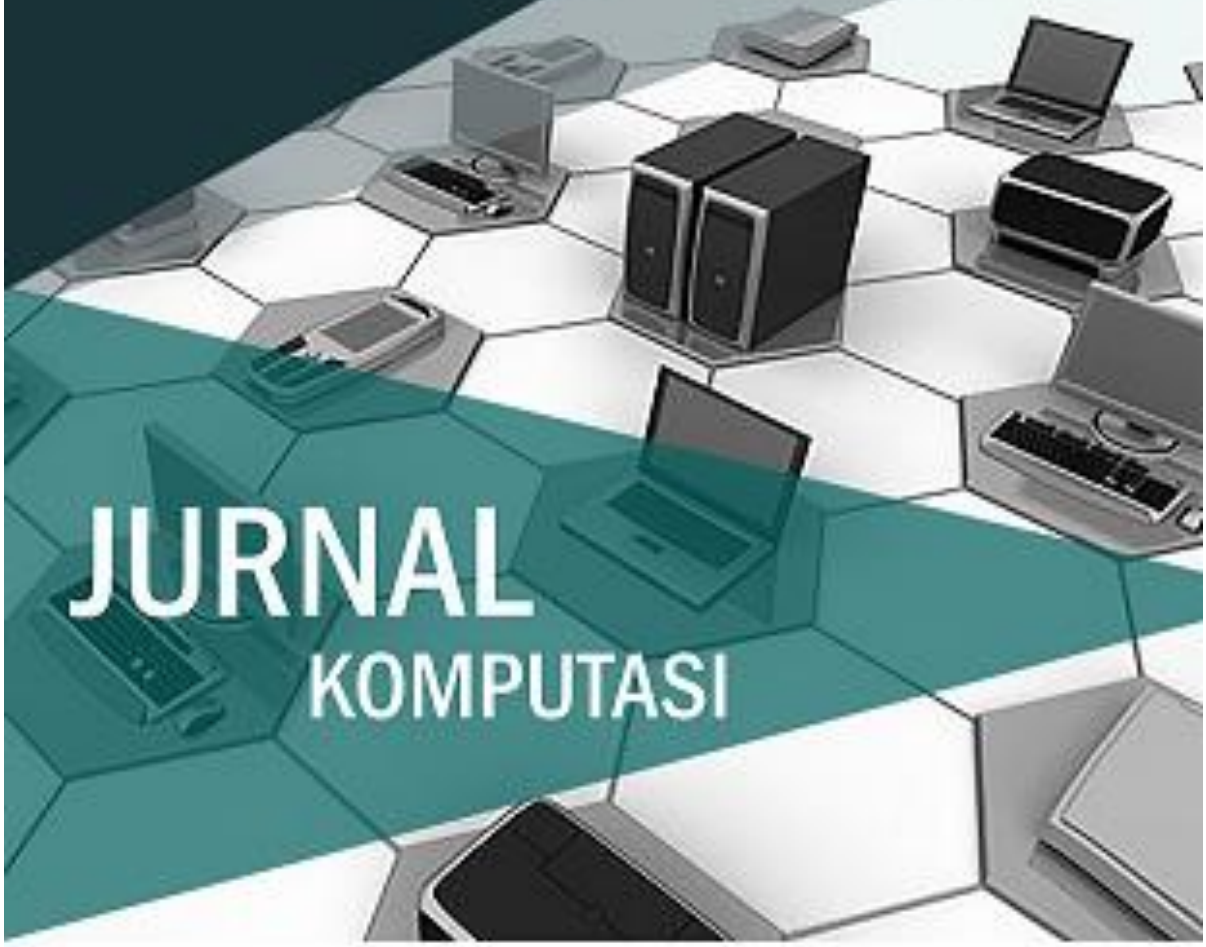
P-ISSN 2541-0296

Volume 11 No 1

2023



JURNAL KOMPUTASI

Ilmu Komputer
Universitas Lampung
Bandar Lampung, April 2023

EDITORIAL TEAM

EDITOR-IN-CHIEF

MANAGING EDITOR

EDITORIAL BOARD

ADDITIONAL MENU
CONTACT JOURNAL
EDTORIAL BOARD
REVIEWER BOARD
SUBMISSION GUIDE
PUBLICATION ETHICS
GOOGLE SCHOLAR INDEXING
GARUDA INDEXING
PLAGIARISM CHECK
AUTHOR GUIDELINES
FOCUS AND SCOPE
ADDITIONAL MENU
USER

Username
Password
Remember me
Login

Recommended Tools

Indexed by

VOL 11, NO 1 (2023)

JURNAL KOMPUTASI

DOI: http://dx.doi.org/10.23960%2Fkomputasi.v11i1

TABLE OF CONTENTS

ARTICLES

Jurnal Komputasi

Vol 11 No. 1, 2023

Peningkatkan Keamanan Transmisi Data *REST API* Menggunakan Enkripsi SHA-512

[1,*)]M. Iqbal Parabi, [2]Didik Kurniawan & [3]Rizky Prabowo

[1,2,3] Jurusan Ilmu Komputer, Universitas Lampung, Jalan Prof. Dr. Sumantri Brojonegoro Nomor 1, Bandar Lampung, Indonesia

