

BACKEND API DATA PROTECTION MENGGUNAKAN JWT TOKEN DAN ALGORITMA AES 256-BIT DENGAN BAHASA PEMROGRAMAN GOLANG

Agustinus Bayu Prasetyo^{1*}, Imam Suharjo²

^{1,2} Universitas Mercu Buana Yogyakarta, Kampus 2, Depok, Sleman, Yogyakarta 55283, Indonesia

Received: 11 Desember 2024

Accepted: 14 Januari 2025

Published: 20 Januari 2025

Keywords:

AES 256-bit;
JSON Web Token (JWT);
Golang;
Golang Data Security;
Backend API.

Correspondent Email:

agustinusbayu77@gmail.com

Abstrak. Keamanan data menjadi perhatian utama dalam pengembangan aplikasi berbasis web dan backend API, terutama dalam melindungi informasi sensitif dari akses yang tidak sah. Penelitian ini bertujuan mengintegrasikan Advanced Encryption Standard (AES) 256-bit untuk enkripsi data dan JSON Web Token (JWT) untuk autentikasi pengguna, yang dikembangkan menggunakan bahasa pemrograman Go. Metode yang digunakan adalah Research and Development (R&D), meliputi tahap analisis kebutuhan, desain sistem, implementasi, dan pengujian. Hasil penelitian menunjukkan bahwa kombinasi AES 256-bit dan JWT memberikan mekanisme keamanan berlapis, memastikan integritas dan kerahasiaan data selama proses penyimpanan dan transmisi. Solusi ini efektif untuk aplikasi yang memerlukan keamanan tinggi seperti e-commerce, sektor kesehatan, dan ujian daring. Selain itu, sistem ini menawarkan integrasi yang mudah dengan aplikasi pihak ketiga, memudahkan adopsi tanpa mengorbankan keamanan. Studi ini menyimpulkan bahwa pendekatan terintegrasi ini meningkatkan perlindungan data dan memfasilitasi interaksi API yang aman di berbagai platform.

Abstract. Data security is a critical concern in the development of web-based applications and backend APIs, particularly in protecting sensitive information from unauthorized access. This research aims to integrate Advanced Encryption Standard (AES) 256-bit for data encryption and JSON Web Token (JWT) for user authentication, developed using the Go programming language. The methodology employed is Research and Development (R&D), encompassing stages of requirement analysis, system design, implementation, and testing. The results show that combining AES 256-bit and JWT provides a dual-layer security mechanism, ensuring data integrity and confidentiality during storage and transmission. This solution proves effective for high-security applications such as e-commerce, healthcare, and online examinations. Additionally, the system offers seamless integration with third-party applications, promoting ease of adoption without compromising security. The study concludes that this integrated approach enhances data protection and facilitates secure API interactions across various platforms.

1 PENDAHULUAN

Di era digital yang semakin berkembang, keamanan data menjadi kebutuhan utama dalam pengembangan aplikasi berbasis web dan backend API, terutama dalam melindungi

informasi dari ancaman akses yang tidak sah. Penerapan enkripsi dan autentikasi yang kuat diperlukan untuk menjaga integritas dan kerahasiaan data pengguna, terutama dalam konteks API yang sering digunakan dalam

integrasi layanan. Penelitian ini memfokuskan pada kombinasi enkripsi Advanced Encryption Standard (AES) 256-bit dan autentikasi menggunakan JSON Web Token (JWT) sebagai solusi keamanan data yang terintegrasi dalam backend API menggunakan bahasa pemrograman Golang.

Latar belakang kebutuhan akan keamanan data yang kuat tercermin dari berbagai penelitian yang menyoroti tantangan keamanan data dalam berbagai sektor. Penelitian sebelumnya menunjukkan bahwa algoritma AES secara luas digunakan untuk melindungi data sensitif dalam berbagai aplikasi digital. Salah satunya adalah penelitian mengenai sistem keamanan file menggunakan AES yang membuktikan bahwa algoritma ini mampu menjaga kerahasiaan dan akses data pribadi dari akses yang tidak sah[1]. Sementara itu Tohari, dkk. dalam penelitiannya mengenai API vaksinasi COVID-19 berbasis Golang juga menekankan pentingnya keamanan data, dimana API ini berhasil memberikan akses yang aman dan cepat untuk layanan vaksinasi[2].

