipher* dalam pesan yang dikirimkan untuk mencegah dan menghalau serangan MITM oleh pihak yang tidak berwenang yang mencoba mengintai dan memanipulasi pesan yang dilakukan antar dua entitas dengan hasil penelitian yang menyatakan bahwa dengan penerapan *Caesar Cipher* yang sederhana namun cukup baik untuk mengamankan suatu pesan.

Zulkarnain (2020) yang menulis penelitiannya dengan judul "Analisis Keamanan FTP Server Menggunakan Serangan *Man-In-The-Middle Attack*" menjelaskan penelitiannya yaitu penerapan MITM untuk menguji keamanan FTP Server yang dimana hasil penerapan serangan dengan MITM dapat mengakses data informasi pada pertukaran pada kasus ini yaitu pertukaran file dengan informasi *username* dan *password* yang jebol dari serangan MITM yang menggunakan teknik *Sniffing* pada jaringan. Dalam upaya pencegahannya, penelitian ini menerapkan OpenSSL untuk pencegahannya, namun hasil penelitian juga menyatakan bahwa

penerapan OpenSSL juga belum terjamin aman karena dapat di jebol oleh seseorang yang ahli atau berpengalaman di bidang *hacking* dengan melakukan pemalsuaan sertifikat untuk autentikasi di server.

Manguling dan Parenreng (2023) yang melakukan penelitian dengan judul "*Security System Analysis Using The HTTP Protocol Against Packet Sniffing Attacks*". Penelitian ini menjelaskan serangan menggunakan MITM berhasil memantau pertukaran informasi pada *web service* yang menggunakan HTTP Protocol dalam layanan *service* nya, dengan menggunakan dua tools yaitu Ettercap dan Wireshark, penelitian ini menyatakan bahwa informasi dapat dibaca pada pertukaran jaringan yang dilakukan pada *website* SIM MBKM, oleh karenanya penelitian ini juga menyarankan solusi untuk penerapan sertifikasi seperti HTTPS sebagai protocol *website*.

Maharani dkk, (2023) dengan judul penelitiannya "*Kekuatan Enkripsi End-to-End: Kajian Literatur Mengenai Kerahasiaan Komunikasi Digital dalam Aplikasi Pesan Instan*". Penelitian ini memaparkan kekuatan metode E2EE dalam melakukan enkripsi pada aplikasi pesan instan untuk menjaga privasi dan keamanan informasi dengan hasil penelitiannya menyatakan E2EE memiliki keunggulan dalam memberikan lapisan keamanan yang kuat dalam mengamankan data.

Cristy dan Riandari (2021) menulis penelitian dengan judul "*Implementasi Metode Advanced Encryption Standard (AES 128 bit) Untuk Mengamankan Data Keuangan*". Penelitian ini memaparkan penjelasan dengan rinci langkah-langkah menggunakan atau mengimplementasikan enkripsi dengan AES 128 bit baik dalam pengenkripsian dan juga pembuatan *key* untuk enkripsi nya, dan juga diikuti langkah-langkah melakukan dekripsi yang diterapkan pada suatu informasi dalam kasus ini diterapkan pada file keuangan pada SMK Harapan Bangsa. Hasil penelitian berhasil menerapkan proses enkripsi dan dekripsi dengan AES 128 Bit.

## **2.2 Landasan Teori**

### **2.2.1 Sistem Keamanan**

Menurut Adam dan Romli (2024) sistem keamanan itu merupakan sekumpulan rangkaian yang melindungi data atau informasi dari suatu kegiatan

yang memodifikasi, merubah ataupun mengungkapkan informasi tersebut oleh orang yang tidak berwenang dan tidak memiliki hak akses yang tidak hanya terbatas pada informasi digital namun dapat juga berupa suatu fisik pada perangkat keras dan juga manajemen informasi tersebut..

Menurut Wulandari (2023) sistem keamanan informasi merupakan suatu upaya untuk mencegah kejadian tertentu yang berkaitan dengan keamanan sistem informasi seperti risiko kehilangan data, kerahasiaan dan integritas dari suatu informasi yang dapat dilakukan oleh pihak tidak bertanggungjawab dengan beberapa tindakan seperti pengaksesan secara tidak sah pada suatu computer, database dan situs web.

Dari beberapa penjelasan terkait sistem keamanan, maka dapat disimpulkan jika sistem keamanan merupakan serangkaian upaya dalam melakukan pencegahan penyerangan oleh pihak yang tidak berwenang yang dapat melakukan kegiatan yang menyebabkan hilangnya data, rahasia dan interitas data dengan memodifikasi, merubah ataupun membocorkan informasi baik itu dalam informasi digital maupun produk fisik.

### **2.2.2 *Man In The Middle Attack***

Menurut Awalsyah dkk, (2023) *Man In The Middle Attack* merupakan kegiatan serangan keamanan yang dilakukan oleh seseorang yang disebut *attacker* yang menjadi pihak yang berada di tengah dan ia bebas mendengarkan bahkan mengubah percakapan antara dua pihak.

Menurut Zulkarnain (2020) *Man In The Middle Attack* ialah serangan yang dilakukan oleh orang yang berada di tengah suatu pertukaran informasi yang memanfaatkan kelemahan *Internet Protocol* (IP) yang memungkinkan penyerang untuk membaca, mengubah hingga mengontrol data atau informasi yang dikirim antar dua computer tersebut.

Menurut Zulkarnain (2020) terdapat beberapa teknik dalam melakukan penyerangan *Man In The Middle Attack* antara lain, *Sniffing*, *Spoofing*, dan *Modification*.

#### **A. *Sniffing***

Menurut Manguling dan Parenreng (2023) *Sniffing* merupakan suatu bentuk kejahatan komputer dimana penyerang dapat mencuri dan mengumpulkan informasi penting yang didapatkan dengan membaca pengiriman informasi yang dilakukan pada suatu jaringan.

### **B. *Spoofing***

Menurut Zulkarnain (2020) *Spoofing* merupakan suatu bentuk kejahatan yang dilakukan oleh penyerang dengan menyamar menjadi *user* dengan melakukan pemalsuan informasi data yang kemudian mendapatkan suatu keuntungan yang tidak sah.

### **C. *Modification***

Menurut Zulkarnain (2020) *Modification* merupakan suatu bentuk kejahatan yang dilakukan oleh penyerang tidak bertanggungjawab yang melakukan perubahan terhadap integritas atau keaslian dari suatu informasi.

### **D. Kriptografi**

Menurut Khasanah dan Sutabri (2023) kriptografi merupakan ilmu matematika dengan seperangkat perhitungan yang berbasis algoritma atau aturan dengan acuan informasi dan teknik komunikasi yang juga menjadi solusi dari suatu masalah keamanan.

Tujuan penerapan kriptografi ialah untuk menjaga aspek keamanan informasi menurut Khasanah dan Sutabri (2023), diantaranya ialah :

1. Kerahasiaan, merupakan suatu layanan untuk menjaga isi informasi dari siapapun yang tidak memiliki otoritas untuk membuka informasinya.
2. Integritas data, merupakan suatu sistem yang wajib memiliki suatu kemampuan untuk mempertahankan isi dari suatu informasi.
3. Autentikasi, merupakan suatu sistem yang dapat mengidentifikasi pihak-pihak yang memiliki akses untuk membaca suatu informasi.

Menurut Herlambang dkk, (2024) kriptografi merupakan bidang ilmu yang menggunakan teknik-teknik matematika yang berhubungan dengan aspek dari keamanan yang diterapkan untuk mengubah pesan asli atau plainteks menjadi pesan yang telah disandikan atau cipherteks begitu pula sebaliknya.

Pada kriptografi terdapat istilah enkripsi, dekripsi, plainteks dan juga cipherteks. Menurut Herlambang dkk, (2024) enkripsi merupakan suatu proses mengubah plainteks (teks awal) menjadi bentuk yang sulit untuk dipahami dengan menggunakan suatu algoritma, sedangkan dekripsi merupakan proses yang mengembalikan cipherteks (teks yang telah terenkripsi) menjadi bentuk awal seperti asal mulanya.

Menurut Herlambang dkk, (2024) ada dua kategori dalam membuat suatu *key* atau kunci dalam melakukan kegiatan enkripsi dan dekripsi yaitu :

1. Cipher alir (*stream cipher*) yang dilakukan dengan memproses setiap bit tunggal untuk dilakukan enkripsi dan dekripsi. Contoh algoritma kategori ini ialah *Caesar Cipher*.

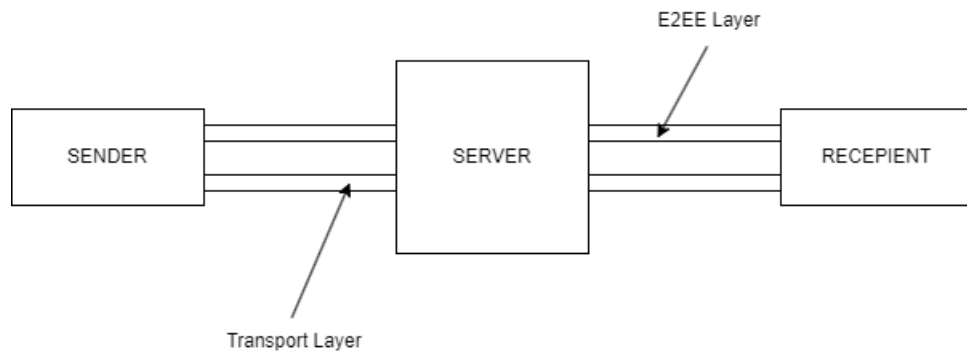
Cipher blok (*block cipher*) yang memproses enkripsi dan dekripsi dalam bentuk blok-blok bit. Contoh algoritma kategori ini ialah *Advanced Encryption Standard*.

### **2.2.3 End To End Encryption**

Menurut Bai dkk, (2020) *End To End Encryption* merupakan suatu cara yang paling baik dan dikenal dalam melakukan perlindungan informasi digital dalam suatu komunikasi dalam mencegah pihak ketiga yang tidak memiliki akses untuk membaca informasi.

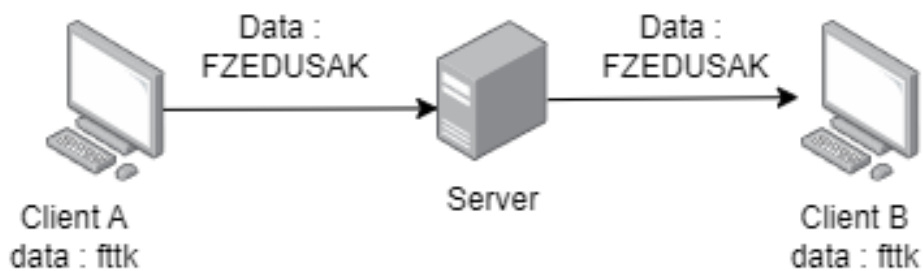
Menurut Maharani dkk, (2023) *End To End Encryption* adalah proses yang melakukan pertukaran data informasi yang aman dari satu pengguna ke pengguna lain dengan proses enkripsi pada informasi sehingga tidak dapat diakses pihak lain.

Menurut Lestari (2022) E2EE seperti memiliki jalur sendiri dalam melakukan komunikasi yang hanya menjadi perumpamaan saja karena jalur tersebut merupakan suatu pembungkus informasi, dan terdapat jalur *Transport* yang menjadi jalur pesan terenkripsi yang menghubungkan *web client* dan *web server*.



**Gambar 1.** Konsep *End To End Encryption* (Lestari, 2022)

Berikut merupakan ilustrasi sistem kerja E2EE pada *web* yang dirujuk pada gambar oleh Liander (2022).



**Gambar 2.** *End To End Encryption* pada *web* (Liander, 2022)

#### 2.2.4 *Advanced Encryption Standard (AES)*

Menurut Cristy dan Riandari (2021) algoritma *Advanced Encryption Standard* merupakan salah satu metode enkripsi dengan menggunakan blok cipher dan menggunakan kunci simetri di proses enkripsi dan juga proses dekripsinya.

Menurut Yaomulfurqqan dan Pramusinto (2023) AES adalah algoritma yang menggunakan transformasi terhadap state yang dilakukan secara berulang dan dilakukan dalam beberapa ronde.

**Tabel 1.** Perbedaan Jenis AES (Yaomulfurqqan dan Pramusinto, 2023)

Tipe	Jumlah Key (Nk)	Besar Blok (Nb)	Jumlah Round (Nr)
------	-----------------------	-----------------------	-------------------------

AES 128	4	4	10
AES 192	6	4	12
AES 256	8	4	14

Menurut Ridho dan Kusumaningish (2023) AES berdasarkan jumlah bloknnya dibagi menjadi AES 128, AES 192 dan AES 256. Perbedaan antar jenis AES terdapat pada jumlah key (Nk), besar blok (Nb), dan jumlah round (Nr).

### 2.2.5 Enkripsi AES 128

Menurut Daemen dan Rijamen (1999) ada beberapa proses yang menjadi spesifikasi algoritma AES 128 dengan beberapa proses transformasi yang dilakukan dalam enkripsi dengan AES 128, yaitu :

#### 1. *SubBytes*

Merupakan suatu proses merubah setiap *byte* pada state dan dilakukan substitusi dengan table subsitusi atau S-Box. Berikut merupakan table S-Box.

		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 3.** Tabel S-Box

#### 2. *ShiftRows*

Merupakan suatu proses menggeser bit dimana bit paling kiri menjadi bit paling kanan atau disebut juga rotasi bit. Pada baris pertama tidak dilakukan rotasi bit, untuk baris kedua dilakukan rotasi bit sekali, untuk

baris ketiga dilakukan rotasi bit dua kali, dan untuk baris keempat dilakukan rotasi bit tiga kali.

### 3. *MixColumns*

Merupakan proses mengkalikan setiap kolom dari matriks state dengan matriks konstan *Galois field MixColumns* dimana proses penjumlahan diganti menjadi proses XOR antar hex.

Matriks konstan *Galois field* merupakan matriks 4x4 berisi nilai hex dengan nilai =

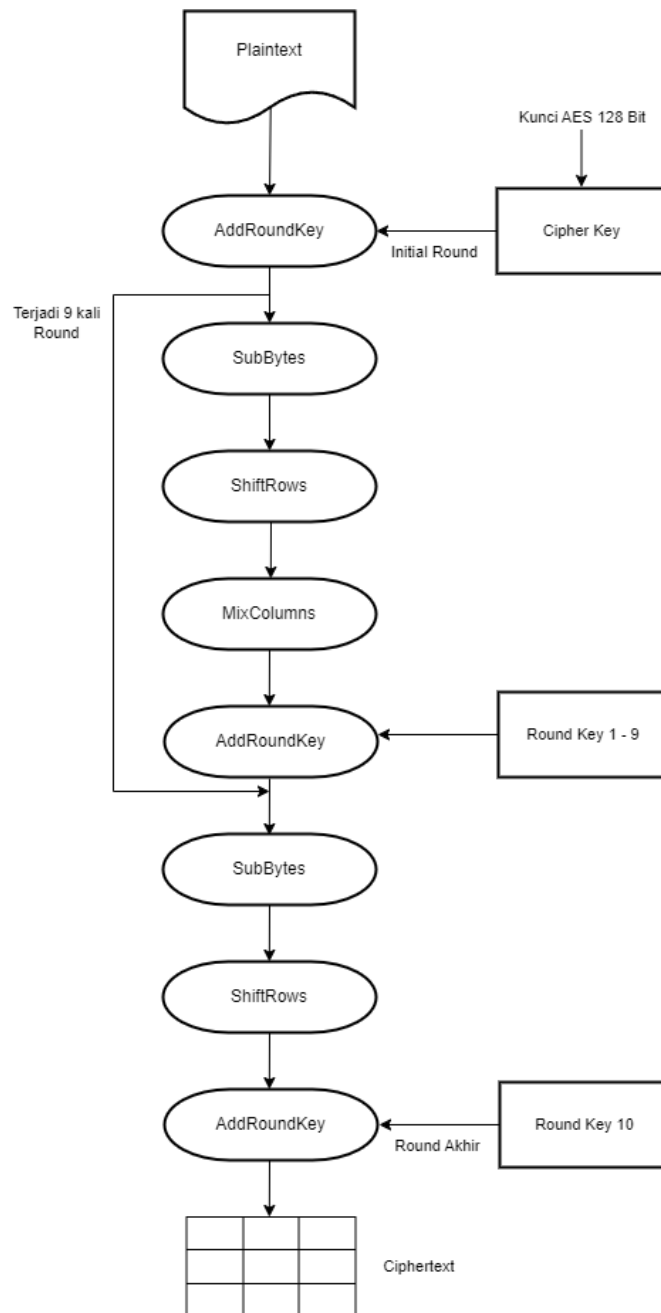
02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

### 4. *AddRoundKey*

Merupakan proses XOR setiap byte pada *byte* matriks state dengan matriks *roundkey* sesuai pada putaran ke-n.

Menurut Yaomulfurqqan dan Pramusinto (2023) berikut alur lengkap proses enkripsi dengan AES 128 :





**Gambar 4.** Alur Enkripsi AES 128 (Yaomulfurqqan dan Pramusinto, 2023)

### 2.2.6 Dekripsi AES 128

Menurut Daemen dan Rijmen (1999) pada proses dekripsi AES merupakan langkah kebalikan dari langkah-langkah proses enkripsi, dan terdapat beberapa proses transformasi yang dikenal dalam algoritma dekripsi AES 128, yaitu :

1. *AddRoundKey*

Merupakan proses XOR setiap *byte* matriks state dari cipherteks dengan *byte* matriks *roundkey* dengan urutan iterasi yang berkebalikan.

2. *Inverse SubBytes*

Merupakan kebalikan proses *SubBytes* dimana dilakukan proses substitusi tiap *byte* dengan table *Inverse S-Box*.

		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 5.** Tabel *Inverse S-Box*

3. *Inverse ShiftRows*

Merupakan proses kebalikan dari *ShiftRows* dimana dilakukan pergeseran bit ke kanan dengan jumlah persgeseran yang sama pada *ShiftRows* sesuai barisnya.

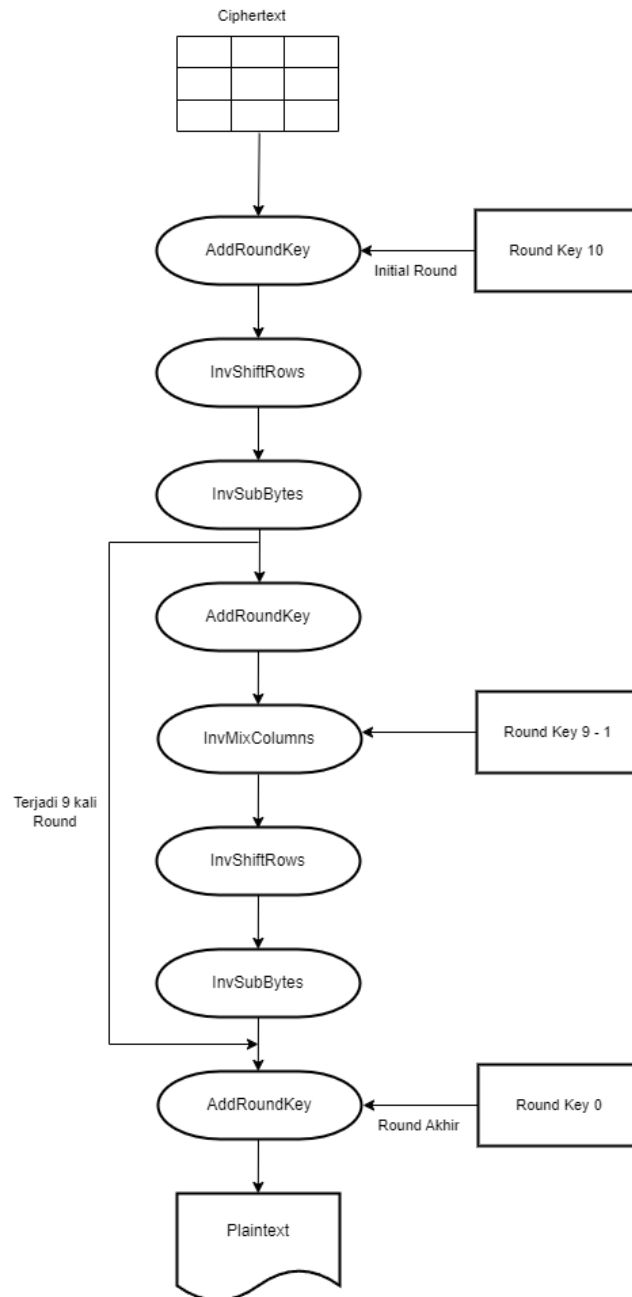
4. *Inverse MixColumns*

Merupakan proses perkalian matriks state dengan matriks konstan *Inverse MixColumns* dengan pergantian penjumlahan menjadi operasi XOR.