Abstrak — Perkembangan teknologi informasi terus meningkat seiring dengan kebutuhan organisasi dalam mengelola proses bisnis menggunakan berbagai sistem yang saling terhubung. Pada proses transmisi data antar sistem, dibutuhkan keamanan tingkat tinggi yang mampu menjamin integritas data dalam hal keutuhan dan konsistensi data, khususnya sistem keuangan yang memiliki data sensitif. *Exposure* data sensitif melalui *Application Programming Interface* (API) merupakan salah satu kerentanan pada keamanan sistem berbasis web. *Man in the Middle Attack* (MitMA) merupakan jenis serangan yang sering terjadi ketika transmisi data menggunakan API. Penelitian ini bertujuan untuk meningkatkan keamanan transmisi data pada *Representational State Transfer* (REST) API menggunakan enkripsi *Secure Hash Algorithm* (SHA)-512. Hasil penelitian menunjukkan bahwa penerapan enkripsi SHA-512 mampu meningkatkan keamanan transmisi data dari *client* ke *server* maupun sebaliknya. Pada metode POST, data *request* yang terenkripsi memiliki performa lebih baik 2,8% dibandingkan dengan data *request* yang tidak terenkripsi. Sedangkan pada metode GET, data *request* yang terenkripsi lebih lambat 12,5% dibandingkan dengan data *request* yang tidak terenkripsi. Hal ini dikarenakan adanya proses enkripsi dan dekripsi yang dilakukan *server* untuk mengelola *response* sehingga membutuhkan waktu lebih lama.

Kata Kunci: api, enkripsi; rest; sha-512

Abstract *— The development of information technology continues to increase along with the need for organizations to manage business processes using various systems. The process of transmitting data between systems requires a high level of security that can guarantee data integrity, especially in financial systems that have sensitive data. Exposure of sensitive data through Application Programming Interface (API) is one of the vulnerabilities in web-based system security. Man in the Middle Attack (MitMA) is a type of attack that often occurs when transmitting data using API. This research aims to improve the security of data transmission on the Representational State Transfer (REST) API using Secure Hash Algorithm (SHA)-512 encryption. The results showed that implementing of SHA-512 encryption algorithm was able to improve the security of data transmission from client to server. In the POST method, encrypted request data has a better performance of 2.8% compared to unencrypted request data. Whereas in the GET method, encrypted request data is 12.5% slower than unencrypted data. This is due to the encryption and decryption process carried out by the server to manage the response, so it takes longer.*

Keywords: api; encryption; rest; sha-512

** Corresponding author :*
M. Iqbal Parabi
Universitas Lampung, Bandar Lampung, Indonesia
iqbal.parabi@fmipa.unila.ac.id

1. PENDAHULUAN

Perkembangan teknologi terus meningkat seiring dengan kebutuhan organisasi dalam mengelola proses bisnis menggunakan berbagai sistem yang saling terhubung. Pada proses pertukaran data antar sistem, dibutuhkan keamanan tingkat tinggi yang mampu menjamin integritas data dalam hal keutuhan dan konsistensi data, terlebih sistem keuangan yang memiliki data sensitif. Teknologi pertukaran data yang saat ini banyak digunakan adalah *Application Programming Interface* (API). API mampu mengintegrasikan sistem tanpa melihat *platform*, bahasa pemrograman, dan arsitektur sistem. Menurut

Open Web Application Security Project (OWASP), *exposure* data sensitif melalui API merupakan salah satu dari sepuluh kerentanan pada keamanan sistem berbasis web. Kerentanan pada *exposure* data sensitif berkaitan dengan kegagalan kriptografi dalam mengelola pertukaran data [1].

Arsitektur API yang saat ini banyak digunakan adalah *Representational State Transfer* (REST) [2] dan *Simple Object Access Protocol* (SOAP) [3]. Keduanya merupakan pendekatan yang dapat digunakan untuk melakukan pertukaran data antar sistem. Berdasarkan ukuran pesan, REST lebih kecil dibandingkan SOAP [4] sehingga lebih efektif dalam melakukan transmisi data, namun dari segi keamanan, REST tidak lebih baik jika dibandingkan SOAP [5]. Untuk menjaga keamanan data, REST API menggunakan *JSON Web Token* (JWT) yang memanfaatkan *token* berbentuk *string* JSON untuk melakukan autentikasi selama pertukaran data berlangsung [6]. JWT dengan metode *Hash-Based Message Authentication Code* (HMAC) algoritme *Secure Hash Algorithm* (SHA)-512 memiliki kinerja 50% lebih baik dibandingkan SHA-256 [7]-[8].