Penelitian ini bertujuan untuk mengembangkan pendekatan keamanan data yang optimal berbasis API dengan mengintegrasikan algoritma AES 256-bit untuk enkripsi dan JSON Web Token (JWT) untuk autentikasi pengguna. Penerapan ini dirancang khusus menggunakan bahasa pemrograman Golang, dengan harapan dapat memenuhi kebutuhan keamanan pada berbagai aplikasi digital yang memproses data sensitif. Dalam konteks API yang sering berinteraksi dengan platform lain, pendekatan ini juga menawarkan fleksibilitas tinggi untuk diintegrasikan dengan platform lain, pendekatan ini juga menawarkan fleksibilitas tinggi untuk diintegrasikan ke dalam aplikasi pihak ketiga tanpa memerlukan pengembangan sistem keamanan tambahan, sehingga memudahkan aplikasi lain untuk mengadopsi standar keamanan yang sama. Dengan demikian, penelitian ini diharapkan mampu memberikan perlindungan menyeluruh bagi data yang ditransmisikan, menjaga integritas dan kerahasiaan data, serta memastikan bahwa akses hanya diberikan kepada pengguna yang telah terautentikasi..

Kebaruan dari penelitian ini terletak pada penerapan kombinasi antara AES 256-bit dan JWT dalam konteks aplikasi backend API berbasis Golang. Meski masing-masing metode

telah banyak digunakan secara terpisah, penelitian ini menggabungkan keduanya untuk menciptakan lapisan keamanan ganda. AES 256-bit, sebagai algoritma enkripsi yang diakui secara luas untuk kekuatan dan keandalannya, memastikan data terlindungi selama proses penyimpanan dan transmisi. Di sisi lain, JWT memberikan autentikasi yang efisien dan aman, membatasi akses ke data backend API hanya untuk pengguna yang sudah terverifikasi. Selain memperkuat keamanan, pendekatan ini juga menawarkan kemudahan integrasi dengan aplikasi lain, memungkinkan pengembang aplikasi pihak ketiga untuk mengakses API yang aman tanpa perlu membangun sistem keamanan mereka sendiri. Dengan memanfaatkan kekuatan AES untuk enkripsi dan JWT untuk autentikasi, penelitian ini menghadirkan solusi keamanan yang tidak hanya dapat diandalkan dan efisien, tetapi juga memudahkan penggunaan di berbagai aplikasi modern[3].

2 TINJAUAN PUSTAKA

Restful API merupakan arsitektur untuk penerapan web service dalam menerapkan konsep peralihan antar negara [9]. Negara disini dapat dilustrasikan sebagai peramban yang meminta halaman web, pada sisi server akan mengirimkan keadaan halaman web saat ini ke peramban. Dengan REST API memungkinkan berbagai sistem untuk dapat berkomunikasi dan mengirim atau menerima data dengan cara yang cukup sederhana. Didalam RESTful API terdapat REST client yang dapat mengakses data atau resource pada REST server dimana setiap resource akan dibedakan berdasarkan dari global ID atau URI (Universal Resource Identifiers). Hal ini membuat RESTful API sangat cocok diterapkan pada aplikasi yang terintegrasi dengan ponsel pintar[4].

