Implementasi Tanda Tangan Digital Pada Pembuatan Surat Permohonan KKP Dengan Metode AES Dan Deflate (Studi Kasus: Fakultas Teknik, UNISMUH Makassar)

SKRIPSI

Diajukan sebagai Salah Satu Syarat untuk
Mendapatkan Gelar Sarjana Komputer (S. Kom)
Program Studi Informatika

MUHAMMAD IKHSAN NUR
1058411100820

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH MAKASSAR
2025

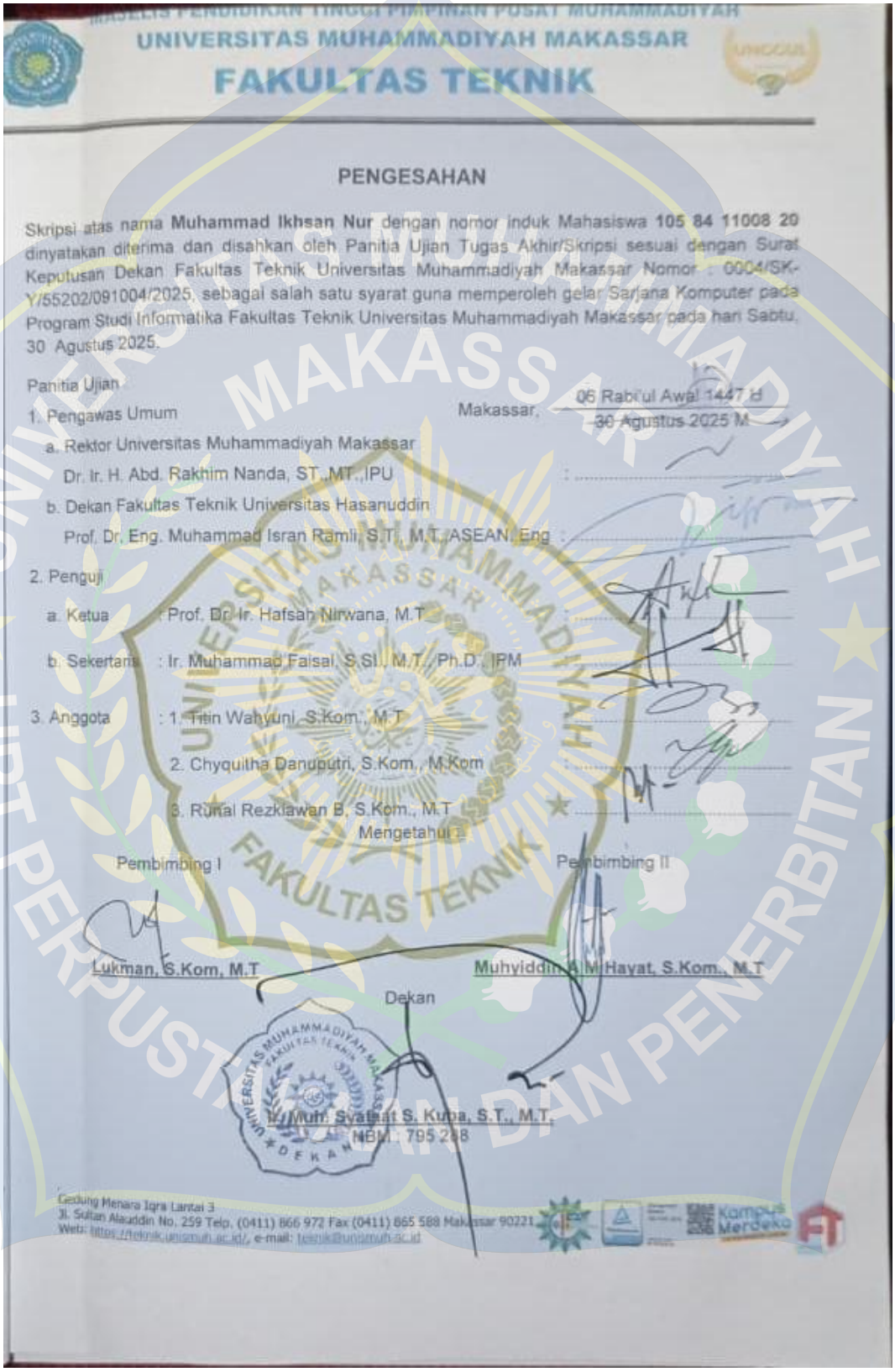
Implementasi Tanda Tangan Digital Pada Pembuatan Surat Permohonan KKP Dengan Metode AES Dan Deflate (Studi Kasus: Fakultas Teknik, UNISMUH Makassar)

