Jurnal Elektronik Ilmu Komputer Udayana
Volume 13, No 1. August 2024

p-ISSN: 2301-5373
e-ISSN: 2654-5101

Performance impact of Hybrid Cryptography in securing RESTful API messages using ECIES

Albertus Ivan Suryawan[a1], Agus Muliantara [a2], I Dewa Made Bayu Atmaja Darmawan[a3], Ngurah Agus Sanjaya ER [a4]

[a]Informatics Department, Udayana University
Bali, Indonesia
[1]albertusivan15@gmail.com
[2] muliantara@unud.ac.id
[3]dewabayu@unud.ac.id
[4] agus_sanjaya@unud.ac.id

[4]Coresponding Author: Agus Muliantara

Abstract

As technology advances this past decades, many businesses start to integrating technology into their business, which making transaction more convenience. However, this convenience also introducing several security threats against these transmitted data that often involving highly private data. While ideally payment services should have some security standards for its users, research show that misconfigured TLS could actually expose some security threats caused by flaw on certain revisions, which could then be potentially used in dictionary attack. In this research, the author tries to implements a hybrid cryptography implementation involving use of Elliptic Curve Cryptography algorithm and AES in form of Elliptic Curve Integrated Encryption Scheme to secure highly private message over REST APIs and assess its impact in term of performance. In the proposed system, every data that contains a personal data will be secured using end-to-end approach, where each data sent and received will be encrypted using ECIES with AES on the top HTTPS connection. As the result, there is a slight performance degradation at rate of 57 to 230 milliseconds or about 15.57% of the original implementation without any encryption involved inside the system. Although this degradation may seem minimal, it underscores the critical trade-off between performance and security. This increased duration is also still under the currently accepted standard for any transaction request maximum duration which is 8 seconds, and estimated duration for certain request to be completed by the proposed system can be predicted using following formula: y=0.01156x+1.23 with RMSE of 3.71*.*

Keywords: *Hybrid Cryptography, Elliptic Curve, AES, REST API, performance analysis, ECIES*

1. Introduction

As technology advances this past decades, many businesses start to integrating technology into their business, which making transaction more convenience. However, this convenience also introducing several security threats against these transmitted data that often involving highly private data. The concern come as this business often relies on other third-parties services that support their business operational such as payment gateway, point of sale software, and management software to name some. While ideally service offering such degree of service should have some security standards, but according to some research, misconfigured TLS [1], [2] could actually expose some security threats caused by flaw on certain TLS revisions [3]. That leaked data can be potentially used in dictionary attack which often relies on high volume of valid data.

Cryptography is often used to secure a secret message to ensure that only the authorized receiver has access to it. Some common cryptography algorithm includes Advanced Encryption Standard (AES), Rivest-Shamir-Adleman (RSA), and Elliptic Curve Cryptography. AES is a symmetric cryptography algorithm in which used a single key for both encryption and decryption [4]. While both RSA and Elliptic Curve Cryptography are an asymmetric cryptography which used 2 type of key, namely private key for decryption and public key for encryption, both also has different underlying structure in how the message are processed. Elliptic Curve Cryptography also has advantage of having a smaller size key for roughly the same security power as larger key of RSA or AES [5].

Table 1. Difference between Hoobi's proposed system and this paper's proposed system

	Hoobi's Proposed System	Proposed System
Key Security	DES key is embedded inside the data, and directly encrypted with ECC	AES encryption key derived from ECDH with KDF
Data Encryption Algorithm	DES	AES

Use of hybrid cryptography using Elliptic Curve has been done by Hoobi [6], where Elliptic Curve is being used with DES to secure a message, and resulted a better security with Elliptic Curve and DES compared to only using DES. Another research with similar interest that has been done is [7] where Elliptic Curve Cryptography in form of ECIES with enhanced hash function is used to improve IoT privacy by utilizing blockchain within. The author found that by changing the hash function, an additional 12% of improved data security in term of coefficient correlation between payload and ciphertext, while using 7% less computing time. Research on effectivity of payload encryption also has been done by Varma and UniKrishnan [8], where they measured the effectivity against Man-in-the-Middle (MiTM) attack between application with payload encrypted using AES, and one without any encryption on application level, which both are tested on HTTPS connection. They found that by encrypting sent payload, attacker that gained access to the connection due to MiTM attack, cannot have direct access to the payload.