Di sektor pendidikan, keamanan data ujian online juga menjadi perhatian utama. Utama, dkk. menerapkan AES 256 CBC untuk menjaga integritas dan kerahasiaan data ujian berbasis online, yang efektif dalam mencegah modifikasi dan kebocoran data ujian selama proses penyimpanan dan distribusi[5]. Selain itu, penerapan AES dalam aplikasi berbasis Android untuk voucher hotspot oleh Irawan, dkk. menunjukkan bahwa AES efektif dalam menjaga keamanan transaksi dan data pengguna dari potensi penyalahgunaan[6]. Dalam rekam

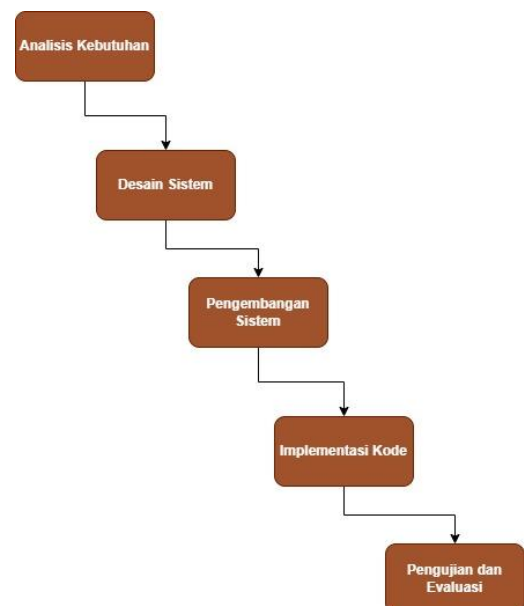
medis di sektor kesehatan juga menunjukkan kebutuhan akan keamanan yang lebih ketat, sebagaimana yang diteliti oleh Khoirunnisa dan Djuniadi, yang menyoroti penggunaan AES untuk melindungi data pasien dari akses yang tidak diinginkan. Dengan penerapan AES, data pasien yang bersifat pribadi tetap terlindungi dan hanya dapat diakses oleh pihak yang berwenang[7].

Sektor *e-commerce* juga menghadapi tantangan keamanan yang sama, terutama dalam melindungi informasi pelanggan. Studi oleh Sodikin dan Hidayat menunjukkan bahwa penerapan AES pada platform *e-commerce* dapat melindungi data pelanggan dari risiko pencurian identitas dan kebocoran data transaksi yang menjadi tantangan besar dalam sistem *e-commerce*[8]. Pentingnya autentikasi yang aman juga ditekankan dalam penelitian Bucko dkk. yang menggunakan JWT berbasis riwayat perilaku pengguna untuk meningkatkan keamanan akses. Hasil penelitian menunjukkan bahwa JWT tidak hanya efektif dalam autentikasi tetapi juga dapat meningkatkan kepercayaan dalam proses otorisasi akses pengguna[9]. Di sisi lain, Gupta dkk. menerapkan JWT bersama dengan Kerberos pada platform Hadoop untuk mengamankan autentikasi dan kontrol akses antar node di lingkungan data besar. Hasilnya menunjukkan bahwa penggunaan JWT sebagai autentikasi berbasis token dapat memberikan keamanan tambahan pada sistem dengan volume data besar[10].

Implementasi AES pada sistem autentikasi login di sistem informasi juga menunjukkan bahwa algoritma ini mampu mengenkripsi data sensitif seperti *password*. Fadlullah dkk. dalam penelitiannya menunjukkan bahwa AES mampu menjaga kerahasiaan *password* pengguna sehingga informasi tersebut dapat terjaga kerahasiaannya dan tidak mudah diakses oleh individu atau pihak yang tidak memiliki izin [11]. Astowo dan Sujarwo menggunakan JWT pada aplikasi multi-masjid untuk melindungi data jamaah dari akses tidak sah. JWT memungkinkan autentikasi yang aman dan efisien pada aplikasi tersebut, menjaga privasi dan keamanan data pengguna dalam konteks layanan yang melibatkan banyak lokasi [12].

3 METODE PENELITIAN

Dalam penelitian ini, diterapkan metode *Research and Development (R&D)* untuk merancang, mengembangkan, dan menguji backend API yang memungkinkan enkripsi dan dekripsi data teks maupun file menggunakan algoritma AES 256-bit serta autentikasi menggunakan JWT (*JSON Web Token*). API ini dikembangkan dengan tujuan agar dapat diakses oleh pengguna terautentikasi melalui JWT dan bisa dipergunakan dalam proses enkripsi dan dekripsi data secara aman. Tahapan utama penelitian ini mencakup analisis kebutuhan, desain sistem, implementasi, serta pengujian API.