Diajukan sebagai Salah Satu Syarat untuk
Mendapatkan Gelar Sarjana Komputer (S. Kom)
Program Studi Informatika

Disusun dan Diajukan oleh:

MUHAMMAD IKHSAN NUR
1058411100820

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH MAKASSAR
2025



MASLIS PENDIDIKAN TINGGI PIMPINAN PUSAT MUHAMMADIYAH
UNIVERSITAS MUHAMMADIYAH MAKASSAR
FAKULTAS TEKNIK

PENGESAHAN

Skripsi atas nama **Muhammad Ikhsan Nur** dengan nomor induk Mahasiswa **105 84 11008 20** dinyatakan diterima dan disahkan oleh Panitia Ujian Tugas Akhir/Skripsi sesuai dengan Surat Keputusan Dekan Fakultas Teknik Universitas Muhammadiyah Makassar Nomor : 0004/SK-Y/55202/091004/2025, sebagai salah satu syarat guna memperoleh gelar Sarjana Komputer pada Program Studi Informatika Fakultas Teknik Universitas Muhammadiyah Makassar pada hari Sabtu, 30 Agustus 2025.

Panitia Ujian

Makassar, 06 Rabi'ul Awal 1447 H / 30 Agustus 2025 M

1. Pengawas Umum
   a. Rektor Universitas Muhammadiyah Makassar
      Dr. Ir. H. Abd. Rakhim Nanda, ST.,MT.,IPU :
   b. Dekan Fakultas Teknik Universitas Hasanuddin
      Prof. Dr. Eng. Muhammad Isran Ramli, S.T., M.T. ASEAN. Eng :
2. Penguji
   a. Ketua : Prof. Dr. Ir. Hafsah Nirwana, M.T.
   b. Sekertaris : Ir. Muhammad Faisal, S.Si., M.T., Ph.D., IPM
3. Anggota : 1. Titin Wahyuni, S.Kom., M.T.
   2. Chyquitha Danuputri, S.Kom., M.Kom
   3. Rinal Rezklawan B, S.Kom., M.T

Mengetahui :

Pembimbing I	Pembimbing II
Lukman, S.Kom, M.T	Muhyiddin A M Hayat, S.Kom., M.T

Dekan

Ir. Muh. Syafaat S. Kuba, S.T., M.T.
NBM : 795 288

Gedung Menara Iqra Lantai 3
Jl. Sultan Alauddin No. 259 Telp. (0411) 866 972 Fax (0411) 865 588 Makassar 90221
Web: https://teknik.unismuh.ac.id/, e-mail: teknik@unismuh.ac.id

MAJELIS PENDIDIKAN TINGGI PIMPINAN PUSAT MUHAMMADIYAH
UNIVERSITAS MUHAMMADIYAH MAKASSAR
FAKULTAS TEKNIK

HALAMAN PENGESAHAN

Tugas Akhir ini diajukan untuk memenuhi syarat ujian guna memperoleh gelar Sarjana Komputer (S.Kom) Program Studi Informatika Fakultas Teknik Universitas Muhammadiyah Makassar.

