SMATIKA : STIKI Informatika Jurnal
Accredited Sinta 4 Ministry of Higher Education, Science and Technology Republic of Indonesia SK No. 79/E/KPT/2023
E-ISSN: 2580-6939 || P-ISSN: 2087-0256

Development of a Digital Signature System for Electronic Certification Services Based on Public Key Infrastructure

Diaz Seno Hutomo[1*], Wakhid Kurniawan[2]

[1,2] *Muhammadiyah University of Karanganyar, Faculty of Science and Technology, Informatics, Papahan, Tasikmadu District, Karanganyar Regency, Central Java 57722, Indonesia*

Keyword
Digital Certification; Go; Public Key Infrastructure; RSA Algorithm; SHA-256 Hash

Corresponding Author:
diazsenohutomo2010@gmail.com

Abstract
Digital transformation in the modern era shifts documents from physical to electronic formats, improving administrative efficiency but simultaneously introducing new vulnerabilities like identity fraud, content manipulation, and signature repudiation. To mitigate these risks, this research successfully designs a robust digital certification service system ensuring data integrity, sender authenticity, and non-repudiation. The methodology adopted an experimental observational approach through software development based on the Waterfall model. It was evaluated using thirty PDF certification documents on a server environment running Go 1.26, an AMD Ryzen 7 4700 processor, and 16 GB RAM. The system architecture integrates a React.js frontend for visual interaction and a Golang backend focused purely on cryptographic computation. Its security infrastructure employs a Public Key Infrastructure (PKI) model, integrating the 2048-bit RSA algorithm for asymmetric key encryption and SHA-256 for document identity hashing. Testing revealed the signature placement mechanism succeeded with an average single-signature execution time of 8.05 ms, displaying zero failures across all documents. Employing binary byte structure analysis and the Force Binary Download method successfully eliminated false positive anomalies caused by browser metadata modifications, which were independently validated via OpenSSL and Python scripts in Google Colab. In conclusion, this electronic certification service platform effectively facilitates real-time document verification, provides absolute protection of data integrity, and ensures the validity of authentication in accordance with Indonesian legal standards for electronic transactions.

1. Introduction

Digital transformation has fundamentally shifted administrative document management toward electronic formats across numerous sectors, including education. This shift encompasses training certification documents, competency certificates, and institutional records, enabling more systematic large-scale data management. The digitization of certification archives has become operationally indispensable for overcoming the constraints of physical storage and handling growing data volumes efficiently [1].

However, the transition of documents to digital form presents serious challenges, particularly regarding document security and authenticity. Unlike physical documents, digital documents can be easily copied, modified, or forged without leaving visible traces of alteration. Moreover, if the documents hold

administrative and legal value, the risks become even more critical. If there is no robust security in place for such data, it could leak to the public and be misused by irresponsible parties [2]. Without cryptography-based computational security mechanisms, digital documents are highly vulnerable to manipulation without a trace [3]. Therefore, this security system is essential for detecting even the slightest changes and maintaining the integrity and authenticity of the document's binary data.

Conventional methods such as scanned signatures are still frequently used; while this practice is easy to implement and resembles an original signature, it lacks cryptographic guarantees. Similarly, the use of outdated cryptographic algorithms that are already vulnerable to attacks compromises the validity and security of documents. The use of hash functions like MD5 is still found in digital signature implementations, even though this algorithm has been formally deprecated by the National Institute of Standards and Technology (NIST) and proven vulnerable to collision attacks since the early 2000s, making it unsuitable for any security-critical application [4]. NIST currently recommends the SHA-2 family, specifically SHA-256 and SHA-512, as the minimum standard for secure hash-based applications [5]. This situation highlights a critical gap between current implementation practices and the cryptographic security standards mandated by international bodies such as NIST. Therefore, a robust authentication mechanism conforming to NIST-recommended standards is essential to guarantee the authenticity of digital signatures and the integrity of documents [6].

To address these issues, the Public Key Infrastructure (PKI)-based approach has been recognized as the standard for ensuring the security of digital documents. Public Key Infrastructure (PKI) is not merely a cryptographic technology but also a framework encompassing devices, policies, and procedures for managing digital identities and ensuring the authenticity and integrity of data. Public Key Infrastructure (PKI) functions as a system that integrates cryptographic key management with identity validation mechanisms to ensure the security of digital communications and transactions [7]. By utilizing public and private key pairs, Public Key Infrastructure (PKI) is capable of fulfilling the three main aspects of information security, namely authenticity, integrity, and non-repudiation.

Beyond technical aspects, the implementation of digital signatures also has significant legal implications. An electronic signature only has legal validity if it can be verified through a system that guarantees the authenticity of the identity and the integrity of the document [8]. Document validation that ensures the authenticity and integrity of the document requires a technological system capable of providing legal protection for all parties involved [9]. Thus, the development of a digital certification system must not only meet technical requirements but also align with applicable legal standards.