Gambar 1. Proses *Research and Development (R&D)*

3.1 Analisis Kebutuhan

Pada tahap ini, dilakukan analisis kebutuhan guna menentukan spesifikasi utama dari sistem backend API, yang meliputi:

a. Proses Registrasi dan Penyimpanan Data Pengguna:

- Pengguna baru harus melalui proses registrasi, dimana data *username* dan *password* mereka akan disimpan secara terenkripsi di *MongoDB*.
- *Username* dan *password* yang disimpan akan dienkripsi menggunakan algoritma AES 256 yang *key* nya berasal dari *password* yang juga dienkripsi menggunakan

algoritma hashing *SHA-256* untuk memastikan keamanan.

b. Autentikasi

3.2 Desain Sistem

Berdasarkan kebutuhan yang telah diidentifikasi, sistem backend *API* dirancang dengan spesifikasi berikut:

- Menggunakan *framework Gin* pada bahasa pemrograman *Golang*.
- Memanfaatkan *MongoDB* sebagai basis data untuk penyimpanan.
- Menerapkan *algoritma AES 256-bit* menggunakan *library aes* di *golang* untuk *enkripsi* data dan *JSON Web Token (JWT)* untuk autentikasi pengguna.

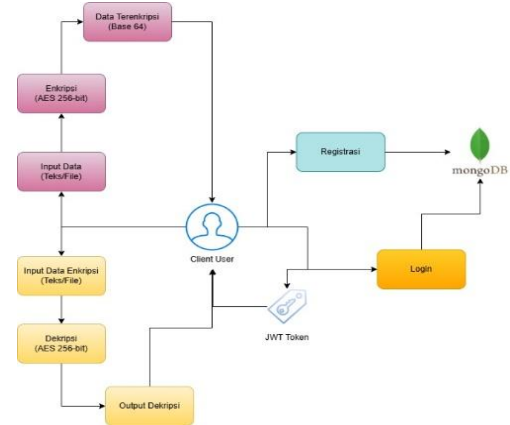
3.3 Pengembangan Sistem

Pengembangan sistem mencakup implementasi fitur-fitur utama berikut:

- Registrasi Pengguna: Menggunakan *endpoint /register* untuk mendaftarkan pengguna baru.
- Login Pengguna: Menggunakan *endpoint /login* untuk autentikasi dan penerbitan token *JWT*.
- Enkripsi dan Dekripsi Data: Menggunakan *endpoint /encrypt* dan */decrypt* yang berfungsi sebagai *enkripsi* dan *dekripsi* teks.
- Enkripsi dan Dekripsi File: Menggunakan *endpoint /encryptFile* dan */decryptFile* untuk mengenkripsi dan mendekripsi file.

3.4 Implementasi Kode

Bagian ini menjelaskan implementasi kode untuk *enkripsi* dan *dekripsi* data menggunakan *algoritma AES 256-bit*, serta penerapan *JSON Web Token (JWT)* untuk autentikasi pengguna. Implementasi dilakukan menggunakan bahasa pemrograman *Golang*.



Gambar 1. Alur proses sistem

Pada gambar 2. merupakan diagram alur proses system yang dibangun Dimana proses enkripsi dan dekripsi menggunakan *library AES 256-bit* bawaan *Golang*.