JWT dapat dikirim melalui URL, parameter HTTP POST, atau disematkan pada *header* HTTP. Proses pertukaran data menggunakan JWT dari *client* ke *server* atau sebaliknya dapat menimbulkan celah keamanan tersendiri karena data *request* atau *response* yang dikirim berbentuk data serial yang tidak terenkripsi. Jika *client* atau *server* tidak menggunakan HTTPS, maka dapat dengan mudah dilakukan *Man in the Middle Attack* (MitMA) [9] yang memungkinkan penyusup menyadap data sensitif yang ditransmisikan dari *client* ke *server* maupun sebaliknya. Berdasarkan hal tersebut, penelitian ini bertujuan untuk meningkatkan keamanan data pada REST API menggunakan JWT SHA-512 dengan melakukan enkripsi pada data *request* atau *response* yang ditransmisikan dari *client* ke *server* atau sebaliknya.

2. METODOLOGI PENELITIAN

Metodologi yang digunakan pada penelitian ini adalah sebagai berikut.

2.1 Studi Literatur

Tahapan studi literatur dilakukan untuk mengumpulkan informasi yang dibutuhkan pada penelitian seperti mencari artikel ilmiah dan jurnal penelitian terdahulu yang berkaitan dengan penelitian yang sedang dilakukan. Tahap studi literatur dilakukan agar penelitian lebih fokus dan mencegah terjadi kesamaan dengan penelitian terdahulu.

2.2 Analisis Kebutuhan Perangkat Lunak

Tahapan analisis kebutuhan perangkat lunak dilakukan untuk mendapatkan kebutuhan perangkat lunak yang akan dibangun dalam penelitian ini. Hasil analisis kebutuhan perangkat lunak dalam penelitian ini meliputi software dan hardware untuk mendukung kegiatan penelitian ini.

2.3 Implementasi Perangkat Lunak

Tahapan implementasi perangkat lunak dilakukan dengan mengembangkan aplikasi berbasis web yang memiliki REST API menggunakan JWT SHA-512. Selanjutnya, dikembangkan *library* atau pustaka enkripsi menggunakan algoritme SHA-512 yang ditanamkan pada aplikasi *client* dan *server* untuk melakukan enkripsi pada data *request* dan melakukan dekripsi pada data *response* yang ditransmisikan *client* dan *server*.

2.4 Pengujian dan Perbandingan

Tahapan pengujian perangkat lunak dilakukan dengan cara menguji integritas data yang dikirim dari *client* ke *server* ataupun sebaliknya. Pengujian dilakukan dengan memberikan data *string* dan *array* untuk dienkripsi dan didekripsi guna mengetahui keutuhan dan konsistensi data. Pengujian selanjutnya, dilakukan perbandingan kinerja dengan cara mengukur waktu *server* dalam memberikan *response* dari *request* pada data serial tidak terenkripsi dan data serial yang terenkripsi.

3. HASIL DAN PEMBAHASAN

3.1 Studi Literatur

Penelitian terkait implementasi dan optimisasi alogitme *hash* kriptografi telah dilakukan oleh beberapa peneliti. Serme et al. (2012) memperkenalkan teknik optimisasi untuk implementasi SHA-512 pada prosesor x86. Teknik yang digunakan termasuk instruksi SSE2 dan AVX serta teknik *unrolling loop*. Hasil pengujian menunjukkan bahwa implementasi SHA-512 yang dioptimisasi mampu meningkatkan *throughput* hingga 3 kali lipat dibandingkan implementasi standar [10].

Sumaghita dan Riadi (2018) juga mengusulkan beberapa teknik optimisasi untuk algoritme hash kriptografi. Penelitian ini memfokuskan pada penggunaan teknik *pipelining* dan *loop unrolling* pada alogitme SHA-512. Hasil pengujian menunjukkan bahwa teknik *pipelining* dan *loop unrolling* mampu meningkatkan *throughput* hingga 4 kali lipat dibandingkan dengan implementasi standar [11].

Grembowski et al. (2007) mengusulkan implementasi SHA-512 yang dioptimisasi untuk platform FPGA. Mereka menggunakan teknik *pipelining* dan optimisasi sumber daya dalam perancangan implementasi FPGA. Hasil pengujian menunjukkan bahwa implementasi SHA-512 pada FPGA yang dioptimisasi dapat mencapai *throughput* hingga 320 Mbps [12].

Bhonge et al. (2020) memperkenalkan pendekatan baru untuk mengatasi kerentanan pada alogitme *hash* kriptografi SHA-512. Pendekatan ini melibatkan penggunaan SHA-512 dengan mode operasi yang berbeda dan evaluasi masing-masing mode dalam hal waktu yang dibutuhkan untuk komputasi *hash*. Hasil pengujian menunjukkan bahwa pendekatan ini berhasil meningkatkan ketahanan *pre-image* dan *collision resistance* pada alogitme SHA-512 [13].