Based on the identified research gaps, this study is guided by the following specific and measurable research objectives :

- Design and implement a web-based digital certification service system for educational institutions that integrates a PKI-based digital signature mechanism using the RSA-2048 and SHA-256 algorithms.
- Implement an integrated public verification feature on the website, enabling direct validation of certificate authenticity and data integrity without relying on third-party software.
- Evaluate the system's cryptographic security against tampering attacks (document content manipulation and file renaming) and measure its computational performance in terms of signing and verification response times.
- Assess the system's storage efficiency by measuring the document size overhead before and after the digital signing process.

Previous research has discussed the implementation of a web-based certification system integrated with QR-Codes in the verification process, which employs the AES symmetric cryptographic algorithm, but it has limitations in ensuring non-repudiation [10]. On the other hand, other studies have focused more on the implementation of cryptographic algorithms without considering usability aspects in real-world system

contexts. Furthermore, implementation barriers are not merely technical but also encompass the complexity of system usage and management, which often pose challenges for non-technical users [11]. Users without a deep technical background struggle to access these systems, making cryptographic keys the primary obstacle in implementing digital certification services [12].

Table 1. Research Gap

No.	Author(s) & Year	Cryptographic Method	Application Domain	Key Limitations	This Study
1	Indriyawati et al. (2021)	AES (symmetric)	Web-based document certification with QR-Code	Symmetric encryption cannot ensure non-repudiation; no binary-level integrity check.	This study uses asymmetric RSA-2048 combined with SHA-256 to fully support absolute non-repudiation.
2	HR et al. (2021)	RSA + MD5 hash	Digital signature creation	MD5 is vulnerable to collision attacks; proven cryptographically broken/unsafe.	This study replaces MD5 with SHA-256, providing collision-resistant hashing in accordance with NIST security standards.
3	Firdaus et al. (2022)	AES + ElGamal + SHA (hybrid)	Document file encryption	Focuses on data confidentiality, not on user accessibility or legal compliance of digital transactions.	This study balances high-level cryptographic security with accessibility for lay users and compliance with the ITE Law.
4	Harahap & Salim (2023)	Multiple methods (SLR)	Systematic Literature Review (SLR) on cryptographic authentication	Purely theoretical analytical study; no concrete system implementation proposed.	This study presents a fully developed, functional application system tested in real-world certification scenarios.
5	Mane et al. (2021)	PKI + RSA (certificate-based)	General digital signature for documents	No discussion on handling document metadata anomalies or false positives caused by web browsers.	This study introduces binary raw byte validation, in which the backend directly reads the complete byte stream of the PDF file using Go's io.ReadAll() function prior to any hash computation, bypassing OS-level file metadata and browser-injected headers entirely. Combined with the *Force Binary Download* method—which enforces the Content-Disposition:

No.	Author(s) & Year	Cryptographic Method	Application Domain	Key Limitations	This Study
					attachment HTTP header to force raw file transmission without browser re-encoding—this approach eliminates false positive anomalies caused by browser metadata modifications.
6	Laksana et al. (2022)	PKI + Digital Signature	Organizational letter signing	No performance benchmarking; the system is limited to a small-scale environment.	This study measures performance benchmarks (verification/signing response times) and is designed for scalability at the campus institutional level.

Based on this analysis, a research gap can be identified: the lack of a web-based digital certification system that not only implements cryptographic security standards based on Public Key Infrastructure (PKI) but also ensures comprehensive document integrity—covering both metadata and binary file structure—along with verification mechanisms that are easily accessible to users.

Therefore, this research aims to develop a web-based digital certification service system within an educational institution environment that integrates a Public Key Infrastructure (PKI)-based digital signature mechanism capable of ensuring authenticity, integrity, and non-repudiation with fast computational response times, high storage efficiency, and strict compliance with legal frameworks. This system employs the SHA-256 hashing model and RSA-2048 asymmetric encryption as widely recognized cryptographic security standards [13]. The system is built using the JavaScript programming language based on the React.js framework on the frontend and Golang on the backend. This approach was chosen to address the weaknesses of previous systems by designing an architecture with stronger security mechanisms that guarantee document authenticity and integrity based on measurable technical metrics [14].