Judul Skripsi : **Implementasi Tanda Tangan Digital Pada Pembuatan Surat Permohonan KKP Dengan Metode AES dan Deflate (Studi Kasus: Fakultas Teknik, UNISMUH Makassar)**

Nama : Muhammad Ikhsan Nur
Stambuk : 105 84 11008 20

Makassar, 30 Agustus 2025

Telah Diperiksa dan Disetujui
Oleh Dosen Pembimbing:

Pembimbing I	Pembimbing II
Lukman, S.Kom, M.T	**Muhyiddin A M Hayat, S.Kom., M.T**

Mengetahui,
Ketua Prodi Informatika

Rizki Yusliana Bakti, S.T., M.T.
NBM : 1307 284

Gedung Menara Iqra Lantai 3
Jl. Sultan Alauddin No. 259 Telp. (0411) 866 972 Fax (0411) 865 588 Makassar 90221
Web: https://teknik.unismuh.ac.id/, e-mail: teknik@unismuh.ac.id

MOTTO DAN PERSEMBAHAN

Motto

"Setiap langkah kecil hari ini adalah bagian dari pencapaian besar esok hari."

Persembahan

Dengan penuh rasa syukur ke hadirat Allah SWT atas limpahan rahmat, taufik, serta hidayah-Nya, akhirnya penulis dapat menyelesaikan skripsi ini dengan segala keterbatasan yang ada. Karya sederhana ini saya persembahkan kepada Ayahanda Bapak Halik dan Ibunda Ibu Marta Rahimahallah, yang do'a, kasih sayang, dan pengorbanannya tiada henti menjadi kekuatan terbesar dalam setiap langkah hidup saya. Kepada keluarga besar, khususnya nenek tercinta Ibu Sitti Hasnah, yang senantiasa memberikan semangat dan do'a tulus, kehadiran kalian selalu menjadi alasan untuk terus berjuang dan tidak menyerah. Kepada dosen pembimbing Lukman, S.Kom, MT. dan Muhyiddin A M Hayat, S.Kom., M.T. dan seluruh pengajar yang dengan sabar membagikan ilmu serta memberikan arahan, bimbingan, dan nasihat berharga yang menjadi bekal penting dalam proses penyusunan karya ini. Juga kepada sahabat seperjuangan, khususnya Wildan dan Aidhil, yang setia menemani, memberikan dukungan moral, serta menghadirkan kebersamaan yang penuh makna selama masa perkuliahan hingga selesainya skripsi ini. Semoga karya ini dapat menjadi ungkapan kecil dari rasa hormat, cinta, dan terima kasih saya yang mendalam kepada semua pihak yang telah menjadi bagian penting dalam perjalanan hidup dan pendidikan saya.

ABSTRAK

MUHAMMAD IKHSAN NUR, "Implementasi Tanda Tangan Digital Pada Pembuatan Surat Permohonan KKP Dengan Metode AES Dan Deflate (Studi Kasus: Fakultas Teknik, UNISMUH Makassar)" (dibimbing oleh Lukman, S.Kom, M.T dan Muhyiddin A.M. Hayat, S.Kom., M.T).

Perkembangan transformasi digital di lingkungan akademik menuntut adanya sistem verifikasi dokumen yang andal dan aman. Penelitian ini mengimplementasikan sistem tanda tangan digital pada proses verifikasi Surat Permohonan Kuliah Kerja Profesi (KKP) di Fakultas Teknik, Universitas Muhammadiyah Makassar. Sistem dirancang menggunakan algotirma *Advanced Encryption Standard (AES)* dengan mode *Cipher Block Chaining (CBC)* 128-bit untuk proses enkripsi serta algoritma Deflate untuk kompresi data. Integrasi AES dan Deflate mampu menjaga kerahasiaan, integritas, dan keaslian dokumen, sekaligus mengoptimalkan efisiensi penyimpanan dan pengiriman. Data yang telah dienkripsi kemudian dikonversi menjadi *QR Code*, sehingga dapat dipindai dan didekripsi melalui aplikasi *mobile* secara *offline*. Pengujian sistem menggunakan metode *black-box* menunjukkan bahwa tanda tangan digital berhasil mempercepat proses verifikasi administrasi, mengurangi potensi kesalahan manusia, serta memastikan keaslian dokumen akademik. Penelitian ini membuktikan bahwa tanda tangan digital berbasis AES dan Deflate dapat meningkatkan efisiensi, keandalan, dan keamanan verifikasi dokumen di institusi akademik.

