

Comparative Analysis of ElGamal and RSA Methods for Digital Signature Implementation in the Integrated Service Unit Application of Universitas Malikussaleh

Helmi Naluri ^{1*}, Nurdin^{2*}, Munirul Ula ^{3*}

*Departement of Information Technology, Universitas Malikussaleh, Lhokseumawe, Indonesia
nurdin@unimal.ac.id²

Article Info

Article history:

Received 2026-08-09

Revised 2026-09-11

Accepted 2026-10-09

Keyword:

*Digital Signature,
RSA,
ElGamal,
Performance Evaluation,
Public-Key Cryptography.*

ABSTRACT

The digital transformation of administrative services in higher education requires systems capable of delivering services that are fast, responsive, and efficient. Universitas Malikussaleh has implemented the Integrated Service Unit (ULT) application as an electronic administrative service platform. The implementation of digital signatures in this system requires the selection of an algorithm that can ensure document authenticity while maintaining efficient computational performance. This study aims to implement and compare the performance of the Rivest–Shamir–Adleman (RSA) and ElGamal algorithms in a digital signature module integrated with the Universitas Malikussaleh ULT application. The main contribution of this research is the integration of RSA- and ElGamal-based digital signature modules into the ULT application and the evaluation of their performance in terms of computational time, CPU utilization, and memory consumption. The system was implemented using Python 3 and the Flask 3 framework. Performance evaluation was conducted using PDF documents with sizes of 500 KB, 1 MB, and 5 MB. Before the signing and verification processes, each document was processed using the SHA-256 hash algorithm to generate a message digest. Performance measurements were conducted over 30 iterations for the key generation, signing, and verification processes. The collected data were analyzed using descriptive statistics, the Shapiro–Wilk normality test, and the Mann–Whitney U test. The experimental results indicate that RSA achieves better performance in the key generation and document verification processes, whereas ElGamal outperforms RSA during the document signing process. Furthermore, the Mann–Whitney U test revealed statistically significant differences between RSA and ElGamal for most of the evaluated performance metrics ($p < 0.05$). Based on these findings, RSA is recommended for deployment in the ULT system because it provides superior verification performance and is better suited to the operational requirements of administrative services that prioritize the verification of electronic document



This is an open access article under the [CC-BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.

I. INTRODUCTION

The development of information and communication technology has driven the transformation of administrative services into electronic-based systems in various sectors, including higher education [1],[2]. Malikussaleh University has implemented the Integrated Service Unit (ULT) system as

an electronic-based administrative service medium that allows academics and the general public to submit various services online and receive the results in the form of electronic documents. Implementing this system simplifies the administrative process, but also requires a mechanism to guarantee the security, authenticity, and integrity of the resulting electronic documents.

In the ULT service process, service-generated documents are generally issued in PDF format. However, these documents are not equipped with a digital signature mechanism, thus still being susceptible to forgery or alteration of their content after publication. Digital signatures have been proven to guarantee the authentication, authenticity, and integrity of electronic documents, making them a crucial security mechanism in electronic-based administrative systems [2]. In addition to security aspects, digital signature implementation also requires consideration of performance aspects because the processes of key generation, signing, and document verification require computational time and use system resources, which can impact service response speed. This is crucial considering that user satisfaction and service effectiveness are indicators of the success of information system implementation in higher education [3].

A digital signature is a cryptographic mechanism used to guarantee the authentication and integrity of electronic documents [4] [5] [6]. The RSA and ElGamal algorithms are two public-key cryptography algorithms that have been widely used in various research and implementations of information security systems. RSA has a high level of implementation maturity and is based on the large integer factorization problem, while ElGamal is based on the discrete logarithm problem [4][7]. In digital signature implementations, documents are first processed using a hash function to generate a message digest before the signing and verification processes are performed. In this study, the SHA-256 algorithm was used as the hash function because it is capable of producing a fixed-size message digest and supports the efficiency of the digital signature process [8], [9] [10].

Several previous studies have compared the RSA and ElGamal algorithms in cryptography and digital signature applications [11], [12], [13], [14]. However, most of these studies were conducted in simulation environments or stand-alone applications and therefore do not fully represent implementations in real-world administrative service systems [5], [15], [16]. Furthermore, some studies focused more on security aspects or only evaluated computation time [17], [13], [18], without considering CPU usage and memory consumption as indicators of algorithm efficiency. Previous research has generally not conducted a comprehensive performance evaluation of the key generation, signing, and verification processes in a real-world system environment [14], [15].

The Malikussaleh University Integrated Service Unit (ULT) system serves various online academic and non-academic administrative needs, requiring a digital signature algorithm that not only guarantees the authenticity of electronic documents but also offers efficient performance. Previous research using the PIECES method has shown that structured system performance measurements can provide an overview of the system's ability to meet user needs [19]. Therefore, the selection of a digital signature algorithm must consider the efficiency of computation time and system resource usage to ensure responsive and optimal service quality.

Based on these conditions, this study implemented an RSA- and ElGamal-based digital signature module integrated with the Malikussaleh University Integrated Service Unit (ULT) system. The key generation, signing, and verification processes were evaluated using parameters such as computation time, CPU usage, and memory consumption. The SHA-256 algorithm is implemented identically in both algorithms as the message digest generation mechanism and therefore is not a variable in the comparison. Furthermore, the differences in performance of the two algorithms were analyzed using descriptive statistics, the Shapiro–Wilk normality test, and the Mann–Whitney test. The results of this study are expected to contribute to the evaluation of the performance of digital signature algorithms and become the basis for recommendations for selecting the most appropriate algorithm to be implemented in the Integrated Service Unit (ULT) system at Malikussaleh University [19].

II. METHODOLOGY

A. Research Procedure

This research was conducted systematically to develop and evaluate a digital signature module based on the RSA and ElGamal algorithms integrated with the Integrated Service Unit (ULT) application of Malikussaleh University. The developed module implements the RSA and ElGamal algorithms as digital signature algorithms and SHA-256 as a hash function to generate message digests. Furthermore, a performance evaluation was conducted based on computation time, CPU usage, and memory consumption to determine the most suitable algorithm to be implemented in the electronic administration service system. The research stages include literature review, system requirements analysis, system development (implementation and integration), performance evaluation, statistical analysis, performance comparison, and conclusion. All research stages are shown in Figure 1.

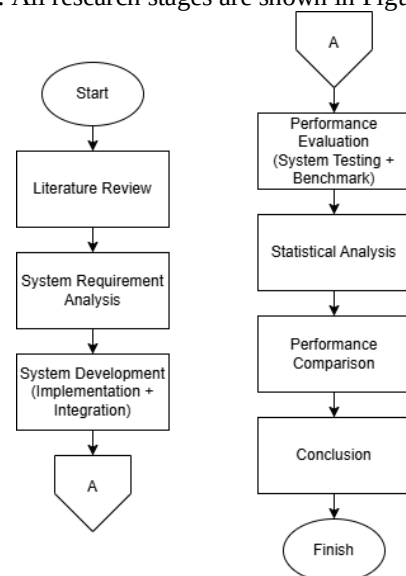


Figure 1. Research Procedure

Figure 1 shows the research flow for this study. The process began with a literature review to obtain a theoretical foundation for cryptography, digital signatures, the RSA, ElGamal, and SHA-256 algorithms, as well as a system performance evaluation. Next, a system requirements analysis was conducted to identify the need for implementing a digital signature module in the Integrated Service Unit (ULT) application.