Based on the related research, this paper aims to measure the performance impact against implementation of hybrid cryptography using AES and ECC with ECIES on existing system, as there are some security advantages can be gained by incorporating a hybrid cryptography into an existing system.

2. Research Methods

Due to the nature of the subject, this research used Waterfall Methodology which consist of 5 stages such as System Requirement Analysis, System Design, Implementation, Testing and Evaluation as shown in Figure 1.

2.1. System Requirement Analysis

System Requirement Analysis is the first stage of System Development Life Cycle (SDLC) where the developer will conduct an analysis to list any requirement needed for the system. Based on observation of technology companies, the system is required to be able to securely transmitted data to another system, while also maintaining stability on high load condition.

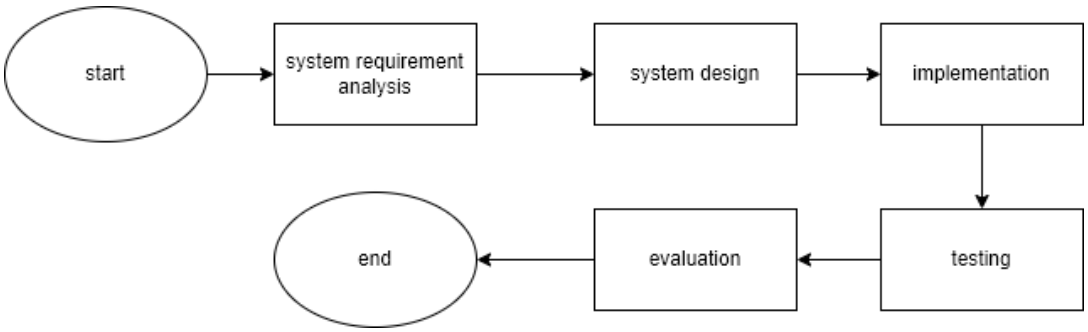

Figure 1. Research Methodology

2.2. System Design

In this research, there are 2 main system to be developed namely Client and Server. Client system is responsible to store user's credentials needed to be able transmitting data securely through the system. This system consists of 2 subsystems, desktop client, and security worker. The desktop client is a GUI-based application used by user to interact with the system. The security worker is program running independently from the main desktop application, and connected by Inter-Process Communication

p-ISSN: 2301-5373
e-ISSN: 2654-5101

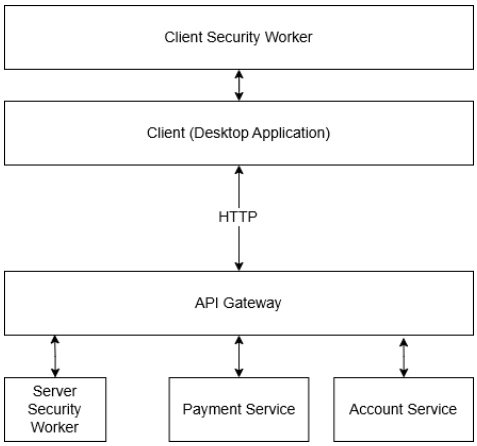

Figure 2. High Level Architecture of Proposed System

(IPC) to the desktop application. It used to handle all cryptographic related process. While on the server side, there is 4 subsystems namely account service, payment service, API gateway, and server security worker. The server security worker behaves almost the same as the client counterpart, except it also stores client's public key to be used in server-side encryption. Account service is used to handle all user related process such as user management, and user session managements. While payment service is used to handle all business logic involving payment services, such as deposit, withdrawal, fund transfer, and payment. All request towards server is handled by API gateway which then encrypt or decrypt payload received based on several condition using security service, before then forward that payload to each respective service. Client and Server are connected using HTTPS in RESTful manner, and utilized JSON as its body content type. High Level Architecture of the proposed system is shown Figure 2.

2.3. Implementation

The proposed system is developed using several technologies. In Client side, desktop subsystem is developed using both Rust language for security worker, and Svelte for desktop frontend. While in Server side, Rust language is being used for security worker, and Go language is being used for other services. The proposed system is also using Hybrid Cryptography implementation of ECIES which consist of ECC and AES to encrypt a message. To encrypt a message from either side of the system which shown in Figure 3, each system will fetch the recipient public key, sender secret key, and message to be encrypted. Then, a shared secret would be calculated using both secret key and public key using ECDH, in which used as key material to derive a MAC key and encryption key.

