

Incorporating rivest-shamir-adleman algorithm and advanced encryption standard in payment gateway system

Veronica¹, Raymond Sunardi Oetama¹, Arief Ramadhan²

¹Department of Information Systems, Faculty of Engineering and Informatics, Universitas Multimedia Nusantara, Tangerang, Indonesia

²School of Computing, Telkom University, Bandung, Indonesia

Article Info

Article history:

Received Jul 31, 2023

Revised Feb 27, 2024

Accepted Mar 5, 2024

Keywords:

Advanced encryption standard

Agile methodology

Credit cards

Payment gateway system

Rivest-shamir-adleman algorithm

ABSTRACT

The rapid development of technology has led to various advancements, including the ability to conduct online payments through applications. Companies providing digital transaction payment services require a payment gateway system as an intermediary for online transactions. The system acts as an intermediary between merchants with digital wallets and banks, involving the company, merchants, and banks in its development. The system includes essential features like bill payment, user credit card verification, and transaction checking, customized to meet the specific requirements of merchants and adhere to security standards. In this study, we incorporated the rivest-shamir-adleman (RSA) algorithm and advanced encryption standard (AES) to ensure the security of a payment gateway system. We adopt the agile methodology in the development process of the system. We test the acceptance of the system to the user, and we also test the performance of the system. The results of this research show that the system can be accepted by the user, fulfill the user's needs, can be executed well, and performs adequately in handling multiple transactions concurrently. The outcome of this study can serve as valuable input for the company in building its own system, providing insights into algorithm implementation techniques and the system workflow.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Arief Ramadhan

School of Computing, Telkom University

Bandung, Indonesia

Email: arieframadhan@telkomuniversity.ac.id

1. INTRODUCTION

The rapid development of technology in accordance with the progression of time has led to various advancements, including the ability to conduct online payments through various applications. According to data obtained from Bank Indonesia at the databoks forum, there was a significant increase of 58.60% in the use of electronic money in December 2021 compared to the previous year [1]. This widespread adoption of electronic money has prompted several companies to offer digital wallets as a facility to their customers. A digital wallet is an electronic application that serves as a tool for conducting online transactions [2]. Its primary function is to replace cash payments with electronic transactions, allowing users to make transactions anywhere quickly. Companies providing digital transaction payment services require a payment gateway as an intermediary for online transactions. This payment is made through an application that connects to a payment gateway to communicate with banks. A payment gateway is a payment system that connects online payments made through e-commerce platforms with financial institutions [3].

The payment gateway plays a vital role in ensuring seamless transactions and protecting customer transaction data as it travels over the internet [4]. Acting as a link between electronic wallets and financial

institutions, the payment gateway securely transfers transaction details to the financial institution, safeguarding them from unauthorized access. Its primary purpose is to establish a secure and encrypted pathway to the financial institution, protecting the encrypted data that includes user card information and transaction details. During the transaction approval process, the payment gateway relays transaction information back to the digital wallet or application, allowing for verification and completion of the payment. Security is very important for this kind of payment system [5], [6]. An encryption algorithm can be used to ensure the security of this kind of system [7], [8].

Rivest shamir adleman (RSA) is an algorithm in encryption system that utilizes asymmetric encryption mechanisms. Asymmetric encryption involves the use of two keys: a public key and a private key [9]. The advanced encryption standard (AES) algorithm is commonly used for data encryption. AES employs symmetric encryption mechanisms. It supports key sizes of 128, 192, or 256 bits. The number of iterations performed increases with larger key sizes: 10, 12, and 14 times for 128, 192, and 256 bits, respectively [10]. Both RSA and AES have been used in banking or payment systems in several research [11]–[14]. However, none of that research focused on the implementation of both RSA and AES in payment gateway system. Therefore, in this study, we focus on incorporating both RSA and AES in a payment gateway system.

A payment gateway is provided by a company, which plays a role in offering online payment solutions to companies associated with digital wallets. The company that become our case study is a new client of a credit card service provider and collaborates with them to facilitate payments between digital wallet providers and banks. The specific type of payment to be executed through this collaboration is credit card payments. Currently, there are several digital wallet clients who will utilize the company's services for credit card payments. As a result, the company aims to develop a secure credit card payment gateway system within their digital wallet application, featuring the logo of the credit card service provider they are partnering with. Previously, the company had a payment gateway for conducting payments, but they did not have a specific service for credit card payments within their range of offerings. This was due to the lack of customers who required a payment gateway service designed specifically for credit card payments. Consequently, the company intends to build a secure payment gateway system tailored for credit card payments. After discussions with the company's manager, the proposed solution to address this issue is to create a payment gateway system. The flow of the payment gateway system involves users making credit card payments from their digital wallets. The payment will be processed through the payment gateway, which will connect to the financial institution responsible for completing the credit card payment. The payment status will be responded to by the financial institution and sent to the merchant for displaying the transaction status. The payment gateway system needs to be integrated smoothly with the company's existing application, and therefore, interviews will be conducted between the researchers and the company to gather the necessary information. Based on the insights gained from these interviews, the payment gateway system can be developed. If the system meets the criteria, further development will be pursued to expand the company's services.

The objective of this paper is to create and execute the development of an electronic payment gateway. The gateway is used for a secure online payment system that includes credit card payments, employing a security mechanism using RSA and AES based on a secure exchange procedure. In this system, the customer's financial information is transmitted directly to a payment gateway.

2. METHOD

We adopted the agile methodology in this study. Agile methodology is a flexible methodology that responds well to changes and aims to improve the quality of software during project development. The development process using agile methodology is in the form a cycle that consists of requirements analysis, design, system building, and testing [15]. We adopted the agile methodology in our study and depicted as research design in Figure 1.

The research design in Figure 1 begins with problem analysis, followed by gathering user requirements through interviews with the chief executive officer (CEO) and project manager. Data is collected to create the payment gateway system, including transaction details, user interactions, product information, pricing, and credit card personal account number (PAN). Examples of the required data are obtained from client companies and customized in collaboration with the company. Once the data is collected, the business process for the payment gateway is designed the system with suggest from the project manager.

Figure 2 explains the business process of a payment gateway being developed by the company for online credit card payments using digital wallets. The main goal of this process is to facilitate connections between merchants with digital wallets and the provision of credit card payment functionalities. The process starts with users accessing the merchant application and using the funds in their digital wallets to make credit

card payments. Transaction details, such as payment rate, PAN, and user number, are then transmitted to the payment gateway, which acts as an intermediary between the merchant, and the involved bank.