The next stage was system development, which included implementing the digital signature module using the Python programming language and the Flask framework, as well as integrating the module with the ULT application through the Representational State Transfer Application Programming Interface (REST API). The module implemented the RSA and ElGamal algorithms as digital signature algorithms, and SHA-256 as a hash function to support key generation, document signing, and signature verification.

After the development process was completed, a performance evaluation was conducted through system testing using PDF documents of various file sizes. Testing was conducted on key generation, signing, and verification processes, along with measurements of computation time, CPU usage, and memory consumption as performance evaluation parameters. The test data were then analyzed using descriptive statistics, the Shapiro–Wilk normality test, and the Mann–Whitney test to determine the significance of the performance differences between the RSA and ElGamal algorithms.

The next stage was a performance comparison, which compared the performance of the two algorithms based on all measured parameters. The results of this comparison were used to formulate conclusions and provide recommendations for the most appropriate algorithm for implementation in the Malikussaleh University Integrated Service Unit (ULT) system.

B. System Architecture

The system architecture developed in this study is designed to integrate the digital signature module with the Integrated Service Unit (ULT) application of Malikussaleh University through the Representational State Transfer Application Programming Interface (REST API). This architecture allows the ULT application to utilize the digital signature service without changing the main mechanisms of the existing system. An overview of the system architecture is shown in Figure 2.

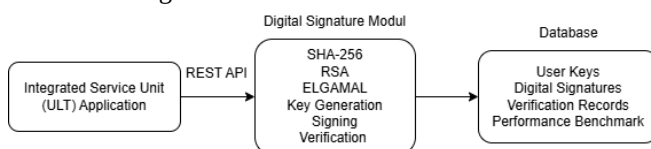


Figure 2. System Architecture

Based on this architecture, the ULT application acts as a client, sending requests to the digital signature module via a REST API. The digital signature module is developed using the Python programming language with the Flask framework

and implements the SHA-256, RSA, and ElGamal algorithms to support key generation, document signing, and digital signature verification. All cryptographic processes are executed within this module, allowing the ULT application to function solely as a service user.

All data generated during the operational process is stored in a database, including user key pairs, digital signature data, verification results, and benchmark data used to evaluate algorithm performance. Storing data in the database allows each signing and verification process to be centrally managed while supporting the performance measurement process conducted in this study.

This architecture separates the ULT application's business processes from the cryptographic processes, allowing the implementation of the digital signature module to be carried out independently without impacting the application's core functionality. This approach also simplifies the development, maintenance, and integration of digital signature algorithms into electronic administration service systems.

C. Digital Signature Scheme and Cryptographic Configuration

1) Digital Signature Workflow

The digital signature workflow comprises three main stages: message digest generation, digital signature generation, and digital signature verification[10]. During the signing process, the electronic document is first read as binary data and processed using SHA-256 to produce a fixed-length message digest. This generated message digest then serves as input for the signature generation process using either an RSA or ElGamal private key.

Once the message digest generation stage is complete, a benchmarking mechanism is activated prior to the execution of the cryptographic signing function. This benchmark records execution time, CPU usage, and memory consumption during the RSA or ElGamal signature generation process. Consequently, the recorded performance metrics reflect only the computational load of the digital signature operation, excluding the document reading and SHA-256 hashing processes.

The verification process follows a similar mechanism. The received electronic document is processed using SHA-256 to regenerate the message digest, which is then used as input for signature verification using the corresponding public key. A benchmarking mechanism is initiated before the RSA or ElGamal verification function runs to measure the computational performance of the verification process. As with the signing stage, the measured verification performance excludes the overhead associated with document processing and message digest generation.

The overall workflow ensures that both RSA and ElGamal are evaluated under identical conditions. Comparisons are conducted using identical document inputs, hash functions, security parameters, and benchmarking procedures, allowing observed performance differences to be attributed to the specific characteristics of each digital signature algorithm.

2) SHA-256 Hash Implementation

In this study, SHA-256 is implemented as the cryptographic hash function used to generate a message digest from an electronic document prior to the digital signature process[2]. This implementation utilizes the Crypto.Hash.SHA256 module provided by the PyCryptodome library. The PDF document is first converted into binary data, which is then processed by SHA-256 to produce a fixed-length message digest.

The SHA-256 implementation is applied consistently across both RSA and ElGamal digital signature schemes. However, the hash function's output format is adapted to the input requirements of each signature algorithm. For RSA, the SHA-256 hash object generated by the SHA256.new() function is passed directly to the PKCS#1 v1.5 signing and verification functions. This approach allows the RSA implementation to utilize the hash metadata required by the signature scheme.

For ElGamal, the SHA-256 output is obtained using the digest() function, which produces a 32-byte message digest. The resulting digest is then converted into an integer representation via byte conversion before being used in the modular arithmetic operations of the ElGamal signature.

3) RSA Implementation

In this study, the RSA algorithm is implemented as a digital signature scheme using the RSA cryptographic module provided by the PyCryptodome library in Python[6]. The implementation utilizes the Crypto.PublicKey.RSA module for key generation and management, while the Crypto.Signature.pkcs1_15 module is employed for digital signature generation and verification.

RSA key pairs are generated using the RSA.generate() function with a key size of 2048 bits. The generated keys consist of a private key and a public key; the private key is used during the signature generation process, whereas the corresponding public key is used for signature verification.

Prior to the RSA signing process, the electronic document is processed using SHA-256 to generate a message digest. The resulting digest object serves as input for the RSA signature operation. Signature generation is performed using the PKCS#1 v1.5 signature scheme via the pkcs1_15.sign() function. The generated signature is stored alongside the document metadata for subsequent verification purposes.

The verification process is carried out using the RSA public key via the pkcs1_15.verify() function. The recomputed SHA-256 digest of the document is compared against the received signature to determine the document's validity. RSA mathematical operations including modular exponentiation and key computation are handled internally by the cryptographic library.

4) ElGamal Implementation

In this study, the ElGamal algorithm is implemented as a digital signature scheme using modular arithmetic operations in Python and cryptographic utilities provided by the PyCryptodome[7]. The ElGamal key generation process

begins by generating a 2048-bit prime number p using the getPrime() function from the PyCryptodome library. The generator parameter g is set to 2, while the private key x is randomly generated within the range of 2 to $p-2$. The corresponding public key y is calculated using:

$$y = g^x \bmod p$$

The generated private key consists of the parameters (p, g, x) , whereas the public key consists of (p, g, y) . The private key is used during the signature generation process, while the public key parameters are used for signature verification.