Kata kunci: Tanda Tangan Digital, AES, Deflate, Enkripsi, Dokumen Akademik.

ABSTRACT

MUHAMMAD IKHSAN NUR, "*Implementation of Digital Signature in the Creation of Internship Request Letters Using AES and* Deflate *Methods (Case Study: Faculty of Engineering, UNISMUH Makassar)*" (*supervised* by Lukman, S.Kom, M.T and Muhyiddin A.M. Hayat, S.Kom., M.T).

The rapid development of digital transformation in academic environments requires a reliable and secure system for document verification. This research implements a digital signature system for the verification of Surat Permohonan Kuliah Kerja Profesi (KKP) at the Faculty of Engineering, Universitas Muhammadiyah Makassar. The system applies the Advanced Encryption Standard (AES) algorithm with 128-bit Cipher Block Chaining (CBC) mode for encryption and the Deflate *algorithm for data compression. The integration of AES and* Deflate *ensures document confidentiality, integrity, and authenticity, while also optimizing storage and transmission efficiency. Data is encrypted and converted into a QR Code, which can be scanned and decrypted on mobile applications, enabling offline verification. System testing using black-box methods demonstrated that the digital signature system successfully accelerated administrative verification, reduced human error, and maintained the authenticity of academic documents. This research confirms that digital signatures with AES and* Deflate *can significantly improve the efficiency, reliability, and security of document verification processes in academic institutions.*

Keywords: *Digital Signature, AES,* Deflate*, Encryption, Academic Documents.*

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Dengan penuh rasa syukur atas kehadirat Allah Subhanallahu Wa Ta'ala, atas nikmat iman, Islam, dan juga nikmat Kesehatan yang senantiasa terlimpahkan. Sehingga penulis dapat menyelesaikan Laporan Tugas Akhir yang berjudul **"Implementasi Tanda Tangan Digital Pada Pembuatan Surat Permohonan KKP Dengan Metode AES Dan Deflate (Studi Kasus: Fakultas Teknik, UNISMUH Makassar)"** Tak lupa pula sholawat serta salam kepada junjungan Nabi Muhammad SAW, sang pembawa rahmat bagi semesta alam. Yang telah membawa kita dari Zaman jahiliyah menuju Zaman yang penuh berkah seperti sekarang ini.

Ucapan terima kasih yang tidak terhingga penulis sampaikan kepada semua yang telah membantu dan memberikan dukungan dalam penyusunan Laporan Tugas Akhir ini, khususnya:

1. Kepada kedua Orang Tua saya tercinta, Bapak **Halik** dan Ibu **Marta** Rahimahallah, persembahan terima kasih yang tak terhingga atas segala pengorbanan, kasih sayang, dan bimbingan yang telah diberikan.
2. Nenek saya tercinta, Ibu **Sitti Hasnah**, yang selalu memberikan bantuan moral dan moril serta kasih sayang tanpa batas.
3. Bapak **Dr. Ir. H. Abd. Rakhim Nanda, S.T., M.T., IPU.,** selaku Rektor Universitas Muhammadiyah Makasar.
4. Bapak **Muh. Syafaat S. Kuba, S.T., M.T.,** selaku Dekan Fakultas Teknik Universitas Muhammadiyah Makassar.
5. Ibu **Rizki Yusliana Bakti, S.T., M.T.,** selaku Ketua Prodi Informatika Universitas Muhammadiyah Makassar.
6. Bapak **Lukman, S.Kom, M.T,** selaku Dosen Pembimbing 1.
7. Bapak **Muhyiddin A M Hayat S.Kom., M.T,** selaku Dosen Pembimbing 2.
8. Seluruh Jajaran Dosen dan Staff Fakultas Teknik Universitas Muhammadiyah Makassar.