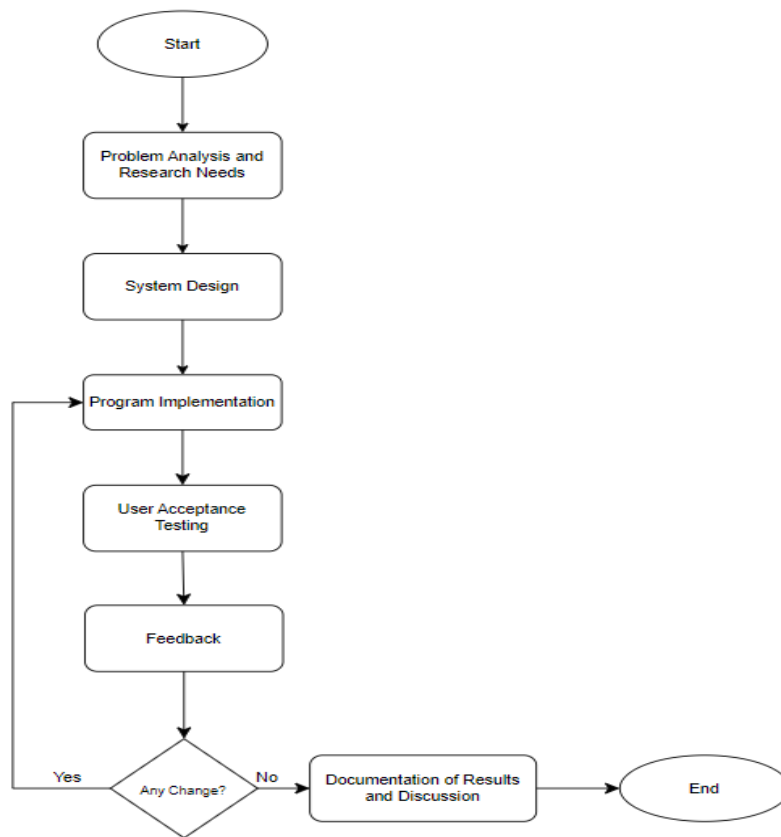


Figure 1. Research design

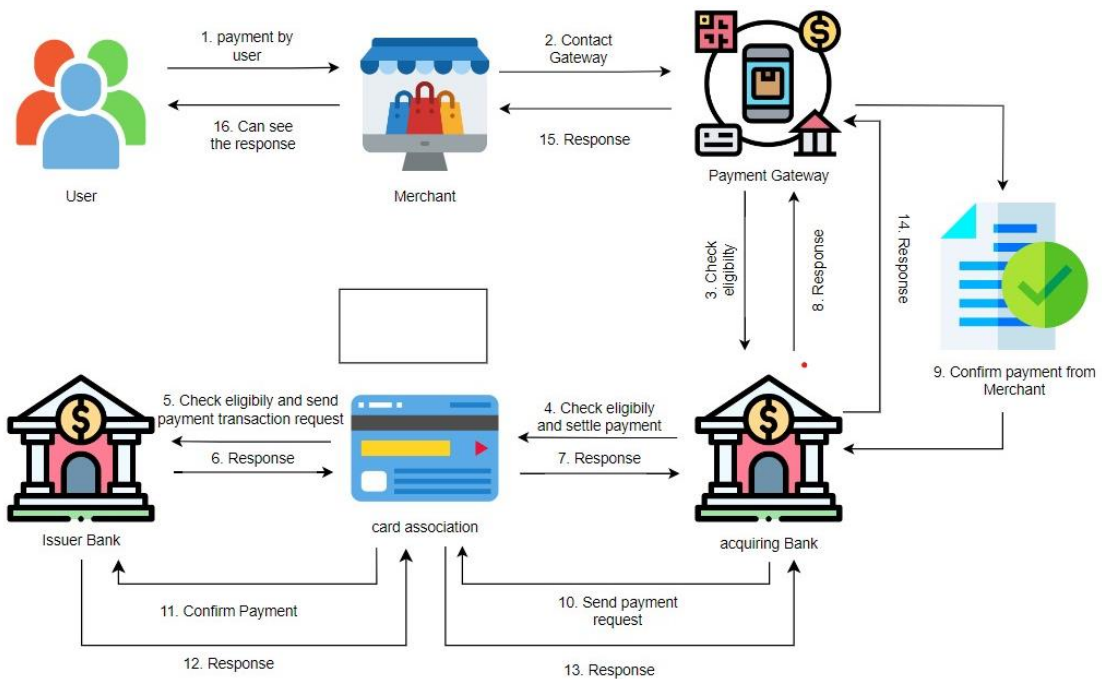


Figure 2. Business process of a payment gateway being developed

In the next process, the payment gateway verifies the user's identity and card details by sending the PAN to the acquiring bank, which, in turn, forwards the request to the corresponding financial institution indicated by the card association shown in Figure 2. The card association collaborates with multiple banks, each associated with a specific card logo (e.g., visa, mastercard). The issuer bank, responsible for the user's card, verifies the credit card's validity and sends a response to the payment gateway, indicating success or failure. If the response is positive, payment confirmation proceeds.

Upon receiving a positive response, the payment gateway sends a settlement request to the acquiring bank to finalize the credit card bill payment. The acquiring bank temporarily holds the funds and confirms the payment with the issuer bank through the card association. Once the payment is confirmed, the user's credit card is charged with the specified amount. The acquiring bank then transfers the previously withheld funds to the issuer bank, and a confirmation response is sent from the acquiring bank to the payment gateway. This response is subsequently relayed to the merchant for display to the user, confirming the successful completion of the transaction.

Several web and network technologies are used in the implementation steps. The application programming interface (API) gateway is used to serve as separate applications from external users by dynamically directing requests from clients and end applications to multiple internal applications [16]. It is a service that authorizes developers to create, implement, and manage APIs on the frontend, enabling access to data from backend services such as database access and storage. The API gateway acts as an intermediary between clients and servers, providing a consistent and uniform interface as a collection of microservices. This allows clients to interact with developers [17].

We also used go lang or go as our programming language. It is an open-source programming language developed by Google. It shares similarities with C and C++ and is known for its excellent features that contribute to its simplicity. Go is easy to learn, thanks to its lightweight syntax and efficient memory management. It offers advantages such as fast execution time and a structured approach. Static typing ensures security when working with go. Furthermore, go provides comprehensive documentation and supports cross compilation [18]. Echo/labstack is also used in the implementation step. It is a go programming language framework that excels in performance, minimalism, and extensibility. It is well-suited for large-scale systems. Notable features of echo include optimized routing, HTTP/2 support, data binding, rendering, middleware, and customizable error handling. These features empower users to tailor the framework to their specific needs and preferences [19].

For the database, we used PostgreSQL and elasticsearch. PostgreSQL is an open-source database that offers excellent services and modern features such as complex queries, triggers, views, and transactional integrity. It allows users to add extensions for data types, functions, operators, and procedural languages. However, PostgreSQL can be used for both medium-scale and large-scale applications [20]. Elasticsearch is a distributed restful search tool that is based on a software library for information retrieval. It differs from relational database management systems (RDBMS) in that elasticsearch is a search and document storage tool based on key-value pairs. Searching in elasticsearch involves not only queries but also filters, which are useful for fast document retrieval [21]. Each object in elasticsearch is represented as JavaScript object notation (JSON), so it doesn't use structured query language (SQL) for database creation. JSON is a lightweight format used for exchanging data between web application clients and servers. Its main purpose is to provide a versatile method for storing and transferring data. JSON is human-readable and straightforward to write, making it user-friendly. It also enables easy processing and construction by machines. The format of JSON involves key-value pairs, where keys are typically strings enclosed in quotation marks. JSON values can be strings, booleans, null, arrays, and other objects [22].

We used vault to implement the RSA dan AES. Vault is an open-source tool for secrets management software package. Developed by HashiCorp, it is designed to securely store and refresh tokens for users across various virtual organizations. Vault supports openID connect (OIDC), kuberos, and offers a flexible plugin architecture and representational state transfer (REST) API. Essentially, vault is used to generate new tokens and provide a secure storage location for them. It employs RSA and AES algorithms in token creation. Vault aims to enhance data security for both large and small enterprises [23]. We also used SmartVista integration platform (SVIP) as middleware that helps the integration of financial services through a single platform. SVIP has several key features of a payment hub, it can act as universal gateway and it supports open API platform.

Once the implementation step is completed, the end user will try the system and may request improvements from the developer. The system testing is conducted to verify the results against expectations and is reviewed by the company. The success of the system testing will be evaluated through user acceptance test (UAT), and the level of transaction speed will be measured using apache JMeter. The apache JMeter can be used to assess the performance of static and dynamic resources, simulating heavy loads on servers, networks, and objects. Its purpose is to analyze overall performance under different types of loads [24]. Any

issues or desired changes after testing. The client will give a feedback to the researcher for refinement. Once all additions and changes are addressed, the research findings can be documented as a research report.