Matriks konstan *Inverse MixColumns* merupakan matriks 4x4 berisi nilai hex dengan nilai =

0E	0B	0D	09
09	0E	0B	0D
0D	09	0E	0B
0B	0D	09	0E

Menurut Yaomulfurqqan dan Pramusinto (2023) berikut alur lengkap proses dekripsi dengan AES 128 :



**Gambar 6.** Alur Dekripsi AES 128 (Yaomulfurqqan dan Pramusinto, 2023)

### 2.2.7 Web Service

Menurut Baharuddin dkk, (2022) *web service* adalah sebuah layanan perangkat lunak yang menjadi penghubung antar berbagai sitem agar dapat saling

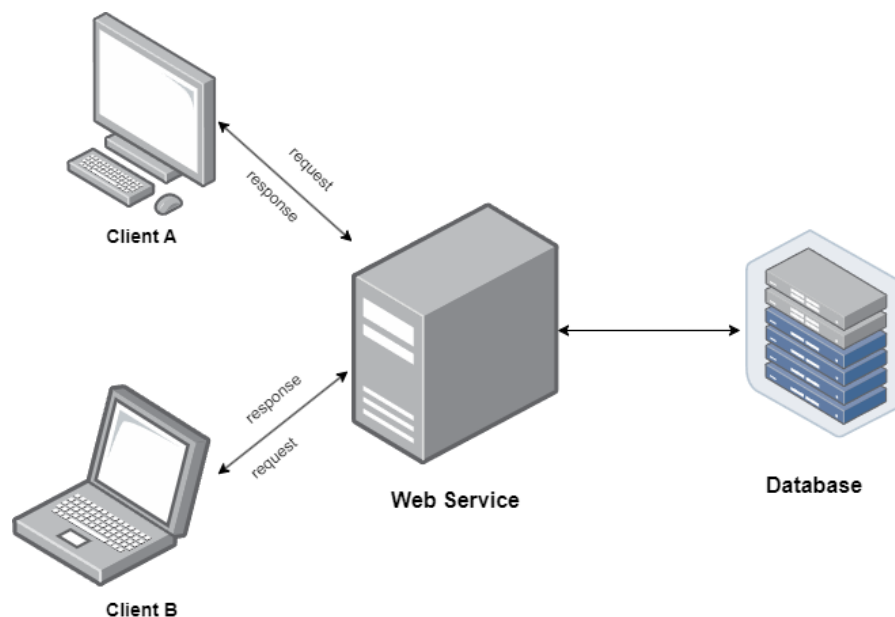
berkomunikasi meskipun dengan perbedaan platform dan tidak secara langsung terhubung dengan *database*.

Menurut Huda (2022) *web service* berdasarkan arsitekturnya terbagi menjadi beberapa jenis, antara lain SOAP (*Simple Object Access Protocol*) yang menggunakan dokumen XML sebagai pertukaran data antar sistem dan REST (*Representational State Transfer*) yang menggunakan data JSON.

Menurut Salahudin (2022) REST menggunakan beberapa metode yang berbasis protokol HTTP dan beberapa yang populer antara lain :

1. POST, yang digunakan untuk menambahkan data baru
2. GET, yang digunakan untuk mengambil data
3. PUT, yang digunakan untuk melakukan pembaruan data yang telah ada
4. DELETE, yang digunakan untuk menghapus suatu data

Berikut merupakan penggambaran cara kerja pada *web service* REST dirujuk pada gambar oleh Simbulan dan Aryanto (2024).



**Gambar 7.** *Web Service* REST (Simbulan dan Aryanto, 2024)

Pada *Web Service* mengenal istilah *Server Backend* sebagai pusat dibentuknya *web service* dan juga *Server Frontend* yang menjadi *website client* yang terhubung dan menjalin transmisi data dengan *server backend*.

### **2.2.8 Burp Suite**

Menurut Elshoush dkk, (2023) Burp Suite merupakan salah satu alat yang populer untuk melakukan suatu serangan pada jaringan yang dapat menemukan kerentanan dan digunakan khusus pada keamanan dari suatu aplikasi *web*. Dimana dengan penggunaan Burp Suite dapat memantau seluruh pesan HTTPS yang telah dikirimkan menggunakan proxy yang tersedia. Dengan proses pemantauan ini maka serangan *Man In The Middle Attack* dapat dilakukan.

## BAB III

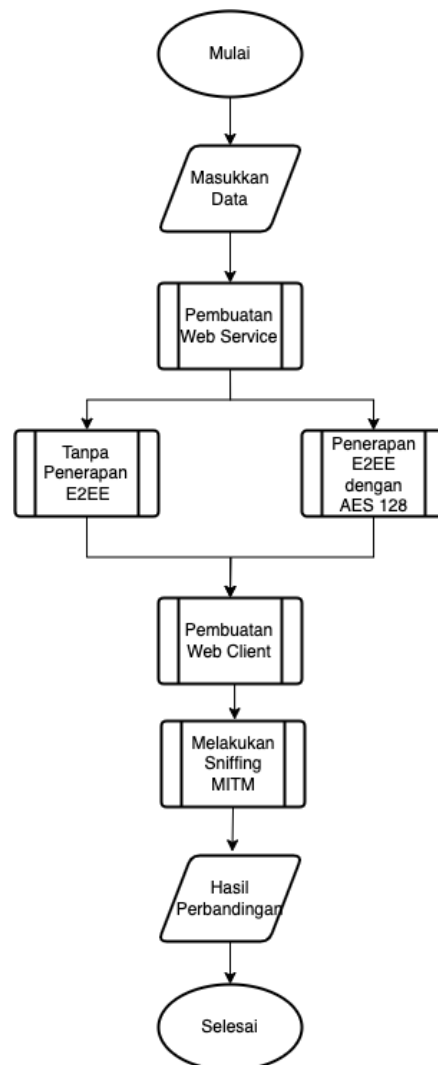
### METODE PENELITIAN

#### 3.1 Waktu Penelitian

Penelitian ini akan dilakukan selama 6 bulan pada bulan Januari s/d Juni tahun 2024.

#### 3.2 Tahapan Penelitian

Penelitian ini akan dilakukan dengan tahapan kegiatan yang dilakukan seperti gambar berikut.



**Gambar 8.** Tahapan Penelitian

Berdasarkan gambar 8, akan dilakukan proses memasukkan data-data pasien terlebih dahulu pada *database* kemudian akan dibuatkan *web service* yang akan menerima *service* untuk mendapatkan kumpulan data pasien stroke tersebut. Disini akan dilakukan atau dibuat dua *service* yang nantinya akan dibandingkan sebelum dan sesudah penerapan E2EE. Kemudian suatu aplikasi sederhana *client* berbasis website akan dibuat, dimana *client* ini akan menggunakan *service* dari *web service*.

Kemudian akan dilakukan percobaan serangan MITM dengan *Sniffing* menggunakan tools Burp Suite. Hasil percobaan serangan ini akan dijadikan hasil penelitian nantinya yang kemudian akan dibandingkan sebelum dan penerapan E2EE pada *web service* tersebut.

### **3.3 Jenis Penelitian**

Penelitian ini memiliki jenis penelitian kuantitatif, yaitu penelitian yang melibatkan suatu pemrosesan terhadap data numerik. Pada penelitian ini akan berfokus pada perubahan suatu *input* yang berupa suatu kalimat dimana setiap *byte* nya akan dirubah menjadi bilangan desimal berdasarkan ASCII dan kemudian dirubah menjadi heksadesimal dan nantinya akan dikenakan beberapa operasi seperti XOR. Adanya perubahan menjadi desimal dan heksadesimal yang kemudian ada operasi XOR ini yang menunjukkan penelitian ini merupakan penelitian terhadap data numerik atau penelitian kuantitatif.

### **3.4 Alat atau Instrumen Penelitian**

Adapun beberapa tools yang akan digunakan selama proses implementasi nya, yaitu bahasa pemrograman Javascript, PostgreSQL sebagai *database* seluruh data yang telah dikumpulkan, NextJS sebagai framework membuat aplikasi *web client*, ExpressJS sebagai framework membuat aplikasi *web service*, dan CryptoJS sebagai package yang akan melakukan enkripsi dan dekripsi AES 128 bit. Lalu adapun tools untuk melakukan uji serangan MITM *Sniffing* yaitu menggunakan aplikasi Burp Suite.

Untuk instrumen penelitian menggunakan laptop penulis yang memiliki spesifikasi seperti berikut :

**Tabel 2. Instrumen Penelitian**

Spesifikasi	Keterangan
<i>Processor</i>	AMD Ryzen 3 3300
<i>Memory</i>	16GB
<i>Operating System</i>	Windows 11 64 bit

### **3.5 Analisis dan Perancangan**

#### **3.5.1 Analisis Data**

Data yang digunakan merupakan data sekunder yang didapatkan dari situs Kaggle berupa data pasien stroke yang didapatkan pada situs <https://www.kaggle.com/code/tilii7/modeling-stroke-dataset-with-lasso-regression/input?select=train.csv>. Data tersebut terdapat beberapa atribut, yaitu : *id*, *gender*, *age*, *hypertension*, *heart\_disease*, *ever\_married*, *work\_type*, *Residence\_type*, *avg\_glucose\_level*, *bmi*. Setelah pengumpulan maka akan digunakan sampel data untuk nantinya digunakan saat implementasi yang disimpan dalam *database* dan kemudian akan digunakan pula dalam proses pengambilan data dengan menggunakan *web service*.

Data yang dikumpulkan menggunakan 16 data sampel dari keseluruhan data yang dimana beberapa data tersebut dimasukkan ke *database* PostgreSQL yang mana terlihat seperti berikut :



no	id	gender	age	hypertension	heart_disease	ever_married	work_type	residence_type	avg_glucose_level	bmi	smoking_status	stroke
1	cltodaelu00005r9impjkaqaz	Male	33	false	false	false	Private	Rural	78,44	23,9	formerly smoked	false
2	cltodaelu00015r9igimb15i	Female	42	false	false	false	Private	Rural	103	40,3	Unknown	false
3	cltodaem300025r9ifyfmmng	Male	56	false	false	false	Private	Urban	64,87	28,8	never smoked	false
4	cltodaem300035r9i8wp06j2u	Female	24	false	false	false	Private	Rural	73,36	28,8	never smoked	false
5	cltodaem300045r9iv0mi6a81	Female	34	false	false	false	Private	Urban	84,35	22,2	Unknown	false
6	cltodaem300055r9i8cj9wn2m	Female	53	false	false	false	Private	Rural	88,97	25,3	never smoked	false
7	cltodaem300065r9i7rozkbnt	Male	78	false	true	false	Self-employed	Rural	75,32	24,8	Unknown	false
8	cltodaem300075r9iju7yovns	Female	45	false	false	false	Private	Rural	107,22	34,1	never smoked	false
9	cltodaem300085r9iyod2cyf6	Female	62	false	false	false	Govt_job	Urban	62,68	18,4	formerly smoked	false
10	cltodaem300095r9i797m9uoi	Male	51	false	false	false	Self-employed	Urban	114,89	20,1	never smoked	false
11	cltodaem3000a5r9igtgm0ar	Female	45	false	false	false	Private	Urban	69,94	23,5	never smoked	false
12	cltodaem3000b5r9ie9ni4e14	Female	4	false	false	false	children	Urban	84,1	14,1	Unknown	false
13	cltodaem3000c5r9iqg4g52wb	Male	23	false	false	false	Private	Urban	112,09	37,3	smokes	false
14	cltodaem4000d5r9iqsepd4j4	Female	36	false	false	false	Private	Rural	73,78	29,6	never smoked	false
15	cltodaem4000e5r9i8gz1v5e9	Female	59	false	false	false	Private	Rural	100,54	28,1	never smoked	false
16	cltodleam00008gu55nsch7v	Male	28	false	false	false	Private	Urban	79,53	31,1	never smoked	false

**Gambar 9.** Data sampel yang digunakan

Data ini yang akan menjadi acuan untuk dilakukannya pengujian keberhasilan proses sistem nantinya selama diterapkan proses *End To End Encryption* dengan algoritma AES 128.

### 3.5.2 Perancangan Algoritma Enkripsi AES 128

Pada proses perancangan enkripsi AES 128 ini akan diterapkan menggunakan sebuah *input* berupa plainteks yang akan dilakukan enkripsi terlebih dahulu. Proses ini melewati dua langkah penting yaitu Ekspansi Key untuk membuat *RoundKey* dan proses Enkripsi.

Disini plainteks yang akan digunakan yaitu “INFORMATIKA HORE” dengan key nya yang wajib 128 bit atau 16 digit maksimal yaitu “FTTKUMRAHBERJAYA”. Disini seluruh karakter akan dirubah menjadi bentuk heksadesimal berdasarkan tabel ketetapan heksadesimal ASCII.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

**Gambar 10.** ASCII Heksadesimal (Hamzah, 2014)

### A. Ekspansi Key

Dalam proses ini akan dibuat 10 ronde untuk *RoundKey* sesuai dengan algoritma AES 128 bit yang membutuhkan 10 *RoundKey*. Berikut langkah-langkah yang dilakukan :

1. Mengurutkan plainteks kunci dan dirubah setiap karakter kunci menjadi bentuk heksadesimal dalam blok 16 digit dari kode ASCII.

F	T	T	K	U	M	R	A	H	B	E	R	J	A	Y	A
46	54	54	4B	55	4D	52	41	48	42	45	52	4A	41	59	41

2. Merubah kunci ke dalam *state* 4x4 dengan urutan penyusunan ke bawah dahulu.

46	55	48	4A
54	4D	42	41
54	52	45	59
4B	41	52	41

3. Melakukan fungsi *RotWord*, yang merupakan fungsi merubah kolom ke empat pada RoundKey ke-0 untuk merubah tiap *byte* naik ke atas satu kali.

4A
41
59
41

Setelah dilakukan *RotWord* menjadi :

41
59
41
4A

4. Melakukan substitusi kolom yang telah dilakukan *RotWord* dengan nilai pada S-Box (*SubBytes*).

Misal heksadesimal 41 disubstitusi yaitu 4 (pada kolom x) dan 1(pada kolom y) pada tabel S-Box menjadi 83.

		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 11.** Substitusi S-BOX

41
59
41
4A

Setelah dilakukan *SubBytes* :

83
CB
83
D6

5. Untuk kolom pertama dalam membuat suatu *RoundKey* Lakukan operasi XOR pada kolom pertama *RoundKey* ke-i dengan state hasil *SubBytes* dan

juga XOR terhadap *Round Constant (Rcon)* yang merupakan matriks konstan dengan nilai matriks sesuai ronde nya.

	01	02	04	08	10	20	40	80	1B	36
	00	00	00	00	00	00	00	00	00	00
	00	00	00	00	00	00	00	00	00	00
	00	00	00	00	00	00	00	00	00	00
i	1	2	3	4	5	6	7	8	9	10

Nilai heksadesimal 46 jika dirubah menjadi desimal maka bernilai 70 dan nilai biner dari 70 yaitu 01000110, lalu heksadesimal 83 nilai desimalnya 131 dan dalam biner yaitu 010000011, dan terakhir heksadesimal 01 dalam desimal bernilai 1 dengan biner nya yaitu 01.

Sehingga operasi XOR pada 1 baris ini yaitu :

heksadesimal	biner
46	01000110
83	010000011
	XOR
Hasil :	11000101
01	01
	XOR
Hasil : <b>C4</b>	11000100

Berikut perhitungan seluruh nya :

46	$\oplus$	83	$\oplus$	01	=	<b>C4</b>
54	$\oplus$	CB	$\oplus$	00	=	<b>9F</b>
54	$\oplus$	83	$\oplus$	00	=	<b>D7</b>

Kolom pertama

$$\boxed{4B} \oplus \boxed{D6} \oplus \boxed{00} = \boxed{9D}$$

Keterangan :

$\oplus$  = Operasi XOR

6. Untuk kolom selanjutnya, dilakukan operasi XOR antar kolom pada *RoundKey* ke-0 dengan kolom hasil *SubBytes* sebelumnya.

55	$\oplus$	C4	=	<b>91</b>
4D	$\oplus$	9F	=	<b>D2</b>
52	$\oplus$	D7	=	<b>85</b>
41	$\oplus$	9D	=	<b>DC</b>

Kolom kedua

48	$\oplus$	91	=	<b>D9</b>
42	$\oplus$	D2	=	<b>90</b>
45	$\oplus$	85	=	<b>C0</b>
52	$\oplus$	DC	=	<b>8E</b>

Kolom ketiga

4A	$\oplus$	D9	=	<b>93</b>
41	$\oplus$	90	=	<b>D1</b>
59	$\oplus$	C0	=	<b>99</b>
41	$\oplus$	8E	=	<b>CF</b>

Kolom keempat

7. Gabungkan seluruh kolom tersebut dan akan terbentuk *RoundKey* 1.

*RoundKey* ke-1 :

<b>C4</b>	<b>91</b>	<b>D9</b>	<b>93</b>
<b>9F</b>	<b>D2</b>	<b>90</b>	<b>D1</b>
<b>D7</b>	<b>85</b>	<b>C0</b>	<b>99</b>
<b>9D</b>	<b>DC</b>	<b>8E</b>	<b>CF</b>

Proses tersebut dilakukan sebanyak 10 ronde untuk mengumpulkan *RoundKey* ke-1 hingga *RoundKey* ke-10.

F8	69	B0	23
71	A3	33	E2
5D	D8	18	81
41	9D	13	DC
<i>RoundKey ke-2</i>			

64	0D	BD	9E
7D	DE	ED	0F
DB	03	1B	9A
67	FA	E9	35
<i>RoundKey ke-3</i>			

1A	17	AA	34
C5	1B	F6	F9
4D	4E	55	CF
6C	96	7F	4A
<i>RoundKey ke-4</i>			

93	84	2E	1A
4F	54	A2	5B
9B	D5	80	4F
74	E2	9D	D7
<i>RoundKey ke-5</i>			

8A	0E	20	3A
CB	9F	3D	66
95	40	C0	8F
D6	34	A9	7E
<i>RoundKey ke-6</i>			

F9	F7	D7	ED
B8	27	1A	7C
66	26	E6	69
56	62	CB	B5
<i>RoundKey ke-7</i>			

69	9E	49	A4
41	66	7C	00
B3	95	73	1A
03	61	AA	1F
<i>RoundKey ke-8</i>			

11	8F	C6	62
E3	85	F9	F9
73	E6	95	8F
4A	2B	81	9E
<i>RoundKey ke-9</i>			

BE	31	F7	95
90	15	EC	15
78	93	0B	84
E0	CB	4A	D4
<i>RoundKey ke-10</i>			

## B. Enkripsi

Proses enkripsi AES 128 akan dilakukan dengan urutan pengerjaan yang telah dijelaskan pada landasan teori untuk melakukan enkripsi tersebut. Berikut penjelasannya :

1. Mengurutkan plainteks *input* yaitu “INFORMATIKA HORE” dan dirubah setiap karakter kunci menjadi bentuk heksadesimal dalam blok 16 digit dari kode ASCII.