9. Teman-teman saya yang khususnya Angkatan 2020 Fakultas Teknik Universitas Muhammadiyah Makassar.
10. Teman-teman kelas A angkatan 2020 Program Studi Informatika Universitas Muhammadiyah Makassar.
11. Teman baik saya **Ahmad Wildan Dzakki Adam dan Aidhil Prima Abdiguna**, yang telah banyak memberikan bantuan maupun beban sebelum, saat, maupun setelah pembuatan proposal ini.
12. Senior sejurusan saya Kak **Andi Agung Dwi Arya, S.Kom,** atas bantuannya sejak awal saya masuk ke jurusan ini.
13. Dan juga orang-orang yang tidak sempat saya sebutkan namanya satu-persatu, terima kasih atas segala bantuan, semangat, dan do'anya.

Penulis menyadari bahwa Laporan Tugas Akhir ini masih jauh dari kesempurnaan. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan demi penyempurnaan laporan ini di masa depan. Harapan penulis, semoga Laporan Tugas Akhir ini dapat memberikan manfaat bagi penyandang disabilitas tunanetra dalam meningkatkan kemandirian dan kualitas hidup mereka. Akhir kata, tunanetra dalam meningkatkan kemandirian dan kualitas hidup mereka. Akhir kata, penulis mohon maaf atas segala kekurangan dan kekhilafan yang terdapat dalam Laporan Tugas Akhir ini.

Billahi fisabililhaq, fastabiqul khairat.
Wassalamualaikum Wr.Wb.

Makassar, Februari 2025

Penulis,

MUHAMMAD IKHSAN NUR

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

DAFTAR ISTILAH

Advanced Encryption Standard	AES merupakan standar enkripsi simetris yang digunakan secara luas untuk mengamankan data digital, dalam skripsi ini digunakan mode AES-128-CBC.
Cipher Block Chaining	CBC adalah mode operasi pada algoritma kriptografi blok yang menghubungkan setiap blok *ciphertext* dengan blok sebelumnya agar lebih aman.
Deflate	Algoritma kompresi data yang digunakan untuk memperkecil ukuran data tanpa menghilangkan informasi.
Inflate	Proses dekompresi data untuk mengembalikan data terkompresi ke bentuk aslinya.
CryptoJS	*Library* JavaScript yang digunakan untuk proses enkripsi dan dekripsi berbasis AES pada sisi klien.
Pako	*Library* JavaScript yang digunakan untuk proses kompresi (Deflate) dan dekompresi (Inflate).
Initialization Vector	IV Adalah nilai awal acak yang digunakan bersama kunci enkripsi untuk memastikan hasil *ciphertext* berbeda meskipun *plaintext* sama.
Ciphertext	Hasil data yang telah dienkripsi sehingga tidak dapat dibaca langsung tanpa proses dekripsi.
Plaintext	Data asli sebelum diproses melalui enkripsi.

Base64	Format encoding data biner menjadi teks ASCII agar dapat ditransmisikan melalui media teks.
WordArray	Format internal yang digunakan oleh CryptoJS untuk merepresentasikan data biner.
Quick Response Code	*QR Code* adalah kode dua dimensi yang dapat menyimpan data terenkripsi dan dapat dipindai untuk validasi.
Digital Signature	Mekanisme untuk menjamin keaslian, integritas, dan autentikasi data dalam dokumen digital.
Hashing	Proses mengubah data menjadi nilai tetap dengan algoritma tertentu yang tidak dapat dikembalikan ke bentuk aslinya.
Public Key Infrastructure	PKI adalah Kerangka kerja yang digunakan untuk mengelola kunci publik dan sertifikat digital agar sistem tanda tangan digital lebih aman.
Black-Box Testing	Metode pengujian perangkat lunak yang berfokus pada masukan dan keluaran sistem tanpa memperhatikan struktur internal program.

BAB I
PENDAHULUAN

A. Latar Belakang

Dalam proses pembuatan Surat Permohonan Kuliah Kerja Profesi (KKP) di Fakultas Teknik Universitas Muhammadiyah Makassar, staf administrasi sering menghadapi tantangan dalam memverifikasi keaslian surat. Proses verifikasi masih dilakukan secara manual, yaitu dengan memeriksa data secara online melalui komputer, sehingga menyebabkan keterlambatan pelayanan dan meningkatkan potensi kesalahan manusia. Penerapan tanda tangan digital menjadi salah satu solusi yang dapat meningkatkan efisiensi dan akurasi proses ini (Fatma dkk., 2023).