3. RESULTS AND DISCUSSION

3.1. Problem analysis and research needs

We identified two problem types exists, those are the process oriented related problem and information oriented related problem. There are three actors involved in all of those problems. Merchant, payment gateway, and acquiring bank. All of those actors have several specific needs related to the two problems type as listed in Table 1.

Table 1. The problem types, actors, and needs

Type	Actor	Needs
Process oriented	Merchant	<ol style="list-style-type: none"> 1. The system is capable of processing transactions for credit card payments that have been aligned with their system. This includes both the database and the flow of transaction entries into the payment gateway. 2. The payment gateway system is expected to provide success or error codes. 3. The payment gateway system is expected to provide success or error codes. 4. The system can perform credit card verification for users. 5. Merchants can view transaction responses by sending a status check request to the payment gateway. 6. The system is able to store transaction data, both successful and unsuccessful ones, that have been performed. 7. Merchants can view detailed daily transaction reports in the form of a comma-separated values (CSV) file.
	Payment gateway	<ol style="list-style-type: none"> 1. Providing verification of the connection between the merchant and the bank using JSON web token (JWT) or signature for receiving and sending APIs. 2. The payment gateway can view the details of both successful and failed transactions for all banks and merchants. 3. Employees can view daily transactions and generate transaction data in the form of a CSV file.
	Acquiring bank	<ol style="list-style-type: none"> 1. The system is expected to send transactions in JSON format. 2. The system is expected to securely transmit data into the bank. 3. The bank can view and search transactions for all registered merchants. 4. The bank can view daily transactions in the form of a CSV file.
Information oriented	Merchant	<ol style="list-style-type: none"> 1. The system must store transaction data as a company recap for future reference. 2. The system must be easily accessible for merchants. 3. The system must store daily transactions that are conducted. 4. The received response should provide an explanation if any issues occur during the transaction process.
	Payment gateway acquiring bank	<ol style="list-style-type: none"> 1. The system stores transaction data in a detailed manner. 2. The system must store daily transactions that are performed.

3.2. System design

We use use case diagram and activity diagram to deliver the system design. Figure 3 presents the use case diagram for the payment gateway system being developed. The diagram includes two actors: the merchant and the bank acquirer. The merchant engages in six activities, with the “verify connection” activity appearing in three of them. These activities involve: 1) “authorize user card” to verify the validity of user credit cards, 2) “check payment” to review transaction details and their status, and 3) “verify connection” to authenticate the merchant's connection to the payment gateway using JWT or signature. Additionally, the merchant performs “bill payment” by sending user transactions to the payment gateway for processing, “check daily report” to examine daily transactions in a static CSV file, and “check detail transaction” to access transaction status on the company website. In the bank acquirer role, two activities are conducted: “check detail transaction” to view transaction details on the company website and “check daily report” to observe daily transactions stored in a static CSV file.

Figure 4 illustrates the authorization process for user credit cards. The merchant initiates the process by sending an authorization request with JWT or signature to the API gateway. The API gateway verifies the merchant's registration and masks the PAN, creating a modified request for the SVIP component. The SVIP further adjusts the request to match the requirements of the specific credit institution. The SVIP then sends the request to the credit card issuer to validate the user's credit card. The final outcome, success or failure, is communicated back to the merchant to be displayed to the user.

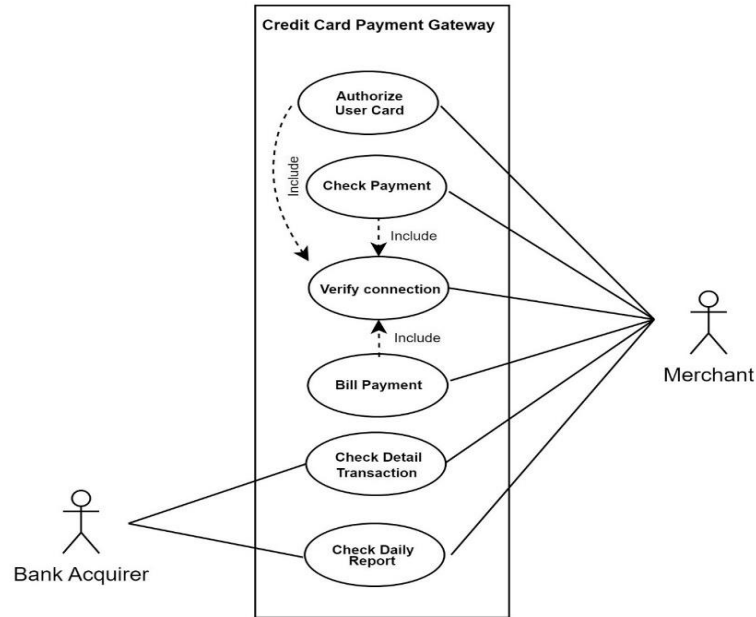


Figure 3. Use case diagram

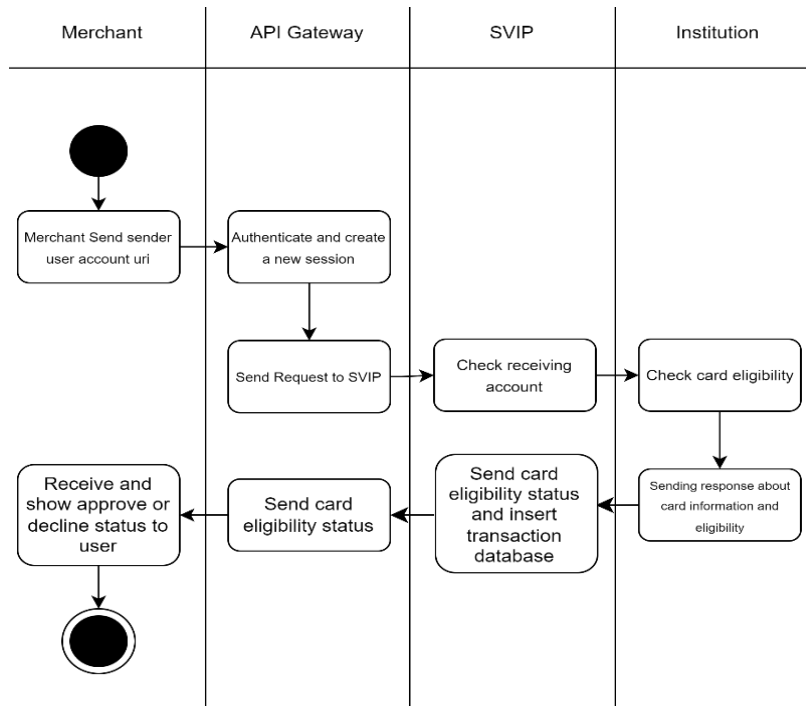


Figure 4. Activity diagram authorization user card

The activity diagram for the bill payment process is presented in Figure 5. Here is an explanation of the flow of bill payment in the system. The initial step involves verifying the validity of the user's credit card, following a similar process as the user card authorization in Figure 4. The difference lies in the continuation of the payment process when the user's credit card is deemed valid. In the payment process, the adjusted request is sent to the SVIP component. The SVIP further customizes the request according to the credit institution's requirements and forwards it to the user's bank. The bank responds with the transaction status, which is then stored in the API gateway's transaction log. Finally, the API gateway returns the response to the merchant for displaying the transaction status to the user.