I	N	F	O	R	M	A	T	I	K	A		H	O	R	E
49	4E	46	4F	52	4D	41	54	49	4B	41	20	48	4F	52	45

2. Merubah plainteks heksadesimal ke dalam *state* 4x4 dengan urutan penyusunan ke bawah dahulu.

49	52	49	48
4E	4D	4B	4F
46	41	41	52
4F	54	20	45

3. Melakukan proses *AddRoundKey* dari *state* plainteks dengan *RoundKey* ke-0.

49	52	49	48
4E	4D	4B	4F
46	41	41	52
4F	54	20	45

 $\oplus$ 

46	55	48	4A
54	4D	42	41
54	52	45	59
4B	41	52	41

 $=$ 

$49 \oplus 46$	$52 \oplus 55$	$49 \oplus 48$	$4A \oplus 48$
$4E \oplus 54$	$4D \oplus 4D$	$4B \oplus 42$	$41 \oplus 4F$
$46 \oplus 54$	$41 \oplus 52$	$41 \oplus 45$	$59 \oplus 52$
$4F \oplus 4B$	$54 \oplus 41$	$20 \oplus 52$	$41 \oplus 45$

 $=$ 

<b>0F</b>	<b>07</b>	<b>01</b>	<b>02</b>
<b>1A</b>	<b>00</b>	<b>09</b>	<b>0E</b>
<b>12</b>	<b>13</b>	<b>04</b>	<b>0B</b>
<b>04</b>	<b>15</b>	<b>72</b>	<b>04</b>

Hasil pengerjaan ini dikatakan telah melalui *initial round*.

4. Hasil dari *AddRoundKey* akan diproses lagi dengan 4 transformasi yaitu *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey*. Dimana setiap proses ini dikatakan melewati ronde ke-i dengan menggunakan *RoundKey* ke-i pula. Proses ini dilakukan dari ronde 1 hingga ronde 9.

- a. Melakukan operasi *SubBytes*

0F	07	01	02
1A	00	09	0E
12	13	04	0B
04	15	72	04

Operasi *SubBytes*

76	C5	7C	77
A2	63	01	AB
C9	7D	F2	2B
F2	59	40	F2

- b. Melakukan proses *ShiftRows*

76	C5	7C	77
A2	63	01	AB
C9	7D	F2	2B
F2	59	40	F2

Operasi *ShiftRows*

76	C5	7C	77
63	01	AB	A2
F2	2B	C9	7D
F2	F2	59	40

- c. Melakukan proses *MixColumns*, operasi ini mirip seperti perkalian matriks, namun operator penjumlahan akan digantikan dengan operator Bitwise XOR.

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

X

76	C5	7C	77
63	01	AB	A2
F2	2B	C9	7D
F2	F2	59	40

Hasil operasi ini akan menghasilkan suatu state 4x4 yang baru

S <sub>1,1</sub>	S <sub>1,2</sub>	S <sub>1,3</sub>	S <sub>1,4</sub>
S <sub>2,1</sub>	S <sub>2,2</sub>	S <sub>2,3</sub>	S <sub>2,4</sub>
S <sub>3,1</sub>	S <sub>3,2</sub>	S <sub>3,3</sub>	S <sub>4,4</sub>
S <sub>4,1</sub>	S <sub>4,2</sub>	S <sub>4,3</sub>	S <sub>4,4</sub>

Perhitungan *Byte* baris 1 kolom 1 (S<sub>1,1</sub>):

$$= (\{02\}.\{76\}) \oplus (\{03\}.\{63\}) \oplus (\{01\}.\{F2\}) \oplus (\{01\}.\{F2\})$$



(dirubah ke bentuk biner)

$$= (\{10\} \cdot \{01110110\}) \oplus (\{11\} \cdot \{01100011\}) \oplus (11110010) \\ \oplus (11110010)$$

(dirubah ke bentuk polinomial)

$$= (x^1(x^6 + x^5 + x^4 + x^2 + x^1)) \oplus ((x^1 + x^0)(x^6 + x^5 + x^1 + x^0)) \oplus \\ (x^7 + x^6 + x^5 + x^4 + x^1) \oplus (x^7 + x^6 + x^5 + x^4 + x^1) \\ = (x^7 + x^6 + x^5 + x^3 + x^2) \oplus ((x^7 + x^6 + x^2 + x^1) \oplus (x^6 + x^5 + x^1 \\ + x^0)) \oplus (x^7 + x^6 + x^5 + x^4 + x^1) \oplus (x^7 + x^6 + x^5 + x^4 + x^1)$$

(coret yang polinom yang sama)

$$= (x^7 + x^6 + x^5 + x^3 + x^2) \oplus (x^7 + x^5 + x^2 + x^0) \\ = x^6 + x^3 + x^0$$

➔ jika terdapat  $x^8$  maka dilakukan operasi tambahan berupa  $\oplus$  terhadap *irreducible polynomial* yang berderajat 8 atau  $(x^8 + x^4 + x^3 + x^1 + x^0)$  agar hasil tidak lebih dari 256bit

(dirubah ke bentuk biner)

$$= 01001001$$

(dirubah ke bentuk heksadesimal)

Maka,  $S_{1,1} = 49$

Perhitungan tersebut dilakukan pada seluruh *byte*  $S$  untuk menghasilkan state hasil dari operasi *MixColumns*.

49	4B	8E	2E
4F	48	28	EF
E7	9F	B5	EF
F4	81	54	C6

- d. Melakukan proses *AddRoundKey* dari state hasil *MixColumns* dengan *RoundKey* ke-i. Untuk ronde (i) saat pemrosesan ini merupakan ronde ke-1. Sehingga akan menggunakan *RoundKey* ke-1 pula.

49	4B	8E	2E	$\oplus$	C4	91	D9	93
4F	48	28	EF	$\oplus$	9F	D2	90	D1

=

E7	9F	B5	EF	$\oplus$	D7	85	C0	99
F4	81	54	C6		9D	DC	8E	CF
State hasil <i>MixColumns</i>					<i>RoundKey</i> ke-1			

<b>8D</b>	<b>DA</b>	<b>57</b>	<b>BD</b>
<b>D0</b>	<b>9A</b>	<b>B8</b>	<b>3E</b>
<b>30</b>	<b>1A</b>	<b>75</b>	<b>76</b>
<b>69</b>	<b>5D</b>	<b>DA</b>	<b>09</b>

5. Lakukan proses pada nomor 4 hingga ronde ke-9 dengan menggunakan state ronde sebelumnya sebagai inputan untuk ronde ke depannya.

<b>8D</b>	<b>DA</b>	<b>57</b>	<b>BD</b>
<b>D0</b>	<b>9A</b>	<b>B8</b>	<b>3E</b>
<b>30</b>	<b>1A</b>	<b>75</b>	<b>76</b>
<b>69</b>	<b>5D</b>	<b>DA</b>	<b>09</b>
Ronde ke-1			

<b>0D</b>	<b>B2</b>	<b>83</b>	<b>B2</b>
<b>FA</b>	<b>9D</b>	<b>57</b>	<b>D2</b>
<b>9A</b>	<b>83</b>	<b>2D</b>	<b>2D</b>
<b>81</b>	<b>D9</b>	<b>D0</b>	<b>2E</b>
Ronde ke-2			

<b>DA</b>	<b>5A</b>	<b>37</b>	<b>1B</b>
<b>54</b>	<b>20</b>	<b>96</b>	<b>3D</b>
<b>AA</b>	<b>D0</b>	<b>76</b>	<b>D3</b>
<b>E1</b>	<b>38</b>	<b>A1</b>	<b>4D</b>
Ronde ke-3			

<b>AD</b>	<b>45</b>	<b>47</b>	<b>53</b>
<b>4C</b>	<b>CC</b>	<b>CA</b>	<b>B4</b>
<b>E3</b>	<b>BF</b>	<b>A2</b>	<b>F6</b>
<b>C7</b>	<b>52</b>	<b>4F</b>	<b>94</b>
Ronde ke-4			

<b>67</b>	<b>40</b>	<b>E8</b>	<b>2C</b>
<b>20</b>	<b>D2</b>	<b>30</b>	<b>78</b>
<b>57</b>	<b>1A</b>	<b>8F</b>	<b>0C</b>
<b>E5</b>	<b>F1</b>	<b>FA</b>	<b>C9</b>
Ronde ke-5			

<b>F1</b>	<b>37</b>	<b>28</b>	<b>95</b>
<b>77</b>	<b>5E</b>	<b>89</b>	<b>B2</b>
<b>3F</b>	<b>DA</b>	<b>A9</b>	<b>61</b>
<b>25</b>	<b>7C</b>	<b>A1</b>	<b>A2</b>
Ronde ke-6			

<b>A1</b>	<b>FA</b>	<b>83</b>	<b>D8</b>
<b>FD</b>	<b>FD</b>	<b>CF</b>	<b>6C</b>
<b>6C</b>	<b>9F</b>	<b>3F</b>	<b>4E</b>

<b>53</b>	<b>BF</b>	<b>D2</b>	<b>A7</b>
<b>93</b>	<b>E4</b>	<b>38</b>	<b>59</b>
<b>5D</b>	<b>04</b>	<b>7C</b>	<b>B3</b>

<b>12</b>	<b>83</b>	<b>DA</b>	<b>C8</b>
<b>85</b>	<b>4C</b>	<b>72</b>	<b>94</b>
<b>6C</b>	<b>BB</b>	<b>DA</b>	<b>4A</b>

<b>51</b>	<b>E1</b>	<b>F5</b>	<b>0D</b>
Ronde ke-7			

<b>C1</b>	<b>0A</b>	<b>6E</b>	<b>E4</b>
Ronde ke-8			

<b>CD</b>	<b>A9</b>	<b>0C</b>	<b>71</b>
Ronde ke-9			

6. Untuk proses pada ronde ke 10 dan juga hasil dari ronde 10 ini akan menjadi cipherteks nya, dilakukan operasi *SubBytes*, *ShiftRows*, dan *AddRoundKey*.

- a. Melakukan operasi *SubBytes*

12	83	DA	C8
85	4C	72	94
6C	BB	DA	4A
CD	A9	0C	71

Operasi *SubBytes*

<b>C9</b>	<b>EC</b>	<b>57</b>	<b>E8</b>
<b>97</b>	<b>29</b>	<b>40</b>	<b>22</b>
<b>50</b>	<b>EA</b>	<b>57</b>	<b>D6</b>
<b>BD</b>	<b>D3</b>	<b>FE</b>	<b>A3</b>

- b. Melakukan proses *ShiftRows*

C9	EC	57	E8
97	29	40	22
50	EA	57	D6
BD	D3	FE	A3

Operasi *ShiftRows*

<b>C9</b>	<b>EC</b>	<b>57</b>	<b>E8</b>
<b>29</b>	<b>40</b>	<b>22</b>	<b>97</b>
<b>57</b>	<b>D6</b>	<b>50</b>	<b>EA</b>
<b>A3</b>	<b>BD</b>	<b>D3</b>	<b>FE</b>

- c. Melakukan proses *AddRoundKey* dengan *RoundKey* ke-10.

C9	EC	57	E8	$\oplus$	BE	31	F7	95	=
29	40	22	97	$\oplus$	90	15	EC	15	
57	D6	50	EA	$\oplus$	78	93	0B	84	
A3	BD	D3	FE	$\oplus$	E0	CB	4A	D4	
State hasil <i>ShiftRows</i>					<i>RoundKey</i> ke-10				

<b>77</b>	<b>DD</b>	<b>A0</b>	<b>7D</b>
-----------	-----------	-----------	-----------

<b>B9</b>	<b>55</b>	<b>CE</b>	<b>82</b>
<b>2F</b>	<b>48</b>	<b>5B</b>	<b>6E</b>
<b>43</b>	<b>76</b>	<b>99</b>	<b>2A</b>

Sehingga, hasil cipherteks yang telah dibuat menggunakan algoritma AES 128 bit, yaitu **“77B92F43DD554878A0CE5B997D826E2A”**.

### 3.5.3 Perancangan Algoritma Dekripsi AES 128

Dalam melakukan proses dekripsi, seluruh langkah yang dilakukan pada saat enkripsi AES 128 bit akan di proses secara kebalikannya (*inverse*). Disini inputan nya akan berupa suatu cipherteks yang akan dicari kembali plainteks utamanya. Cipherteksnya yaitu **“77B92F43DD554878A0CE5B997D826E2A”**.

1. Mengubah cipherteks ke bentuk state 4x4.

77	DD	A0	7D
B9	55	CE	82
2F	48	5B	6E
43	78	99	2A

2. Melakukan operasi XOR antara state cipherteks dengan *RoundKey* ke-10.

77	DD	A0	7D	$\oplus$	BE	31	F7	95	=
B9	55	CE	82	$\oplus$	90	15	EC	15	
2F	48	5B	6E	$\oplus$	78	93	0B	84	
43	78	99	2A	$\oplus$	E0	CB	4A	D4	
Cipherteks					RoundKey ke-10				

<b>C9</b>	<b>EC</b>	<b>57</b>	<b>E8</b>
<b>29</b>	<b>40</b>	<b>22</b>	<b>97</b>
<b>57</b>	<b>D6</b>	<b>50</b>	<b>EA</b>

A3	BD	D3	FE
----	----	----	----

3. Melakukan proses *Inverse ShiftRows*

C9	EC	57	E8
29	40	22	97
57	D6	50	EA
A3	BD	D3	FE

*Inverse ShiftRows*

C9	EC	57	E8
97	29	40	22
50	EA	57	D6
BD	DE	FE	A3

4. Melakukan proses *Inverse SubBytes* dengan table *Inverse S-Box*

C9	EC	57	E8
97	29	40	22
50	EA	57	D6
BD	DE	FE	A3

*Inverse SubBytes*

12	83	DA	C8
85	4C	72	94
6C	BB	DA	4A
CD	A9	0C	71

5. Hasil dari *Inverse SubBytes* akan diproses lagi dengan 4 transformasi yaitu *AddRoundKey*, *Inverse MixColumns*, *Inverse ShiftRows*, dan *Inverse SubBytes*. Dimana setiap proses ini dikatakan melewati ronde ke-i dengan menggunakan *RoundKey* ke-i pula. Proses ini dilakukan dari ronde 9 hingga ronde 2.

- a. Melakukan proses *AddRoundKey*

12	83	DA	C8
85	4C	72	94
6C	BB	DA	4A
CD	A9	0C	71
State hasil <i>Inverse SubBytes</i>			

 $\oplus$ 

11	8F	C6	62
E3	85	F9	F9
73	E6	95	8F
4A	2B	81	9E
<i>RoundKey</i> ke-9			

=

03	0C	1C	AA
66	C9	8B	6D

1F	5D	4F	C5
87	82	8D	EF

- b. Melakukan proses *Inverse MixColumns* dengan table *Inverse MixColumns* State.

0E	0B	0D	09	X	03	0C	1C	AA
09	0E	0B	0D		66	C9	8B	6D
0D	09	0E	0B		1F	5D	4F	C5
0B	0D	09	0E		87	82	8D	EF

Hasil operasi ini akan menghasilkan suatu state 4x4

S <sub>1,1</sub>	S <sub>1,2</sub>	S <sub>1,3</sub>	S <sub>1,4</sub>
S <sub>2,1</sub>	S <sub>2,2</sub>	S <sub>2,3</sub>	S <sub>2,4</sub>
S <sub>3,1</sub>	S <sub>3,2</sub>	S <sub>3,3</sub>	S <sub>4,4</sub>
S <sub>4,1</sub>	S <sub>4,2</sub>	S <sub>4,3</sub>	S <sub>4,4</sub>

Perhitungan Byte baris 1 kolom 1 (S<sub>1,1</sub>):

$$\begin{aligned}
 &= (\{0E\} \cdot \{03\}) \oplus (\{0B\} \cdot \{66\}) \oplus (\{0D\} \cdot \{1F\}) \oplus (\{09\} \cdot \{87\}) \\
 &(0E) \cdot (03) \\
 &= (1110) \cdot (0011) \\
 &= (x^3 + x^2 + x^1) \cdot (x^1 + x^0) \\
 &= (x^4 + x^3) \oplus (x^3 + x^2) \oplus (x^2 + x^1) \\
 &= (x^4 + x^1) \\
 &= 10010 \\
 &= 12
 \end{aligned}$$

$$\begin{aligned}
 &(0B) \cdot (66) \\
 &= (1011) \cdot (01100110) \\
 &= (x^3 + x^1 + x^0) \cdot (x^6 + x^5 + x^2 + x^1)
 \end{aligned}$$

$$=(x^9 + x^8 + x^5 + x^4) \oplus (x^7 + x^6 + x^3 + x^2) \oplus (x^6 + x^5 + x^2 + x^1)$$

$$=(x^9 + x^8 + x^7 + x^4 + x^3 + x^1)$$

$$\Rightarrow x^8 = \text{irreducible polynomial} \text{ atau } (x^8 + x^4 + x^3 + x^1 + x^0)$$

Bisa menggunakan modulus terhadap  $(x^8 + x^4 + x^3 + x^1 + x^0)$  atau konversi nilai  $x^8$  menjadi  $x^4 + x^3 + x^1 + x^0$

$$=(x^8 \cdot x^1) + (x^8 + x^7 + x^4 + x^3 + x^1)$$

$$=(x^4 + x^3 + x^1 + x^0) \cdot (x^1) \oplus (x^4 + x^3 + x^1 + x^0) \oplus (x^7 + x^4 + x^3 + x^1)$$

$$=(x^5 + x^4 + x^2 + x^1) \oplus (x^7 + x^0)$$

$$=(x^7 + x^5 + x^4 + x^2 + x^1 + x^0)$$

$$= 10110111$$

$$= B7$$

$$(0D).(1F)$$

$$= (1101).(00011111)$$

$$=(x^3 + x^2 + x^0).(x^4 + x^3 + x^2 + x^1 + x^0)$$

$$=(x^7 + x^6 + x^5 + x^4 + x^3) \oplus (x^6 + x^5 + x^4 + x^3 + x^2) \oplus (x^4 + x^3 + x^2 + x^1 + x^0)$$

$$=(x^7 + x^4 + x^3 + x^1 + x^0)$$

$$= 10011011$$

$$= 9B$$

$$(09).(87)$$

$$= (00001001).(10000111)$$

$$=(x^3 + x^0) \cdot (x^7 + x^2 + x^1 + x^0)$$

$$=(x^{10} + x^5 + x^4 + x^3) \oplus (x^7 + x^2 + x^1 + x^0)$$

$$=((x^2 \cdot x^8) + x^5 + x^4 + x^3) \oplus (x^7 + x^2 + x^1 + x^0)$$

$$=(x^2 \cdot (x^4 + x^3 + x^1 + x^0)) \oplus (x^5 + x^4 + x^3) \oplus (x^7 + x^2 + x^1 + x^0)$$

$$=(x^6 + x^5 + x^3 + x^2) \oplus (x^5 + x^4 + x^3) \oplus (x^7 + x^2 + x^1 + x^0)$$

$$=(x^7 + x^6 + x^4 + x^1 + x^0)$$

$$= 11010011$$

= D3

Sehingga nilai  $(S_{1,1})$  yaitu

$$\begin{aligned}(S_{1,1}) &= 10010 \oplus 10110111 \oplus 10011011 \oplus 11010011 \\ &= 10100101 \oplus 1001000 \\ &= 11101101\end{aligned}$$

$(S_{1,1}) = \mathbf{ED}$

Perhitungan tersebut dilakukan pada seluruh *byte* *S* untuk menghasilkan state hasil dari operasi *Inverse MixColumns*.