Tanda tangan digital merupakan skema matematis yang memungkinkan validasi keaslian suatu dokumen elektronik tanpa perlu pemeriksaan manual. Keunggulan utama tanda tangan digital adalah kemampuannya dalam menjamin integritas dan autentikasi dokumen, serta melindungi data dari pemalsuan (Pemayun & Dewi, 2025). Dalam implementasinya, tanda tangan digital bekerja dengan kriptografi asimetris, yang memanfaatkan kunci privat dan kunci publik untuk mengamankan dokumen.

Dalam konteks keamanan dokumen elektronik, algoritma *Advanced Encryption Standard (AES)* menjadi pilihan yang efektif untuk melindungi informasi dari akses tidak sah. AES adalah algoritma enkripsi simetris yang menggunakan ukuran kunci 128, 192, atau 256 bit, sehingga memberikan tingkat keamanan yang tinggi (Fahlevvi dkk., 2025). Selain itu, teknik kompresi Deflate dapat diterapkan sebelum enkripsi untuk mengurangi ukuran file, sehingga mempercepat proses enkripsi dan dekripsi tanpa mengurangi keabsahan data (Sujono dkk., 2020).

Dengan mengintegrasikan tanda tangan digital berbasis AES dan kompresi Deflate, proses verifikasi keaslian surat dapat dilakukan secara *offline*, tanpa memerlukan akses langsung ke komputer. Hal ini dapat mempercepat pelayanan, mengurangi risiko kesalahan, serta memastikan dokumen tetap aman dan dapat dipertanggungjawabkan secara hukum (Indriani dkk., 2024). Penerapan sistem ini sejalan dengan transformasi digital di lingkungan akademik, meningkatkan

efisiensi operasional, serta memastikan keabsahan dokumen secara elektronik (Fahlevvi dkk., 2025).

B. Rumusan Masalah

1. Bagaimana penerapan metode *Advanced Encryption Standard (AES)* dan Deflate dalam memastikan keamanan serta keabsahan tanda tangan digital pada dokumen akademik?
2. Bagaimana merancang dan mengembangkan aplikasi *mobile* untuk tanda tangan digital guna mempercepat proses verifikasi keaslian Surat permohonan KKP di Fakultas Teknik UNISMUH Makassar?
3. Seberapa efektif penerapan tanda tangan digital dalam mengurangi keterlambatan proses administrasi dibandingkan dengan metode verifikasi konvensional?

C. Tujuan Penelitian

1. Merancang dan mengembangkan aplikasi *mobile* yang mendukung penggunaan tanda tangan digital guna mempercepat proses verifikasi keaslian Surat Permohonan KKP di Fakultas Teknik UNISMUH Makassar.
2. Menganalisis dan mengimplementasikan metode *Advanced Encryption Standard (AES)* dan Deflate dalam sistem tanda tangan digital untuk memastikan keamanan, integritas, dan keabsahan dokumen akademik.
3. Mengevaluasi efektivitas penerapan tanda tangan digital dalam mengurangi keterlambatan proses administrasi dibandingkan dengan metode verifikasi konvensional, baik dari segi kecepatan, efisiensi, maupun tingkat keamanannya.

D. Manfaat Penelitian

Manfaat dari penelitian ini adalah:

1. Bagi penulis, penelitian ini akan memberikan pemahaman yang mendalam tentang algoritma *Advanced Encryption Standard (AES) dan* Deflate, serta bagaimana penerapannya dalam tanda tangan digital, serta memberikan pemahaman dalam menerapkan teknologi ke dalam sistem penyuratan.
2. Bagi Universitas, hasil penelitian ini dapat meningkatkan reputasi Universitas sebagai Lembaga yang mendukung inovasi dalam penerapan

tanda tangan digital dan juga dapat sebagai sumber referensi untuk penelitian selanjutnya yang berkaitan dengan *Advanced Encryption Standard (AES)* dan Deflate serta tanda tangan digital.