Before generating the ElGamal signature, the SHA-256 message digest of the document is converted into an integer representation and reduced modulo $p-1$. The signature generation process produces a random value k that satisfies:

$$\gcd(k, p-1) = 1$$

A digital signature is represented as a pair (r, s) , where:

$$r = g^k \bmod p$$

$$s = k^{-1} (h - xr) \bmod (p-1)$$

During verification, the signature validity is determined by comparing the following values:

$$v_1 = g^h \bmod p$$

$$v_2 = (y^r r^s) \bmod p$$

The signature is considered valid if $v_1 = v_2$. Through this implementation, ElGamal performs signing and verification operations on the SHA-256 message digest, rather than processing the original PDF document directly.

D. Data Collection

This study gathered the data necessary to implement and evaluate the performance of the RSA and ElGamal algorithms within a digital signature module integrated into the Malikussaleh University Integrated Service Unit (ULT) application. The collected data comprised PDF electronic documents, generated key pairs, digital signatures, verification results, and benchmark data derived from the cryptographic processes.

PDF documents were selected as test subjects because they represent the document format used in electronic administrative service workflows. Three document sizes 500 KB, 1 MB, and 5 MB were utilized to assess how document characteristics influence the digital signature workflow. Although the RSA and ElGamal signing and verification operations were performed on fixed-length SHA-256 message digests (rather than the entire document content), document size variations were still considered to analyze potential differences during document processing and hash generation.

For each test scenario, the RSA and ElGamal algorithms were evaluated across three primary processes: key generation, digital signing, and signature verification. SHA-256 hashing was performed prior to the signing and verification stages to

generate the message digests required by both algorithms. Consequently, the execution times measured during signing and verification reflected the computational characteristics of the cryptographic operations themselves, excluding the time required for hashing.

Each experimental scenario was executed 30 times to obtain representative performance data and minimize the impact of measurement variations. The collected benchmark data included execution time, CPU usage, and memory consumption for each cryptographic process. All collected data were stored in a database and served as the basis for statistical analysis and performance comparison between RSA and ElGamal.

E. Experimental Environment

The performance of the RSA and ElGamal algorithms was conducted in a controlled hardware and software environment to ensure consistent and reproducible measurement results. The entire testing process used the same system configuration in each experimental scenario so that the performance differences obtained reflect the characteristics of the algorithms being tested, not influenced by changes in the testing environment. The specifications of the testing environment used in this study are shown in Table II.

TABLE I
EXPERIMENTAL ENVIRONMENT SPECIFICATION

Component	Specification
Operating System	Windows 11 64-bit
Processor	13th Gen Intel® Core™ i5-13420H
CPU Configuration	8 Cores / 12 Threads
Processor Speed	2.10 GHz
RAM	24 GB DDR5
Storage	512 GB SSD
System Architecture	64-bit
Programming Language	Python 3.14.2
ULT Application Framework	CodeIgniter 3
Digital Signature Framework	Flask 3
Database	MySQL
Cryptographic Library	PyCryptodome
Time Measurement Library	time (perf_counter)
CPU and Memory Monitoring	psutil

The testing environment was used consistently throughout the key generation, signing, and verification processes. All computational time measurements were performed using the `perf_counter` function from the time library, while CPU usage and memory consumption were monitored using the `psutil` library. The use of a uniform environment ensures that the performance evaluation results objectively reflect the characteristics of the RSA and ElGamal algorithms.

F. Data Analysis and Processing

Data obtained from the testing process was analyzed to evaluate the performance differences between the RSA and ElGamal algorithms in implementing digital signatures in the Integrated Service Unit (ULT) application at Malikussaleh University. The analysis was conducted on benchmark data generated during the key generation, signing, and verification processes. The analysis stages included a data collection scheme, determining performance parameters, processing data using statistical methods, and presenting the results in tables, graphs, and discussions.

1) Data Acquisition Scheme

In this study, data was obtained through a digital signature module testing process integrated with the Integrated Service Unit (ULT) application. Testing was conducted using the RSA and ElGamal algorithms across three main processes: key generation, signing, and verification. Each test scenario was run 30 times using PDF documents of 500 KB, 1 MB, and 5 MB sizes to obtain consistent and representative data.

2) Performance Metrics

Performance evaluation was conducted using three main parameters: computation time, CPU usage, and memory consumption. These three parameters were chosen because they can describe the algorithm's efficiency in terms of processing speed and system resource usage during key generation, signing, and verification.

TABLE II
PERFORMANCE EVALUATION METRICS

Metric	Description
Execution Time	The time required to complete each process (key generation, signing, atau verification).
CPU Utilization	CPU usage percentage during the process
Memory Consumption	The amount of memory used during the process.

3) Data Processing

The test data were processed using descriptive statistics to obtain the mean and standard deviation for each performance parameter. Furthermore, a normality test was performed using the Shapiro–Wilk method to determine the data distribution. Based on the normality test results, an analysis of performance differences between the RSA and ElGamal algorithms was performed using the Mann–Whitney U Test due to the non-normal distribution of the data [18], [19].

4) Data Presentation

The analysis results are presented in tables and graphs to facilitate interpretation of the performance differences between the RSA and ElGamal algorithms. Data are presented for each key generation, signing, and verification process based on parameters such as computation time, CPU usage, and memory consumption. Furthermore, the statistical analysis results are used to evaluate the performance of both algorithms and to determine the most appropriate algorithm for

implementation on the Integrated Service Unit (ULT) system at Malikussaleh University.

III. RESULT AND DISCUSSION

A. Implementation of Digital Signature Module

The digital signature module was successfully implemented and integrated with the Malikussaleh University Integrated Service Unit (ULT) application through the Representational State Transfer Application Programming Interface (REST API). This module was developed using the Python programming language with the Flask framework, while the ULT application was built using the CodeIgniter 3 framework. This integration allows the ULT application to utilize cryptographic services without changing the main mechanisms of the existing system. The implementation of the digital signature module is shown in Figure 3.

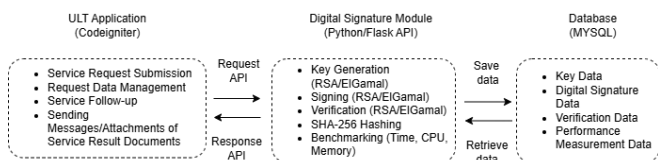


Figure 3. Implementation of the Digital Signature Module Integrated with the ULT Application

Based on Figure 3, the ULT application acts as the primary system that handles administrative service processes, including submitting service requests, managing request data, following up on service processes, and sending service-received documents to users. When a digital signature process is required, the ULT application sends a request to the digital signature module via the REST API. Once the process is complete, the module returns a response to the ULT application so that the signing and verification results can be used in the administrative service flow.