3.2 Analisis Kebutuhan Perangkat Lunak

Berdasarkan literatur yang diperoleh, kebutuhan perangkat lunak yang digunakan pada penelitian ini adalah sebagai berikut.

a. *Software*

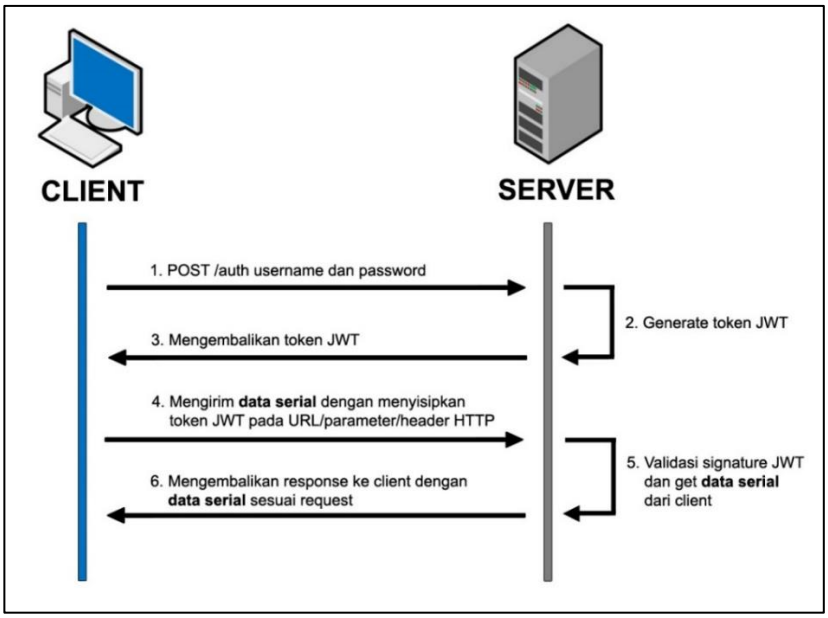
- *Web server*: *web server* untuk menjalankan REST API yang akan diuji dan dianalisis dalam penelitian ini adalah Apache.
- Bahasa pemrograman: bahasa pemrograman untuk mengembangkan aplikasi *web* yang digunakan adalah PHP.
- *Library* Enkripsi SHA-512
- *Tool* analisis: *tool* untuk melakukan analisis keamanan pada REST API yang telah diimplementasikan dengan enkripsi SHA-512 yang digunakan adalah Postman.

b. *Hardware*

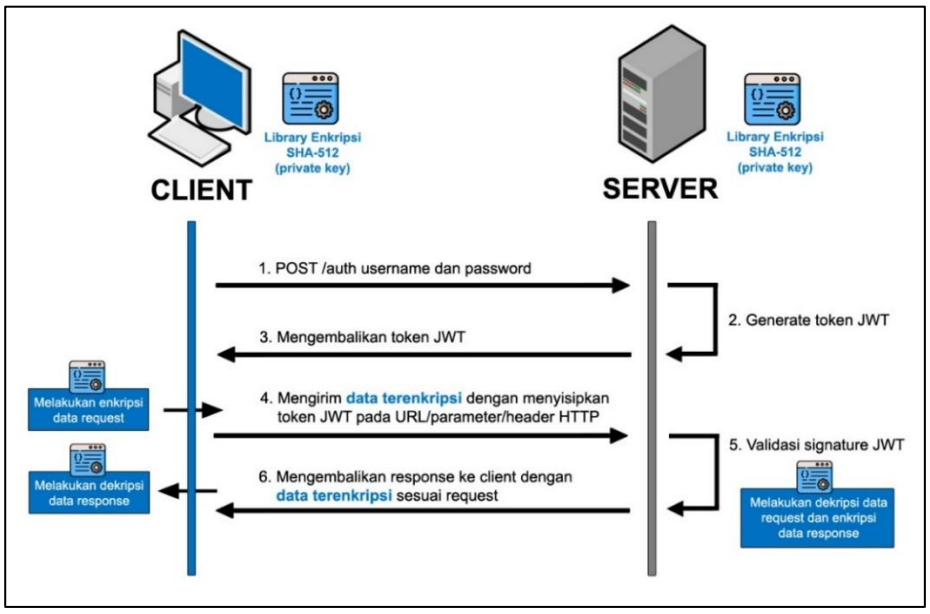
- Komputer: komputer atau laptop untuk mengembangkan aplikasi *web* dan menjalankan *tool* analisis keamanan.
- *Server*: untuk menjalankan aplikasi *web* yang akan diuji dan dianalisis pada penelitian ini.

3.3 Implementasi Perangkat Lunak

Untuk meningkatkan keamanan data pada REST API menggunakan JWT, dikembangkan pustaka enkripsi dengan bahasa pemrograman PHP algoritme SHA-512. Pustaka enkripsi ini digunakan untuk melakukan enkripsi data *request* yang dikirim *client* ke *server* maupun sebaliknya. Mekanisme REST API menggunakan JWT dapat dilihat pada gambar 1. Sedangkan mekanisme REST API menggunakan JWT dengan data terenkripsi dapat dilihat pada gambar 2.