<b>ED</b>	<b>08</b>	<b>B5</b>	<b>5C</b>
<b>69</b>	<b>07</b>	<b>CB</b>	<b>DC</b>
<b>10</b>	<b>6D</b>	<b>4C</b>	<b>F2</b>
<b>69</b>	<b>78</b>	<b>67</b>	<b>9F</b>

c. Melakukan proses *Inverse ShiftRows*

ED	08	B5	5C
69	07	CB	DC
10	6D	4C	F2
69	78	67	9F

*Inverse ShiftRows*

<b>ED</b>	<b>08</b>	<b>B5</b>	<b>5C</b>
<b>DC</b>	<b>69</b>	<b>07</b>	<b>CB</b>
<b>4C</b>	<b>F2</b>	<b>10</b>	<b>6D</b>
<b>78</b>	<b>67</b>	<b>9F</b>	<b>69</b>

d. Melakukan proses *Inverse SubBytes*

ED	08	B5	5C
DC	69	07	CB
4C	F2	10	6D
78	67	9F	69

*Inverse SubBytes*

<b>53</b>	<b>BF</b>	<b>D2</b>	<b>A7</b>
<b>93</b>	<b>E4</b>	<b>38</b>	<b>59</b>
<b>5D</b>	<b>04</b>	<b>7C</b>	<b>B3</b>
<b>C1</b>	<b>0A</b>	<b>6E</b>	<b>E4</b>

6. Lakukan proses pada nomor 5 hingga ronde ke-1 dengan menggunakan state ronde sebelumnya sebagai inputan untuk ronde ke depannya.



53	BF	D2	A7
93	E4	38	59
5D	04	7C	B3
C1	0A	6E	E4
Ronde ke-9			

A1	FA	83	D8
FD	FD	CF	6C
6C	9F	3F	4E
51	E1	F5	0D
Ronde ke-8			

F1	37	28	95
77	5E	89	B2
3F	DA	A9	61
25	7C	A1	A2
Ronde ke-7			

67	40	E8	2C
20	D2	30	78
57	1A	8F	0C
E5	F1	FA	C9
Ronde ke-6			

AD	45	47	53
4C	CC	CA	B4
E3	BF	A2	F6
C7	52	4F	94
Ronde ke-5			

DA	5A	37	1B
54	20	96	3D
AA	D0	76	D3
E1	38	A1	4D
Ronde ke-4			

0D	B2	83	B2
FA	9D	57	D2
9A	83	2D	2D
81	D9	D0	2E
Ronde ke-3			

8D	DA	57	BD
D0	9A	B8	3E
30	1A	75	76
69	5D	DA	09
Ronde ke-2			

0F	07	01	02
1A	00	09	0E
12	13	04	0B
04	15	72	04
Ronde ke-1			

7. Melakukan proses *AddRoundKey* dengan *initial key* atau *RoundKey* ke-0

0F	07	01	02	$\oplus$	46	55	48	4A	=
1A	00	09	0E	$\oplus$	54	4D	42	41	
12	13	04	0B	$\oplus$	54	52	45	59	
04	15	72	04	$\oplus$	4B	41	52	41	
State ronde 1					RoundKey ke-0				

49	52	49	48
4E	4D	4B	4F
46	41	41	52
4F	54	20	45

8. Merubah state menjadi plainteks dengan merubah heksadesimal menjadi bentuk ASCII.

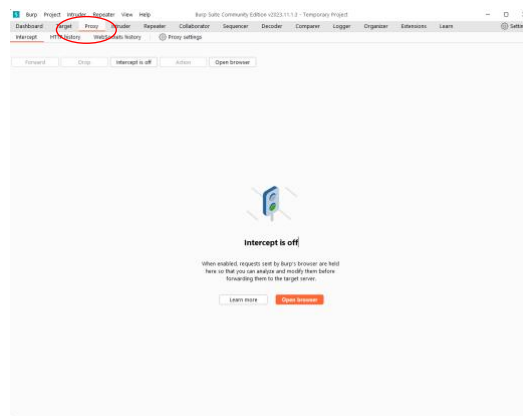
49	4E	46	4F	52	4D	41	54	49	4B	41	20	48	4F	52	45
I	N	F	O	R	M	A	T	I	K	A		H	O	R	E

Sehingga, hasil plainteks yang telah di dekripsi menggunakan algoritma AES 128 bit, yaitu **“INFORMATIKA HORE”**.

### 3.5.4 Perancangan Skenario Serangan MITM

Dengan menggunakan aplikasi Burp Suite, serangan *Sniffing* MITM dapat dilakukan dengan memanfaatkan *proxy* dan *intercept* dari aplikasi ini. Berikut merupakan langkah-langkah skenario nya :

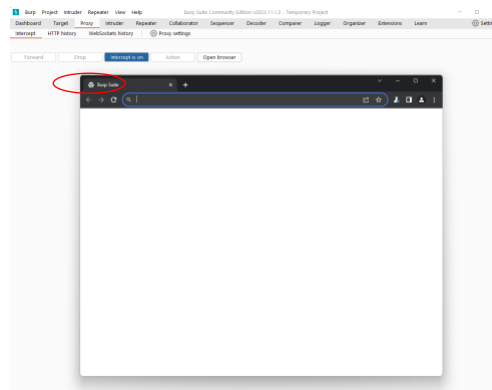
1. Mengaktifkan *proxy* dan fitur *intercept* Burp Suite.



**Gambar 12. MITM - 1**

Pada gambar tersebut kita akan menuju fitur yang disediakan oleh Burp Suite pada tab *proxy* dimana fitur ini memungkinkan pengguna untuk mengakses suatu URL menggunakan browser Burp Suite. *Intercept* merupakan fitur untuk mengambil semua informasi *request* yang dilakukan pada URL browser Burp Suite.

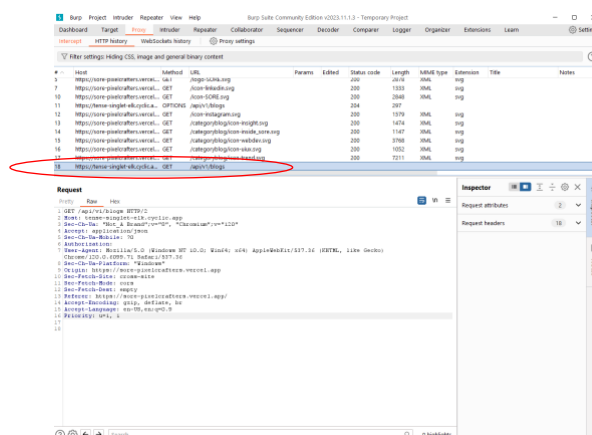
2. Buka browser yang telah diterapkan *proxy* dan *intercept*.



**Gambar 13. MITM - 2**

Pada gambar tersebut merupakan tampilan browser Burp Suite yang kemudian akan kita gunakan untuk membuka suatu URL *website* yang akan kita serang.

3. Pada browser masukkan URL yang ingin di lakukan serangan.
4. Untuk melanjutkan proses, pada halaman *intercept* terus klik *forward* untuk melanjutkan proses pengumpulan informasi yang terjadi pada *web service* dan *client*.
5. Pada tab *Proxy* dan *HTTP History* Burp Suite informasi tersebut dikumpulkan.

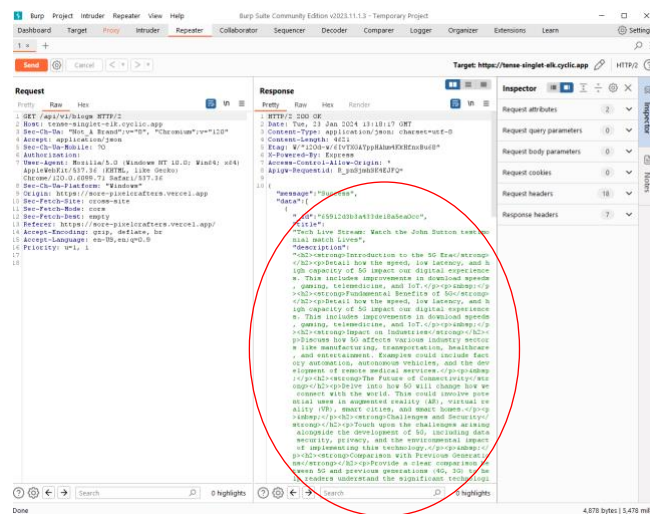


**Gambar 14. MITM - 3**

Pada gambar tersebut kita akan melakukan pemilihan informasi *request* yang menjadi suatu informasi pengambilan data dari *web server*. Setelah

diambil kita akan mengirimkan informasi tersebut kembali menggunakan fitur *repeater* untuk mendapatkan hasil serangan *sniffing* pada sistem *web service* sehingga akan mendapatkan data informasi dari *web server*.

6. Kirim *request* ke fitur *Repeater* pada Burp Suite untuk dilakukan pengiriman ulang.
7. Hasil data yang dikirimkan akan tampil.



**Gambar 15. MITM – 4**

Dengan mengirimkan *request* dan membaca *response* yang dikirimkan kita telah berhasil melakukan serangan sniffing pada suatu aplikasi yang berbasis *web service* dan *client*.

### 3.5.5 Pengujian

Algoritma yang telah dibentuk akan dibandingkan dengan sistem uji hasil enkripsi dan dekripsi sesuai atau tidak dengan plainteks dan juga cipherteksnya.

Dengan plainteks yaitu “INFORMATIKA HORE” dan kunci “FTTKUMRAHBERJAYA” setelah dilakukan enkripsi dengan AES 128 bit, didapatkan hasil enkripsi atau cipherteks yaitu “77B92F43DD554878A0CE5B997D826E2A”.

Dengan cipherteks “77B92F43DD554878A0CE5B997D826E2A” dan kunci “FTTKUMRAHBERJAYA” setelah dilakukan dekripsi dengan AES 128 bit, didapatkan kembali plainteksnya yaitu “INFORMATIKA HORE”.

**Tabel 3. Tabel Pengujian Enkripsi dan Dekripsi**

		Keterangan
Plainteks	INFORMATIKA HORE	
Kunci	FTTKUMRAHBERJAYA	
Hasil Enkripsi	77B92F43DD554878A0CE5B997D826E2A	Berhasil membuat cipherteks
Hasil Dekripsi	INFORMATIKA HORE	Berhasil mengembalikan cipherteks Kembali ke plainteks

Pengujian serangan *Man In The Middle Attack* dilakukan menggunakan tools Burp Suite. Pada penelitian ini, *website* yang akan diserang masih berbentuk *dummy* dan merupakan contoh yang akan dihasilkan saat proses serangan dilakukan pada sistem *web service* tanpa penerapan AES 128 sebagai algoritma enkripsi *End To End Encryption* dan sistem *web service* dengan penerapan AES 128 sebagai algoritma enkripsi *End To End Encryption*.

**Tabel 4. Tabel Pengujian Skenario *Man In The Middle Attack***

No	Langkah-langkah	Keterangan	Tanpa E2EE	Dengan E2EE
1.	Menggunakan Burp Suite			
2.	Mengaktifkan <i>proxy</i> dan fitur <i>intercept</i>			

No	Langkah-langkah	Keterangan	Tanpa E2EE	Dengan E2EE
3.	Buka URL <i>website</i> yang ingin diserang			
4.	Pada halaman <i>intercept</i> terus klik <i>forward</i> untuk melanjutkan proses pengumpulan informasi yang terjadi pada <i>web service</i> dan <i>client</i> .			
5.	Cari HTTP <i>History</i> yang ingin diserang	/api/users		
6.	Kirim ulang <i>request</i> HTTP dengan fitur <i>repeater</i> dari Burp Suite	GET /api/pasien		
7.	Hasil response <i>Sniffing Man In The Middle Attack</i> akan muncul		{ message : "success", status : 200, data : { "nama":"oriast anjung", "nik":2012590 21951 } }	7e24229 5b6305a b864310 e5c475d 643f521 c22c818 4d60ffbc f913081 dc5ebe7 0a2b038 39a8ba1 1a51adf 0ccdc60 584f

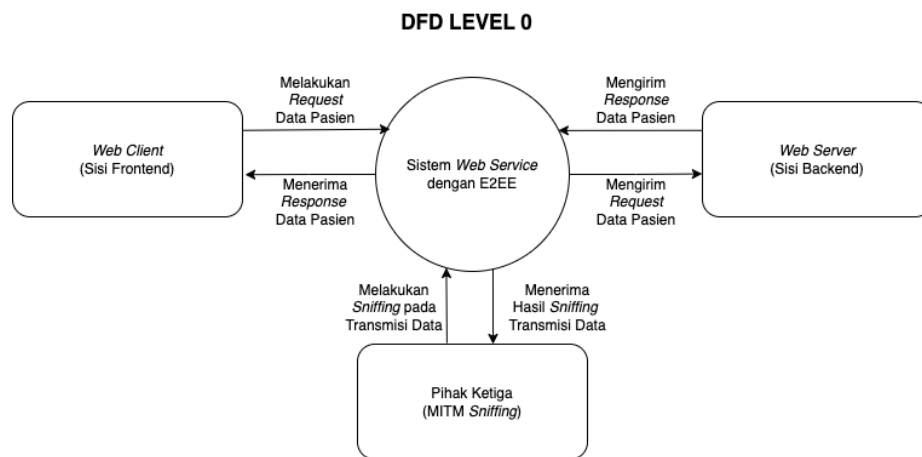
Pengujian serangan *Man In The Middle Attack* dilakukan menggunakan tools Burp Suite menghasilkan perbandingan sistem tanpa E2EE dan dengan E2EE dimana dengan penerapan E2EE serangan *Man In The Middle Attack* tidak dapat mengambil informasi dari *web service*.

### 3.6 Perancangan *Data Flow Diagram* (DFD)

Perancangan DFD akan memudahkan dalam membaca alur sistem yang akan dirancang. Untuk itu dibuat DFD dalam dua tingkat yaitu tingkat 0 dan tingkat 1.

#### 3.6.1 DFD Tingkat 0

Berikut merupakan diagram rancangan DFD pada tingkat 0.

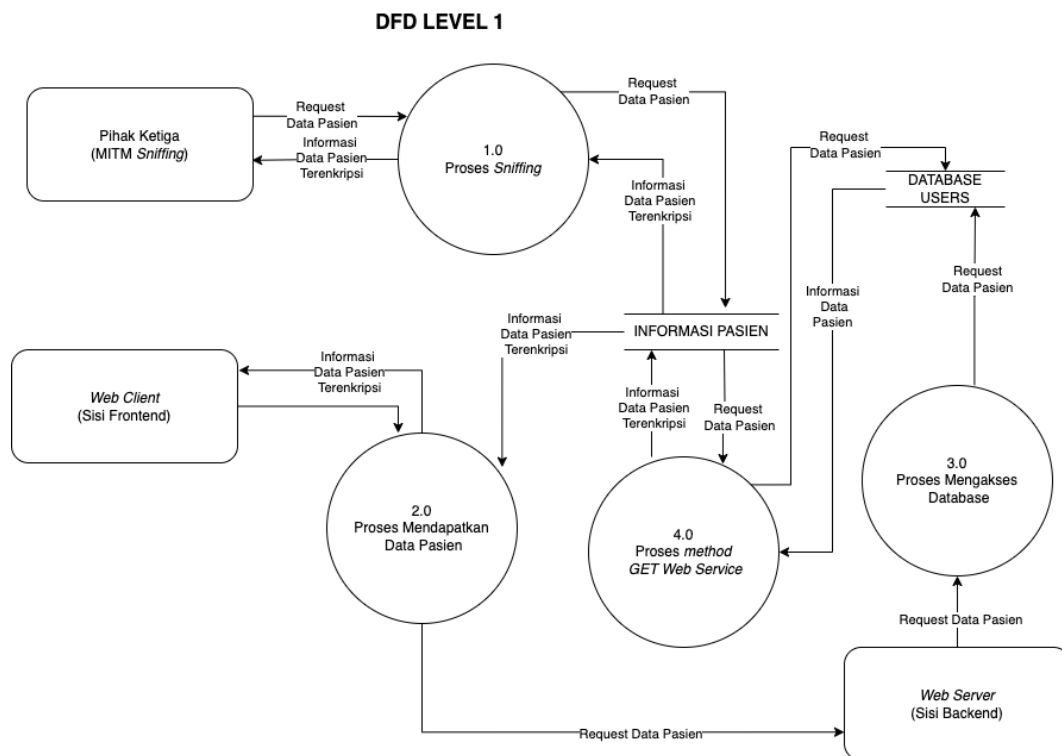


**Gambar 16.** DFD Tingkat 0

Pada DFD tingkat 0 dapat dilihat beberapa proses utama yang terjadi dan dilakukan pada sistem dimana terdapat 3 entitas utama yaitu *web client*, *web server*, dan pihak ketiga atau pihak yang melakukan serangan MITM *sniffing*. Beberapa proses utama tersebut antara lain proses mendapatkan data pasien dan proses melakukan *sniffing* data pasien.

#### 3.6.2 DFD Tingkat 1

Berikut merupakan diagram rancangan DFD pada tingkat 1. Pada diagram ini terbagi menjadi beberapa proses sesuai dengan proses yang terjadi.



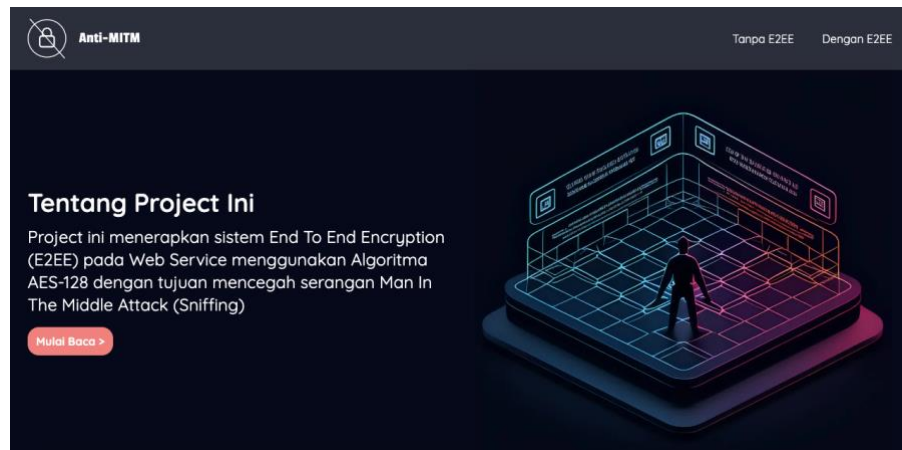
**Gambar 17.** DFD Tingkat 1

Diagram Aliran Data (DFD) Tingkat 1 ini menggambarkan alur yang terjadi dengan lebih rinci pada sistem yang ada pada penelitian ini. Proses utama yang terlibat meliputi proses *sniffing*, pengambilan data pasien, proses mengakses *database*, dan permintaan layanan *web service* dengan metode GET. Data pasien yang terenkripsi mengalir dari *web client* ke *web server*, yang kemudian mengakses basis data dan mengembalikan data tersebut ke *web client*. "Informasi Pasien" berfungsi sebagai penyimpanan sementara yang mengelola data pasien yang terenkripsi.