The digital signature module provides key generation, signing, signature verification, and message digest generation using the SHA-256 algorithm. In addition to executing cryptographic processes, the module also records benchmark data in the form of computation time, CPU usage, and memory consumption, which are used for performance evaluation in this study.

All operational data is stored in a MySQL database, including user key pairs, digital signature data, verification results, and performance benchmark data. During the verification process, the module retrieves the necessary data from the database before executing the verification algorithm. This architecture separates the cryptographic functions from the main application, allowing the digital signature module to be developed, maintained, and integrated independently without impacting the business processes in the ULT application.

B. Performance Evaluation of Key Generation

Key generation performance evaluation was conducted to compare the efficiency of the RSA and ElGamal algorithms in generating public and private key pairs. The test was performed 30 times using the same test environment. Parameters analyzed included computation time, CPU usage, and memory consumption. Furthermore, the data was analyzed using descriptive statistics and statistical tests to determine the significance of differences in the performance of the two algorithms.

1) Execution Time

Computation time was measured to evaluate the efficiency of the RSA and ElGamal algorithms in generating key pairs. Measurements were performed 30 times for each algorithm. Test result statistics include minimum, maximum, average, and standard deviation values, as shown in Table III.

TABLE III
EXECUTION TIME STATISTICS OF KEY GENERATION

Algorithm	Minimum (ms)	Maximum (ms)	Mean (ms)	Std Dev (ms)
RSA	387,29	5460,59	1807,86	1126,54
EGAMAL	636,35	24616,56	6446,15	5320,57

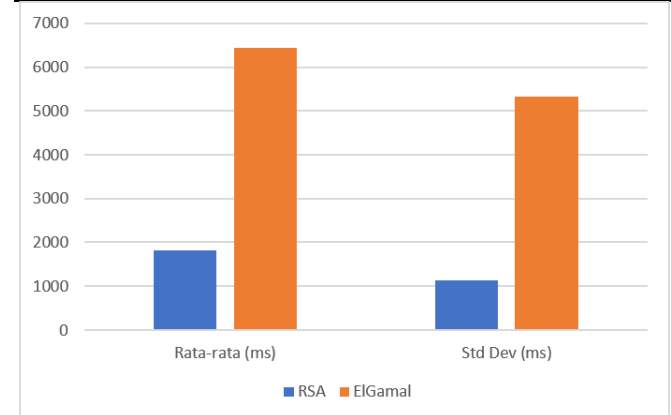


Figure 4. Comparison of Key Generation Execution Time

Based on Table III and Figure 4, the RSA algorithm produces a lower key generation time than ElGamal. The average key generation time for RSA was 1,807.86 ms, while ElGamal required an average of 6,446.15 ms. These results indicate that RSA is capable of generating key pairs faster in the test environment used.

In addition to having a lower average time, RSA also exhibits smaller time variations, with a standard deviation of 1,126.54 ms, while ElGamal has a standard deviation of 5,320.57 ms. This indicates that RSA key generation times tend to be more stable than ElGamal during the testing process.

Before conducting the hypothesis testing, a normality test using the Shapiro–Wilk method was performed to determine the appropriate statistical analysis method. The test results are shown in Table IV.

TABLE IV
SHAPIRO–WILK NORMALITY TEST RESULTS FOR KEY GENERATION
EXECUTION TIME

Metric	Algorithm	p-value	Decision
Time	RSA	0,002	Not Normal
Time	ElGamal	< 0,001	Not Normal

Based on the Shapiro–Wilk test results, the key generation time data for the RSA algorithm has a p-value of 0.002, while ElGamal has a p-value <0.001. Because all p-values are <0.05, the data are not normally distributed. Therefore, the analysis of performance differences continued using the Mann–Whitney test.

The results of the Mann–Whitney test for key generation time are shown in Table V.

TABLE V
MANN–WHITNEY TEST RESULTS FOR KEY GENERATION EXECUTION TIME

Parameter	p-value	Decision
Execution Time	< 0,001	Reject H_0

The Mann–Whitney test results show a p-value <0.001, thus rejecting H_0 . This indicates a significant difference between the key generation times of the RSA and ElGamal algorithms. Based on these results, it can be concluded that the RSA algorithm shows a lower key pair generation time than ElGamal.

2) CPU Utilization

CPU usage was measured to evaluate the level of processor resource utilization during the key pair generation process using the RSA and ElGamal algorithms. Measurements were performed 30 times for each algorithm. Test result statistics include minimum, maximum, average, and standard deviation values, as shown in Table VI.

TABLE VI
CPU UTILIZATION STATISTICS OF KEY GENERATION

Algorithm	Minimum (%)	Maximum (%)	Mean (%)	Std. Dev. (%)
RSA	18.10	37.90	24.35	4.98
ElGamal	17.80	42.30	24.49	4.88

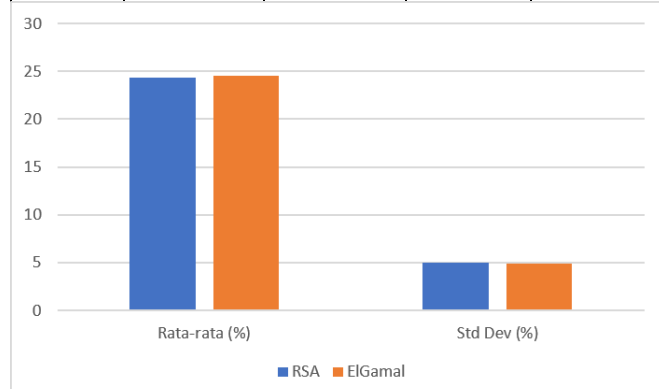


Figure 5. Comparison of Key Generation CPU Utilization

Based on Table VI and Figure 5, the CPU usage of the RSA and ElGamal algorithms shows relatively similar values. The average CPU usage for RSA is 24.35%, while for ElGamal it is 24.49%. The difference in average CPU usage between the two algorithms is only 0.14%, indicating that both algorithms utilize nearly the same level of processor resources during the key generation process.

In addition to having nearly identical average CPU usage, the two algorithms also exhibit relatively similar levels of data variation. RSA has a standard deviation of 4.98%, while ElGamal has a standard deviation of 4.88%. This indicates that the CPU usage of both algorithms tends to be consistent throughout the testing process.

Before testing the hypotheses, a normality test using the Shapiro–Wilk method was performed to determine the appropriate statistical analysis method. The test results are shown in Table VII.

TABLE VII
SHAPIRO–WILK NORMALITY TEST RESULTS FOR KEY GENERATION CPU
UTILIZATION

Metric	Algorithm	p-value	Decision
CPU Utilization	RSA	0.001	Not Normal
CPU Utilization	ElGamal	< 0.001	Not Normal

Based on the Shapiro–Wilk test results, the CPU usage data for the RSA algorithm has a p-value of 0.001, while the ElGamal algorithm has a p-value <0.001. Since all p-values are <0.05, the data are not normally distributed. Therefore, the analysis of performance differences continued using the Mann–Whitney test.