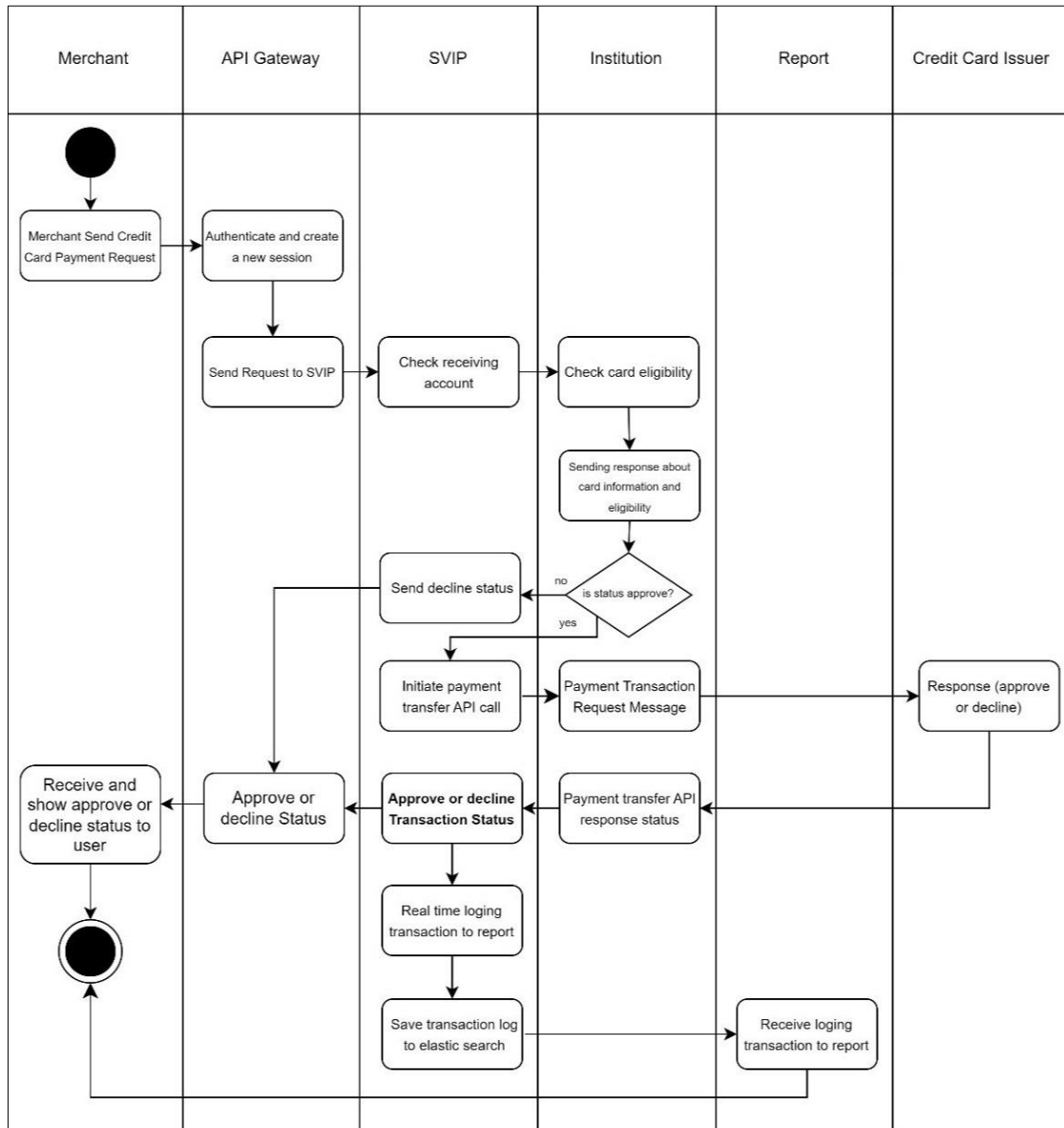


Figure 5. Activity diagram billpayment

In Figure 6, the activity of checking the transaction status is depicted. Here is an explanation of the flow in the check payment activity. Initially, the check payment activity verifies the authorization of the user card. Once the validation is successful, the API gateway sends a request to check the transaction. The API gateway forwards the request to the SVIP for adjustments, and then it is sent to the credit card issuer. The credit card issuer sends the request to the relevant bank for verification. The bank performs the check and returns the transaction status to the merchant via the API gateway. The merchant can then proceed with further actions based on the received status.

Activity diagram in Figure 7 presents the steps for users to check detail transaction. They must log in to the web-based application and navigate to the transaction tab on the main page. By selecting the static option, users can apply personalized filters. The website generates a report, presenting users with the desired transaction details.

Activity diagram in Figure 8 presents the process of checking daily reports in a web-based application. Users must log in to access transaction details. After logging in, they navigate to the main page, select the transaction tab, and choose the static option. On this page, users can customize filters to their liking. The website generates a daily report in CSV format, which is then displayed for users to review.