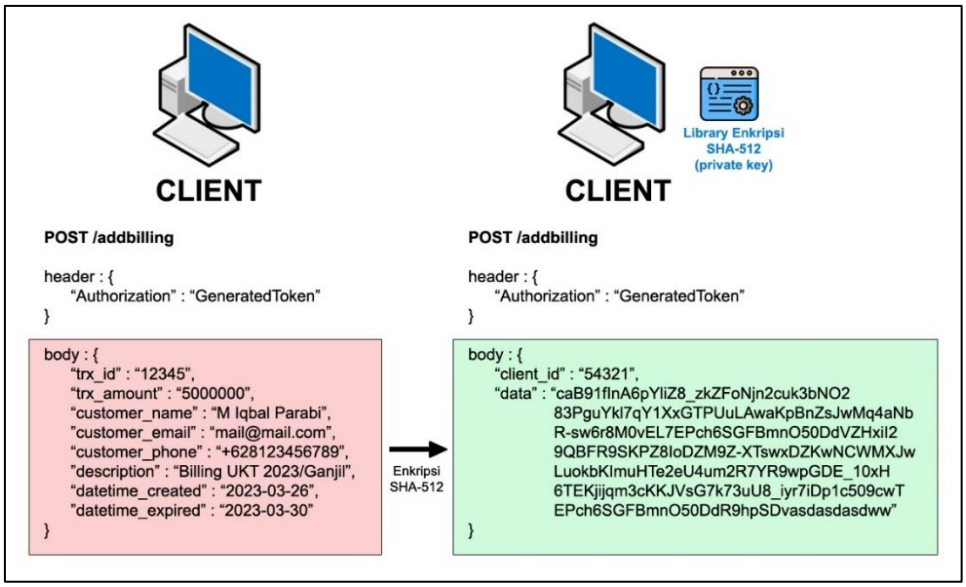
Gambar 1. Mekanisme REST API menggunakan JWT



Gambar 2. Mekanisme REST API menggunakan JWT dengan data terenkripsi

Gambar 1 menjelaskan mekanisme REST API menggunakan JWT yang dimulai dari proses POST /auth dengan mengirim *username* dan *password* dari *client* ke *server*. Kemudian *Server* memeriksa kesesuaian akun dan melakukan *generate token* JWT dan mengembalikan *token* kepada *client*. *Token* digunakan *client* untuk autentikasi setiap melakukan transmisi data yang dapat disisipkan pada URL/parameter POST/header HTTP.

JWT konvensional melakukan *request* dalam bentuk data serial yang tidak terenkripsi, hal tersebut menimbulkan celah keamanan MitMA jika *client* atau *server* tidak menggunakan HTTPS. Penelitian ini mengajukan mekanisme baru yang dijelaskan pada gambar 2 dimana sebelum *client* melakukan *request*, dilakukan enkripsi terlebih dahulu pada data serial yang akan dikirim, begitu juga pada sisi *server* dimana terdapat proses dekripsi data *request* dan enkripsi data *response*. Perbedaan mekanisme REST API menggunakan JWT dengan data serial tidak terenkripsi dan data serial terenkripsi dapat dilihat pada gambar 3.



Gambar 3. Perbedaan data serial tidak dienkripsi dan data serial terenkripsi

Gambar 3 merupakan ilustrasi proses *request* POST /addbilling yang dilakukan *client* ke *server*. Data *request* pada REST API menggunakan JWT berbentuk data serial tidak terenkripsi (kotak merah) dengan parameter: *trx_id, trx_amount, customer_name, customer_email, customer_phone, description, datetime_created,* dan *datetime_expired*. Sedangkan pada REST API menggunakan JWT dengan data serial terenkripsi (kotak hijau) memiliki parameter: *client*_id dan *data*. Kedua *request* memiliki nilai yang sama meskipun parameter berbeda.

Pada data serial terenkripsi, terdapat parameter *client_id* yang merupakan id unik yang digunakan *server* untuk mencari *private key* di dalam *database*. Data serial terenkripsi memiliki tingkat keamanan lebih tinggi dibandingkan data serial tidak terenkripsi. Jika terjadi serangan MitMA dan berhasil mendapatkan data *request*, penyusup tidak dapat dengan mudah langsung mengambil data sensitif seperti *customer_phone* dan *customer_email*. Dibutuhkan lebih banyak usaha dan waktu yang lama untuk mengekstrak data sensitif yang diinginkan.

3.4 Pengujian dan Perbandingan

Pengujian dibagi ke dalam 2 bagian, pengujian pertama memastikan integritas data serial yang dienkrip menggunakan pustaka enkripsi yang telah dikembangkan, sedangkan pengujian kedua mengukur kinerja waktu menggunakan perangkat lunak API Postman.