- User Registrasi** : Saat *client* atau *user* pertama kali akses akan registrasi terlebih dahulu dimana nanti data *username* dan *password* akan disimpan di *mongodb*.
- Login** : Pada saat *user* sudah melakukan registrasi, *user* dapat login menggunakan *username password* yang sudah terdaftar dan nanti *response* akan berbentuk token *JWT* dimana nanti digunakan sebagai *header autentikasi* saat menggunakan *endpoin enkripsi* atau *deskripsi*
- Enkripsi** : *User* melakukan *enkripsi* melalui *endpoint api enkripsi* memasukan teks atau *file* dan nanti akan di proses di system menggunakan *library AES* dan *outputnya* akan di *convert* terlebih dahulu dalam bentuk *base64*
- Dekripsi** : *User* melakukan *dekripsi* melalui *endpoint api dekripsi* memasukan teks atau *file* yang sudah *terenkripsi* sebelumnya nanti dari system akan memproses *dekripsinya* dan memberi *output* ke *user text* atau *file* yang asli
- Kode Enkripsi dan Dekripsi**

Kode berikut merupakan inti untuk *enkripsi* dan *dekripsi* data menggunakan *algoritma AES 256-bit*. Kode ini dirancang untuk menjaga keamanan data selama proses penyimpanan dan transmisi.

a. Fungsi Enkripsi :

```

package aes

import (
    "crypto/aes"
    "crypto/cipher"
    "crypto/rand"
    "encoding/base64"
    "errors"
    "io"
    "golang-api/models"
)

// Fungsi untuk menghasilkan hash dari teks
func GenerateHash(text []byte) []byte {
    hash := sha256.Sum256(text)
    return hash[:]
}

// Fungsi untuk mengenkripsi data dengan AES 256-bit
func Encrypt(plainText []byte, password string) (string, error) {
    plainText = append(plainText, []byte(models.ValidKeyword)...)
    key := GenerateHash([]byte(password))
    block, err := aes.NewCipher(key)
    if err != nil {
        return "", err
    }

    cipherText := make([]byte, aes.BlockSize+len(plainText))
    iv := cipherText[:aes.BlockSize]
    if _, err := io.ReadFull(rand.Reader, iv); err != nil {
        return "", err
    }

    stream := cipher.NewCFBEncrypter(block, iv)
    stream.XORKeyStream(cipherText[aes.BlockSize:], plainText)

    return base64.StdEncoding.EncodeToString(cipherText), nil
}

```

b. Fungsi Dekripsi :

```

package aes

import (
    "crypto/aes"
    "crypto/cipher"
    "encoding/base64"
    "errors"
)

// Fungsi untuk mendekripsi data yang terenkripsi
func Decrypt(encryptedText string, password string) ([]byte, error) {
    key := GenerateHash([]byte(password))
    cipherText, err := base64.StdEncoding.DecodeString(encryptedText)
    if err != nil {
        return nil, err
    }

    block, err := aes.NewCipher(key)
    if err != nil {
        return nil, err
    }

    if len(cipherText) < aes.BlockSize {
        return nil, errors.New("ciphertext too short")
    }

    iv := cipherText[:aes.BlockSize]

```

```

cipherText = cipherText[aes.BlockSize:]

stream := cipher.NewCFBDecrypter(block, iv)
stream.XORKeyStream(cipherText, cipherText)

if string(cipherText[len(cipherText)-len(models.ValidKeyword):]) != models.ValidKeyword {
    return nil, errors.New("invalid password or data")
}

return cipherText[:len(cipherText)-len(models.ValidKeyword)], nil
}

```

c. *JWT Token untuk Autentikasi :*

```

package auth

import (
    "crypto/rsa"
    "crypto/x509"
    "encoding/pem"
    "errors"
    "os"
    "time"

    "golang-api/models"
    "github.com/dgrijalva/jwt-go"
)

// Fungsi untuk menghasilkan token JWT
func GenerateToken(username string) (string, error) {
    claims := models.TokenClaims{
        Username: username,
        StandardClaims: jwt.StandardClaims{
            ExpiresAt: time.Now().Add(1 * time.Hour).Unix(),
        },
    },
}

token := jwt.NewWithClaims(jwt.SigningMethodRS256, claims)
return token.SignedString(privateKey)
}

```

3.5 Pengujian dan Evaluasi

Pengujian sistem dilakukan menggunakan *Postman* untuk memastikan bahwa semua *endpoint* berfungsi dengan baik dan sesuai dengan *spesifikasi*. Pengujian mencakup *autentikasi* menggunakan *JSON Web Token (JWT)* dan *enkripsi* serta *dekripsi* data menggunakan *algoritma AES 256-bit*.