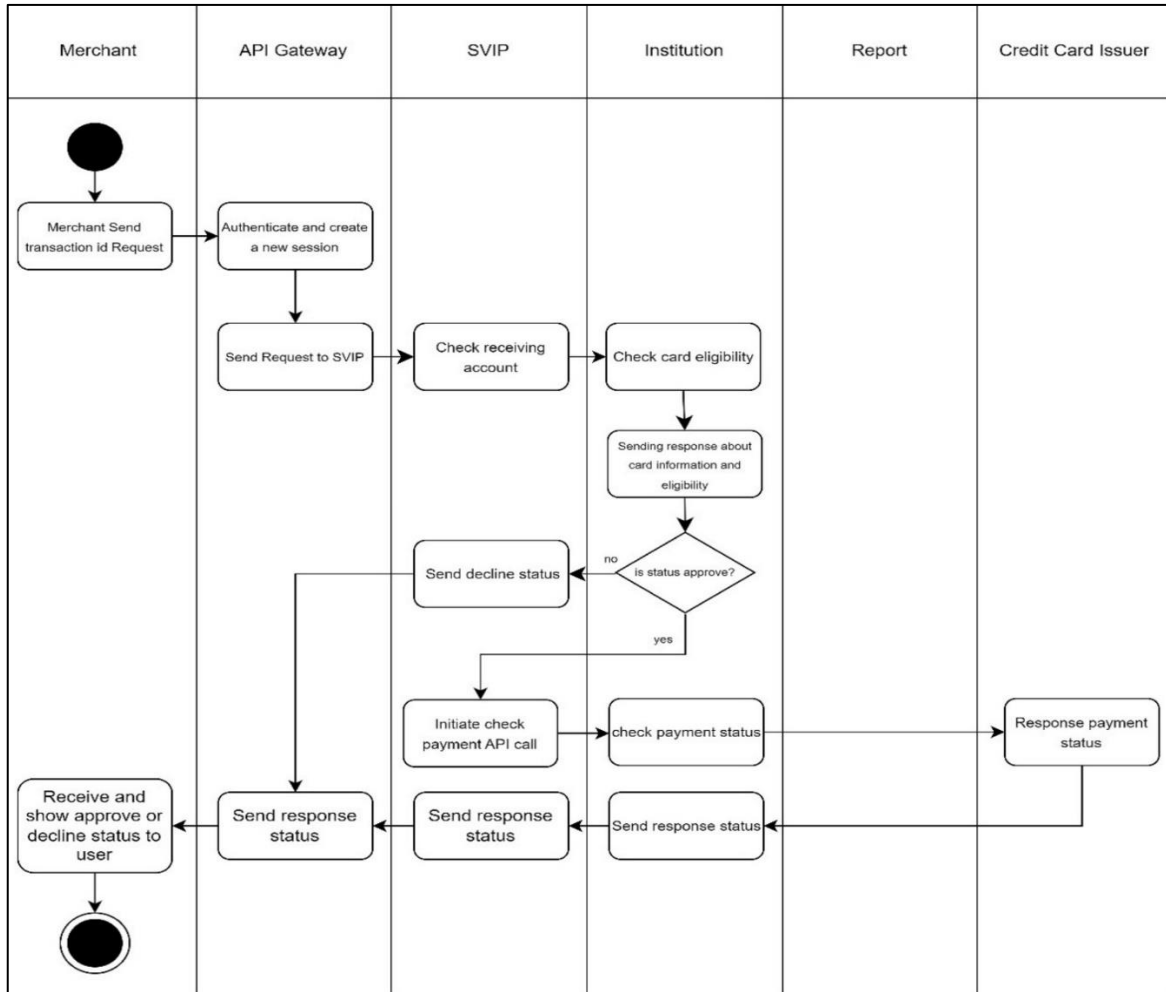


Figure 6. Activity diagram check payment

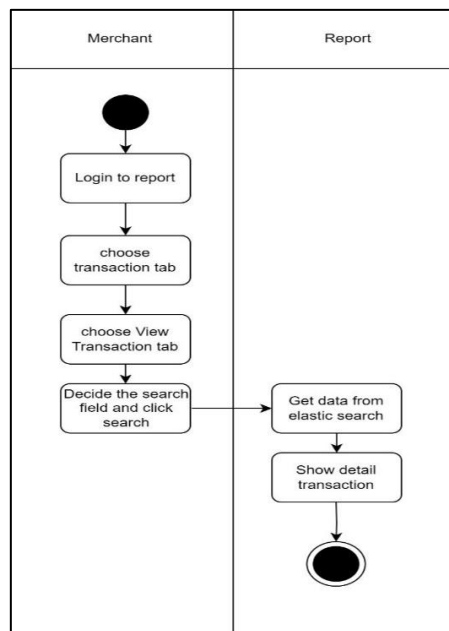


Figure 7. Activity diagram check detail transaction in dashboard

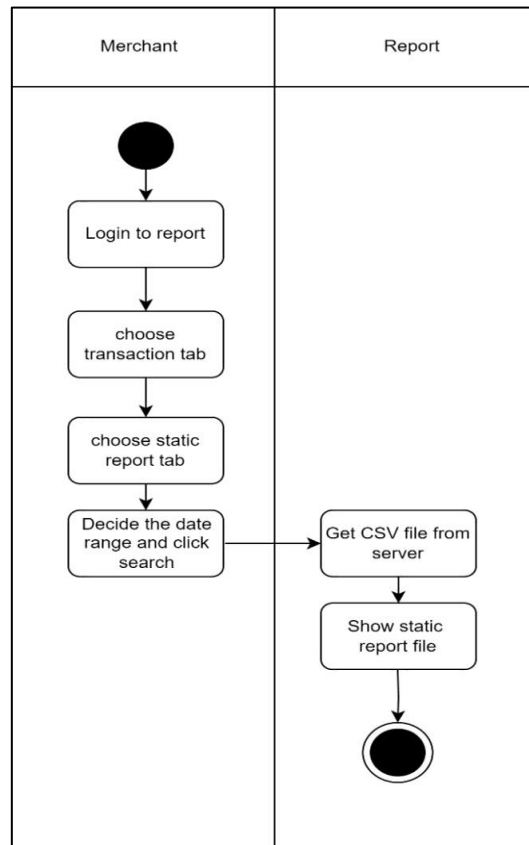


Figure 8. Activity diagram check daily transaction in dashboard

3.3. Payment gateway implementation

The following is an explanation of the flow of using the AES and RSA algorithms in a payment gateway for encrypting user credit card numbers. First, the AES algorithm will be used. The following outlines the steps performed on the user's credit card number using the AES algorithm [25]:

- 1) A key size of 256 bits will be generated, and the block size for encrypting the algorithm will be 128 bits. Therefore, each block of data to be encrypted using the 256-bit key will be divided into 128-bit blocks, and each block will be encrypted separately. For example, if there is a file of size 512 bits (64 bytes) and it is encrypted using AES with a 256-bit key, the file will be divided into 4 blocks, each block being 128 bits (16 bytes) in size.
- 2) Based on the predetermined key size of 256 bits, 14 rounds of encryption will be performed. Each round has its own key obtained through key scheduling. The following provides an overview of the key usage in each round [26].

The explanation pertains to the utilization of the key schedule algorithm in AES. Since there are 14 rounds, a total of 15 keys are required. These 15 keys consist of one input key and 14 keys derived from the input key. The input key is referred to as key 0 (the first key). Key 0 generates 14 different keys. The generation of these 14 keys involves matrix calculations. Figure 9 provides an illustration of a single block as mentioned in the previous explanation.

The image shows that each byte row contains 1 word, resulting in 4 words per block. With 14 rounds of AES 256-bit, there are a total of 96 words. Therefore, 95 words need to be searched for. The keys are represented by w_1 to w_7 for key 0, w_8 to w_{16} for key 1, and so on. Figure 10 illustrates the AES key schedule flow for key 1.

The algorithm in Figure 11 follows a basic formula: $w_i = w_{(i-N_k)} \oplus w_{(i-1)}$. In this formula, w represents the word in column i , and N is the starting word index in each round. Here's an example of the basic formula for key scheduling in the second block, for the 13th word:

$$w_{13} = w_{13 - N_8} \oplus w_{13-1}$$

$$w_{13} = w_5 \oplus w_{12}$$

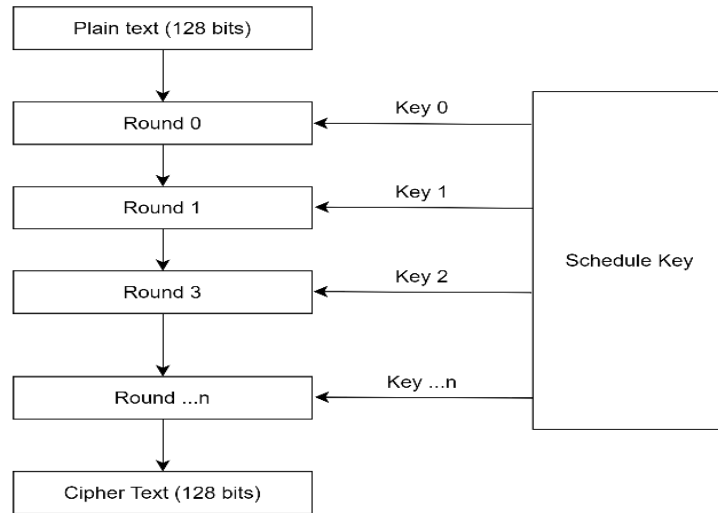


Figure 9. Block matrix in AES algorithm

w_0	w_1	w_2	w_3	
b_1	b_5	b_9	b_{13}	$w = \text{Word}$ $b = \text{Bytes}$
b_2	b_6	b_{10}	b_{14}	
b_3	b_7	b_{11}	b_{15}	
b_4	b_8	b_{12}	b_{16}	

Figure 10. Block matrix in AES algorithm

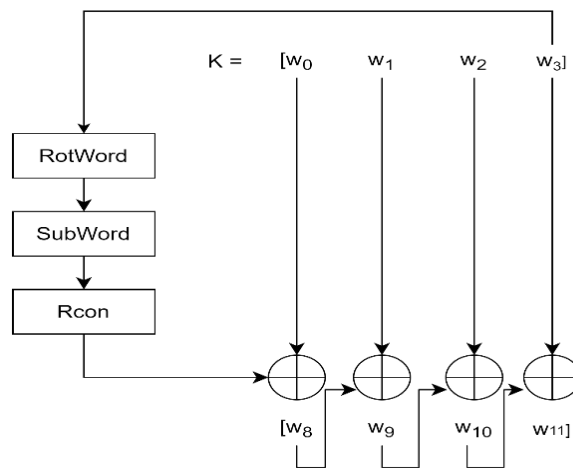


Figure 11. Block matrix in AES algorithm

The following are the steps to obtain key 1 and subsequent keys [27].