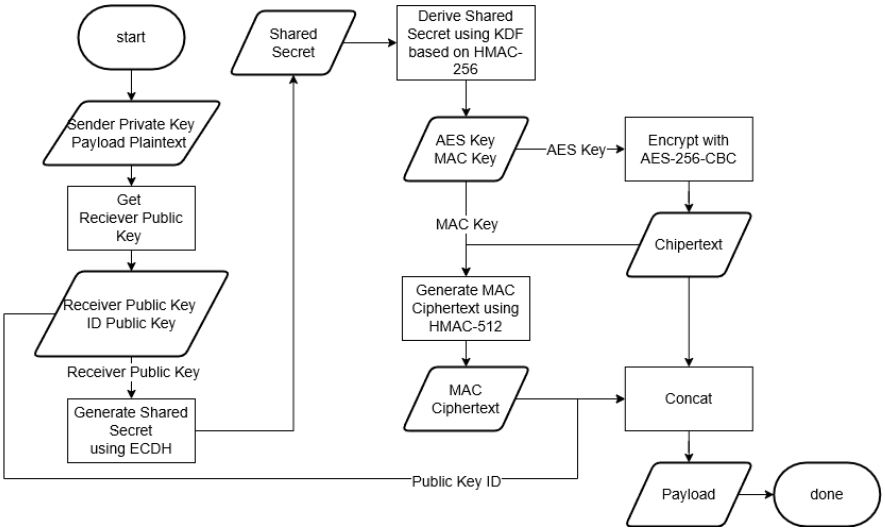

Figure 3. ECIES Encryption Scheme

Message is then encrypted using respective AES configuration using derived encryption key. MAC value of the ciphertext is then calculated using HMAC and MAC key. MAC, Ciphertext and Public Key ID then concatenated together as a single message to be send to the authorized recipient.

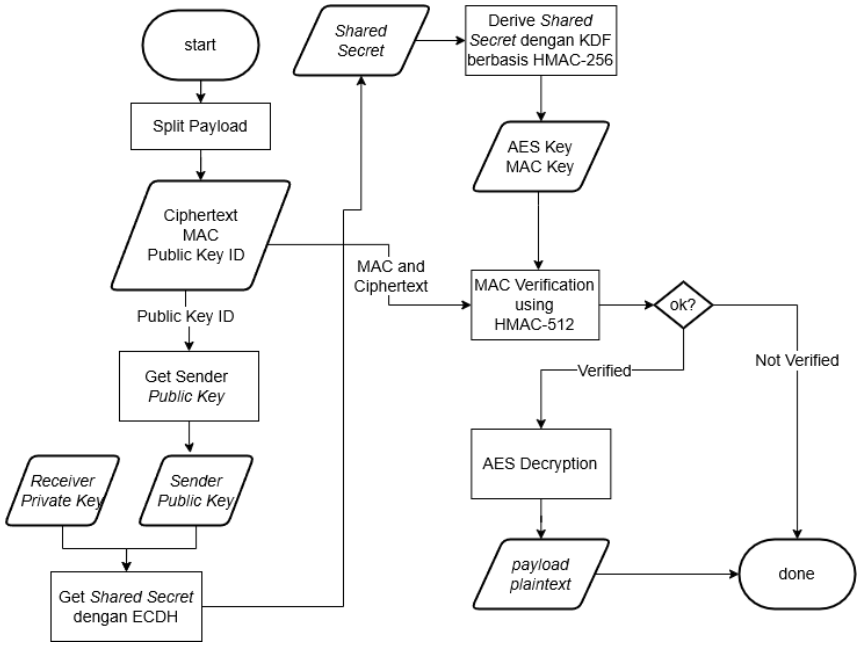

Figure 4. ECIES Decryption Scheme

To decrypt a message from either side of the system which shown in Figure 4, each system will fetch the recipient public key and encrypted message. The concatenated message then split into 3 parts consist of MAC, Encrypted Message, and Public Key ID. A shared secret then calculated from public key and secret key, which then used to derive both MAC key and Encryption Key. Message's MAC value then validated to ensure encrypted message is still intact. After validation passed, message then decrypted using encryption key.

3. Result and Discussion

In this research, the performance of the proposed application is measured by running a load test between proposed implementation of ECIES in the system and similarly built application without implementation of ECIES. Load test is done using k6 with fixed time of 60 seconds and range of virtual users. Virtual User is used to simulate number of users connecting to the Server side of application concurrently in a time. For this proposed system, the number of virtual users used is 100, 200 and 500 which denoted the time from sending requests to first byte received. This load test measures the time from sending request to first byte received in 3 types of number of virtual users across 2 scenario which is encrypted traffic and unencrypted traffic. The overall result is shown in Table 2.