4 HASIL DAN PEMBAHASAN

4.1 Hasil Pengujian Sistem

Pada bagian ini, akan disajikan pengujian yang dilakukan untuk memastikan sistem backend *API* berfungsi sebagaimana dengan tujuan yang diharapkan. Pengujian ini akan dilakukan menggunakan *software Postman*, dengan fokus pada validasi setiap *endpoint* dan kesesuaian fungsionalitasnya.

a. Pengujian Endpoint Registrasi Pengguna

- Endpoint : `/register`
- Metode : `POST`

Data yang dikirim:

```

{
  "username": "user1",
  "password": "password123"
}

```

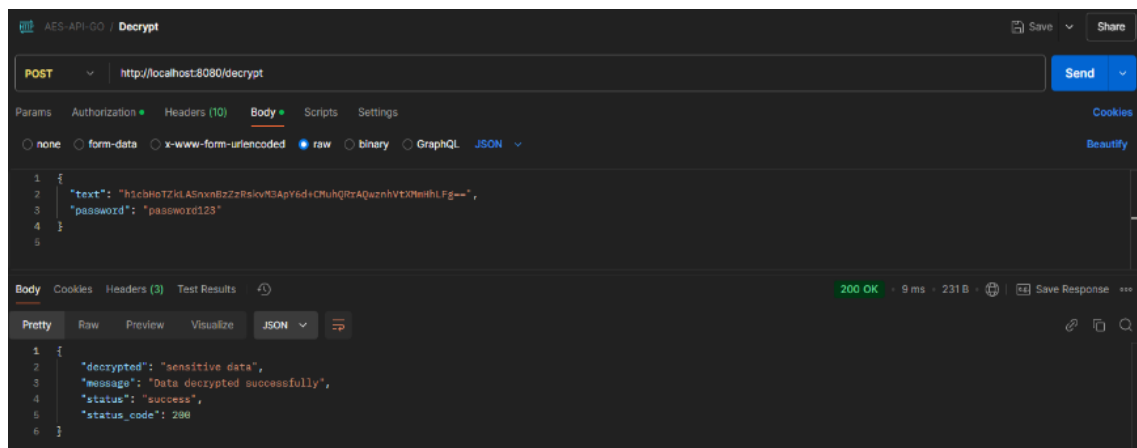
Hasil Pengujian:

- Status Kode : 201 (Created)
- Respons :

```

{
  "message": "User
  Succesfully registered",
  "status": "success",
  "status_code": 201
}

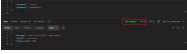
```

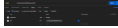
Gambar 4. Hasil Test Endpoint Deskripsi Data

Pengujian ini dilakukan untuk memastikan sitem dapat menagnai berbagai sekenario,sperti penggunaan kunci yang berbeda,teks yang berbeda, dan kecepatan waktu proses nya.Berikut adalah hasil pengujian yang dirangkum dalam table:

TABEL 1
PENGUJIAN SISTEM

No	Sekenario Pengujian	Data yang Dikirim	Hasil yang Diharapkan	Hasil Pengujian	Response Time
1	Registrasi pengguna baru	{ "username": "user1", "password": "password1" }	Penguna berhasil terdaftar dan data terenkripsi	Berhasil,pengguna berhasil terdaftar dan response nya cepat 	44 ms
2	Login dengan pengguna yang valid	{ "userame": "user1", "passw ord": "password1" }	Pengguna berhasil login dan mendapat token	Berhasil,login berhasil,token JWT diterimas 	54 ms
3	Enkripsi dengan token yang tidak sesuai		Pengguna gagal untuk melakukan enkripsi	Gagal,penguna gagal karena tokenya tidak valid	6 ms