- a. RotWord: in the RotWord step, a positional shift is performed by shifting the contents of each word column. Each word consists of 4 bytes and the word chosen is the last word in the block.

- b. SubWord: in the SubWord step, the obtained words are transformed using the S-box table, which is a 16x16 block. The S-box is a constant table. For example, if we have a 4-byte word $W_3 = [B_{14}, B_{15}, B_{16}, B_{13}]$ where B_{14} is 14, it is read as row 1 and column 4 in the S-box table, resulting in the value fractional anisotropy (FA). Thus, B_{14} becomes FA. This process is applied to other bytes as well, resulting in $W_3 = [FA, D2, 77, 0C]$.
- c. Round constant: the round constant is a 4x1 matrix where the values in the last 3 rows are 0, and in each different round, the value in the first-row changes. The round constant matrix is used to perform XOR with the result of the SubWord operation. The following are the steps involved in the round constant process:
- After obtaining the value from SubWord, it is XORed with the round constant for the round: $g(w_3) = \text{SubWord}(w_3) \oplus R1$.
 - Converting all hexadecimal values in the matrix to binary:
SubWord (w_3) = 11111010 11010010 01110111 00001100
R1 = 00000001 00000000 00000000 00000000
 - The XOR operation results in:
11111011 11010010 01110111 00001100
 - The binary result is then converted back to hexadecimal. This becomes the round 1 key:
 $k_1 = [FB, D2, 77, 0C]$
- 3) After obtaining the key, each block will go through four steps in each round: SubBytes, ShiftRows, MixColumns, and add round key. Here is an example implementation for each step [10]:
- a. SubBytes: in the SubBytes step, each byte is substituted with another byte using the S-box table. An example implementation of SubBytes is performed by replacing specific bytes with their corresponding values from the S-box.
- b. ShiftRows: in the ShiftRows step, each row is shuffled according to the following rules.
- The first row remains unchanged.
 - The second row is shifted one position to the left.
 - The third row is shifted two positions to the left.
 - The fourth row is shifted three positions to the left.
- An example implementation of ShiftRows is performed by shifting the rows of the block.
- c. MixColumns: in the MixColumns step, each column undergoes matrix multiplication with a specific matrix. An example implementation of MixColumns is performed by multiplying each column with the predefined matrix.
- d. Add round key: in the add round key step, the block is XORed with the round key. An example implementation of add round key is performed by XORing the block with the corresponding round key. By following these steps, the block undergoes the transformations of SubBytes, ShiftRows, MixColumns, and add round key for each round in the AES algorithm. The AES encryption process is followed by encrypting the 256-bit key using the RSA algorithm. Here's a simplified explanation of the RSA implementation [16]:
- a) Key generation:
- Select two complex prime numbers, p and q, as the private key.
 - Compute $n = p * q$ as the public key.
 - Calculate $\Phi(n) = (p-1) * (q-1)$ and store the result.
- b) Choose a random value e as the public key, which is not related to $\Phi(n)$.
- c) Calculate d as the private key using the equation $d.e \pmod{\Phi(n)} = 1$.
- d) Define the communication function $h(m, k) = m^k \pmod{100}$.
- e) Encryption:
- Convert the AES key into text representation.
 - Provide the public key (e and n) to the payment gateway.
 - Choose a random value k.
 - Perform calculations $A = k^e \pmod{n}$, $B = m \cdot (k + 1)^e \pmod{n}$, and $H = h(m, k)$
 - Send the values A, B, and H.
- By following these steps, the AES key is encrypted using RSA, and the resulting values are transmitted for communication. When the receiving party receives the ciphertext consisting of variables A, B, and H, they perform the following calculations to decrypt the key:
- a. Calculate k using the formula [28]:
- $k = A^d \pmod{n}$
- b. Calculate the original message value m:
- $m = B / (k + 1)^e \pmod{n}$

c. Check if $H=h(m, k)$

If the calculated H matches the received value, the process proceeds successfully.

The details of a transaction can be seen in Figure 12. It illustrates the system's capability to connect to the transaction detail dashboard and effectively present successfully executed transactions. The system can also show the record of failed or successful transactions as depicted in Figure 13.

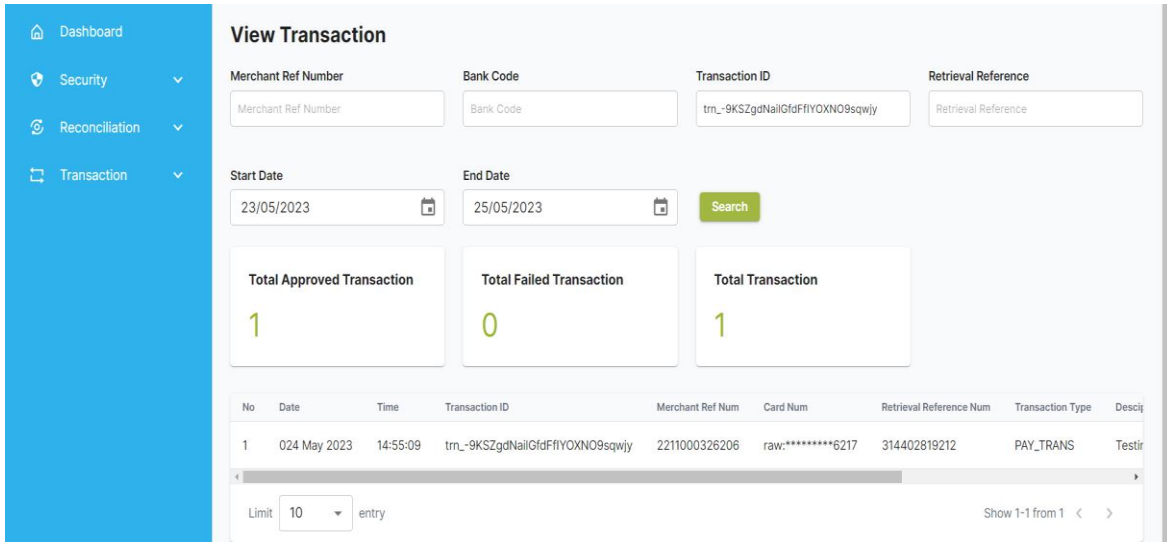


Figure 12. Detail transaction dashboard

Transaction ID	User Refno	Operation ID	Date	Time	Trans Type	Amount	Status	BTN_Fee
trn_N9UtVVTrl	1.5E+09	2.3052E+15	5/24/2023	20:11:24	Testing	5.8E+08	success	5000
trn_szCpVdMm	1.5E+09	2.3052E+15	5/24/2023	20:14:06	Testing	5.8E+08	success	5000
trn_ep2HzgB0C	1.5E+09	2.3052E+15	5/24/2023	20:14:06	Testing	5.8E+08	success	5000
trn_CtTQ5XgM	1.5E+09	2.3052E+15	5/24/2023	20:14:06	Testing	5.8E+08	success	5000
trn_vlaXz4SYSh	1.5E+09	2.3052E+15	5/24/2023	20:14:06	Testing	5.8E+08	failed	5000
trn_m-pTelTYN	1.5E+09	2.3052E+15	5/24/2023	20:14:07	Testing	5.8E+08	success	5000
trn_a1Hsk-333	1.5E+09	2.3052E+15	5/24/2023	20:14:07	Testing	5.8E+08	success	5000
trn_9PG4MvIFl	1.5E+09	2.3052E+15	5/24/2023	20:14:07	Testing	5.8E+08	success	5000

Figure 13. Static report transaction value

3.3. System testing

3.2.1. User acceptance testing

UAT was conducted to test the features by end users without knowledge of the underlying program. This testing will involve 7 participants, including 3 merchants, 2 tester from company staffs, and 2 testers from the bank acquirer. In the Table 2 are the scores represent the user acceptance level. Meanwhile, the Table 3 represents the results of the UAT conducted.