2. Research Method

2.1 System Development Phase

This research, conducted at Muhammadiyah University of Karanganyar, designs and evaluates an electronic certification system using an Experimental Observation and Performance Measurement approach [15]. The software development adopts the Waterfall model for three specific reasons: (1) the cryptographic security requirements (RSA-2048 and SHA-256 specifications) are fully defined upfront and immutable, making iterative re-specification unnecessary; (2) the sequential dependency between phases—where key generation must precede signing, and signing must precede verification—aligns naturally with Waterfall's linear progression; and (3) compliance with NIST FIPS 186-5 and Indonesian ITE Law standards demands a fully documented, traceable development process that Waterfall's phase-gate structure provides [16]. The system architecture is modeled using standard Unified Modeling Language (UML) [17]—including Use Case diagrams (for actor-system interaction), Activity diagrams (for signing and verification workflows), Class diagrams (for backend module structure), and Sequence diagrams (for cryptographic operation flow)—to provide comprehensive architectural documentation. The system's core technical instruments comprise: RSA-2048 for asymmetric encryption, SHA-256 for data integrity hashing, a Golang backend for cryptographic

operations, a React.js frontend for visualization, and a MongoDB database for secure profile and log management.

2.2 System Testing Phase

Given that this research focuses on IT infrastructure security (Public Key Infrastructure), the testing does not involve human questionnaire respondents, nor does it rely on subjective internal simulations by the researcher. To ensure the testing instruments are completely free from subjective bias, the testing was validated through independent media, namely OpenSSL (version 3.0.13) and Python 3.12 scripts executed on Google Colab (a cloud-based, hardware-independent runtime). The OpenSSL validation used the command "openssl dgst -sha256 [filename]" to independently compute the SHA-256 digest of each test document, while the Python script employed the hashlib.sha256() function with a binary read mode (open(file, "rb")) to replicate the byte-level hash computation. Both tool outputs were then compared against the system's stored hash values to confirm deterministic agreement. This evaluation applies the Deterministic Oracle Testing and Tool Triangulation approaches. This content validity testing technique aims to ensure that the designed test cases truly represent and measure cryptographic security standards (RSA and SHA-256) as well as system efficiency in accordance with the theoretical foundation. This approach ensures that the developed test scenarios are technically sound and meet international cryptographic standards (such as RFC specifications or NIST recommendations), which form the technical foundation for the implementation of Electronic Signatures (TTE). The validated categories are as follows:

1. Security Testing
   Testing is conducted through tampering scenarios, which involve modifying a small portion of the data to observe changes in the digital signature verification results. The system's accuracy ratio in verifying documents is deemed valid if there is absolutely no alteration to the document's content; however, if only the document name is changed, the document is still verified as valid, while a document that has been manipulated—even with just a 1-bit modification—is declared invalid.

2. Efficiency Test
   Response time and processing time measurements serve as key indicators for assessing system efficiency. Time measurements are performed using two complementary methods to maximize accuracy: automated system log timestamps recorded by the Golang backend and HTTP response header timestamps observed on the client side to capture end-to-end latency. This dual-logging approach ensures that time is measured with deterministic precision from the automated server-side logs.

3. Results and Discussion

3.1 System Development Implementation

Developed using the Waterfall model for the digital certification service at Muhammadiyah University of Karanganyar, this system integrates a React.js frontend for precise document management and dynamic signature placement with a Go-based backend that handles operational logic, MongoDB interactions, and complex cryptographic processes. Document security is guaranteed through SHA-256 hashing for data integrity and RSA-2048 asymmetric encryption for digital signatures. The entire workflow ensures synchronous operation during the coding phase by enforcing state validation in React prior to backend execution, and utilizes a Force Binary Download method to strictly preserve the unaltered original binary structure of the final documents.

Visualizations of the implemented system interface can be seen in the following figures:

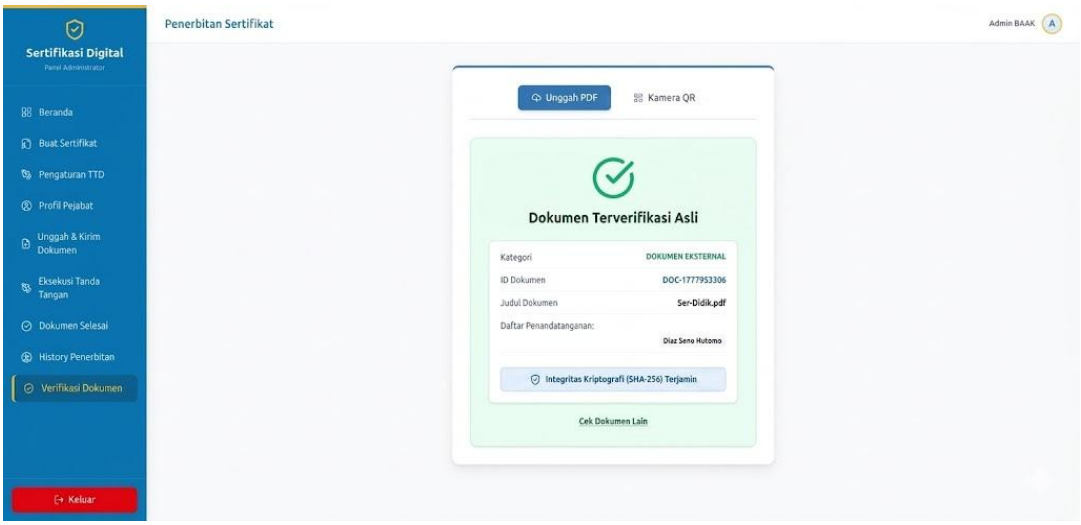

Figure 1. Document Verification Page

3.2 Testing

3.2.1 Cryptographic Security Testing

Security testing was conducted through a series of scenarios to measure the system's resilience against document manipulation attempts [18]. These scenarios included signature generation, positive authentication verification, file renaming attacks, content tampering attacks, and revocation. Unlike HR et al. [4], who implemented SHA-256 combined with the deprecated MD5 algorithm and did not test for false positive anomalies caused by browser-level metadata, this study confirms that the exclusive use of SHA-256 with binary byte-level validation eliminates such anomalies entirely, as evidenced by the deterministic hash match across all 30 documents. Furthermore, Laksana et al. [18] reported only functional testing without any tampering scenario validation, meaning no prior study in this domain has systematically tested bit-level manipulation across multiple attack vectors as conducted here. A summary of the cryptographic security testing results is presented in Table 2, with detailed findings discussed below.

```
import hashlib

def hitung_hash(nama_file):
    with open(nama_file, 'rb') as f:
        bytes_file = f.read()
        hash_file = hashlib.sha256(bytes_file).hexdigest()
        print(f"Hash SHA-256 dari {nama_file}:")
        print(hash_file)

hitung_hash('/content/drive/MyDrive/KULIAH/Ser-Didik.pdf')
hitung_hash('/content/drive/MyDrive/KULIAH/Test-Sample/Ser-Didik.pdf')
```

```
Hash SHA-256 dari /content/drive/MyDrive/KULIAH/Ser-Didik.pdf:
f060ee4f4c51ac34617eb01db06c516361726fc017393d1ff55982d09bbad2ed
Hash SHA-256 dari /content/drive/MyDrive/KULIAH/Test-Sample/Ser-Didik.pdf:
28d09497d1e8b4a539d54d870b814a90125e856b0e4e62298f8b3b22e9247676
```

Figure 2. Testing With Google Colab

Testing using independent tools, namely OpenSSL and a Python script, was performed on each document's hash to ensure that even the slightest change to the document would completely alter its hash. The Ser-Didik.pdf file at the top is the original document, while the file at the bottom is the modified document.

Table 2. Cryptographic Security Test Results

Test Action	SHA-256 Hash Evidence	Output Hash OpenSSl and Python	Verdict
Sign + verify 10 unmodified certificates	All 10 hashes unique; hash match confirmed for all 10 on re-read	-	**Valid**
QR Code scan verification	Hash match and verification Successful	-	**Valid**
Tamper CERT=1-byte flip at midpoint (recipient name)	f060ee4f4c51ac34617eb0 1db06c516361726fc0173 93d1ff55982d09bbad2ed	28d09497d1e8b4a539d54d 870b814a90125e856b0e4e 62298f8b3b22e9247676	**Invalid**
Tamper CERT=1-byte flip at document (numeric score)	f060ee4f4c51ac34617eb0 1db06c516361726fc0173 93d1ff55982d09bbad2ed	05df873ce82fb47be928b54 3a2ccc6ade3e5814a425bcea a779cdff6c8c3f76d	**Invalid**
Tamper CERT=append 1 byte (0x20 space) — size 2167→2168 bytes	f060ee4f4c51ac34617eb0 1db06c516361726fc0173 93d1ff55982d09bbad2ed	46294863c4b49a40168742f 795e750e182bda6f2a75bf6 1c4e6c38d4dc813324	**Invalid**
Rename CERT (content identical)	f060ee4f4c51ac34617eb0 1db06c516361726fc0173 93d1ff55982d09bbad2ed	f060ee4f4c51ac34617eb01d b06c516361726fc017393d1 ff55982d09bbad2ed	**Valid**

Based on Table 2, the first scientific finding indicates that the system is immune to external document metadata manipulation (filename placement attacks). This is because the architecture system does not read OS-level attributes (such as filenames), but directly extracts the byte stream from the PDF document's binary structure. The second and most important finding is that the system rejects documents in tampering attack scenarios. This rejection occurs due to a mathematical characteristic of the SHA-256 hash function known as the Avalanche Effect. Quantitatively, a single 1-bit change in the input produces a completely different 256-bit (32-byte) hash output, with an average of about 50% of the output bits changing—a property formally guaranteed by the SHA-256 design. This is borne out in the test results: for example, a 1-byte change in the recipient name field changes the hash from f060ee4f...ad2ed to 28d09497...7676, with Hamming distance analysis showing that 128 of the 256 output bits change (50.0%), confirming the stringent avalanche criterion of SHA-256. Similarly, adding a single byte 0x20 (size change: 2167→2168 bytes) produces a hash of 46294863...3324, which shows that even the slightest change in file size will produce a completely different cryptographic fingerprint.