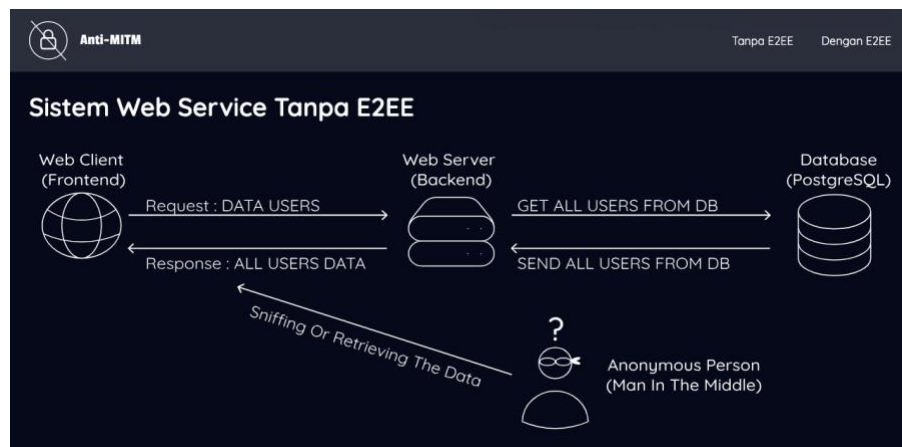
### 3.7 Tampilan *User Interface* (UI)

Untuk tampilan *website* akan dibangun seperti beberapa tampilan gambar berikut.

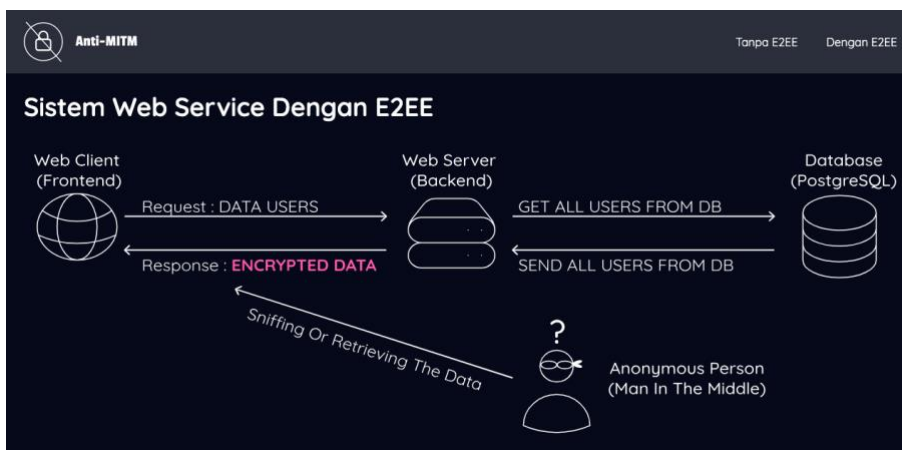




**Gambar 18.** UI website - 1



**Gambar 19.** UI website - 2



**Gambar 20.** UI website - 3

Tampilan beberapa gambar tersebut merupakan halaman awal implementasi tampilan dimana akan terdapat informasi terkait pengajuan sistem tanpa E2EE dan sistem dengan penerapan E2EE. Disini juga terdapat dua *route* berbeda yang akan

digunakan untuk proses perbandingan serangan sistem yang menerapkan dan tidak menerapkan E2EE.

### 3.8 Implementasi Sistem *End To End Encryption*

#### 3.8.1 Ekspansi Kunci AES 128

Proses ekspansi kunci AES 128 akan digunakan pada proses baik itu sisi *web server* maupun *web client*. Dimana kode dari proses ini akan selalu ada di kedua sisi tersebut dengan kode yang sama pula.

```
//utils/roundkey.js
const { hexXORoneDimensional } = require("./XOR");
const makingBlockOfChars = require("./block");
const { getDataByColumns } = require("./helper");
const { stringToHexArray, hexArrayToString } = require("./hexString");
const RCON = require("./rcon");
const { rotateWord, subBytesLastColumns } = require("./rotateWord");
const { sbboxOneDimensionalArray } = require("./sbox");
function makeRoundItem(lastRound, roundkeys, round, tempColumns) {
  const roundItem = [
    ["00", "01", "02", "03"],
    ["10", "11", "12", "13"],
    ["20", "21", "22", "23"],
    ["30", "31", "32", "33"],
  ];
  let lastColumnDataRotWord = rotateWord(lastRound);
  let subBytesLastColumn = sbboxOneDimensionalArray(lastColumnDataRotWord);
  for (let i = 0; i < lastRound.length; i++) {
    if (i === 0) {
      let firstColumn = getDataByColumns(lastRound, 0);
      let xorFirstWithLast = hexXORoneDimensional(
        firstColumn,
        subBytesLastColumn
      );
    }
  }
}
```

```

        let xorWithRCON =
hexXORoneDimensional(xorFirstWithLast,RCON[round]);
        tempColumns = xorWithRCON;
    } else
    let column = getDataByColumns(lastRound, i);
        tempColumns = hexXORoneDimensional(column, tempColumns);
    }
    roundItem[0][i] = tempColumns[0];
    roundItem[1][i] = tempColumns[1];
    roundItem[2][i] = tempColumns[2];
    roundItem[3][i] = tempColumns[3];
}

    roundkeys.push(roundItem);
}

function makeRoundKeys(keyString) {
    const hexKey = stringToHexArray(keyString);
    let tempColumns = [];
    const roundkeys = [hexKey];
    for (let j = 0; j < 10; j++) {
        let lastRound = roundkeys[j]
        makeRoundItem(lastRound, roundkeys, j, tempColumns);
    }
    return roundkeys
}

module.exports = makeRoundKeys;

```

**Gambar 21.** Kode Pembuatan *Roundkey*

Kode pada gambar 21 tersebut merupakan implementasi dalam pembuatan *RoundKey* dimana menerima sebuah masukan kunci dalam tipe data string, kemudian dilakukan proses ekspansi kunci hingga 10 kunci yang akan digunakan pada proses enkripsi maupun dekripsi AES 128.

```
//utils/XOR.js
function hexXORoneDimensional(hex1, hex2) {
  const result = []
  for(let i = 0; i < hex1.length; i++){
    const int1 = parseInt(hex1[i], 16);
    const int2 = parseInt(hex2[i], 16);
    const resultInt = int1 ^ int2;
    const resultHex = resultInt
      .toString(16).padStart(hex1[i].length, "0");
    result.push(resultHex)
  }
  return result;
}
module.exports = {hexXORoneDimensional}
```

**Gambar 22.** Kode operasi XOR 1 baris

Kode pada gambar 22 tersebut digunakan untuk melakukan XOR 1 baris heksadesimal dengan 1 baris tujuan heksadesimal lainnya. Kode ini juga akan dibutuhkan pada proses enkripsi maupun dekripsi nantinya.

```
//utils/helper.js
const getLastColumns = (arrayHex) => {
  // posisi 3,7,11,14
  return [arrayHex[0][3], arrayHex[1][3], arrayHex[2][3],
arrayHex[3][3]];
};
const updateLastColumns = (arrayHex, arrayInput) => {
  arrayHex[0][3] = arrayInput[0];
  arrayHex[1][3] = arrayInput[1];
  arrayHex[2][3] = arrayInput[2];
  arrayHex[3][3] = arrayInput[3];
};
const getDataByColumns = (arrayHex, indexColumn) => {
  return [
    arrayHex[0][indexColumn],
    arrayHex[1][indexColumn],
    arrayHex[2][indexColumn],
```

```

        arrayHex[3][indexColumn],
    ];
};

function flattenHexData(data) {
    return data.map((hexArray) => {
        let result = "";
        for (let j = 0; j < hexArray[0].length; j++) {
            for (let i = 0; i < hexArray.length; i++) {
                result += hexArray[i][j];
            }
        }
        return result;
    });
}

module.exports =
{getLastColumns,updateLastColumns,getDataByColumns,flattenHexData,
};

```

**Gambar 23.** Kode pembantu dalam *RoundKey*

Adapun kode pada gambar 23 yang terdiri dari beberapa operasi yang digunakan selama operasi pada proses pembuatan kunci di AES 128 yang tertera pada gambar di bawah. Kode tersebut merupakan kode pembantu.

```

//utils/hexString.js
function charToHex(charInput) {
    return charInput.charCodeAt(0).toString(16).toUpperCase();
}

function stringToHexArray(inputString) {
    // Konversi string menjadi array karakter
    const charArray = inputString.split("");
    // Inisialisasi array 4x4
    const hexArray = [];
    for (let i = 0; i < 4; i++) {
        hexArray[i] = [];
        for (let j = 0; j < 4; j++) {
            // Ambil karakter dari array input
            const char = charArray[i + j * 4] || " "; // Ubah indeks
            // Konversi karakter menjadi kode hex dan tambahkan ke array

```

```

        hexArray[i][j] = charToHex(char);
    }
}
return hexArray;
}
function split2CharAsHex(inputString) {
    // Konversi string menjadi array karakter per 2 hex
    const charArray = []
    for(let i = 0; i < inputString.length; i+=2){
        charArray.push(inputString[i]+inputString[i+1])
    }
    // Inisialisasi array 4x4
    const hexArray = [];
    for (let i = 0; i < 4; i++) {
        hexArray[i] = [];
        for (let j = 0; j < 4; j++) {
            // Ambil karakter dari array input
            const char = charArray[i + j * 4] || " "; // Ubah indeks
            // Konversi karakter menjadi kode hex dan tambahkan ke array
            hexArray[i][j] = char;
        }
    }
    return hexArray;
}
function hexToChar(hexInput) {
    return String.fromCharCode(parseInt(hexInput, 16));
}
function hexArrayToString(hexArray) {
    let result = "";
    for (let j = 0; j < hexArray[0].length; j++) {
        for (let i = 0; i < hexArray.length; i++) {
            result += hexToChar(hexArray[i][j]);
        }
    }
    return result;
}
module.exports = {
    stringToHexArray,
    hexArrayToString,
    split2CharAsHex
};

```

**Gambar 24.** Kode konversi heksadesimal ke string dan sebaliknya

Kode pada gambar 24 tersebut merupakan kode yang digunakan untuk melakukan konversi suatu masukan berupa tipe data string menjadi heksadesimal pada setiap elemen dan disimpan pada suatu bentuk tipe data array, dan juga sebaliknya yaitu tipe data heksadesimal ke bentuk tipe data string pada setiap elemen dalam bentuk array.

```
//utils/RCON.js
const RCON = [
  ["01", "00", "00", "00"],
  ["02", "00", "00", "00"],
  ["04", "00", "00", "00"],
  ["08", "00", "00", "00"],
  ["10", "00", "00", "00"],
  ["20", "00", "00", "00"],
  ["40", "00", "00", "00"],
  ["80", "00", "00", "00"],
  ["1B", "00", "00", "00"],
  ["36", "00", "00", "00"],
];
module.exports = RCON
```

**Gambar 25.** Kode RCON

Kode pada gambar 25 tersebut merupakan kode yang menyimpan matriks dari RCON pada pembuatan suatu kunci.

```
//utils/sbox.js

const sBoxTable = [
  [
    "63","7C","77","7B","F2","6B","6F","C5","30","01","67","2B","FE","
D7","AB","76",
  ],
  ..... isi dari tabel S-Box
]

function sboxOneDimensionalArray(hexArray) {
```

```

    const substitutedArray = hexArray.map((hexValue) => {
      const row = parseInt(hexValue.charAt(0), 16);
      const col = parseInt(hexValue.charAt(1), 16);
      return sBoxTable[row][col];
    });
    return substitutedArray;
  }

function substitutionSBox(inputArray) {
  const substitutedArray = inputArray.map(row =>
    row.map(hexValue => {
      const rowIndex = parseInt(hexValue.charAt(0), 16);
      const colIndex = parseInt(hexValue.charAt(1), 16);
      return sBoxTable[rowIndex][colIndex];
    })
  );
  return substitutedArray;
}

module.exports={sBoxTable,substitutionSBox,sboxOneDimensionalArray
};

```

**Gambar 26.** Kode *SubBytes*

Kode pada gambar 26 tersebut merupakan penerapan operasi *SubBytes* sesuai dengan algoritma dalam AES 128, dimana variabel `sBoxTable` merupakan data yang dibuat dari tabel s-box dari AES 128, kemudian *function* `sboxOneDimensionalArray` merupakan proses *SubBytes* untuk 1 baris, dan *function* `substitutionSBox` merupakan *function* utama dalam melakukan operasi *SubBytes*.

```

//utils/sbox.js
const { getLastColumns, updateLastColumns } = require("../helper");
const { substitutionSBox } = require("../sbox");
function rotateWord(arrayHex) {
  const lastColumn = getLastColumns(arrayHex);
  const updatedData = [
    lastColumn[1],
    lastColumn[2],

```



```

        lastColumn[3],
        lastColumn[0],
    ];
    return updatedData}
function subBytesLastColumns(arrayHex) {
    const lastColumn = getLastColumns(arrayHex);
    const afterSbox = substitutionSBox(lastColumn);
    updateLastColumns(arrayHex, afterSbox)
}
module.exports = { rotateWord, subBytesLastColumns };

```

**Gambar 27.** Kode RotWord di ekspansi kunci

Kode pada gambar 27 tersebut mengimplementasikan operasi *Rotate Word* atau RotWord pada proses pembuatan kunci.

```

//utils/sbox.js
const { stringToHexArray } = require("../hexString");

function splitStringIntoChunks(str, chunkSize) {
    const chunks = [];
    for (let i = 0; i < str.length; i += chunkSize) {
        chunks.push(str.substr(i, chunkSize));
    }
    return chunks;
}

function padString(str, length, char) {
    while (str.length < length) {
        str += char;
    }
    return str;
}

function splitStringInto16CharChunks(str) {
    const chunks = splitStringIntoChunks(str, 16);
    const paddedChunks = chunks.map(chunk => padString(chunk, 16,
' '));
    return paddedChunks;
}

```

```

}

//menyusun 4 huruf ke bawah bukan ke samping seperti aturan AES
function rearrangeData(data) {
  // Reshape the data into a 4x4 grid
  let grid = [];
  for (let i = 0; i < data.length; i += 4) {
    grid.push(data.slice(i, i + 4));
  }

  // Transpose the grid to change the order
  let transposedGrid = [];
  for (let i = 0; i < grid[0].length; i++) {
    let row = [];
    for (let j = 0; j < grid.length; j++) {
      row.push(grid[j][i]);
    }
    transposedGrid.push(row);
  }
  // Convert the transposed grid back to a flat list
  let rearrangedData = [];
  for (let i = 0; i < transposedGrid.length; i++) {
    rearrangedData = rearrangedData.concat(transposedGrid[i]);
  }
  return rearrangedData;
}

function makingBlockOfChars(inputString){
  const arrayStr = splitStringInto16CharChunks(inputString);
  const result = arrayStr.map((item) => stringToHexArray(item));
  return result;
}

module.exports = makingBlockOfChars;

```

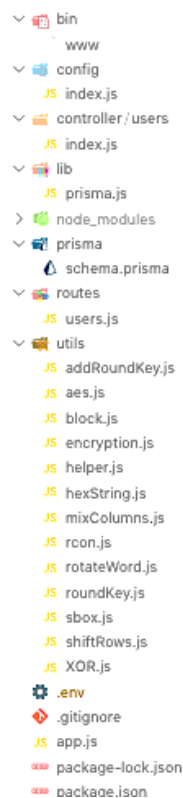
### Gambar 28. Kode pembuatan *Block Cipher*

Kode pada gambar 28 tersebut akan digunakan nantinya pada proses enkripsi dan dekripsi dalam pembagian suatu masukan menjadi *block* yang dibentuk pada array.

#### 3.8.2 *Web Server (Sisi Backend)*

Dari *web service* salah satu sisi yang dibutuhkan dalam pembuatan sistem ini ialah sisi *backend* atau biasa disebut *web server*. Disini pembuatan *web server* akan menggunakan *framework* ExpressJS, dengan bantuan ORM Prisma.io untuk koneksi dari *web server* dengan *database* PostgreSQL.

Berikut struktur proyek yang dirancang dalam membangun sisi *backend*



**Gambar 29.** Struktur folder proyek sisi *backend*

Pada gambar 29 tersebut folder proyek dipecah agar terorganisir dan berjalan dengan kode minimalis, disini folder untuk *web server* berfokus pada folder *controller*, *routes*, *prisma*, *lib* dan file *app.js*. Khusus untuk kode penerapan algoritma AES 128 terletak pada folder *utils*.

### A. Kode Web Server

```
//app.js
const express = require('express');
const path = require('path');
const cookieParser = require('cookie-parser');
const logger = require('morgan');
const usersRouter = require("./routes/users")
const cors = require("cors")

const app = express();
app.use(cors())
app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/users', usersRouter);

module.exports = app;
```

**Gambar 30.** Kode utama *web server*

Kode pada gambar 30 ini merupakan kode utama dalam sisi *backend* yang menjalankan *web service* dengan penggunaan *routes* yang terpisah kodenya.