3.4.1 Pengujian Integritas Data

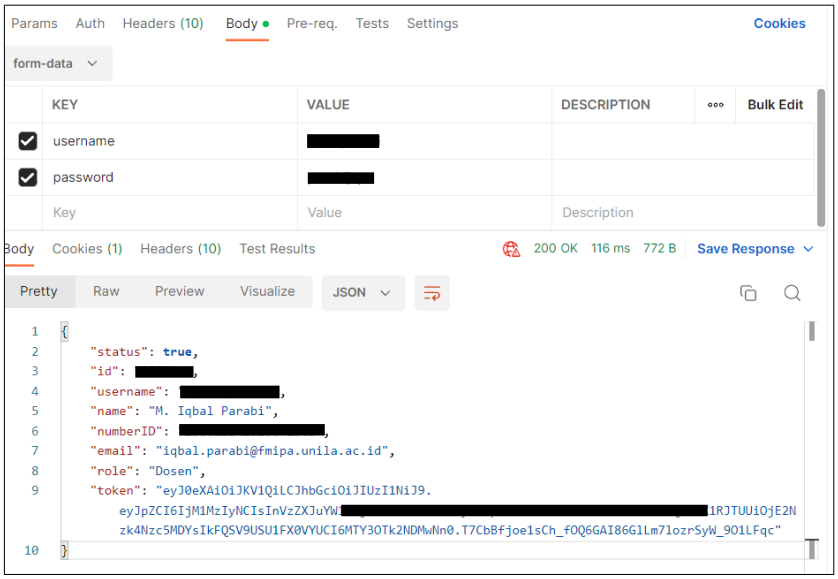
Pengujian integritas data dilakukan untuk mengetahui keutuhan dan konsistensi data selama proses enkripsi dan dekripsi. Pengujian dilakukan dengan cara memberikan masukan berupa data *string* dan *array* yang kemudian dilakukan proses enkripsi dan dekripsi menggunakan *library* yang telah dikembangkan. Pengujian dilakukan menggunkan metode *black box testing* dengan hasil pada Tabel 1.

Tabel 1. Pengujian *black box* integritas data

Masukan	Hasil yang diharapkan	Hasil Pengujian	Kesimpulan
String "Keamanan data Unila"	*String* yang dimasukkan terenkripsi	*String* berhasil terenkripsi	Ok
String yang terenkripsi	*String* yang dimasukkan didekripsi	*String* berhasil didekripsi "Keamanan data Unila"	Ok
Array: {"id":"12345", "pin":"543210"}	*Array* yang dimasukkan terenkripsi	*Array* berhasil terenkripsi	Ok
Array yang terenkripsi	*Array* yang dimasukkan didekripsi	*Array* berhasil didekripsi *{"id":"12345", "pin":"543210"}*	Ok

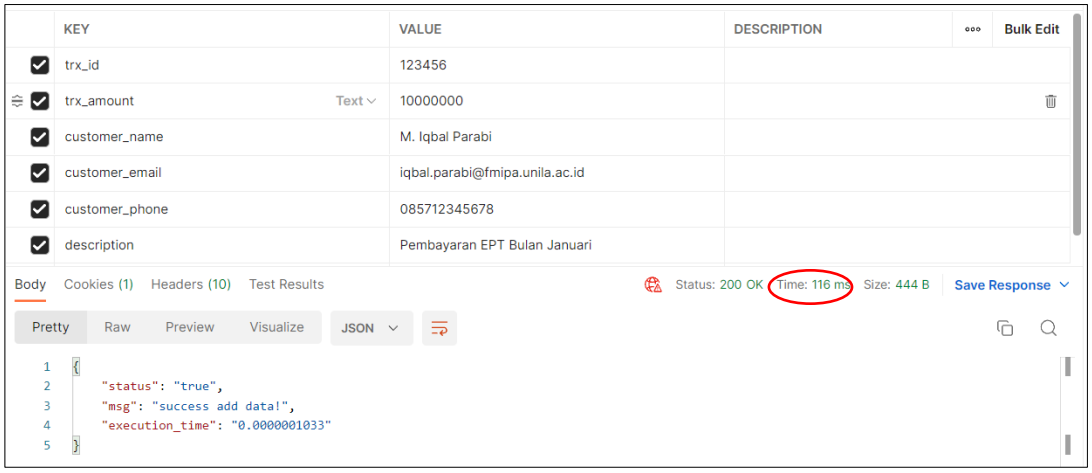
3.4.2 Pengukuran dan Perbandingan Kinerja

Pengukuran kinerja dilakukan dengan cara mengukur waktu transmisi data yang dimulai dari *client* mengirim *request* metode POST atau GET ke *server* REST API sampai dengan *client* menerima *response* dari *server*. Perbandingan kinerja dilakukan pada data serial tidak terenskripsi dan data serial terenkripsi yang dikirim oleh *client*. Perangkat lunak pendukung yang digunakan untuk pengujian REST API adalah Postman. Sebelum melakukan pengukuran kinerja, langkah pertama yang dilakukan adalah autentikasi *user* untuk mendapatkan *token* JWT dengan cara *request* metode POST menggunakan parameter *username* dan *password* yang dapat dilihat pada Gambar 4.



Gambar 4. Proses autentikasi REST API menggunakan JWT

Langkah selanjutnya dilakukan *request* dengan metode POST dan GET pada data serial tidak terenkripsi dan data serial terenkripsi. Pengujian dilakukan sebanyak 10 kali pada masing-masing metode POST dan GET. Untuk mengukur waktu transmisi data, perhitungan dimulai dari *client* mengirim *request* ke *server* sampai *client* menerima *response* dari *server*. Pengujian dilakukan menggunakan perangkat lunak Postman yang dapat dilihat pada Gambar 5 dan Gambar 6.