Table 2. Overall Performance Result on Proposed System

Client Count	Request Duration (*ms*)					
	Avg	Min	Max	Avg	Min	Max
	Encrypted			Not Encrypted		
100	174.36	136.69	311.99	162.14	124.23	255.69
200	345.75	198.25	725.54	328.21	179.89	651.90
500	663.70	278.90	2003.48	637.87	257.68	1733.10

3.1. 100 Virtual Users

This scenario simulates low traffic of users interacting with the server. As shown in Figure 5, there are some minor bumps of average duration taken for each API call between encrypted and unencrypted traffic. Unencrypted traffic has average duration between 124ms and 255ms on the last iteration, while encrypted traffic has higher average duration between 136ms and 312ms.

p-ISSN: 2301-5373
e-ISSN: 2654-5101

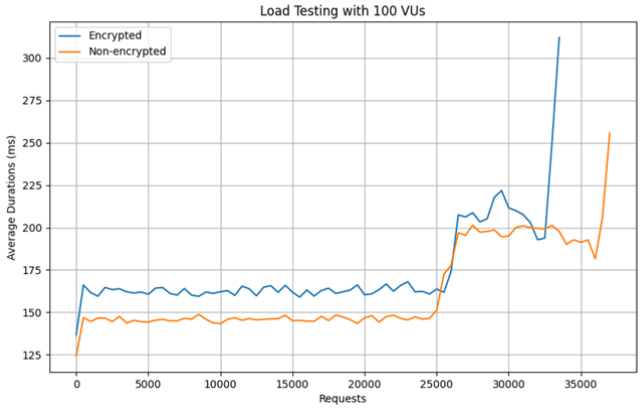

Figure 5. Load testing with 100 VUs

3.2. 200 Virtual Users

This scenario simulates moderate traffic of users interacting with the server. As shown in Figure 6, there are also some minor bumps of average duration taken for each API call between encrypted and unencrypted traffic. Unencrypted traffic has average duration between 180ms and 652ms on the last iteration, while encrypted traffic has higher average duration between 198ms and 725ms.

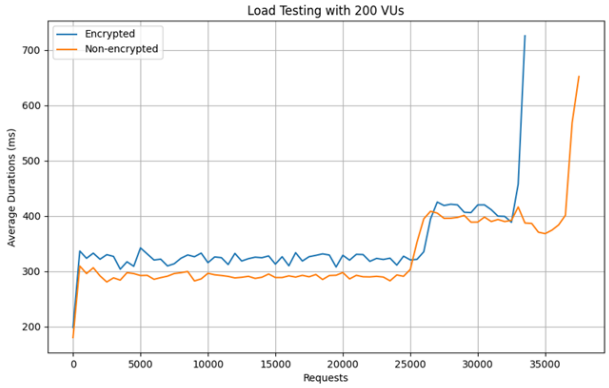

Figure 6. Load testing with 200 VUs

3.3. 300 Virtual Users

This scenario simulates a slight high traffic of users interacting with the server. As shown in Figure 7, there are some minor bumps of average duration taken for each API call between encrypted and unencrypted traffic. Unencrypted traffic has average duration between 258ms and 1733ms on the last iteration, while encrypted traffic has higher average duration between 279ms and 2003ms.

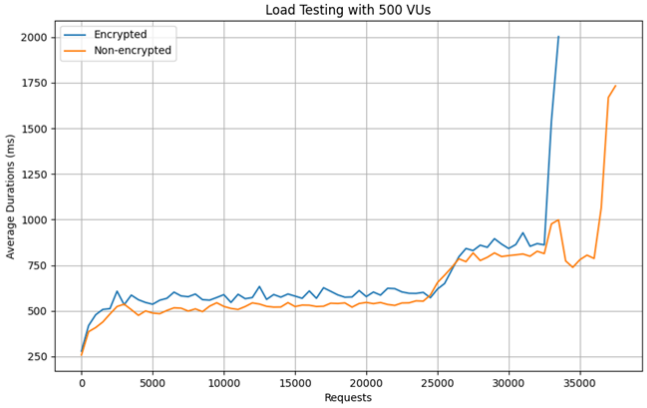

Figure 7. Load testing with 500 VUs

3.4. Performance in term of data size

This section shows the computation time required to process certain number of transactions in the proposed system incorporating encrypted payload. The system is tested against 10.000, 30.000, 60.000, 120.000, and 240.000 requests. As shown in Figure 8, the time it takes for the system to process such data range from 115 seconds (1 minutes and 55 seconds) for 10.000 data to 2.776 seconds (46 minutes and 16 seconds). The computation time required for certain number of transactions in this proposed system can also be predicted using following linear regression formula: y=0.01156x+1.23, where y is the computation time required in seconds, and x is the number of data to be processed, with RMSE of 3,70.