The results of the Mann–Whitney test for CPU usage in the key generation process are shown in Table VIII.

TABLE VIII
MANN–WHITNEY TEST RESULTS FOR KEY GENERATION CPU UTILIZATION

Metric	p-value	Decision
CPU Utilization	0.767	Accept H_0

The Mann–Whitney test results show a p-value of 0.767, thus H_0 is accepted. This indicates that there is no significant difference in CPU usage between the RSA and ElGamal algorithms during the key generation process. Thus, both algorithms exhibit relatively similar levels of processor resource utilization in the testing environment used.

3) Memory Consumption

Memory consumption was measured to evaluate the amount of memory used by the RSA and ElGamal algorithms during the key pair generation process. Measurements were repeated 30 times for each algorithm. Test result statistics include minimum, maximum, average, and standard deviation values, as shown in Table IX.

TABLE IX
MEMORY CONSUMPTION STATISTICS OF KEY GENERATION

Algorithm	Minimum (Byte)	Maximum (Byte)	Mean (Byte)	Std. Dev. (Byte)
RSA	0	700,416	25,122.13	127,622.39
ElGamal	0	32,768	2,321.07	6,860.95

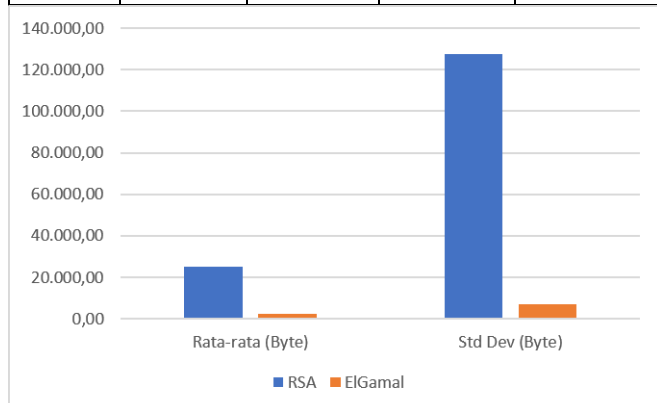


Figure 6. Comparison of Key Generation Memory Consumption

Based on Table IX and Figure 6, the ElGamal algorithm shows lower average memory consumption than RSA. ElGamal's average memory usage is 2,321.07 bytes, while RSA uses an average of 25,122.13 bytes. Descriptively, these results indicate that ElGamal is more efficient in memory usage during the key generation process.

In addition to having a higher average memory consumption, RSA also shows significantly greater variation in memory usage than ElGamal. The standard deviation for RSA is 127,622.39 bytes, while ElGamal's is only 6,860.95 bytes. This difference indicates that RSA's memory usage varies more during the testing process, while ElGamal tends to be more consistent.

Before testing the hypotheses, a normality test using the Shapiro–Wilk method was performed to determine the appropriate statistical analysis method. The test results are shown in Table X.

TABLE X
SHAPIRO–WILK NORMALITY TEST RESULTS FOR KEY GENERATION MEMORY CONSUMPTION

Metric	Algorithm	p-value	Decision
Memory Consumption	RSA	< 0.001	Not Normal
Memory Consumption	ElGamal	< 0.001	Not Normal

Based on the Shapiro–Wilk test results, the memory consumption data for both the RSA and ElGamal algorithms have p-values < 0.001. Since all p-values are < 0.05, the data are not normally distributed. Therefore, the analysis of performance differences continued using the Mann–Whitney test.

The results of the Mann–Whitney test for memory consumption in the key generation process are shown in Table XI.

TABLE XI
MANN–WHITNEY TEST RESULTS FOR KEY GENERATION MEMORY CONSUMPTION

Metric	p-value	Decision
Memory Consumption	0.546	Accept H_0

The Mann–Whitney test results show a p-value of 0.546, thus H_0 is accepted. This indicates that there is no significant difference in memory consumption between the RSA and ElGamal algorithms during the key generation process.

C. Performance Evaluation of Document Signing

A performance evaluation of the document signing process was conducted to compare the efficiency of the RSA and ElGamal algorithms in generating digital signatures on electronic documents. The test was repeated 30 times using PDF documents of 500 KB, 1 MB, and 5 MB in the same test environment to obtain consistent and representative results. Parameters evaluated included computation time, CPU usage, and memory consumption. The test results were then analyzed using descriptive statistics and statistical tests to determine the significance of the performance differences between the two algorithms.

1) Execution Time

Computation time was measured to evaluate the efficiency of the RSA and ElGamal algorithms in generating digital signatures on electronic documents. Measurements were repeated 30 times for each document size: 500 KB, 1 MB, and 5 MB. Test result statistics include minimum, maximum, average, and standard deviation values, as shown in Table XII.

TABLE XII
EXECUTION TIME STATISTICS OF DOCUMENT SIGNING

Document Size	Algorithm	Min (ms)	Max (ms)	Mean (ms)	Std. Dev. (ms)
500 KB	RSA	57.46	511.90	126.26	86.58
	ElGamal	54.23	65.15	59.95	2.66
1 MB	RSA	107.81	131.36	122.13	6.18
	ElGamal	54.48	65.58	59.13	3.16
5 MB	RSA	100.05	139.27	121.04	8.12
	ElGamal	52.09	61.04	56.79	2.31

Based on Table XII, the ElGamal algorithm consistently produces lower signing times than RSA across all tested document sizes. On a 500 KB document, the average RSA signing time is 126.26 ms, while ElGamal only requires 59.95 ms. Similar results are also obtained on a 1 MB document, with an average RSA time of 122.13 ms and ElGamal of 59.13 ms, and on a 5 MB document, where RSA requires an average of 121.04 ms, while ElGamal requires 56.79 ms.

In addition to having a lower average time, the ElGamal algorithm also exhibited smaller time variations than RSA across all document sizes. This is demonstrated by the ElGamal standard deviation, which ranged from 2.31–3.16 ms, while RSA had a larger standard deviation, particularly for a

500 KB document, which reached 86.58 ms. These results indicate that the signing process using ElGamal tends to be more consistent than RSA.

Before testing the hypothesis, a normality test was performed using the Shapiro–Wilk method on all test data (90 samples for each algorithm). The test results showed that the RSA data were not normally distributed ($p < 0.001$), while the ElGamal data were normally distributed ($p = 0.512$). Because one of the data sets did not meet the assumption of normality, the analysis of performance differences continued using the Mann–Whitney test.

TABLE XIII
MANN–WHITNEY TEST RESULTS FOR DOCUMENT SIGNING EXECUTION TIME

Metric	p-value	Decision
Execution Time	< 0.001	Reject H_0

The Mann–Whitney test results show a p-value < 0.001 , so H_0 is rejected. This indicates that there is a significant difference between the computational time of the RSA and ElGamal algorithms in the document signing process. Based on the results of descriptive statistics and the Mann–Whitney test, it can be concluded that the ElGamal algorithm has a significantly lower signing time than RSA for all tested document sizes.