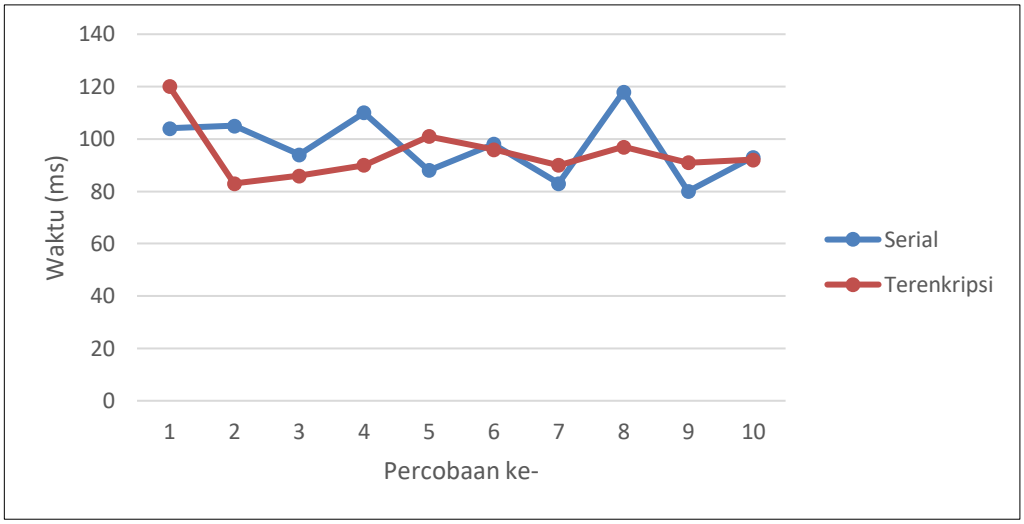
Gambar 5. Pengujian menggunakan Postman pada data serial

Gambar 6. Pengujian menggunakan Postman pada data terenkripsi

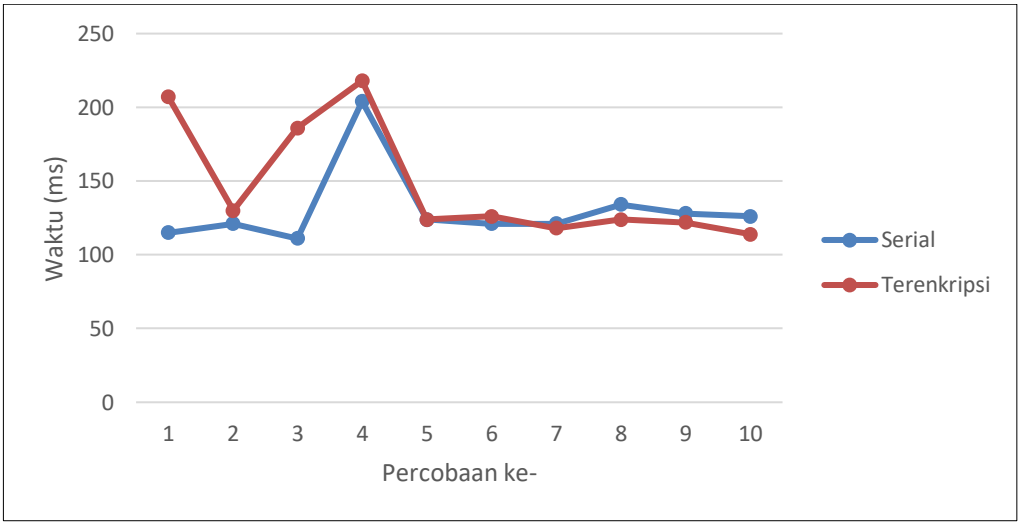
Berdasarkan pengujian pada gambar 5 dan gambar 6 yang dilakukan sebanyak 10 kali percobaan, didapatkan hasil pengukutan yang dapat dilihat pada Tabel 2.

Tabel 2. Hasil pengukuran waktu data serial tidak terenkripsi dan data serial terenkripsi

#	POST (ms)		GET (ms)	
	Data Serial	**Data Terenkripsi**	**Data Serial**	**Data Terenkripsi**
1.	104	120	115	207
2.	105	83	121	130
3.	94	86	111	186
4.	110	90	204	218
5.	88	101	124	124
6.	98	96	121	126
7.	83	90	121	118
8.	118	97	134	124
9.	80	91	128	122
10.	93	92	126	114
AVG	**97,3**	**94,6**	**130,5**	**146,9**



Gambar 7. Pengujian menggunakan metode POST



Gambar 8. Pengujian menggunakan metode GET

Tabel 2 merupakan hasil pengujian yang mengukur waktu transmisi data serial tidak terenskripsi dan data serial terenkripsi pada proses POST dan GET sebanyak 10 kali. Dari hasil percobaan, dapat dilihat pada Gambar 7 dan Gambar 8 yang merupakan grafik perbandingan antara data serial tidak terenkripsi dan data serial terenkripsi dalam hal performa waktu (ms). Pada metode POST data serial tidak terenkripsi, rata-rata waktu yang dibutuhkan untuk menerima *response* dari *server* adalah 97,3 ms, sedangkan pada data serial terenkripsi, rata-rata waktu yang dibutuhkan adalah 94,6 ms. Berdasarkan informasi tersebut, dapat diketahui bahwa kecepatan data serial terenkripsi lebih cepat 2,7 ms dibandingkan data serial tidak terenkripsi.