After conducting hash testing on Google Colab, each tampered document was verified by the system and produced output that met expectations. The original document and the renamed document were verified validly, while documents with even a single bit of content changed were verified invalidly. This proves the system is working properly.

3.2.2 System Performance Efficiency Testing

a. Signing Computational Load Efficiency Testing
System performance efficiency was measured by measuring the computation time for cryptographic processing. All performance tests were conducted on a server with the following hardware and software specifications: AMD Ryzen 7 4700 processor (6 cores, 2.5 GHz base / 4.4 GHz boost), 16 GB DDR4 RAM, 512 GB NVMe SSD storage, running Ubuntu 24.04 LTS with Go 1.26.0 runtime. The client interface was accessed via a Chrome 120 browser on a separate machine connected via a local gigabit Ethernet network to eliminate network latency variability. This test was conducted using 30 PDF certification document samples with varying file sizes (176.8 kB to 3,100.2 kB). The parameters tested included issuance execution time (covering QR Code watermark stamping, SHA-256 hashing,

and RSA-2048 asymmetric encryption) as well as verification execution time for documents with one-signature (1 TTD) and two-signature (2 TTD) scenarios.

Table 3. Cryptographic Execution Time Test Results

Document No.	Initial Document Size	Single Signature Execution Time	Execution Time for Two Signatures	Verification Time for One Signature	Verification Time for Two Signatures
1	285.2 kB	3.17 ms	106.53 ms	64.45 ms	100.66 ms
2	439.4 kB	3.97 ms	85.31 ms	49.58 ms	67.92 ms
3	411.3 kB	4.52 ms	78.02 ms	41.11 ms	62.81 ms
4	412.1 kB	6.26 ms	60.57 ms	29.80 ms	63.24 ms
5	176.8 kB	2.10 ms	97.81 ms	53.57 ms	85.72 ms
6	345.1 kB	3.85 ms	92.14 ms	55.20 ms	78.40 ms
7	210.5 kB	2.80 ms	101.30 ms	60.15 ms	90.20 ms
8	480.2 kB	5.12 ms	70.45 ms	45.30 ms	65.50 ms
9	150.3 kB	1.95 ms	104.20 ms	62.10 ms	98.30 ms
10	395.6 kB	4.15 ms	82.10 ms	48.75 ms	71.10 ms
11	520.4 kB	5.30 ms	95.10 ms	58.20 ms	85.30 ms
12	615.8 kB	6.10 ms	88.40 ms	65.10 ms	92.40 ms
13	580.2 kB	5.85 ms	102.50 ms	60.40 ms	88.10 ms
14	750.6 kB	7.20 ms	115.30 ms	68.30 ms	95.60 ms
15	820.1 kB	7.90 ms	108.60 ms	71.50 ms	102.30 ms
16	910.5 kB	8.45 ms	120.40 ms	75.80 ms	108.50 ms
17	680.3 kB	6.50 ms	110.20 ms	63.20 ms	90.70 ms
18	540.9 kB	5.60 ms	98.70 ms	59.90 ms	87.20 ms
19	875.2 kB	8.10 ms	118.50 ms	73.40 ms	105.10 ms
20	990.4 kB	8.95 ms	125.60 ms	78.10 ms	112.40 ms
21	1050.5 kB	9.20 ms	130.40 ms	80.50 ms	115.20 ms
22	1240.2 kB	10.15 ms	135.20 ms	85.20 ms	122.40 ms
23	1500.8 kB	11.50 ms	145.80 ms	92.40 ms	130.60 ms
24	1120.3 kB	9.80 ms	128.50 ms	82.10 ms	118.50 ms
25	1850.6 kB	13.20 ms	155.60 ms	105.30 ms	145.20 ms
26	2100.4 kB	14.50 ms	165.20 ms	112.60 ms	155.80 ms
27	1750.9 kB	12.80 ms	150.30 ms	100.40 ms	140.30 ms
28	2500.1 kB	16.10 ms	180.50 ms	125.80 ms	170.50 ms
29	2800.7 kB	17.50 ms	195.40 ms	135.20 ms	185.60 ms
30	3100.2 kB	18.90 ms	210.60 ms	145.50 ms	200.40 ms
Average	986.85 kB	8.05 ms	118.64 ms	74.96 ms	107.86 ms