4	Enkripsi dengan token yang valid		Pengguna berhasil melakukan enkripsi	Berhasil, pengguna berhasil enkripsi data	8 ms
5	Dekripsi dengan input password yang tidak sesuai	{ "text": "h1cbHo TZkLAS nxnBzZz RskvM3 ApY6d+ CMuhQ RrAQwz nhVtXM mHhLFg ==", "password": "password salah" }	Pengguna gagal untuk dekripsi	Gagal, pengguna gagal dekripsi data karena passwordnya tidak valid	7 ms
6	Dekripsi dengan data enkripsi yang tidak sesuai	{ "text": "h1cbHo TZkLAS nxnBzZz RskvM3s fswrwrw rsfsfsfAp Y6d+=", "password": "password123" }	Pengguna gagal untuk dekripsi	Gagal, pengguna gagal dekripsi data karena text tidak valid	5 ms
7	Enkripsi file		Pengguna berhasil enkripsi file dan file hasil enkripsi dapat disimpan namun isinya sudah terenkripsi	Berhasil, pengguna berhasil enkripsi file dan menyimpan file hasilnya	134 ms

8	Dekripsi file		Pengguna berhasil dekripsi file dan hasilnya sesuai file aslinya	Berhasil, pengguna berhasil dekripsi file dan hasil filenya sesuai	125 ms
---	---------------	---	--	--	--------



4.2 Pembahasan

Berdasarkan hasil pengujian, dapat disimpulkan bahwa system yang dikembangkan mampu mengenkripsi dan mendekripsi data menggunakan algoritma AES 256-bit secara efektif. Selain itu, implementasi JSON Web Token (JWT) untuk autentikasi berfungsi dengan baik, sesuai dengan yang diharapkan

- Keamanan: Algoritma AES 256-bit memberikan perlindungan yang kuat untuk data sensitif, mencegah akses oleh pihak yang tidak berwenang.
- Autentikasi: JWT Token terbukti efektif dalam memastikan akses hanya diberikan kepada pengguna yang telah terverifikasi, menjaga keamanan endpoint API.
- Kinerja: Pengujian menunjukkan bahwa sistem dapat menangani permintaan API dengan baik, termasuk proses enkripsi dan dekripsi data, tanpa mengorbankan performa.

5 KESIMPULAN

- Pengembangan Sistem *Backend API*: Penelitian ini berhasil mengembangkan sistem backend API yang menggabungkan dua teknologi utama untuk memberikan perlindungan data, yaitu algoritma AES 256-bit untuk enkripsi data dan JSON Web Token (JWT) untuk autentikasi. Dengan menggunakan kombinasi ini, sistem dapat mengamankan data sensitif dan memastikan hanya pengguna yang terverifikasi yang memiliki akses terhadap informasi yang dilindungi.
- Keamanan dan Autentikasi Data: Hasil pengujian sistem menunjukkan bahwa proses enkripsi dan dekripsi data berjalan dengan aman, sementara autentikasi

menggunakan JWT memastikan bahwa hanya pengguna yang terverifikasi yang dapat mengakses data sensitif. Hal ini menjadikan sistem lebih aman dalam mengelola dan melindungi informasi yang bersifat sensitif.

- Kinerja dan Efektivitas Sistem: Sistem ini juga menunjukkan kinerja yang optimal dalam menangani permintaan API, dengan response API yang cukup cepat selama pengujian. Kecepatan dan efisiensi ini menjadikannya solusi yang efektif untuk aplikasi yang membutuhkan tingkat keamanan tinggi dan respons cepat dalam pengolahan data.
- Saran untuk Penelitian Selanjutnya: Penelitian selanjutnya dapat difokuskan pada pengujian skalabilitas sistem untuk mengukur kemampuannya dalam menangani beban yang lebih besar. Selain itu, integrasi dengan platform berbasis cloud dapat menjadi fokus utama untuk meningkatkan efisiensi dan fleksibilitas penerapan sistem, memungkinkan sistem beradaptasi dengan lebih baik pada berbagai kebutuhan dan lingkungan operasional.