Lalu terdapat pula kode berupa *routes* yang mana merupakan pembuatan suatu *endpoint* untuk dikenakan beberapa *method* HTTP seperti GET, DELETE, UPDATE, dan POST. Untuk *method* tersebut, dipisahkan dalam suatu file *controller*. Dikarenakan fokus penelitian itu untuk mencegah penyerangan MITM

jenis *Sniffing* dimana serangan ini dilakukan ketika pihak tidak berwenang memantau transmisi data di *web service* terutama pada *method* GET, maka disini berfokuskan pada kode dengan *controller* GET yang mana berfungsi untuk mendapatkan suatu data.

```
//routes/users.js
const express = require("express")
const { getAllUsers, createOneUser, createManyUsers, getOneUser,
deleteOneUser, updateOneUser, getAllUsersTanpaE2EE } =
require("../controller/users")
const router = express.Router()
router.get("/",getAllUsers)
router.get("/no-e2ee",getAllUsersTanpaE2EE)
router.get("/:id",getOneUser)
router.delete("/:id",deleteOneUser)
router.put("/:id",updateOneUser)
router.post('/',createOneUser)
router.post('/many',createManyUsers)

module.exports = router;
```

**Gambar 31.** *Routes endpoint data pasien*

Pada gambar 31 untuk *endpoint* yang difokuskan yaitu pada *"/users/"* dan *"/users/no-e2ee"*, dimana *endpoint* ini yang akan dilakukan uji serangan MITM *Sniffing* nantinya. *Endpoint* *"/users/"* menerapkan sistem *End To End Encryption* dimana data yang dikirimkan sebagai *response* dari suatu *web client* (sisi *frontend*) yang melakukan *request* data pasien. Sedangkan *Endpoint* *"/users/no-e2ee"* tidak menerapkan sistem *End To End Encryption*.

```
//controller/users/index.js
const prisma = require("../..lib/prisma");
const { AESEncryption } = require("../..utils/aes");
const getAllUsers = async (req, res, next) => {
  try {
    const allUsers = await prisma.user.findMany();
    const stringUsers = JSON.stringify(allUsers);
    const encryptedData = AESEncryption(stringUsers)
```

```

        return res.json(encryptedData);
    } catch (error) {
        next(error);
    }
};

const getAllUsersTanpaE2EE = async (req, res, next) => {
    try {
        const allUsers = await prisma.user.findMany();
        return res.json(allUsers);
    } catch (error) {
        next(error);
    }
};

module.exports = {
    getAllUsers,
    getAllUsersTanpaE2EE
};

```

**Gambar 32.** *Controller* data pasien

Pada gambar 32 terdapat dua *controller* utama yaitu *function* `getAllUsers` yang merupakan penerapan kode *web service* yang menggunakan enkripsi data dengan algoritma AES 128, sedangkan `getAllUsersTanpaE2EE` merupakan *function* yang tidak menerapkan AES 128

## B. Kode Enkripsi AES 128

Kode ini merupakan kode yang melakukan proses enkripsi data dimana dilakukan pada suatu masukan data sebagai parameter yang dijadikan dalam bentuk tipe data string. Kode ini dipisah menjadi berbagai *function* sesuai dengan algoritma AES 128.

```

//utils/addRoundKey.js
const { hexXORoneDimensional } = require("../XOR")

function addRoundKey(hex1, hex2){
    const result = []

```

```

    for(let i = 0; i < hex1.length ; i++){
        let temp = hexXORoneDimensional(hex1[i],hex2[i])
        result.push(temp)
    }

    return result
}
module.exports = {
    addRoundKey
}

```

**Gambar 33.** Kode *AddRoundKey*

Kode tersebut digunakan untuk melakukan operasi yang terdapat pada algoritma AES 128 yaitu *AddRoundKey*. Disini variabel `hex1` merupakan heksadesimal *block cipher* dan `hex2` merupakan heksadesimal *block cipher* dari *RoundKey* yang akan diterapkan.

```

//utils/shiftRows.js
function shiftRows(hexArrayOfChars) {
    let state = hexArrayOfChars;
    // Baris pertama tidak diubah, tetap seperti itu
    // Baris kedua di shift ke kiri sebanyak 1 langkah
    state[1] = [state[1][1], state[1][2], state[1][3], state[1][0]];
    // Baris ketiga di shift ke kiri sebanyak 2 langkah
    state[2] = [state[2][2], state[2][3], state[2][0], state[2][1]];
    // Baris keempat di shift ke kiri sebanyak 3 langkah
    state[3] = [state[3][3], state[3][0], state[3][1], state[3][2]];
    return state;
}
module.exports = {
    shiftRows,
};

```

**Gambar 34.** Kode *ShiftRows*

Kode tersebut merupakan kode melakukan operasi *ShiftRows* pada algoritma AES 128.

```

//utils/mixColumns.js
const R_GALOIS_FIELD_MATRIX = [
  ["02", "03", "01", "01"],
  ["01", "02", "03", "01"],
  ["01", "01", "02", "03"],
  ["03", "01", "01", "02"],
];

function galoisMultiply(a, b) {
  let result = 0;
  let mask = 0x80; // 0b10000000
  for (let i = 0; i < 8; i++) {
    if ((b & 1) === 1) {
      result ^= a;
    }
    const carry = a & mask;
    a <<= 1;
    if (carry !== 0) {
      a ^= 0x1b; // XOR dengan 0x1B jika carry terjadi
    }
    b >>= 1;
  }
  return result;
}

function mixColumns(state) {
  const result = [];
  for (let i = 0; i < 4; i++) {
    result[i] = [];
    for (let j = 0; j < 4; j++) {
      result[i][j] = 0; // Inisialisasi dengan 0 untuk hasil akhir
      for (let k = 0; k < 4; k++) {
        const stateValue = parseInt(state[k][j], 16);
        const galoisValue = parseInt(R_GALOIS_FIELD_MATRIX[i][k],
16);
        result[i][j] ^= galoisMultiply(stateValue, galoisValue) %
256;
      }
    }
  }
}

```



```

        result[i][j]
result[i][j].toString(16).toUpperCase().padStart(2, "0");
    }
}
return result;
}
module.exports = {
    mixColumns,
};

```

**Gambar 35.** Kode *MixColumns*

Kode tersebut merupakan proses melakukan operasi *MixColumns* pada algoritma AES 128, dimana untuk enkripsi menggunakan matriks Galois Field. *Function* `galoisMultiply` merupakan proses perkalian di dalam medan Galois  $GF(2^8)$ , yang merupakan operasi matematika dasar dalam AES. Dalam fungsi ini:

1. *a* dan *b* adalah dua angka yang akan dikalikan.
2. Perkalian dilakukan bit per bit menggunakan metode shift dan XOR.
3. Jika bit paling signifikan dari *a* ( $\text{mask} = 0x80$ ) adalah 1, maka *a* digeser ke kiri dan dilakukan XOR dengan  $0x1b$  (polinomial reduksi).
4. *b* digeser ke kanan satu bit dalam setiap iterasi.

*Function* `mixColumns` merupakan proses perkalian Galois pada setiap heksadesimal yang ada pada suatu matriks heksadesimal dengan nilai dari matriks Galois Field.

```

//utils/encryption.js
const { addRoundKey } = require("./addRoundKey");
const { mixColumns } = require("./mixColumns");
const { substitutionSBox } = require("./sbox");
const { shiftRows } = require("./shiftRows");

function encryptBlock(hexArrayOfChars, roundKeys) {
    // add Round Key state plainteks dengan roundkey ke 0
    const stateAddRoundKeyWithState = addRoundKey(hexArrayOfChars,
roundKeys[0]);

```

```

// transformasi berulang dari ronde 1 hingga ronde ke 9
let temp = stateAddRoundKeyWithState;

for (let i = 1; i <= 9; i++) {
  // subbytes
  const subbytesState = substitutionSBox(temp);

  // shiftrows
  const shiftrowsState = shiftRows(subbytesState);

  // mix columns
  const mixColumnsState = mixColumns(shiftrowsState);

  // add round key dengan ronde ke i
  const addRoundKeyState = addRoundKey(mixColumnsState,
roundKeys[i]);
  temp = addRoundKeyState;
}

const round9State = [...temp];

// untuk round ke 10, lakukan operasi yang sama namun tanpa mix
column
// subbytes
const subbytesState10 = substitutionSBox(round9State);
// shiftrows
const shiftrowsState10 = shiftRows(subbytesState10);
// add round key dengan ronde ke 10
const addRoundKeyState10 = addRoundKey(shiftrowsState10,
roundKeys[10]);
const result = [...addRoundKeyState10]
return result
}
module.exports = {
  encryptBlock,
};

```

**Gambar 36.** Kode Enkripsi per *block* AES 128

Kode tersebut merupakan kode untuk melakukan enkripsi suatu data berupa *block* dan dilakukan sesuai dengan tahapan yang dilakukan pada algoritma AES 128.

```
//utils/aes.js
const { KEY } = require("../config");
const makingBlockOfChars = require("./block");
const makeRoundKeys = require("./roundKey");
const { encryptBlock } = require("./encryption");
const { flattenHexData } = require("./helper");
// membuat roundkeys
const key = KEY;
const roundKeys = makeRoundKeys(key);

const AESEncryption = (inputDataPlaintext) => {
  const blocks = makingBlockOfChars(inputDataPlaintext); //
  kumpulan plain text per huruf dan per blocks

  const encryptedBlocks = blocks.map((item) =>
    encryptBlock(item, roundKeys))
  const flattenData = flattenHexData(encryptedBlocks)
  const finalOutput = flattenData.join('').replace(/ /g, '');
  return finalOutput;
};

module.exports = {
  AESEncryption,
};;
```

**Gambar 37.** Kode enkripsi AES 128

Pada kode gambar 37 *function* ini lah yang digunakan pada *web server* untuk mengirimkan data pasien dalam bentuk data yang telah di enkripsi ke sisi *frontend* atau *web client* sebagai pihak yang melakukan *request* data pasien. Proses ini telah menyelesaikan satu tahap *End To End Encryption* pada satu arah.

### **3.8.3    *Web Client (Sisi Frontend)***

Pada *web service* salah satu sisi lainnya pada sistem ini ialah sisi *frontend* atau biasa disebut *web client*. Disini pembuatan *web client* diterapkan menggunakan *framework* NextJS.

#### **A.    Struktur Proyek *Web Client***

Berikut struktur proyek yang dirancang dalam membangun sisi *frontend*



**Gambar 38.** Struktur proyek sisi *web client*

Folder untuk *website* berfokus pada folder *app* dimana terdapat dua *route* tampilan website yaitu "no\_e2ee" dan "with\_e2ee". Khusus untuk kode penerapan algoritma AES 128 terletak pada folder *utils*.

```
//src/api/services/Users/index.js
import api, {ENDPOINT} from "@api";

const getAllUsers = async () => {
```

```

    try {
      const response = await api.get(`${ENDPOINT.USERS}`)
      return response.data
    } catch (error) {
      throw error
    }
  }
}

const getAllUsersNoE2EE = async () => {
  try {
    const response = await api.get(`${ENDPOINT.USERS}/no-e2ee`)
    return response.data
  } catch (error) {
    throw error
  }
}

export {
  getAllUsers,
  getAllUsersNoE2EE
}

```

**Gambar 39.** Kode *web service method GET* sisi *web client* dalam request data

Kode pada gambar 39 tersebut digunakan untuk melakukan *request* data pasien ke *web server* dimana sesuai dengan *routes* yang telah dibuat sebelumnya pada sisi *web server* untuk penggunaan sistem yang menerapkan *End To End Encryption* dan yang tidak menerapkannya.

```

//src/app/with_e2ee/page.js
"use client"
import { getAllUsers } from "@api/services/Users";
import UsersList from "@components/organism/UsersList";
import { AESDecryption } from "@utils/aes";
import { useEffect, useState } from "react";
export default function Home() {
  const [data, setData] = useState("")
  const [decryptedData, setDecryptedData] = useState()
  useEffect(() => {

```

```

(async() => {
    const resdata = await getAllUsers();
    const resdecryptedData = await AESDecryption(resdata)
    setData(resdata)
    setDecryptedData(resdecryptedData)
})()
},[])
return (
    <main className="container mx-auto p-7">
        <UsersList data={data && data} decryptedData={decryptedData
&& decryptedData} />
    </main>
);
}

```

**Gambar 40.** Kode *website* dan penerapan dekripsi AES 128

Kode pada gambar 40 tersebut merupakan kode *website* untuk *route* `"/with_e2ee"` yang mana menerapkan sistem *End To End Encryption* pada arah lainnya setelah sisi *web server*. Untuk kode pada *route* `"/no_e2ee"` melalui proses yang sama, namun menggunakan *function* yang berbeda untuk proses GET nya dan tidak melakukan proses dekripsi disini.

## B. Kode Dekripsi AES 128

Kode ini merupakan kode yang melakukan proses dekripsi data dimana dilakukan pada suatu masukan data sebagai parameter yang dijadikan dalam bentuk tipe data array heksadesimal. Kode ini dipisah menjadi berbagai *function* sesuai dengan algoritma AES 128.

```

//utils/shiftRows.js
function inverseShiftRows(hexArrayOfChars) {
    let state = hexArrayOfChars;
    // Baris pertama tidak diubah, tetap seperti itu

    // Baris kedua di shift ke kiri sebanyak 1 langkah
    state[1] = [state[1][3], state[1][0], state[1][1], state[1][2]];

```

```

// Baris ketiga di shift ke kiri sebanyak 2 langkah
state[2] = [state[2][2], state[2][3], state[2][0], state[2][1]];

// Baris keempat di shift ke kiri sebanyak 3 langkah
state[3] = [state[3][1], state[3][2], state[3][3], state[3][0]];

return state;
}

export{
  inverseShiftRows
};

```

**Gambar 41.** Kode *InverseShiftRows*

Kode tersebut merupakan kode melakukan operasi *InverseShiftRows* pada algoritma AES 128.

```

//utils/mixColumns.js
const INVERSE_MIXCOLUMNS_MATRIX = [
  ["0E", "0B", "0D", "09"],
  ["09", "0E", "0B", "0D"],
  ["0D", "09", "0E", "0B"],
  ["0B", "0D", "09", "0E"],
];

function galoisMultiply(a, b) {
  let result = 0;
  let mask = 0x80; // 0b10000000
  for (let i = 0; i < 8; i++) {
    if ((b & 1) === 1) {
      result ^= a;
    }
    const carry = a & mask;
    a <<= 1;
    if (carry !== 0) {
      a ^= 0x1b; // XOR dengan 0x1B jika carry terjad
    }
    b >>= 1;
  }
}

```



```

    }
    return result;
}
function inverseMixColumns(state) {
    const result = [];
    for (let i = 0; i < 4; i++) {
        result[i] = [];
        for (let j = 0; j < 4; j++) {
            result[i][j] = 0; // Inisialisasi dengan 0 untuk hasil akhir
            for (let k = 0; k < 4; k++) {
                const stateValue = parseInt(state[k][j], 16);
                const galoisValue =
                parseInt(INVERSE_MIXCOLUMNS_MATRIX[i][k], 16);
                result[i][j] ^= galoisMultiply(stateValue, galoisValue) %
256;
            }
            result[i][j] =
result[i][j].toString(16).toUpperCase().padStart(2, "0");
        }
    }
    return result;
}
module.exports = {inverseMixColumns};

```

**Gambar 42.** Kode *InverseMixColumns*

Kode pada gambar 42 tersebut merupakan proses melakukan operasi *InverseMixColumns* pada algoritma AES 128, dimana untuk dekripsi menggunakan matriks Inverse MixColumns.

*Function* *inverseMixColumns* merupakan proses perkalian Galois pada setiap heksadesimal yang ada pada suatu matriks heksadesimal dengan nilai dari matriks Inverse MixColumns.

```

//utils/decryption.js
import { addRoundKey } from "../addRoundKey";
import { inverseMixColumns } from "../mixColumns";
import { inverseSubstitutionSBox } from "../sbox";
import { inverseShiftRows } from "../shiftRows";

```

```

function decryptBlock(hexArrayOfChars, roundKeys) {
  // add Round Key state plainteks dengan roundkey ke 0
  const stateAddRoundKeyWithState = addRoundKey(hexArrayOfChars,
roundKeys[10]);
  // inverseShiftRows
  const stateAfterInverseShiftRows = inverseShiftRows(
    stateAddRoundKeyWithState
  );
  // inverseSubBytes
  const stateAfterInverseSubBytes = inverseSubstitutionSBox(
    stateAfterInverseShiftRows
  );
  // transformasi berulang dari ronde 9 hingga ronde ke 1
  let temp = stateAfterInverseSubBytes;
  for (let i = 9; i >= 1; i--) {
    // add round key dengan ronde ke i
    const addRoundKeyState = addRoundKey(temp, roundKeys[i]);
    // Inverse mix columns
    const inverseMixColumnsState =
inverseMixColumns(addRoundKeyState);
    // Inverse shiftrows
    const shiftrowsState =
inverseShiftRows(inverseMixColumnsState);
    // Inverse subbytes
    const subbytesState = inverseSubstitutionSBox(shiftrowsState);
    temp = subbytesState;
  }
  const decryptedData = addRoundKey(temp, roundKeys[0])
  return decryptedData
}
export { decryptBlock };

```

**Gambar 43.** Kode dekripsi per *block* AES 128

Kode pada gambar 43 tersebut merupakan kode untuk melakukan dekripsi suatu data berupa *block* dan dilakukan sesuai dengan tahapan yang dilakukan pada algoritma AES 128.

```
//utils/aes.js
import { KEY } from "@config";
import { makingBlockOfHex } from "../blockOfHex";
import makeRoundKeys from "../roundKey";
import { decryptBlock } from "../decryption";
import { hexArrayToString } from "../hexString";
// membuat roundkeys
const key = KEY;
const roundKeys = makeRoundKeys(key);

export const AESDecryption = async (encryptedData) => {
  const blocks = makingBlockOfHex(encryptedData)
  const decryptedBlocks = blocks.map((item) =>
    hexArrayToString(decryptBlock(item, roundKeys)))
  const finalOutput = decryptedBlocks.join("")
  return JSON.parse(finalOutput)
};
```

**Gambar 44.** Kode dekripsi AES 128

Kode pada gambar 44 dengan *function* ini lah yang digunakan pada *web client* untuk melakukan dekripsi pesan yang telah di terima setelah melakukan *request* data ke *web server* dimana data yang diterima berupa data yang telah di enkripsi lalu dilakukan proses dekripsi di sisi *web client* dan data pasien berhasil di dapatkan dengan aman. Proses ini telah menyelesaikan proses *End To End Encryption* pada dua arah.

### 3.9 Implementasi Serangan MITM *Sniffing*

Proses melakukan serangan MITM dilakukan untuk mencoba memantau proses transmisi data pada *web service* dengan *sniffing* sehingga mendapatkan data yang dikirimkan dari kedua sisi dari *web client* dan *web server*. Proses ini akan dilakukan menggunakan aplikasi Burp Suite dengan memanfaatkan fitur *proxy* dan *http history* dari aplikasi ini yang bisa melakukan proses pengiriman ulang suatu *request* dari *web client* yang tersimpan pada *http history*.

Untuk melakukan serangan ini juga perlu memastikan suatu *web server* dan *web client* telah berjalan dengan baik dan *web client* dapat diakses secara terbuka.

Disini untuk meneruskan *web client* ke port terbuka dengan suatu domain khusus yang mana domain ini lah yang akan digunakan pada aplikasi Burp Suite.

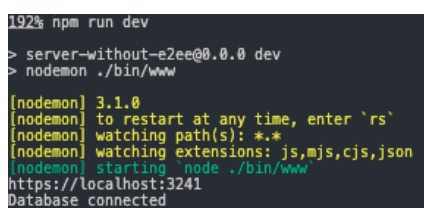
Pada sisi *web server* juga perlu diterapkan hal tersebut, namun kita hanya perlu menerapkan proses membuka *port* dari *device* sisi lokal agar *port* tersebut terbaca sebagai suatu lokasi yang dapat diakses *web client* agar dapat berhubungan dengan *web server*.

Oleh karena itu proses ini membutuhkan dua tahapan menjalankan *web client* dan *web server* agar dapat diakses. Untuk *web client* akan menggunakan suatu bantuan tambahan, yaitu "ngrok" yang merupakan suatu tempat melakukan pemajuan *port* suatu aplikasi *website* pada *device* tertentu agar dapat diakses secara terbuka.

Dikarenakan "ngrok" hanya dapat melakukan pemajuan *port* pada satu *port* saja, maka untuk *web server* akan menggunakan pemajuan *port* dari bantuan aplikasi VSCode.

### 3.9.1 Menjalankan Web Client dan Web Server

Untuk menjalankan *web client* dan *web server* sesuai dengan tools yang digunakan selama pembuatan dan implementasi pada masing-masing *framework*, dimana untuk menjalankan nya pada *device* masing-masing atau dikenal juga dengan berjalan pada sisi lokal, yaitu dengan perintah pada *terminal* atau *command prompt* berupa "npm run dev".



```
192% npm run dev
> server-without-e2ee@0.0.0 dev
> nodemon ./bin/www

[nodemon] 3.1.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting 'node ./bin/www'
https://localhost:3241
Database connected
```

**Gambar 45.** Proses menjalankan *web server* pada sisi lokal

Pada kode gambar 45 tersebut menunjukkan *web server* yang telah berjalan dengan baik yang ditandai dengan munculnya informasi "*Database connected*" dengan *web server* yang berjalan pada *port* 3241.

```
192% npm run dev
> frontend-website@0.1.0 dev
> next dev

▲ Next.js 14.1.3
- Local:      http://localhost:3000
- Environments: .env

✓ Ready in 3.2s
```

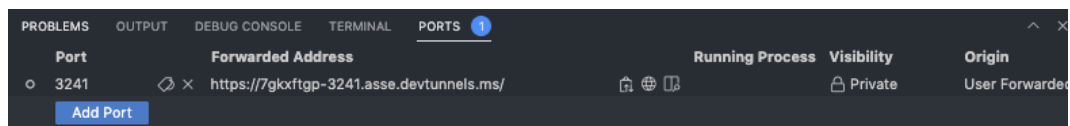
**Gambar 46.** Proses menjalankan *web client* pada sisi lokal

Pada kode gambar 46 tersebut menunjukkan *web client* yang telah berjalan dengan baik yang ditandai dengan munculnya informasi "*Ready in 3.2s*" dengan *web client* yang berjalan pada *port* 3000.