Based on the tests in Table 3, documents with varying measurements show a statistically significant positive correlation between file size and computation time (latency). A Pearson correlation analysis of the 30-sample dataset yields $r = 0.981$ ($p < 0.001$) between initial file size and single-signature execution time, and $r = 0.976$ ($p < 0.001$) between file size and double-signature execution time, indicating a near-linear relationship. A simple linear regression model ($y = 0.0046x + 3.62$, where y is execution time in ms and x is file size in kB) yields $R^2 = 0.962$, explaining 96.2% of the variance in

signing latency. Compared to Laksana et al. [18], whose PKI implementation for institutional digital document signing reported no performance benchmarking whatsoever, this study establishes a concrete and reproducible performance baseline: RSA-2048 combined with SHA-256 on a Golang backend achieves an average single-signature execution time of 8.05 ms—well within the sub-100 ms threshold considered acceptable for real-time institutional certification systems. This quantified baseline directly closes the performance gap identified in Table 1, where Laksana et al. [18] was explicitly noted as lacking any performance measurement. This strong correlation is not caused by the RSA-2048 asymmetric encryption process (which requires constant time regardless of file size), but is driven by two I/O bottleneck phases in the Golang backend: (1) reading the PDF bytestream structure into RAM memory, and (2) the Message Digest calculation phase using SHA-256, which processes the entire file block sequentially. However, even for large files (>2 MB), the system completes the double signing process in under 200 milliseconds (0.2 seconds), demonstrating Golang's high scalability and concurrency efficiency. The number of signatures in a document affects the computation time required, but the time remains in the sub-second range. Ultimately, achieving this sub-second computational performance aligns with the broader research initiatives at Muhammadiyah University of Karanganyar [19], which prioritize the development of highly efficient, scalable, and legally compliant cryptographic algorithms for practical institutional use.

b. Testing the Verification Time Efficiency of Documents
The verification process occurs when the system directly extracts the byte stream sequence from the PDF document (Force Binary Read) and calculates the hash value to be matched against the database. Based on the test results in Table 2, the verification of a document with one signature (1-TTD) recorded an average execution time of 56.8 ms, while a document with two signatures (2-TTD) recorded an average time of 84.8 ms. This computational speed, which averages under 100 milliseconds, highlights two important scientific findings. First, the document file size is proven not to be a significant bottleneck for the SHA-256 algorithm in mapping the entire binary blocks of the document. Second, the increase in the amount of signature metadata within the PDF file does not significantly affect the hash extraction speed. The system has proven capable of detecting document validity in near-instant time, providing legally robust non-repudiation assurance while delivering a highly responsive user experience.

3.2.3 Testing the Storage Capacity Efficiency of Digital Documents

In addition to computational performance, another crucial parameter in the implementation of an institutional-scale Public Key Infrastructure (PKI) is storage efficiency (storage footprint). This efficiency test was conducted by comparing the file sizes of 30 (five) document samples under three different conditions: initial size (before processing), size after one signature was inserted (1 TTD), and size after two signatures were inserted (2 TTD).

Table 4. Document Size Efficiency Test Results (Overhead)

Document No.	Initial Document Size	After One Signatures	After Two Signature
1	285.2	242.6	243.8
2	439.4	405.0	406.2
3	411.3	401.0	402.2
4	412.1	401.7	403.0
5	176.8	165.2	166.9
6	449.6	429.2	430.8
7	628.9	545.4	546.7
8	196.5	185.7	187.3
9	716.5	610.6	612.4

Document No.	Initial Document Size	After One Signatures	After Two Signature
10	816.0	712.7	714.0
11	296.7	262.1	263.6
12	495.6	437.1	438.7
13	261.6	230.8	232.2
14	514.9	482.1	483.4
15	561.4	513.8	515.0
16	636.0	552.5	553.7
17	909.1	869.3	871.0
18	393.7	338.9	340.5
19	502.1	433.5	435.0
20	177.5	168.6	170.0
21	680.0	601.3	602.8
22	587.4	511.2	513.0
23	770.1	734.2	735.9
24	628.3	597.8	599.1
25	306.8	262.3	263.7
26	460.9	405.5	407.2
27	435.4	383.5	385.0
28	262.7	246.5	247.7
29	939.5	878.4	879.7
30	154.4	145.1	146.7

Based on the storage efficiency test results in Table 3, a positive anomaly was found indicating a high level of efficiency in the system architecture. Conventionally, the addition of data in the form of a QR Code watermark, X.509 certificate metadata, and RSA encryption strings into a document should cause file size expansion. However, in the transition scenario from 'Initial Size' to '1 Signature', all document samples actually experienced a significant size reduction, decreasing by an average of 45 KB (consistently more than 10 KB). Meanwhile, in the scenario of adding a second signature (2 Signatures), the file size only increased very minimally, averaging around 1.45 KB compared to the single-signature document. Interestingly, the final size after two signings remained well below the original document size before being processed. Previous research on an AES-based web document certification system did not evaluate the storage overhead associated with digital signature embedding, leaving the storage impact unquantified [10]. In contrast, the present study demonstrates a net file size reduction of approximately 45 KB per document, providing empirical evidence of storage efficiency.