Table 2. User acceptance level score

Score	User acceptance level
1	Totally disagree
2	Disagree
3	Doubfull
4	Agree
5	Strongly agree

Table 3. User acceptance testing

Feature	Totally disagree	Disagree	Doubfull	Agree	Strongly agree	Total score (%)
Staff and merchant						
Making a payment	-	-	-	2	3	92
Performing user credit card verification	-	-	1	1	3	88
Performing transaction verification	-	-	-	1	4	96
Connection with the acquiring bank is established using jwt or signature	-	-	-	2	3	92
Staff, merchant, and bank acquirer						
Checking transaction details on the website.	-	-	1	2	4	88,57
Checking the status of transactions, both successful and failed.	-	-	-	3	4	91,42

From the results in Table 3, it can be seen that all of the features got a decent score and reflects the user acceptance of all of those features. The evaluation results of the UAT indicated that all of the 6 features were successfully executed for the merchant section using JWT and signature. The company and merchant reported 6 successful features for their side. The acquirer bank testing showed that 2 features could be properly executed. The UAT process achieved a 91,33% success rate based on the calculated percentages.

3.2.2. Load testing

In this step, testing was conducted with a range of 20 to 100 transactions per second (TPS) to evaluate the speed and capability of the system in handling multiple transactions concurrently. The server system specifications in development server for running this payment gateway consist of 4 cores, 8 GB random access memory (RAM), and a 100 GB hard disk drive (HDD). Assuming that the transaction processing time limit is influenced by the processor in server development, it is considered good if transactions are completed within 2 to 6 seconds [29]. If the transaction exceeds 6 seconds, it is considered poor performance. The apache JMeter is considered appropriate for those kind of tests [30]–[32]. In the Table 4, the results of testing using apache JMeter are presented in development server, showing the time range of transactions. In the Table 5, the results of testing using JMeter are presented in production server, showing the time range of transactions.

Table 4. Load testing in development server

Sample	Avarage	Minimal	Maximal	Throughput	Result
10	3213	1772	7506	1.3/sec	Good performance
20	3662	2268	4762	4.2/sec	Good performance
30	3732	1832	5398	5.5/sec	Good performance
40	5713	2625	8144	4.9/sec	Good performance
50	6868	2234	11070	4.3/sec	Poor performance
60	4962	1795	8169	7.3/sec	Good performance
70	6202	1837	10640	6.5/sec	Poor performance
80	6947	2579	11672	6.8/sec	Poor performance
90	8343	3281	13782	6.5/sec	Poor performance
100	6493	2047	12235	8.1/sec	Poor performance

Table 5. Load testing in production server

Sample	Avarage	Minimal	Maximal	Throughput	Result
20	552.26	262	824	32.84/sec	Good performance
50	1917.34	1432	2388	20.34/sec	Good performance
100	3297.95	2271	4026	24.76/sec	Poor performance

In this load testing experiment, it will display samples, average, minimum, maximum, throughput, and results. “Sample” represents the number of transactions conducted, “average” indicates the average transaction time in milliseconds, “minimum” represents the fastest transaction time, and “maximum” indicates the longest transaction time. “Throughput” shows how many transactions can be performed per second. “Result” represents the evaluation or assessment.

In terms of load testing on the development server, with a minimum transaction time of 1.77 seconds and a maximum of 4.7 seconds, the average transaction time was 3.7 seconds for 20 transactions. For up to 50 transactions, the transaction times were within a good range, with a minimum of 2.23 seconds and a maximum of 11.07 seconds, averaging at 6.5 seconds. However, when running 100 simultaneous transactions, the transaction times varied between a minimum of 2.04 seconds and a maximum of 12.23

seconds, with an average of 6.5 seconds for load testing on the production server, transaction times were good for 20 to 50 transactions, ranging from 0.5 to 2.4 seconds. However, when running 100 simultaneous transactions, the transaction speed decreased, with transaction times ranging from 2.2 to 4 seconds and an average of 3.3 seconds.

4. CONCLUSION

In this study, the researcher focuses on developing a payment gateway system for a company that serves as a system for credit card payments. The system acts as an intermediary between merchants with digital wallets and banks, involving the company, merchants, and banks in its development. The system includes essential features like bill payment, user credit card verification, and transaction checking, customized to meet the specific requirements of merchants and adhere to the company's security standards. The security measures include authorization during request transmission and the utilization of AES and RSA algorithms for encrypting user PAN. The choice between JWT and signature for authorization is determined through discussions between the merchants and the company.

For user acceptance testing, it achieved a good score of 91.33%. As for load testing in both development and production environments, it can be considered to have good performance when handling 50-60 transactions. In terms of security, the system is considered secure as the user's PAN is protected using AES and RSA algorithm. The outcome of this payment gateway development serves as valuable input for the company in building their own system, providing insights into algorithm implementation techniques and the system workflow. We elaborate on several key points as follows. From an academic perspective, it assists in the aspect of sequencing steps in system development, including the selection of methods used and the stages of creating unified modeling language (UML) as a guide. In a practical context, it greatly aids in the logical aspect of program development, as well as a deep understanding of the optimal system to be implemented in building the payment gateway. Overall, this research outcome also contributes to providing an overview for companies collaborating with each other regarding the developed payment gateway.

For future research, we recommend further exploration in the development of a payment gateway system by expanding the system to accommodate debit payments in addition to credit card payments. Additionally, we suggest developing the payment gateway system using alternative methods, such as different algorithms and different programming languages. The security and transaction processing speed of the payment gateway system should also be explored in further research.

ACKNOWLEDGEMENTS

This study was conducted effectively due to the assistance provided by Multimedia Nusantara University, colleagues, relatives, and the supervising professor who offered valuable feedback and guidance throughout the research. We express our gratitude for the support and aid received during the writing of this article.




REFERENCES

- [1] C. M. Annur, "Bank Indonesia: Transaksi Uang Elektronik RI Tembus Rp 35 Triliun per Desember 2021," [Online]. Available: <https://databoks.katadata.co.id/datapublish/2022/01/31/bank-indonesia-transaksi-uang-elektronik-ri-tembus-rp-35-triliun-per-desember-2021>. (accessed, Jan. 31, 2022).
- [2] S. I. Wahjono, A. Marina, and T. Kurniawati, *Crowdfunding to Fund SMEs and Start-Up Businesses*. Syiah Kuala University Press, Banda Aceh, 2021.
- [3] V. P. Gulati and S. Srivastava, "The empowered internet payment gateway," *International Conference on E-Governance*, pp. 98-107, 2007.
- [4] R. Nagasubramanian and S. P. Rajagopalan, "Payment Gateway Innovation in Multiple Payments," *International Journal of Computer Applications*, vol. 59, no. 16, pp. 33-43, 2012, doi: 10.5120/9635-4391.
- [5] Y. Liu, S. Tang, R. Liu, L. Zhang, and Z. Ma, "Secure and robust digital image watermarking scheme using logistic and RSA encryption," *Expert Systems with Applications*, vol. 97, pp. 95-105, 2018, doi: 10.1016/j.eswa.2017.12.003.
- [6] M. A. Hassan and Z. Shukur, "Device identity-based user authentication on electronic payment system for secure e-wallet apps," *Electronics (Switzerland)*, vol. 11, no. 1, p. 4, 2022, doi: 10.3390/electronics11010004.
- [7] F. Wang, N. Yang, P. M. Shakeel, and V. Saravanan, "Machine learning for mobile network payment security evaluation system," *Transactions on Emerging Telecommunications Technologies*, 2021, doi: 10.1002/ett.4226.
- [8] K. Z. Oo, "Design and Implementation of Electronic Payment Gateway for Secure Online Payment System the Creative Commons Attribution License (CC BY 4.0)," *International Journal of Trend in Scientific Research and Development*, vol. 3, no. 5, pp. 1329-1334, 2019.
- [9] M. A. Hassan, Z. Shukur, and M. K. Hasan, "An efficient secure electronic payment system for e-commerce," *Computers*, vol. 9, no. 3, pp. 1-13, 2020, doi: 10.3390/computers9030066.
- [10] T. Hidayat and R. Mahardiko, "A Systematic Literature Review Method On AES Algorithm for Data Sharing Encryption On Cloud Computing," *International Journal of Artificial Intelligence Research*, vol. 4, no. 1, 2020, doi: 10.29099/ijair.v4i1.154.