2) CPU Utilization

CPU usage was measured to evaluate the level of processor resource utilization during the document signing process using the RSA and ElGamal algorithms. Measurements were repeated 30 times for each document size: 500 KB, 1 MB, and 5 MB. Test result statistics include minimum, maximum, average, and standard deviation values, as shown in Table XIV.

TABLE XIV
CPU UTILIZATION STATISTICS OF DOCUMENT SIGNING

Document Size	Algorithm	Min (%)	Max (%)	Mean (%)	Std. Dev. (%)
500 KB	RSA	24.00	66.70	37.11	10.88
	ElGamal	14.00	37.20	26.10	5.55
1 MB	RSA	16.70	27.20	20.09	2.72
	ElGamal	14.60	43.20	22.16	6.56
5 MB	RSA	15.80	36.80	23.75	4.72
	ElGamal	20.50	39.00	27.71	4.61

Based on Table XIV, CPU usage in the document signing process shows significant variation across document sizes. For a 500 KB document, the RSA algorithm had an average CPU usage of 37.11%, higher than ElGamal's 26.10%. Conversely, for 1 MB and 5 MB documents, ElGamal showed slightly higher average CPU usage than RSA, at 22.16% and 27.71%, respectively, compared to 20.09% and 23.75% for RSA.

The differences in CPU usage between the algorithms did not show a consistent pattern with changes in document size.

However, the standard deviations for both algorithms were relatively small, except for RSA for a 500 KB document, which showed a greater variation in CPU usage than the other tests.

Before testing the hypotheses, a normality test using the Shapiro–Wilk method was performed on all test data (90 samples for each algorithm). The test results showed that the CPU usage data for the RSA ($p < 0.001$) and ElGamal ($p = 0.017$) algorithms were not normally distributed. Therefore, the analysis of performance differences was continued using the Mann–Whitney test.

TABLE XV
MANN–WHITNEY TEST RESULTS FOR DOCUMENT SIGNING CPU UTILIZATION

Metric	p-value	Decision
CPU Utilization	< 0.001	Reject H_0

The Mann–Whitney test results show a p-value < 0.001 , so H_0 is rejected. This indicates that there is a significant difference in CPU usage between the RSA and ElGamal algorithms during the document signing process. Although the average CPU usage varies with document size and does not indicate a consistently lower algorithm, the statistical test results indicate that the overall difference in CPU usage between the two algorithms is significant during the document signing process.

3) Memory Consumption

Memory consumption was measured to evaluate the amount of memory used by the RSA and ElGamal algorithms during the document signing process. Measurements were repeated 30 times for each document size: 500 KB, 1 MB, and 5 MB. Test result statistics include minimum, maximum, average, and standard deviation values, as shown in Table XVI.

TABLE XVI
MEMORY CONSUMPTION STATISTICS OF DOCUMENT SIGNING

Document Size	Algorithm	Min (Byte)	Max (Byte)	Mean (Byte)	Std. Dev. (Byte)
500 KB	RSA	0	524,288	20,480.00	95,909.03
	ElGamal	0	4,096	136.53	747.82
1 MB	RSA	0	610,304	27,989.33	110,683.94
	ElGamal	0	28,672	1,501.87	5,430.05
5 MB	RSA	0	671,744	27,033.60	121,880.66
	ElGamal	0	28,672	1,365.33	5,306.12

Based on Table XVI, the ElGamal algorithm consistently shows lower memory consumption than RSA across all tested document sizes. For a 500 KB document, RSA's average memory consumption is 20,480.00 bytes, compared to only 136.53 bytes for ElGamal. A similar difference is also seen for a 1 MB document, where RSA uses an average of 27,989.33 bytes, compared to 1,501.87 bytes for ElGamal. For a 5 MB document, RSA has an average memory consumption of 27,033.60 bytes, compared to 1,365.33 bytes for ElGamal.

In addition to having a lower average memory consumption, the ElGamal algorithm also exhibits significantly smaller memory usage variations than RSA. This is indicated by the ElGamal standard deviation values, which range from 747.82 to 5,430.05 bytes, while RSA has a much larger standard deviation, ranging from 95,909.03 to 121,880.66 bytes. These results indicate that ElGamal's memory usage is more consistent throughout the document signing process.

Before testing the hypothesis, a normality test was performed using the Shapiro–Wilk method on all test data (90 samples for each algorithm). The test results showed that the memory consumption data for both the RSA and ElGamal algorithms were not normally distributed ($p < 0.001$), so the analysis of performance differences was continued using the Mann–Whitney test.

TABLE XVII
MANN–WHITNEY TEST RESULTS FOR DOCUMENT SIGNING MEMORY CONSUMPTION

Metric	p-value	Decision
Memory Consumption	< 0.001	Reject H_0

The Mann–Whitney test results show a p-value <0.001, so H_0 is rejected. This indicates that there is a significant difference in memory consumption between the RSA and ElGamal algorithms during the document signing process. Based on the descriptive statistics and the Mann–Whitney test, it can be concluded that the ElGamal algorithm uses significantly less memory than RSA for all tested document sizes.

D. Performance Evaluation of Signature Verification

A performance evaluation of the digital signature verification process was conducted to compare the efficiency of the RSA and ElGamal algorithms in verifying the authenticity and integrity of electronic documents. The test was repeated 30 times using PDF documents of 500 KB, 1 MB, and 5 MB in the same test environment to obtain consistent and representative results. Parameters evaluated included computation time, CPU usage, and memory consumption. The test results were then analyzed using descriptive statistics and statistical tests to determine the significance of the performance differences between the two algorithms.

1) Execution Time

Computation time was measured to evaluate the efficiency of the RSA and ElGamal algorithms in the digital signature verification process. Measurements were repeated 30 times for each document size: 500 KB, 1 MB, and 5 MB. Test result statistics include minimum, maximum, average, and standard deviation values, as shown in Table XVIII.

TABLE XVIII
EXECUTION TIME STATISTICS OF SIGNATURE VERIFICATION

Document Size	Algorithm	Min (ms)	Max (ms)	Mean (ms)	Std. Dev. (ms)
500 KB	RSA	2.48	4.03	3.30	0.35
	ElGamal	128.07	155.91	138.40	6.23
1 MB	RSA	1.91	4.74	3.10	0.57
	ElGamal	135.36	154.96	146.16	4.60
5 MB	RSA	1.81	3.43	2.32	0.37
	ElGamal	103.66	115.79	107.81	2.47

Based on Table XVIII, the RSA algorithm consistently produced significantly lower verification times than ElGamal across all tested document sizes. For a 500 KB document, the average RSA verification time was 3.30 ms, while ElGamal required an average of 138.40 ms. Similar results were obtained for a 1 MB document, where RSA had an average verification time of 3.10 ms, while ElGamal required 146.16 ms. For a 5 MB document, RSA required an average of 2.32 ms, while ElGamal required 107.81 ms.