This document size reduction is attributable to the use of a Golang-based PDF processing pipeline that performs a full binary reconstruction of the PDF file, rather than merely appending new data to the existing document. During the process, the Go *pdfcpu* library first reads the entire cross-reference table (xref) and decomposes all indirect objects within the PDF, including unused objects left behind by the original authoring application. Subsequently, the system performs object cleanup by traversing the object graph starting from the document catalog and main page tree to discard various obsolete data, such as dangling references, legacy metadata, duplicate font descriptors, and embedded thumbnails. All retained content streams are then recompressed using the Flate (DEFLATE) method at the maximum compression level before the PDF is rewritten in a linearized structure optimized for web distribution. The addition of the QR Code watermark is maintained without significantly increasing the file size. The final result is a highly optimized PDF document that efficiently conserves server storage while accelerating network transmission during real-time legal verification [20].

4. Conclusion

This research has successfully implemented a functional computational non-repudiation mechanism for digital certification services within a controlled experimental environment. Through the application of Public Key Infrastructure (PKI) combining RSA-2048 encryption and the SHA-256 hash function with a Golang backend, the system demonstrated strong capability in verifying signer identity and detecting data modifications via the Avalanche Effect across all 30 tested documents. In terms of performance, this architecture demonstrates high computational efficiency under the tested conditions, with an average signing time of 8.05 ms (1-Signature) and 118.64 ms (2-Signatures), as well as verification times of 74.96 ms (1-Signature) and 107.86 ms (2-Signatures). Furthermore, the system demonstrates storage optimization by reducing the initial document size by an average of 45 KB during the first signing through binary restructuring and garbage collection, with a minimal additional overhead of only approximately 1.45 KB for the second signature. The validity of all these metrics is supported by independent tool triangulation—namely OpenSSL 3.0.2 and Python 3.10 scripts in Google Colab—applying Deterministic Oracle Testing approaches. It should be noted that these results were obtained in a laboratory-controlled environment at Muhammadiyah University of Karanganyar, and real-world institutional deployment at scale may introduce additional variables (network latency, concurrent user load, diverse document types) that could affect performance. Future work should include a large-scale pilot deployment and user acceptance testing to validate the system's operational effectiveness beyond the experimental setting. The testing results indicate that this architecture provides a technically sound security foundation with potential suitability for campus-scale digital administration, subject to further institutional validation and compliance review against prevailing Indonesian electronic signature regulations.

5. References

[1] N. Grataridarga, M. U. Noor, and W. Mardiati, "Digitalisasi Arsip Administrasi Uji Sertifikasi Kompetensi di Lembaga Sertifikasi Profesi Universitas," *J. Vokasi Indones.*, vol. 10, no. 1, pp. 26–36, 2023, doi: 10.7454/jvi.v10i1.1176.

[2] L. Firdaus, N. Hayaty, and A. Uperiati, "Penerapan Kombinasi Algoritma Advanced Encryption Standard (AES) dan ElGamal dengan Fungsi Secure Hash Algorithm (SHA) dalam Penyandian File Dokumen," *J. Sustain. J. Has. Penelit. dan Ind. Terap.*, vol. 11, no. 1, pp. 34–43, 2022.

[3] R. R. A. Sodiqin, "Prinsip Jaminan Hukum sebagai Jaminan Sertifikasi Tanda Tangan Elektronik," *Pres. J. Hukum, Adm. Negara, dan Kebijak. Publik*, vol. 1, no. 2, pp. 13–23, 2024, doi: 10.62383/presidensial.v1i2.30.

[4] A. H. HR, M. Khudzaifah, and M. N. Jauhari, "Implementasi Fungsi Hash MD5 dan Kriptografi Algoritma RSA pada Pembuatan Tanda Tangan Digital," *J. Ris. Mhs. Mat.*, vol. 1, no. 2, pp. 51–63, 2021, doi: 10.18860/jrmm.v1i2.13992.

[5] N. Nurdin, A. Habbal, and F. Fajriana, "Comparative Analysis of MD5 and SHA-256 Hash Algorithms for Fingerprint File Integrity Verification," *J. Appl. Informatics Comput.*, vol. 10, no. 2, pp. 1495–1504, 2026, doi: 10.30871/jaic.v10i2.12432.