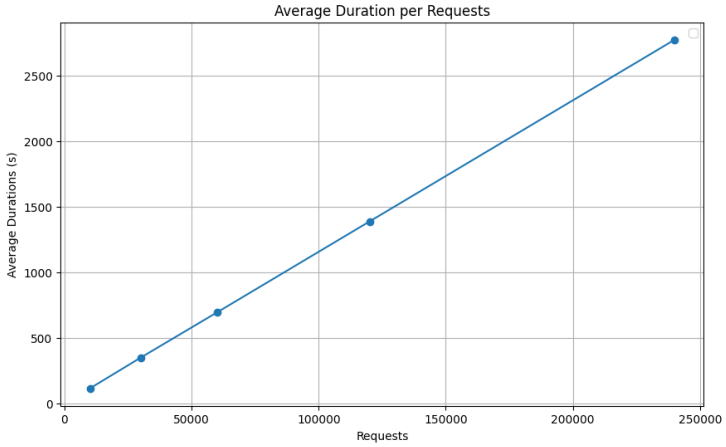

Figure 8. Comparison between number of request and computation time

4. Conclusion

There are many ways to improve security of a system, especially one with wide access to the internet. One of those are by incorporating cryptography into the system, by either encrypting the data, ensuring data integrity with hash, or even both. This research aimed to measure performance impact on incorporating encryption scheme which include both hashing (by using MAC) and encryption, into an existing system. Based on the result shown in the previous section, there are certainly a performance degradation ranging from 57ms until 230ms or roughly 15.57% compared to system without any implementation of ECIES, and can be predicted using linear regression with following equation $y = 0.01156x + 1.23$ with RMSE of 3.71. Such degradation of performance also still below the standard needed for transactional request in the industries. Although this degradation may seem minimal, it underscores the critical trade-off between performance and security.

References

[1] I. Ali, “Examining cyber security implementation through TLS/SSL on academic institutional repository in Indonesia,” *Berkala Ilmu Perpustakaan dan Informasi*, vol. 17, no. 2, pp. 238–249, 2021, doi: 10.22146/bip.v17i1.2082.

[2] J. K. Huang, Z. X. Zhang, W. J. Li, and Y. Xin, “Assessment of the impacts of TLS vulnerabilities in the HTTPS ecosystem of China,” *Procedia Comput Sci*, vol. 147, pp. 512–518, Jan. 2019, doi: 10.1016/J.PROCS.2019.01.238.

[3] A. Satapathy and J. Livingston, “A Comprehensive Survey on SSL/ TLS and their Vulnerabilities,” *Int J Comput Appl*, vol. 153, no. 5, pp. 31–38, Nov. 2016, doi: 10.5120/ijca2016912063.

[4] M. J. Dworkin *et al.*, “Advanced Encryption Standard (AES),” Nov. 2001, doi: 10.6028/NIST.FIPS.197.

[5] D. Mahto and D. Kumar Yadav, “RSA and ECC: A Comparative Analysis,” 2017. [Online]. Available: http://www.ripublication.com

[6] M. M. Hoobi, “EFFICIENT HYBRID CRYPTOGRAPHY ALGORITHM,” *Journal of Southwest Jiaotong University*, vol. 55, no. 3, 2020, doi: 10.35741/issn.0258-2724.55.3.

[7] Y. P. Khanal *et al.*, “Utilizing Blockchain for IoT Privacy through Enhanced ECIES with Secure Hash Function,” *Future Internet*, vol. 14, no. 3, Mar. 2022, doi: 10.3390/fi14030077.

[8] A. Varma and S. UniKrishnan, “Effect of payload security in MQTT protocol over transport and application layer,” *IOP Conf Ser Mater Sci Eng*, vol. 1166, no. 1, p. 012019, Jul. 2021, doi: 10.1088/1757-899x/1166/1/012019.