In addition to having a significantly lower average time, the RSA algorithm also exhibited smaller time variations than ElGamal. The standard deviation for RSA ranged from 0.35–0.57 ms, while ElGamal had a standard deviation between 2.47–6.23 ms. These results indicate that the verification process using RSA is not only faster but also more consistent throughout the testing process.

Before conducting the hypothesis testing, a normality test was performed using the Shapiro–Wilk method on all test data (90 samples for each algorithm). The test results showed that the computation time data for both the RSA ($p = 0.004$) and ElGamal ($p < 0.001$) algorithms were not normally distributed. Therefore, the analysis of performance differences was continued using the Mann–Whitney test.

TABLE XIX
MANN–WHITNEY TEST RESULTS FOR SIGNATURE VERIFICATION EXECUTION TIME

Metric	p-value	Decision
Execution Time	< 0.001	Reject H_0

The Mann–Whitney test results show a p-value <0.001, thus H_0 is rejected. This indicates that there is a significant difference between the computational time of the RSA and ElGamal algorithms in the digital signature verification process. Based on the results of descriptive statistics and the Mann–Whitney test, it can be concluded that the RSA algorithm has a significantly lower verification time than ElGamal for all tested document sizes.

2) CPU Utilization

CPU usage was measured to evaluate the level of processor resource utilization during the digital signature verification process using the RSA and ElGamal algorithms. Measurements were repeated 30 times for each document size: 500 KB, 1 MB, and 5 MB. Test result statistics include

minimum, maximum, average, and standard deviation values, as shown in Table XX.

TABLE XX
CPU UTILIZATION STATISTICS OF SIGNATURE VERIFICATION

Document Size	Algorithm	Min (%)	Max (%)	Mean (%)	Std. Dev. (%)
500 KB	RSA	0.00	60.00	7.67	15.53
	ElGamal	22.30	46.10	31.75	5.36
1 MB	RSA	0.00	100.00	14.02	29.46
	ElGamal	18.50	41.90	27.45	5.10
5 MB	RSA	0.00	55.60	6.84	16.19
	ElGamal	17.20	39.30	24.04	5.06

Based on Table XX, the RSA algorithm shows lower average CPU usage than ElGamal across all tested document sizes. For a 500 KB document, RSA's average CPU usage is 7.67%, while ElGamal's is 31.75%. The same trend is also seen for 1 MB and 5 MB documents, where RSA uses an average CPU usage of 14.02% and 6.84%, respectively, while ElGamal uses 27.45% and 24.04%.

Before hypothesis testing, a normality test was performed using the Shapiro–Wilk method on all test data (90 samples for each algorithm). The test results showed that the CPU usage data for the RSA and ElGamal algorithms were not normally distributed ($p < 0.001$), so the analysis of performance differences was continued using the Mann–Whitney test.

TABLE XXI
MANN–WHITNEY TEST RESULTS FOR SIGNATURE VERIFICATION CPU UTILIZATION

Metric	p-value	Decision
CPU Utilization	< 0.001	Reject H_0

The Mann–Whitney test results show a p-value <0.001, so H_0 is rejected. This indicates that there is a significant difference in CPU usage between the RSA and ElGamal algorithms during the digital signature verification process. Based on the descriptive statistics and the Mann–Whitney test, it can be concluded that the RSA algorithm uses significantly less CPU than ElGamal in the document verification process.

3) Memory Consumption

Memory consumption was measured to evaluate the amount of memory used by the RSA and ElGamal algorithms during the digital signature verification process. Measurements were repeated 30 times for each document size: 500 KB, 1 MB, and 5 MB. Test result statistics include minimum, maximum, average, and standard deviation values, as shown in Table XXII.

TABLE XXII
MEMORY CONSUMPTION STATISTICS OF SIGNATURE VERIFICATION

Document Size	Algorithm	Min (Byte)	Max (Byte)	Mean (Byte)	Std. Dev. (Byte)
500 KB	RSA	0	8,192	682.67	1,888.80
	ElGamal	0	8,192	273.07	1,495.65
1 MB	RSA	0	512,000	20,206.93	92,968.18
	ElGamal	0	12,288	546.13	2,340.24
5 MB	RSA	0	139,264	7,918.93	25,224.99
	ElGamal	0	12,288	819.20	2,499.61

Based on Table XXII, the ElGamal algorithm shows a lower average memory consumption than RSA across all tested document sizes. The largest difference in memory consumption is seen in 1 MB documents, where RSA uses an average of 20,206.93 bytes of memory, while ElGamal only uses 546.13 bytes. A similar pattern is also seen in 500 KB and 5 MB documents, indicating that ElGamal tends to be more efficient in memory usage during the verification process.

Before hypothesis testing, a normality test was performed using the Shapiro–Wilk method on all test data (90 samples for each algorithm). The test results showed that the memory consumption data for the RSA and ElGamal algorithms were not normally distributed ($p < 0.001$), so the analysis of performance differences was continued using the Mann–Whitney test.

TABLE XXIII
MANN–WHITNEY TEST RESULTS FOR SIGNATURE VERIFICATION MEMORY CONSUMPTION

Metric	p-value	Decision
Memory Consumption	< 0.001	Reject H_0

The Mann–Whitney test results showed a p-value <0.001, thus rejecting H_0 . This indicates a significant difference in memory consumption between the RSA and ElGamal algorithms during the digital signature verification process. Based on the descriptive statistics and the Mann–Whitney test, it can be concluded that the ElGamal algorithm uses significantly less memory than RSA in the document verification process.

Based on all test results, the RSA and ElGamal algorithms exhibit different performance characteristics at each stage of the digital signature process. RSA performed better in key generation and document verification, while ElGamal excelled in the document signing process. These results indicate that no single algorithm is superior across all test parameters, so algorithm selection must consider the characteristics of the system implementation.

Algorithm selection is not solely based on statistical significance but also considers the practical impact on system operation. A statistically significant difference is not necessarily a major factor if its impact on business processes is relatively small. Conversely, parameters that directly affect service speed are given greater weight in the decision-making process.

In the key generation process, RSA produces lower computation time than ElGamal, but this process is only performed during key pair creation or updating, so its execution frequency is relatively low. On the other hand, ElGamal exhibits faster signing times and lower memory consumption than RSA. However, the signing process is only performed once when a document is issued, so performance improvements at this stage do not significantly impact system operation.

In contrast, RSA demonstrates a clear advantage in the verification process, with lower computation time and CPU usage compared to ElGamal. In an Integrated Service Unit (ULT) system, the verification process can be repeated as long as the document is in use, by both the operator and the recipient. Therefore, the efficiency of the verification process has a greater impact on system responsiveness than the signing process.