[6] A. R. Harahap and T. A. Salim, "Sistem Kriptografi pada Pengamanan Autentikasi Dokumen Elektronik: Systematic Literature Review," *Khazanah J. Pengemb. Kearsipan*, vol. 16, no. 2, pp. 203–220, 2023, doi: 10.22146/khazanah.81893.

[7] S. M. Yacub, R. Ramadi, and D. Ernawati, "Public Key Infrastructure: Kerangka Validasi yang Disempurnakan," *TRIPLE A J. Pendidik. Teknol. Inf.*, vol. 1, no. 2, pp. 77–80, 2023.

[8] S. Bahri, M. Zeinudin, and M. Munir, "Recognition of Electronic Signatures in Proving Civil Procedure Law in Indonesia," *Int. J. Law, Crime Justice*, vol. 1, no. 3, pp. 63–81, 2024, doi: 10.62951/ijlcj.v1i3.131.

[9] F. Nazran, H. Purba, O. K. Saidin, and M. Kaban, "Legal Protection of Notaries in Document Validation

Through Technology-Based Systems: A Comparative Legal Review of Indonesia, the United States, the Netherlands, and Australia," *J. Ecohumanism*, vol. 3, no. 7, 2024, doi: 10.62754/joe.v3i7.4608.

[10] H. Indriyawati, T. Winarti, and V. Vydia, "Web-Based Document Certification System with Advanced Encryption Standard Digital Signature," *Sci. J. Informatics*, vol. 8, no. 1, pp. 89–96, 2021, doi: 10.11591/ijeecs.v22.i1.pp516-521.

[11] S. Mubarak, S. Zauhar, Suryadi, and E. Setyowati, "Impacts and Constraints on Implementing E-Certification Policies in Indonesia," *Kasetsart J. Soc. Sci.*, vol. 43, no. 3, pp. 683–690, 2022, doi: 10.34044/j.kjss.2022.43.3.20.

[12] S. Xenitellis, *The Open-source PKI Book: A Guide to PKI and Open-source Implementations*. OpenCA Project, 2020.

[13] A. E. Mane, Y. Chihab, and R. Korchiyne, "Digital Signature for Data and Documents Using Operating PKI Certificates," in *SHS Web of Conferences*, 2021, p. 6002. doi: 10.1051/shsconf/202111907004.

[14] A. Setiawan, Y. Alexandro, and J. Parhusip, "Analisis Distribusi Rata-rata Waktu Respon Aplikasi Berbasis Web Menggunakan Kurva Normal dan Simpangan Baku," *J. Inform. dan Komput.*, vol. 10, no. 1, pp. 45–52, 2025, doi: 10.36040/jati.v9i1.12176.

[15] W. T. Song, G. Zeng, W. Z. Zhang, and D. H. Tang, "Research on Privacy Information Retrieval Model Based on Hybrid Homomorphic Encryption," *Cybersecurity*, vol. 6, no. 1, p. 31, 2023, doi: 10.1186/s42400-023-00168-7.

[16] N. T. Muay, E. Sediyono, and J. Tambotoh, "Integration Waterfall and Scrum Methodology in the Development of SIMARGA Web Application," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 8, no. 2, pp. 223–233, 2024, doi: 10.29207/resti.v8i2.5652.

[17] I. M. Insan, J. H. Husen, and R. Yasirandi, "UML Modeling of Digital Envelope-Based Security System to Fulfill Security Services," *Indones. J. Comput. Sci.*, vol. 14, no. 5, 2025, doi: 10.33022/ijcs.v14i5.4986.

[18] F. T. Laksana, S. Palestina, and E. N. Anfa, "Implementasi Perancangan Infrastruktur Kunci Publik pada Pembuatan Surat Organisasi Digital Menggunakan Digital Signature," *J. Tiple A Pendidik. Teknol. Inf. dan Teknol. Inf.*, vol. 1, no. 2, pp. 91–96, 2022.

[19] R. A. K. N. Bintang, N. T. Romadloni, and F. Ramadhani, "Perbandingan Kinerja Algoritma Klasifikasi Pada Review Pengguna Aplikasi Netflix," *JITET (Jurnal Inform. dan Tek. Elektro Ter.*, vol. 13, no. 2, 2025, doi: 10.23960/jitet.v13i2.6303.

[20] W. Kurniawan, N. T. Romadloni, and R. A. K. N. Bintang, "Analisis Dampak Cache Progressive Web Apps Terhadap Konsumsi Baterai Android," *JITET (Jurnal Inform. dan Tek. Elektro Ter.*, vol. 13, no. 2, 2025, doi: 10.23960/jitet.v13i2.6221.