Pada metode GET, rata-rata waktu yang dibutuhkan data serial tidak terenkripsi adalah 130,5 ms, sedangkan pada data serial terenkripsi membutuhkan waktu yang lebih lama, yaitu 146,9 ms. Berdasarkan informasi tersebut, data serial tidak terenkripsi lebih cepat 16,4 ms dibandingkan data serial terenkripsi dalam menerima *response* dari *server*. Hal ini disebabkan karena dibutuhkan waktu untuk melakukan proses dekripsi data serial terenkripsi yang dikirim oleh *client* dan juga waktu proses enkripsi pada data *response* dikirim oleh *server*.

4. KESIMPULAN

Kesimpulan yang dapat diambil dari penerapan enkripsi SHA-512 pada REST API adalah mampu meningkatkan keamanan transmisi data. Enkripsi dapat meminimalisasi dampak serangan MitMA. Meskipun penyusup berhasil menyadap data yang ditransmisikan dari *client* ke *server*, dibutuhkan waktu yang lebih lama dan proses yang lebih panjang untuk mengekstrak data sensitif. Pada metode POST, data serial terenkripsi lebih cepat 2,8% dibandingkan data serial tidak terenkripsi. Namun, pada metode GET, data serial terenkripsi lebih lambat 12,5% dibandingkan data serial tidak terenkripsi. Meskipun data serial terenkripsi lebih lambat, penerapan enkripsi sangat penting dilakukan pada transmisi data REST API.

DAFTAR PUSTAKA

[1] OWASP, "The OWASP Top 10 2021," 2021. [Online]. Available: https://owasp.org/Top10/

[2] R. T. Fielding and R. N. Taylor, "Principled Design of the Modern Web Architecture," *ACM Trans. Internet Technol.*, vol. 2, no. 2, pp. 115–150, 2002, doi: 10.1145/514183.514185.

[3] W3C, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)," 2007. https://www.w3.org/TR/soap12-part1/

[4] S. Mumbaikar and P. Padiya, "Web Services Based On SOAP and REST Principles," vol. 3, no. 5, pp. 3–6, 2013.

[5] S. Chatterjee, "A comparative study on SOAP and RESTful web services," *Int. Res. J. Eng. Technol.*, no. May, pp. 2881–2885, 2020, [Online]. Available: https://www.irjet.net/archives/V7/i5/IRJET-V7I5553.pdf

[6] S. Peyrott, "The JWT Handbook," *Auth0 Inc.*, 2018, [Online]. Available: https://auth0.com/resources/ebooks/jwt-handbook

[7] S. Gueron, S. Johnson, and J. Walker, "SHA-512/256," pp. 3–8, 2011.

[8] A. Rahmatulloh, H. Sulastri, and R. Nugroho, "Keamanan RESTful Web Service Menggunakan JSON Web Token (JWT) HMAC SHA-512," vol. 7, no. 2, 2018.

[9] A. Mallik, A. Ahsan, M. M. Z. Shahadat, and J. C. Tsou, "Man-in-the-middle-attack: Understanding in simple words," *Int. J. Data Netw. Sci.*, vol. 3, no. 2, pp. 77–92, 2019, doi: 10.5267/j.ijdns.2019.1.001.

[10] G. Serme, A. S. de Oliveira, J. Massiera, and Y. Roudier, "Enabling Message Security for RESTful Services," in *2012 IEEE 19th International Conference on Web Services*, Jun. 2012, pp. 114–121. doi: 10.1109/ICWS.2012.94.

[11] M. Sumagita and I. Riadi, "Analysis of Secure Hash Algorithm (SHA) 512 for Encryption Process on Web Based Application," *Int. J. Cyber-Security Digit. Forensics*, vol. 7, no. 4, pp. 373–381, 2018, [Online]. Available: https://www.researchgate.net/publication/327392778

[12] T. Grembowski *et al.*, "Comparative Analysis of the Hardware Implementations of Hash Functions SHA-1 and SHA-512," 2002, pp. 75–89. doi: 10.1007/3-540-45811-5_6.

[13] H. N. Bhonge, M. K. Ambat, and B. R. Chandavarkar, "An Experimental Evaluation of SHA-512 for Different Modes of Operation," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, Jul. 2020, pp. 1–6. doi: 10.1109/ICCCNT49239.2020.9225559.