### 3.9.2 Melakukan *Forward Port* Pada *Web Client* dan *Web Server*

Memajukan atau melakukan *forward port* akan memungkinkan suatu *web client* atau *web server* dapat diakses secara terbuka dengan *device* lokal sebagai *server* nya. Hal ini sangat tepat untuk dilakukan sehingga dapat memberikan gambaran nyata proses implementasi penyerangan *sniffing* nantinya.

Untuk memajukan *port* pada *web server* kita menggunakan bantuan dari aplikasi VSCode yang dilakukan dengan cara setelah berjalannya *web server* maka pada bagian *Port* pada VSCode kita hanya perlu memasukkan *port web server* yang diinginkan, pada hal ini yaitu *port* ":3241". Sehingga *web server* yang hanya diterapkan pada lokal sudah dapat diakses terbuka.



**Gambar 47.** Proses *forward port* pada *web server*

Pada kode gambar 47 tersebut menunjukkan proses *forward port* menjadi *public* telah berhasil yang ditandai dengan adanya *forwarded address* yang dapat digunakan.

Setelah melakukan *forward port* pada *web server*, lakukan perubahan konfigurasi *web client* berupa pengaturan URL yang perlu dihubungkan ke *web server* tersebut.

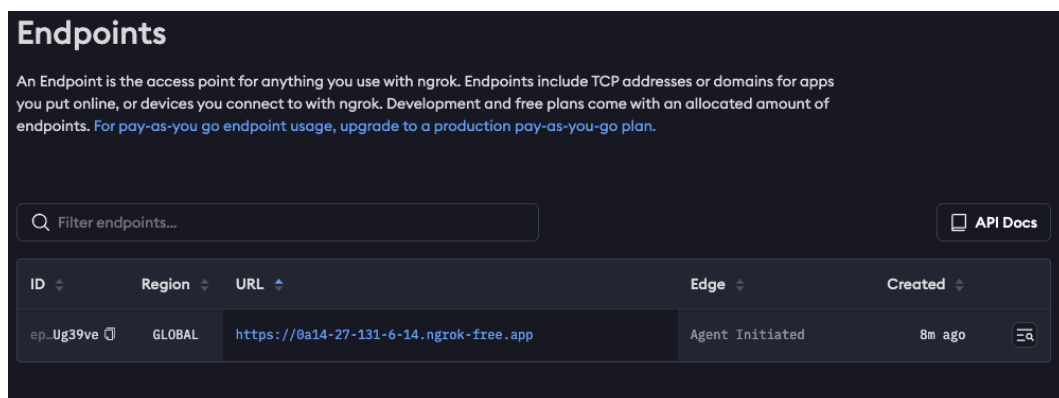
```
// .env
//NEXT_PUBLIC_BACKEND_URL=http://localhost:3241 awal secara lokal
```

```
NEXT_PUBLIC_BACKEND_URL= https://7gkxftgp-3241.asse.devtunnels.ms
NEXT_PUBLIC_KEY=FTTKUMRAHBERJAYA
```

**Gambar 48.** Konfigurasi *web client* setelah *web server* di *forward*

Pada kode gambar 48 dilakukan konfigurasi URL *web server* dari sisi lokal ke sisi *forward address* yang sebelumnya telah dilakukan.

Kemudian kita hanya perlu memajukan *port* pada sisi *web client* dengan bantuan "ngrok". Untuk melakukannya diperlukan suatu akun yang telah terdaftar pada situs resmi "ngrok" dan dapat menggunakan perintah berikut untuk melakukan *forward port* sisi *web client* yaitu "ngrok http 3000". Lalu bukalah situs "ngrok" dan pada tampilan situs *dashboard* terdapat suatu pilihan *endpoints* yang berisi situs URL *web client* yang berjalan.



**Gambar 49.** Proses *forward port* pada *web client*

Pada kode gambar 49 tersebut menunjukkan *web client* yang telah berhasil dilakukan *forward port* dengan adanya *web address* pada kolom URL.

### 3.9.3 Melakukan *Sniffing* Dengan Burp Suite

Setelah mendapatkan URL *web client* yang telah terbuka, selanjutnya kita lakukan serangan MITM *sniffing*. Dengan menggunakan Burp Suite setelah membuka aplikasi tersebut akan terdapat pilihan menu tab yaitu *Proxy* dan didalam menu tersebut akan terdapat pilihan *intercept* dan perlu kita hidupkan. Setelah dihidupkan maka kita gunakan *browser* dengan *proxy* Burp Suite untuk membuka URL dari *web client* dan akan tersimpan seluruh proses *http history* yang terjadi.

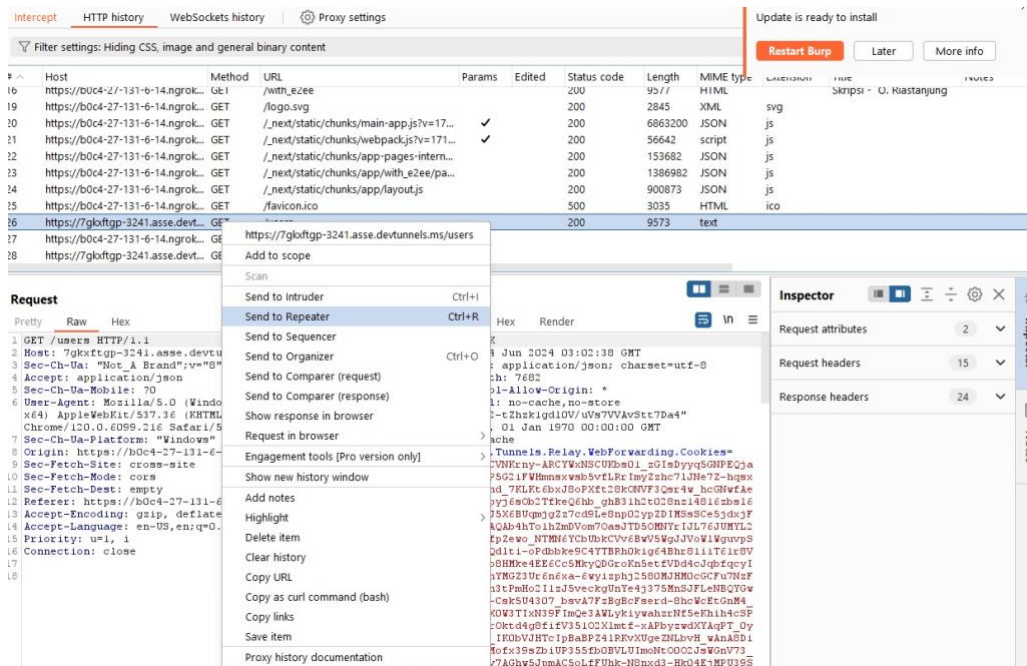
Untuk sistem *web service* yang menggunakan sistem E2EE terletak pada URL `"/with_e2ee"` dari *web client* sehingga URL tersebut yang kita gunakan pada *browser* Burp Suite ini. Seluruh proses *web service* yang terjadi pada *web client* akan tersimpan di tab *http history* seperti berikut.

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP
1	http://www.gstatic.com	GET	/generate_204			204	127						172.217.194.94
2	https://0a14-27-131-6-14.n...	GET	/with_e2ee			200	40139	HTML		Skripsi - O. Riastanj...		✓	52.220.121.212
3	https://0a14-27-131-6-14.n...	GET	/with_e2ee			200	40139	HTML		Skripsi - O. Riastanj...		✓	52.220.121.212
4	https://0a14-27-131-6-14.n...	GET	/with_e2ee			200	40139	HTML		Skripsi - O. Riastanj...		✓	52.220.121.212
6	https://0a14-27-131-6-14.n...	GET	/logo.svg			304	193		svg			✓	52.220.121.212
7	https://0a14-27-131-6-14.n...	GET	/_next/static/chunks/webpack.js?v...		✓	200	56642	script	js			✓	52.220.121.212
8	https://0a14-27-131-6-14.n...	GET	/_next/static/chunks/main-app.js?...		✓	200	6863200	JSON	js			✓	52.220.121.212
9	https://0a14-27-131-6-14.n...	GET	/_next/static/chunks/app-pages-in...			200	153682	JSON	js			✓	52.220.121.212
10	https://0a14-27-131-6-14.n...	GET	/_next/static/chunks/app/layout.js			200	900873	JSON	js			✓	52.220.121.212
11	https://0a14-27-131-6-14.n...	GET	/_next/static/chunks/app/with_e2e...			200	152559	JSON	js			✓	52.220.121.212
13	https://0a14-27-131-6-14.n...	GET	/with_e2ee			200	40139	HTML		Skripsi - O. Riastanj...		✓	52.220.121.212
15	https://0a14-27-131-6-14.n...	GET	/_next/static/chunks/webpack.js?v...		✓	200	56642	script	js			✓	52.220.121.212
16	https://0a14-27-131-6-14.n...	GET	/_next/static/chunks/main-app.js?...		✓	200	6863200	JSON	js			✓	52.220.121.212
17	https://0a14-27-131-6-14.n...	GET	/_next/static/chunks/app/with_e2e...			200	152559	JSON	js			✓	52.220.121.212

**Gambar 50.** Riwayat proses transmisi data *web service*

Pada gambar 50 seluruh proses transmisi data dari *web service* akan terlihat dan disini pula akan dilakukan pemilihan suatu *request* yang dianggap proses mendapatkan suatu informasi.

Kemudian pada *endpoint* yang dipilih kita kirimkan ke *repeater* Burp Suite untuk melakukan proses pengulangan *web service*.

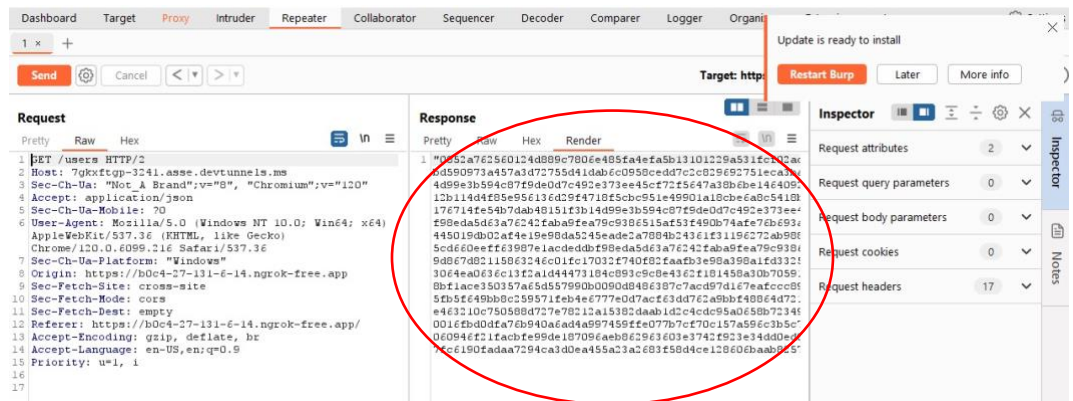


**Gambar 51.** Proses melakukan *sniffing*

Pada gambar 51 dengan fitur *repeater* kita akan melakukan proses pengiriman informasi kembali ke *web service* sehingga kita dalam proses pengambilan informasi sebagai pihak yang tidak berwenang atau *man in the middle* yang melakukan serangan *sniffing*.

Pada *repeater* kita lakukan pengriman *request* lagi ke *web service* sehingga mengembalikan suatu data yang dikirimkan pada suatu transmisi data *web service*. Setelah melakukan *repeater* selanjutnya kita akan mendapatkan hasil dari *sniffing* berupa data yang terbaca dan disinilah pihak MITM mengambil suatu data tersebut.

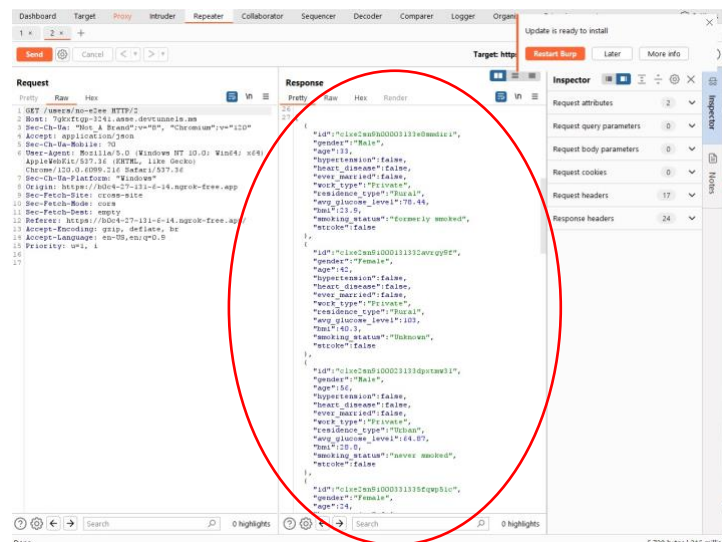




**Gambar 52.** Hasil *sniffing* pada *web service* dengan E2EE

Pada gambar 52 dapat dilihat hasil *sniffing* pada sistem E2EE menghasilkan suatu data yang terenkripsi, sehingga serangan tidak dapat menghasilkan suatu informasi yang secara langsung dapat dibaca oleh orang yang tidak berwenang.

Jika kita lakukan hal yang sama pada sistem *web service* yang tidak diterapkan E2EE, maka hasilnya akan ditampilkan suatu data pasien secara langsung.



**Gambar 53.** Proses *sniffing* pada *web service* tanpa E2EE

Pada gambar 53 dapat dilihat hasil *sniffing* pada sistem tanpa E2EE menghasilkan suatu informasi yang secara langsung dapat dibaca oleh orang yang tidak berwenang.

## BAB IV

### HASIL DAN PEMBAHASAN

#### 4.1 Analisis Sistem *End To End Encryption*

Penerapan sistem *End-to-End Encryption* (E2EE) pada *web service* memiliki hasil yang menunjukkan berperan penting dalam menjaga kerahasiaan data yang ditransmisikan antara *web client* dan *web server*. Dalam penelitian ini, algoritma *Advanced Encryption Standard* (AES) dengan panjang kunci 128-bit digunakan untuk memastikan keamanan data selama proses transmisi. E2EE bekerja dengan cara mengenkripsi data di sisi pengirim dan hanya dapat didekripsi oleh penerima yang memiliki kunci yang sesuai, sehingga pihak ketiga yang tidak berwenang tidak dapat mengakses data tersebut.

Implementasi E2EE pada *web service* diuji dengan mengirimkan suatu transmisi data yang kemudian dibaca oleh pihak ketiga sebagai pihak yang melakukan serangan MITM *sniffing*. Hasil pengujian menunjukkan bahwa data yang dienkripsi menggunakan AES-128 tetap aman selama proses transmisi, bahkan ketika data tersebut diambil oleh pihak yang tidak berwenang. Ini menunjukkan bahwa E2EE efektif dalam melindungi integritas dan kerahasiaan data. Berkaitan dengan integritas data, proses data yang terenkripsi berhasil di dekripsi oleh pihak *web client* menunjukkan pula keberhasilan sistem untuk menjaga suatu data agar dapat digunakan sisi *frontend* atau *web client* dengan tidak terjadinya masalah.

Dalam penerapannya, E2EE memerlukan penanganan kunci kriptografi yang aman dan efisien. Setiap entitas yang terlibat yaitu *web client* dan *web server* memiliki kunci pribadi yang disimpan pada *environment* berjalannya proyek sehingga dapat melakukan proses enkripsi dan dekripsi. Penyimpanan kunci dengan cara seperti ini sudah sering diterapkan dan terbukti sangat aman.

Pada proses penerapannya, terdapat beberapa kekurangan yang dirasakan selama penelitian berlangsung. Salah satunya terkait jumlah pengiriman data yang perlu diperhatikan mengingat banyak nya proses yang terjadi selama satu transmisi data. Hal ini mungkin dapat menjadi salah satu kekurangan dimana proses transmisi

data dengan *method* GET biasanya memerlukan suatu waktu pengiriman yang efisien dan efektif. Hal ini bisa diatasi dengan beberapa cara yang terdapat pada *web service* seperti konsep paginasi atau proses pengiriman beberapa data sesuai yang dibutuhkan saja, lalu adapun solusi lain dengan pemilihan algoritma yang lebih ringan.

#### 4.2 Hasil Serangan Pada Web Service

Pada penelitian ini, diterapkan suatu serangan pada *web service* menggunakan serangan *Man In The Middle Attack* dengan jenis *sniffing* yang mana proses serangan ini bertujuan untuk mendapatkan suatu data yang terjadi selama proses transmisi data antar dua pihak pada *web service* yaitu sisi *frontend* atau *web client* dan sisi *backend* atau *web server*. Penelitian ini telah berhasil melakukan uji coba serangan pada dua sistem *web service* yaitu sistem yang menerapkan E2EE sebagai sistem yang diajukan dan juga sistem *web service* konvensional.

Untuk menunjukkan hasil dari penerapan *web service* yang menerapkan E2EE dan yang tidak menerapkannya, kita dapat melihat hasil yang didapatkan ketika melakukan serangan dengan MITM *sniffing* pada tabel berikut.