IV. CONCLUSION

This research successfully implemented a digital signature module using the RSA and ElGamal algorithms, integrated with the Malikussaleh University Integrated Service Unit (ULT) application through a REST API service. The developed module is capable of key generation, document signing, and signature verification on electronic documents using the SHA-256 algorithm as a hash function. Performance evaluation was conducted based on execution time, CPU utilization, and memory consumption at each stage of the process.

The results showed that RSA and ElGamal have different performance characteristics. RSA produced faster computation times for key generation and signature verification, while ElGamal demonstrated better performance for document signing and lower memory consumption. Statistical analysis using the Shapiro–Wilk and Mann–Whitney tests showed that most test parameters had statistically significant differences, except for CPU utilization and memory consumption in the key generation process.

Based on the performance evaluation results and operational characteristics of the Integrated Service Unit (ULT) application, the RSA algorithm is recommended for implementation in the developed digital signature module. Although ElGamal excels in document signing, this process is only performed once for each document. Conversely, the verification process can be repeated as long as the document is still in use, so the advantages of RSA in the verification process provide greater benefits to the operational efficiency of the system. This recommendation applies to the context of the implementation of ULT applications and cannot be generalized as the best algorithm for all digital service systems, because the choice of algorithm remains dependent on the needs and implementation characteristics of each system.

REFERENCES

- [1] K. Rima, A. Suari, and I. M. Sarjana, “Menjaga Privasi di Era Digital : Perlindungan Data Pribadi di Indonesia,” *J. Anal. Huk.*, no. 1, pp. 132–146, 2023, doi: 10.38043/jah.v6i1.4484.
- [2] N. Sitorus, J. S. G. Sinaga, and S. L. Samosir, “Analisis Kinerja Algoritma Hash pada Keamanan Data : Perbandingan,” *Anal. Kinerja Algoritma Hash pada Keamanan Data Perbandingan Antara SHA-256, SHA-3, dan Blake2*, vol. 2, no. 2, pp. 9–16, 2024.
- [3] N. A. N. Siregar, “Analisis Tingkat Kepuasan Mahasiswa Terhadap Pelayanan Sistem Informasi Perpustakaan Universitas Malikussaleh Menggunakan Metode Fuzzy Service Quality,” *JITET (Jurnal Inform. dan Tek. Elektro Ter.)*, vol. 12, no. 3, pp. 4660–4673, 2024.
- [4] A. H. Hasugian, R. A. Putri, and A. S. Zuhri, “Penerapan Metode Elgamal Untuk Mengamankan Teks Pada File Dokumen Word,” *KLIK Kaji. Ilm. Inform. dan Komput.*, vol. 4, no. 2, pp. 1143–1153, 2023, doi: 10.30865/klik.v4i2.1277.
- [5] N. Nurdin, A. Habbal and F. Fajriana, “Comparative Analysis of MD5 and SHA-256 Hash Algorithms for Fingerprint File Integrity Verification,” *JAIC*, vol. 10, no. 2, pp. 1495–1504, Apr. 2026, doi: 10.30871/jaic.v10i2.12432.
- [6] N. Nurdin, “Implementasi Algoritma Tea dan Fungsi Hash Md4 Untuk Enkripsi dan Deskripsi Data,” *Jurnal TECHSI.*, vol.5, no. pp. 97–112, 2017.
- [7] M. S. Asih, P. Studi, T. Informatika, and U. H. Medan, “Implementasi Algoritma Kriptografi RSA Dalam Aplikasi Sistem Informasi Perpustakaan,” *J. Ilmu Komput. dan Sist. Inf.*, vol. 2, pp. 98–107, 2023.
- [8] A. Aryasanti, M. Hardjianto, G. Brotsaputro, and R. Roeswidiyah, “Implementasi Tanda Tangan Digital Menggunakan Algoritme RSA dan SHA-512 dengan Salt Berbasis Web,” *Technol. Inf. Commun.*, vol. 10, pp. 181–186, 2022.
- [9] E. A. Adeniyi, P. B. Falola, M. S. Maashi, M. Aljebreen, and S. Bharany, “Secure Sensitive Data Sharing Using RSA and ElGamal Cryptographic Algorithms with Hash Functions,” *Information*, pp. 1–14, 2022.
- [10] Nurdin, “Comparative Analysis of AES and FHE Encryption Algorithms in Financial Technology Applications,” *Jurnal CoreIT.*, vol. 10, no. 2, pp. 145–159, 2024.
- [11] P. K. A. Jnr, “A Comparative Study of RSA and ELGAMAL Cryptosystems A Comparative Study of RSA and ELGAMAL Cryptosystems,” *Int. J. Comput. Eng.*, vol. 4, no. 1, pp. 33–41, 2023.
- [12] B. Harjito, T. Setyawati, and A. Wijayanto, “Comparative Analysis between Elgamal and NTRU Algorithms and their implementation of Digital Signature for Electronic Certificate,” vol. 13, pp. 729–739, 2022.
- [13] A. N. Sari *et al.*, “Implementasi Digital Signature Pada Laporan Tugas Akhir,” *(Jurnal Inform. Polinema)*, pp. 99–106, 2023.
- [14] M. Fikry, “Aplikasi Android untuk Tanda Tangan Digital (Studi Kasus di Jurusan Teknik Informatika , UIN Suska Riau),” *J. Sains, Teknol. dan Ind.*, vol. 19, no. 2, pp. 430–435, 2022.
- [15] R. K. Lubis, A. M. H. Pardede, and H. Khair, “Digital Signature Security Analysis By Applying The Elgamal Algorithm And The Idea Method,” *J. Artif. Intell. Eng. Appl.*, vol. 3, no. 1, pp. 373–382, 2023.
- [16] M. Iavich and D. A. Sairangazhykyzy, “Post-quantum Digital Signatures using ElGamal Approach,” *Comput. Netw. Inf. Secur.*, vol. 17, no. 3, pp. 89–108, 2025, doi: 10.5815/ijcnis.2025.03.06.
- [17] S. S. M. N. Taufiq, “JITE (Journal of Informatics and Telecommunication Engineering) Aceh Using the PIECES Method,” vol. 7, no. July, pp. 71–83, 2023, doi: 10.31289/jite.v7i1.9167.
- [18] K. Pasa and N. Aini, “Pengaruh Model Role Playing Terhadap Hasil Belajar Peserta Didik Materi Perumusan Pancasila Kelas 4 Sdn Sidokepung I,” *J. Ilm. Pendidik. Dasar*, vol. 10, pp. 194–206, 2025.
- [19] Yulisnawati, I. Yusratwati, and R. M. Zaenal, “Efektivitas Model Pembelajaran Project Based Learning (Pjbl) Melalui Game Edukasi Terhadap Kemampuan Pemecahan Masalah Matematis Siswa,” *Pas. J. Res. Math. Learn. Educ.*, vol. 10, pp. 90–101, 2025, doi: 10.23969/symmetry.v10i1.29621.