6 UCAPAN TERIMA KASIH

Puji syukur dipanjatkan kepada Tuhan Yang Maha Esa, yang telah memberikan rahmat dan karunia-Nya, sehingga dapat diselesaikannya jurnal skripsi yang berjudul "*Backend API Data Protection Menggunakan JWT Token dan Algoritma AES 256-bit Dengan Bahasa Pemrograman Golang*".
Diucapkan terima kasih yang sebesar-besarnya kepada:

- a. **Bapak/Ibu Dosen Pembimbing** yang telah memberikan bimbingan, masukan, serta motivasi selama proses pembuatan jurnal ini.
- b. **Keluarga tercinta**, yang selalu senantiasa mendoakan, memberi dukungan, dan semangat tanpa henti.
- c. **Teman-teman dan rekan seperjuangan di Universitas Mercu Buana Yogyakarta (UMBY)**, yang telah memberikan dukungan moral, ide, dan diskusi yang bermanfaat selama proses pengerjaan skripsi ini.

Semua pihak yang tidak dapat disebutkan satu per satu, yang telah memberikan kontribusi berharga dalam berbagai bentuk, sehingga jurnal ini mampu diselesaikan dengan baik

7 DAFTAR PUSTAKA

- [1] “[138-148]+Implementasi+Sistem+Keamanan+File+Menggunakan+Algoritma+AES+untuk+Mengamankan+File+Pribadi”.
- [2] D. M. F. Thohari and S. Damerianta, “PERANCANGAN APPLICATION PROGRAMMING INTERFACE (API) UNTUK MENGAKSES LAYANAN VAKSINASI COVID-19 MENGGUNAKAN GOLANG ECHO FRAMEWORK” *Jurnal Ilmiah Informatika Komputer*, vol. 28 no 3, 2023, doi: 10.35760/ik.2023.v28i3.9423.
- [3] “Application of RESTful Method with JWT Security and Haversine Algorithm on Web Service-Based Teacher Attendance System”.
- [4] W. Hadinata dan L. Stianingsih, “ANALISIS PERBANDINGAN PERFORMA RESTFULL API ANTARA EXPRESS.JS DENGAN LARAVEL FRAMEWORK” *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 1, Art. no. 1, Jan 2024, doi: 10.23960/jitet.v12i1.3845.
- [5] G. W. F. Putra Utama R. Faurina, A. Vatesia, U. Bengkulu, and P. Korespondensi, “IMPLEMENTASI ALGORITMA AES 256 CBC, BASE 64, DAN SHA 256 DALAM”, doi: 10.25126/jtiik.2023106558.
- [6] “20817-Article Text-47419-2-10-20240724”.
- [7] O. G. Khoirunnisa and D. Djuniadi, “Implementasi Algoritma AES untuk Keamanan Data Rekam Medis” *PETIR*, vol. 15 no 1, 2021, doi: 10.33322/petir.v15i1.1333.
- [8] T. H. L. Sodikin, “ANALISA KEAMANAN E-COMMERCE MENGGUNAKAN METODE AES ALGORITMA” vol. 3. no 2, 2020.
- [9] K. V. A. Bucko B. Krasniqi, and B. Rexha, “Enhancing JWT Authentication and Authorization in Web Applications Based on User Behavior History” *computers*, vol. 12 no, 4, 2023, doi: : 10.3390/computers12040078.
- [10] M. Gupta, A. Gupta, B. Raj S., dan A. Sharma, “JWTAMH: JSON Web Tokens Based Authentication Mechanism for HADOOP” *ICST Transactions on Scalable Information Systems*, vol. 11, 2024, doi: 10.4108/eetsis.5429.
- [11] F. Fadlullah, M. Tahir, B. P. Bintari, dan M. L. Dewi, and M. F. Ilmy, “Implementasi Algoritma AES pada Autentikasi Login Sistem Informasi” 2023.
- [12] U. Budi dan A. 1✉, and A. Sujarwo, “Penerapan JSON Web Token sebagai Strategi Pengamanan Data pada Aplikasi MultiMasjid”.