**Tabel 5. Perbandingan Hasil Serangan Pada Sistem**

Percobaan	Jenis Sistem	Hasil Serangan
Percobaan 1	Sistem Dengan E2EE	0852a762560124d889c7806e485fa4efa5b13101229a 531fcf02ac0347ea886ea83b4779cb946188aeed 1bad922ce6e2f726ab2b2eebbf7382c1972c31ce6 5734911561f63d51b74bc47c06396fb0bbc2c0 ab5773e22fb1315694357a6c58442248b36c9997786847 ..... aede6cc0216a03d5aa24f7072b1fb3 75b376e37386a5a30b547b991b5bd3f4c4d407a97 77fc6190fadaa7294ca3d0ea455a23a2683 f58d4ce128606baab8257374c160600f8a

		289634be48159b003cacce36949f39b3729b22 7e3b59d1a6eb00f05848321f7db6b3da34ed8aa 5928cb4d8004adb62d967e440f16818d2697
	Sistem Tanpa E2EE	[ {"id":"clxe2sn9h00003l33e8smdir", "gender":"Male","age":33, "hypertension":false,"heart_disease":false, "ever_married":false, "work_type":"Private","residence_type":"Rural", "avg_glucose_level":78.44, "bmi":23.9, "smoking_status":"formerly smoked", "stroke":false}, ..... {"id":"clxe2sn9j000e3l3336rrbi26", "gender":"Female","age":59, "hypertension":false,"heart_disease":false, "ever_married":false, "work_type":"Private", "residence_type":"Rural", "avg_glucose_level":100.54, "bmi":28.1, "smoking_status":"never smoked", "stroke":false} ]

Percobaan 2	Sistem Dengan E2EE	0852a762560124d889c7806e485fa4efa5b13101229a 531fcf02ac0347ea886ea83b4779cb946188aecd 1bad922ce6e2f726ab2b2eebbf7382c1972c31ce6 5734911561f63d51b74bc47c06396fb0bbc2c0 ab5773e22fb1315694357a6c58442248b36c9997786847 ..... aede6cc0216a03d5aa24f7072b1fb3 75b376e37386a5a30b547b991b5bd3f4c4d407a97 77fc6190fadaa7294ca3d0ea455a23a2683 f58d4ce128606baab8257374c160600f8a 289634be48159b003cacce36949f39b3729b22 7e3b59d1a6eb00f05848321f7db6b3da34ed8aa 5928cb4d8004adb62d967e440f16818d2697
	Sistem Tanpa E2EE	[ {"id":"clxe2sn9h00003l33e8smdir", "gender":"Male","age":33, "hypertension":false,"heart_disease":false, "ever_married":false, "work_type":"Private","residence_type":"Rural", "avg_glucose_level":78.44, "bmi":23.9, "smoking_status":"formerly smoked", "stroke":false}, ..... {"id":"clxe2sn9j000e3l3336rrbi26", "gender":"Female","age":59, "hypertension":false,"heart_disease":false, "ever_married":false,

		<pre> "work_type":"Private", "residence_type":"Rural", "avg_glucose_level":100.54, "bmi":28.1, "smoking_status":"never smoked", "stroke":false} ] </pre>
Percobaan 3	Sistem Dengan E2EE	<pre> 0852a762560124d889c7806e485fa4efa5b13101229a 531fcf02ac0347ea886ea83b4779cb946188aecd 1bad922ce6e2f726ab2b2eebbf7382c1972c31ce6 5734911561f63d51b74bc47c06396fb0bbc2c0 ab5773e22fb1315694357a6c58442248b36c9997786847 ..... aede6cc0216a03d5aa24f7072b1fb3 75b376e37386a5a30b547b991b5bd3f4c4d407a97 77fc6190fadaa7294ca3d0ea455a23a2683 f58d4ce128606baab8257374c160600f8a 289634be48159b003cacce36949f39b3729b22 7e3b59d1a6eb00f05848321f7db6b3da34ed8aa 5928cb4d8004adb62d967e440f16818d2697 </pre>
	Sistem Tanpa E2EE	<pre> [ {"id":"clxe2sn9h00003133e8smdiri", "gender":"Male","age":33, "hypertension":false,"heart_disease":false, "ever_married":false, "work_type":"Private","residence_type":"Rural", "avg_glucose_level":78.44, "bmi":23.9, "smoking_status":"formerly smoked", </pre>

		<pre> "stroke":false}, .....  {"id":"clxe2sn9j000e3l3336rrbi26", "gender":"Female","age":59, "hypertension":false,"heart_disease":false, "ever_married":false, "work_type":"Private", "residence_type":"Rural", "avg_glucose_level":100.54, "bmi":28.1, "smoking_status":"never smoked", "stroke":false} ] </pre>
Percobaan 4	Sistem Dengan E2EE	<pre> 0852a762560124d889c7806e485fa4efa5b13101229a 531fcf02ac0347ea886ea83b4779cb946188aeed 1bad922ce6e2f726ab2b2eebbf7382c1972c31ce6 5734911561f63d51b74bc47c06396fb0bbc2c0 ab5773e22fb1315694357a6c58442248b36c9997786847 ..... aede6cc0216a03d5aa24f7072b1fb3 75b376e37386a5a30b547b991b5bd3f4c4d407a97 77fc6190fadaa7294ca3d0ea455a23a2683 f58d4ce128606baab8257374c160600f8a 289634be48159b003cacce36949f39b3729b22 7e3b59d1a6eb00f05848321f7db6b3da34ed8aa 5928cb4d8004adb62d967e440f16818d2697 </pre>

	Sistem Tanpa E2EE	[ <pre> {"id":"clxe2sn9h00003l33e8smdiri", "gender":"Male","age":33, "hypertension":false,"heart_disease":false, "ever_married":false, "work_type":"Private","residence_type":"Rural", "avg_glucose_level":78.44, "bmi":23.9, "smoking_status":"formerly smoked", "stroke":false}, .....  {"id":"clxe2sn9j000e3l3336rrbi26", "gender":"Female","age":59, "hypertension":false,"heart_disease":false, "ever_married":false, "work_type":"Private", "residence_type":"Rural", "avg_glucose_level":100.54, "bmi":28.1, "smoking_status":"never smoked", "stroke":false} </pre> ]
Percobaan 5	Sistem Dengan E2EE	0852a762560124d889c7806e485fa4efa5b13101229a 531fcf02ac0347ea886ea83b4779cb946188aeed 1bad922ce6e2f726ab2b2eebbf7382c1972c31ce6 5734911561f63d51b74bc47c06396fb0bbc2c0 ab5773e22fb1315694357a6c58442248b36c9997786847



		<p>.....</p> <p>aede6cc0216a03d5aa24f7072b1fb3</p> <p>75b376e37386a5a30b547b991b5bd3f4c4d407a97</p> <p>77fc6190fadaa7294ca3d0ea455a23a2683</p> <p>f58d4ce128606baab8257374c160600f8a</p> <p>289634be48159b003cacce36949f39b3729b22</p> <p>7e3b59d1a6eb00f05848321f7db6b3da34ed8aa</p> <p>5928cb4d8004adb62d967e440f16818d2697</p>
	<p>Sistem</p> <p>Tanpa</p> <p>E2EE</p>	<p>[</p> <p>{ "id": "clxe2sn9h00003l33e8smdiri",</p> <p>"gender": "Male", "age": 33,</p> <p>"hypertension": false, "heart_disease": false,</p> <p>"ever_married": false,</p> <p>"work_type": "Private", "residence_type": "Rural",</p> <p>"avg_glucose_level": 78.44,</p> <p>"bmi": 23.9,</p> <p>"smoking_status": "formerly smoked",</p> <p>"stroke": false},</p> <p>.....</p> <p>{ "id": "clxe2sn9j000e3l3336rrbi26",</p> <p>"gender": "Female", "age": 59,</p> <p>"hypertension": false, "heart_disease": false,</p> <p>"ever_married": false,</p> <p>"work_type": "Private",</p> <p>"residence_type": "Rural",</p> <p>"avg_glucose_level": 100.54,</p> <p>"bmi": 28.1,</p>

		<pre>"smoking_status":"never smoked", "stroke":false} ]</pre>
--	--	---

Dari analisis yang dilakukan, dapat disimpulkan bahwa penerapan E2EE pada *web service* dengan AES-128 berhasil memberikan perlindungan yang memadai terhadap data yang ditransmisikan yang dapat dilihat setelah uji coba serangan sebanyak lima kali dan menghasilkan hasil serangan yang sama pula. Integritas data tetap terjaga, dan sistem dapat beroperasi dengan baik tanpa gangguan yang berarti. Keberhasilan ini menunjukkan potensi besar penggunaan E2EE dalam berbagai aplikasi *web service*.

## **BAB V**

### **PENUTUP**

#### **5.1 Kesimpulan**

Penelitian ini berhasil mengimplementasikan *web service* dengan sistem *End-to-End Encryption* (E2EE) menggunakan metode *Advanced Encryption Standard* (AES) 128. Penerapan E2EE pada *web service* menunjukkan bahwa data yang dikirim melalui jaringan lebih aman dari serangan *Man In The Middle* (MITM) dibandingkan dengan *web service* tanpa E2EE. Hasil pengujian menunjukkan bahwa *sniffing* pada *web service* yang menggunakan E2EE menghasilkan data yang terenkripsi, sehingga tidak dapat dibaca oleh pihak yang tidak berwenang. Sebaliknya, *web service* tanpa E2EE rentan terhadap serangan MITM, di mana data dapat dengan mudah diakses dan disalahgunakan oleh penyerang.

Penggunaan AES 128 sebagai metode enkripsi terbukti efektif dalam menjaga kerahasiaan dan integritas data selama transmisi dimana algoritma enkripsi ini sudah memberikan tingkat keamanan yang memadai untuk aplikasi *web service* yang diujikan dalam penelitian ini.

Secara keseluruhan, penelitian ini memberikan pengetahuan penting dalam bidang keamanan informasi, terutama dalam konteks perlindungan data pada *web service*. Penelitian ini memberikan kontribusi pengetahuan dengan memaparkan proses pencegahan serangan pada transmisi data *web service* dengan menerapkan sistem E2EE dan sistem tanpa E2EE yang mana pada penerapannya menggunakan algoritma AES 128, sehingga risiko pencurian data dan penyalahgunaan informasi dapat diminimalkan secara signifikan dari serangan pihak yang tidak berwenang.

#### **5.2 Saran**

Berdasarkan kekurangan yang ditemukan selama penelitian dan pada analisis penerapan sistem, disarankan untuk mengoptimalkan proses transmisi data dalam *web service* dengan beberapa pendekatan. Salah satu cara untuk mengatasi isu efisiensi dan efektivitas waktu pengiriman adalah dengan menerapkan konsep paginasi atau *pagination* yang memungkinkan pengiriman data dalam jumlah kecil

dan bertahap, sehingga mengurangi beban transmisi. Selain itu, penggunaan metode pengiriman yang hanya mengirimkan data sesuai kebutuhan dapat membantu meningkatkan performa. Disarankan juga untuk mempertimbangkan penggunaan algoritma enkripsi yang lebih ringan namun tetap aman, guna mempercepat proses enkripsi dan dekripsi tanpa mengorbankan keamanan data.

## DAFTAR PUSTAKA

- Adam, P., & Romli, M. A. (2024). Implementasi Sistem Keamanan Dokumen Kepegawaian Menggunakan Metode Aes-256 Dan Vigenere Chiper. *Jurnal Komputer dan Teknologi*, 2(2), 20-26.
- Amalia, P., & Nasution, M. I. P. (2024). Tantangan Terkini Dalam Keamanan Informasi Analisis Resiko Dan Upaya Perlindungan. *Jurnal Ekonomi Bisnis dan Manajemen*, 2(1), 24-37.
- Ardiansyah, A. Y., Sibarani, A. J., Pramusinto, W., & Yudha, D. V. S. (2023, April). Rancang Bangun E-Learning Berbasis Web Service Dengan Metode Rest-Api. In *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)*, (Vol. 2, No. 1, pp. 474-481).
- Arnaldy, D., & Perdana, A. R. (2019, September). Implementation and analysis of penetration techniques using the man-in-the-middle attack. In *2019 2nd International Conference of Computer and Informatics Engineering (IC2IE)* (pp. 188-192). *IEEE*.
- Awalsyah, R. M. S. A., Harahap, P. S., & Dono, M. (2023). Implementasi Caesar Cipher Dalam Mengenkripsikan Pesan Pada Serangan Man In The Middle Attack. *JOCITIS-Journal Science Infomatica and Robotics*, 1(1), 64-72.
- Baharuddin, B., Wakkang, H., & Irianto, B. (2022). Implementasi Web Service Dengan Metode Rest Api Untuk Integrasi Data Covid 19 Di Sulawesi Selatan. *Jurnal Sintaks Logika*, 2(1), 236-241.
- Bai, W., Pearson, M., Kelley, P. G., & Mazurek, M. L. (2020). Improving non-experts' understanding of end-to-end encryption: an exploratory study. In *2020 IEEE european symposium on security and privacy workshops (EuroS&PW)* (pp. 210-219). *IEEE*.

- Basatwar, G. (2023). *Understanding AES-128 Encryption And Its Significance In The Current Threat Landscape*. [Online]. Tersedia di: <https://appsealing.com/aes-128-encryption/#:~:text=If%20ypu%20ask%20how%20long,a%20128-bit%20AES%20key>. Diakses tanggal 16 Januari 2024.
- Cristy, N., & Riandari, F. (2021). Implementasi Metode Advanced Encryption Standard (AES 128 Bit) Untuk Mengamankan Data Keuangan. *Jurnal Ilmu Komputer dan Sistem Informasi (JIKOMSI)*, 4(2), 75-85.
- Meko, D. A. (2018). Perbandingan Algoritma DES, AES, IDEA Dan Blowfish dalam Enkripsi dan Dekripsi Data. *Jurnal Teknologi Terpadu (JTT)*, 4(1).
- Daemen, J., & Rijmen, V. (1999). AES proposal: Rijndael. [Online]. Tersedia di [https://www.cs.miami.edu/home/burt/learning/Csc688.012/rijndael/rijndael\\_doc\\_V2.pdf](https://www.cs.miami.edu/home/burt/learning/Csc688.012/rijndael/rijndael_doc_V2.pdf). Diakses tanggal 18 Juni 2024.
- Elshoush, H. T., Mohammed, R. M., Abdelhameed, M. T., & Mohammed, A. F. (2023). *Mitigating Man-in-the-middle Attack In Online Payment System Transaction Using Polymorphic AES Encryption Algorithm*.
- Hamzah, F. M. (2014). *ASCII*. [Online]. Tersedia di <https://mahasiswa.ung.ac.id/531414009/home/2014/10/4/ascii.html>. Diakses tanggal 23 Januari 2024.
- Herlambang, D. N. R., Nilma, N., & Pravitasari, N. (2024). Penerapan Kriptografi AES untuk Keamanan Data Aplikasi Pemesanan Bibit Ternak pada BPSI UAT. *REMIK: Riset dan E-Jurnal Manajemen Informatika Komputer*, 8(1), 29-44.

- Huda, N. (2022). *Pengertian Web Service, Cara Kerja, dan Fungsinya*. [Online]. Tersedia di <https://www.dewaweb.com/blog/apa-itu-web-service/>. Diakses tanggal 21 Januari 2024.
- Ikhwanuzaki, M. F., & Handayani, I. (2024). Implementasi Web Service Menggunakan Restful Api pada Aplikasi Pemesanan Sarung Goyor Suhutex. *Jurnal Riset dan Aplikasi Mahasiswa Informatika (JRAMI)*, 5(1), 191-199.
- Khasanah, S., & Sutabri, T. (2023). Analisis Pencegahan Pencurian Data Melalui Aplikasi Whatsapp Menggunakan Metode Kriptografi. *Sainteks: Jurnal Sain dan Teknik*, 5(2), 145-153.
- Lestari, S. P., Fadlan, H. N., Purba, R. A., & Gunawan, I. (2022). Realisasi Kriptografi Pada Fitur Enkripsi End-To-End Pesan Whatsapp. *Jurnal Media Informatika*, 4(1), 1-8.
- Liander, G. V. (2022). Penggunaan Algoritma Elliptic Curve Diffie Hellman dan AES 256 pada Implementasi End-to-End Encryption WhatsApp. *Sumber*, 7680(384), 15360.
- Luqman, F., Nurul, H., & Alena, U. (2022). Penerapan Kombinasi Algoritma Advanced Encryption Standard (Aes) Dan Elgamal Dengan Fungsi Hash Secure Hash Algorithm (Sha) Dalam Penyandian File Dokumen, (*Doctoral dissertation, Universitas Maritim Raja Ali Haji*).
- Maharani, Y. S., Trisdiatin, S., Ihsanuddin, M. R., & Rahma, F. (2023, November). Kekuatan Enkripsi End-to-End: Kajian Literatur Mengenai Kerahasiaan Komunikasi Digital dalam Aplikasi Pesan Instan. In *Seminar Nasional Teknologi Informasi dan Matematika (SEMIOTIKA)* (Vol. 2, No. 1, pp. 1-7).

- Manguling, I. S. W., & Parenreng, J. M. (2023). Security System Analysis Using the HTTP Protocol Against Packet Sniffing Attacks. *Internet of Things and Artificial Intelligence Journal*, 3(4), 325-340.
- Mouli, V. R., & Jevitha, K. P. (2016). Web services attacks and security-a systematic literature review. *Procedia Computer Science*, 93, 870-877.
- Muzakir, A. (2013). Sistem Keamanan Data Pada Web Service Menggunakan Xml Encryption. *SEMNASTEKNOMEDIA ONLINE*, 1(1), 25-7.
- Ridho, M., & Kusumaningsih, D. (2023, October). Implementasi Kriptografi Menggunakan Algoritme Advanced Encryption Standart (Aes-128) Mengamankan Data Customer Pt. Sarana Teknik Internusa. In *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)* (Vol. 2, No. 2, pp. 269-278).
- Salahuddin, S. (2022). Analisis Kinerja Grpc Dan Rest Api Pada Pertukaran Data Antar Microservices Performance Analysis Of GRPC and REST API On Data Exchange Between Microservices, *Skripsi Universitas Hasanudin*.
- Selimis, G. N., Kakarountas, A. P., Fournaris, A. P., Milidonis, A., & Koufopavlou, O. (2007). A low power design for sbox cryptographic primitive of advanced encryption standard for mobile end-users. *Journal of Low Power Electronics*, 3(3), 327-336.
- Setyowardhani, A. F., Nurlela, I., Primaranti, J., Ghrandiaz, V., & Yulhendri, Y. (2024). Analisis Resiko Keamanan Informasi Website Repository Digital Library Menggunakan Framework ISO/IEC 27001 & 27002: Studi Kasus



Perguruan tinggi X. *Jurnal Riset Multidisiplin dan Inovasi Teknologi*, 2(01), 327-373.

Simbulan, R., & Aryanto, J. (2024). Implementasi REST API Web Services pada Aplikasi Sumber Daya Manusia. *Jurnal Indonesia: Manajemen Informatika dan Komunikasi*, 5(1), 552-560.

Wulandari, I. W., & Hwihanus, H. (2023). Peran Sistem Informasi Akuntansi Dalam Pengaplikasian Enkripsi Terhadap Peningkatan Keamanan Perusahaan. *Jurnal Kajian dan Penalaran Ilmu Manajemen*, 1(1), 11-25.

Yaomulfurqqan, A., & Pramusinto, W. (2023). Implementasi Algoritme Aes 128 Untuk Keamanan File Berbasis Web. In *Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI)* (Vol. 2, No. 2, pp. 196-205).

Zulkarnain, Z. (2020). Analisis Keamanan FTP server Menggunakan Serangan Man-In-The-Middle Attack. *Telcomatics*, 5(1).



## LAMPIRAN

Lampiran 1. Data Pasien Dari Kaggle

id	gender	age	hypertension	heart_disease	ever_married	work_type	residence_type	avg_glucose_level	bmi	smoking_status	stroke
clxe2sn9h00003l33e8smdir	Male	33	FALSE	FALSE	FALSE	Private	Rural	78.44.00	23.09	formerly smoked	FALSE
clxe2sn9i00013l332avrgy9f	Female	42	FALSE	FALSE	FALSE	Private	Rural	103	40.03.00	Unknown	FALSE
clxe2sn9i00023l33dpxtmw3l	Male	56	FALSE	FALSE	FALSE	Private	Urban	64.87	28.08.00	never smoked	FALSE
clxe2sn9i00033l335fqwp51c	Female	24	FALSE	FALSE	FALSE	Private	Rural	73.36.00	28.08.00	never smoked	FALSE
clxe2sn9i00043l33k54qf6si	Female	34	FALSE	FALSE	FALSE	Private	Urban	84.35.00	22.02	Unknown	FALSE
clxe2sn9i00053l33gqkj9zr	Female	53	FALSE	FALSE	FALSE	Private	Rural	88.97	25.03.00	never smoked	FALSE
clxe2sn9i00063l33j7vlss0s	Male	78	FALSE	TRUE	FALSE	Self-employed	Rural	75.32.00	24.08.00	Unknown	FALSE
clxe2sn9i00073l33m0izfxna	Female	45	FALSE	FALSE	FALSE	Private	Rural	107.22.00	34.01.00	never smoked	FALSE
clxe2sn9i00083l33pdd4u8hg	Female	62	FALSE	FALSE	FALSE	Govt_job	Urban	62.68	18.04	formerly smoked	FALSE
clxe2sn9i00093l33x7k5xpxa	Male	51	FALSE	FALSE	FALSE	Self-employed	Urban	114.89	20.01	never smoked	FALSE
clxe2sn9i000a3l33v8sb57bg	Female	45	FALSE	FALSE	FALSE	Private	Urban	69.94	23.05	never smoked	FALSE
clxe2sn9i000b3l33vkqifw7x	Female	4	FALSE	FALSE	FALSE	children	Urban	84.01.00	14.01	Unknown	FALSE
clxe2sn9j000c3l33r7mqbwcb	Male	23	FALSE	FALSE	FALSE	Private	Urban	112.09.00	37.03.00	smokes	FALSE
clxe2sn9j000d3l337o6zy348	Female	36	FALSE	FALSE	FALSE	Private	Rural	73.78	29.06.00	never smoked	FALSE
clxe2sn9j000e3l3336rbi26	Female	59	FALSE	FALSE	FALSE	Private	Rural	100.54.00	28.01.00	never smoked	FALSE