- [11] R. O. Akinyede and O. A. Esese, "Development of a Secure Mobile E-Banking System," *International Journal of Computer*, vol. 26, no. 1, pp. 23–42, 2017.
- [12] Y. Zhou, B. Hu, Y. Zhang, and W. Cai, "Implementation of Cryptographic Algorithm in Dynamic QR Code Payment System and Its Performance," *IEEE Access*, vol. 9, pp. 122362–122372, 2021, doi: 10.1109/ACCESS.2021.3108189.
- [13] S. Bojjagani and V. N. Sastry, "A secure end-to-end proximity NFC-based mobile payment protocol," *Computer Standards and Interfaces*, vol. 66, p. 103348, 2019, doi: 10.1016/j.csi.2019.04.007.
- [14] S. Bobba, "Enhancing the Security of Online Card Payment System," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 2, pp. 2055–2059, 2020, doi: 10.30534/ijatcse/2020/178922020.
- [15] K. Bernsmed, D. S. Cruzes, M. G. Jaatun, and M. Iovan, "Adopting threat modelling in agile software development projects," *Journal of Systems and Software*, vol. 183, p. 111090, 2022, doi: 10.1016/j.jss.2021.111090.
- [16] F. Hussain, W. Li, B. Noye, S. Sharieh, and A. Ferworn, "Intelligent Service Mesh Framework for API Security and Management," in *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference*, IEEE, pp. 735–742, 2019, doi: 10.1109/IEMCON.2019.8936216.
- [17] A. Akbulut and H. G. Perros, "Software Versioning with Microservices through the API Gateway Design Pattern," in *Proceedings-International Conference on Advanced Computer Information Technologies*, IEEE, pp. 289–292, 2019, doi: 10.1109/ACITT.2019.8779952.
- [18] N. K. D. Sabrina, D. Pramana, and T. M. Kusuma, "Implementation of Golang and ReactJS in the COVID-19 Vaccination Reservation System," *ADI Journal on Recent Innovation*, vol. 5, no. 1, pp. 1–12, 2023, doi: 10.34306/ajri.v5i1.877.
- [19] A. W. Subagio and F. Subagio, "Implementation of Clean Architecture in the Development of Online Bank Payment Point Systems," *Tekno: Jurnal Teknologi Elektro dan Kejuruan*, vol. 32, no. 2, pp. 324–333, 2022.
- [20] D. A. B. Prasetyo, "Implementasi Information Schema Database Pada PostgreSQL untuk Pembuatan Tabel Informasi dengan Menggunakan Python Di PT XYZ," *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 9, no. 3, pp. 1961–1972, 2022, doi: 10.35957/jatisi.v9i3.2221.
- [21] N. Kathare, O. V. Reddy, and V. Prabhu, "A Comprehensive Study of Elastic Search," *Journal of Research in Science and Engineering*, vol. 4, no. 11, pp. 34–38, 2022, doi: 10.53469/jrse.2022.04(11).07.
- [22] P. Bourhis, J. L. Reutter, and D. Vrgoč, "JSON: Data model and query languages," *Information Systems*, vol. 89, p. 101478, 2020, doi: 10.1016/j.is.2019.101478.
- [23] D. Dykstra, M. Altunay, and J. Teheran, "Secure Command Line Solution for Token-based Authentication," *EPJ Web of Conferences*, vol. 251, p. 02036, 2021, doi: 10.1051/epjconf/202125102036.
- [24] N. Jha and R. Popli, "Comparative Analysis of Web Applications using JMeter," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 3, pp. 774–777, 2017, doi: 10.26483/ijarcs.v8i3.3095.
- [25] J. Zhang, N. Wu, J. Li, and F. Zhou, "A novel differential fault analysis using two-byte fault model on AES Key schedule," *IET Circuits, Devices, and Systems*, vol. 13, no. 5, pp. 661–666, 2019, doi: 10.1049/iet-cds.2018.5428.
- [26] A. Abdullah, "Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data," *Cryptogr. Netw. Secur.*, vol. 16, no. 11, 2017.
- [27] K. Eachová and P. Trebuña, "Modelling of electronic kanban system by using of entity relationship diagrams," *Acta Logistica*, vol. 6, no. 3, pp. 63–66, 2019, doi: 10.22306/al.v6i3.115.
- [28] D. Rachmawati and M. A. Budiman, "On Using the First Variant of Dependent RSA Encryption Scheme to Secure Text: A Tutorial," *Journal of Physics: Conference Series*, vol. 1542, no. 1, p. 012024, 2020, doi: 10.1088/1742-6596/1542/1/012024.
- [29] X. S. Ha, T. H. Le, T. T. Phan, H. H. D. Nguyen, H. K. Vo, and N. Duong-Trung, "Scrutinizing Trust and Transparency in Cash on Delivery Systems," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 214–227, 2021, doi: 10.1007/978-3-030-68851-6_15.
- [30] E. Alam and F. Dewi, "Performance Testing Analysis of Bandungtanginas Application With Jmeter," *International Journal of Innovation in Enterprise System*, vol. 6, no. 01, pp. 85–94, 2022.
- [31] M. A. R. Maniyar, M. A. Hakeem, M. Khalid, and M. Zafar, "Bottom-up Approach for Performance Testing of Software Applications or Products," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 6, no. 9, pp. 7812–7817, 2018.
- [32] Y. Cong, B. Wang, and T. Du, "Simulation Experiment and Teaching Research of a Land- Based Ship Engine Room," *International Journal of Information and Communication Technology Education*, vol. 19, no. 1, pp. 1–9, 2023, doi: 10.4018/IJICTE.331801.

BIOGRAPHIES OF AUTHORS







Veronica    is an accomplished professional who holds an undergraduate degree from Universitas Multimedia Nusantara, specializing in System Information. With her expertise as a backend developer in the payment gateway industry, Veronica has honed her skills in designing and maintaining secure server-side logic. Her in-depth knowledge of system information enables her to optimize performance and create innovative solutions, making her a valuable asset in the world of backend development. She can be contacted at email: veronica4@student.umn.ac.id.



Raymond Sunardi Oetama     is a lecturer specializing in Mathematics and Computer and Information Science. He completed his bachelor's degree (S1) in Mathematics at Institut Teknologi Sepuluh Nopember in 1992. Recognizing the importance of further education, he pursued his master's degree (S2) at Auckland University of Technology, graduating in 2007. His contributions to education and research enhance his standing within the field. He can be contacted at email: raymond@umn.ac.id.



Arief Ramadhan     is an Assistant Professor at School of Computing, Telkom University, Bandung, Indonesia. He is also a Senior Member of IEEE. His areas of research focus include gamification, metaverse, information technology, information systems, enterprise architecture, e-business, data analytics, e-learning, e-tourism, e-government, and business intelligence. Currently, he is supervising several doctoral students in those field of study. He can be contacted at email: arieframadhan@telkomuniversity.ac.id.