3. Bagi pembaca, memberikan pemahaman tentang algoritma *Advanced Encryption Standard (AES)* dan Deflate serta bagaimana algoritma ini bekerja ketika digunakan dalam sistem tanda tangan digital, juga sebagai referensi untuk mengeksplorasi lebih lanjut tentang penelitian ini.

E. Ruang Lingkup Penelitian

Agar penelitian tetap berada pada jalur yang telah ditentukan, maka penelitian ini dibuatkan ruang lingkup penelitian, diantaranya yaitu:

1. Cara penerapan algoritma aes-128-cbc dan Deflate untuk tanda tangan digital.
2. Mengevaluasi efektifitas tanda tangan digital ketika digunakan dalam sistem penyuratan akademik.

F. Sistematika Penulisan

Secara umum, laporan tugas akhir ini disusun dalam lima bab, yaitu:

BAB I PENDAHULUAN

Berisi latar belakang, rumusan masalah, tujuan, manfaat, batasan, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Memaparkan teori dan kajian literatur yang relevan sebagai dasar penelitian.

BAB III METODE PENELITIAN

Menjelaskan metode penelitian serta perangkat yang digunakan dalam pembuatan sistem.

BAB IV HASIL PENELITIAN

Menyajikan hasil implementasi sistem serta pengujian.

BAB V PENUTUP

Memuat kesimpulan dari penelitian yang telah dilakukan.

BAB II
TINJAUAN PUSTAKA

A. Landasan Teori

1. Tanda Tangan Digital

Tanda tangan digital merupakan skema matematis yang digunakan untuk memverifikasi keaslian dan integritas suatu dokumen elektronik. Teknologi ini berfungsi sebagai mekanisme autentikasi yang dapat memastikan bahwa dokumen yang dikirim berasal dari pengirim yang sah dan tidak mengalami perubahan selama proses transmisi. Tanda tangan digital menggunakan algoritma kriptografi, khususnya kriptografi asimetris, yang melibatkan penggunaan pasangan kunci privat dan kunci publik. Kunci privat digunakan untuk menandatangani dokumen, sementara kunci publik digunakan untuk memverifikasi tanda tangan tersebut (Pemayun & Dewi, 2025).

2. Enkripsi dan Dekripsi

Enkripsi adalah sebuah proses untuk mengamankan data dengan menyembunyikannya atau mengubah data menjadi bentuk yang tidak dapat dibaca atau dimengerti (Nirmala, 2020). Enkripsi telah dimanfaatkan untuk melindungi komunikasi di berbagai negara, namun hanya organisasi tertentu dan individu dengan kebutuhan yang sangat mendesak akan kerahasiaan yang menggunakannya.

Sedangkan dekripsi adalah kebalikan dari enkripsi. Dekripsi diartikan sebagai proses mengubah data yang telah dienkripsi menjadi data aslinya sehingga dapat dibaca atau dimengerti kembali (Nirmala, 2020). Pesan yang akan dienkripsi disebut *plaintext* dan dilambangkan sebagai P, proses enkripsi dilambangkan sebagai E, proses dekripsi dilambangkan sebagai D, dan pesan yang telah dienkripsi disebut *ciphertext* dan dilambangkan sebagai C.

3. Kriptografi

Kriptografi awalnya didefinisikan sebagai ilmu yang mempelajari cara menyembunyikan pesan. Namun, dalam pengertian modern kriptografi adalah ilmu yang didasarkan pada teknik matematika untuk menangani keamanan informasi, termasuk kerahasiaan, keutuhan data dan otentikasi entitas (Fauzi,

2024). Dengan demikian, pengertian kriptografi secara modern tidak hanya berkaitan dengan penyembunyian pesan, tetapi juga melibatkan sekumpulan Teknik untuk menyediakan keamanan informasi.

4. Algoritma *Advanced Encryption Stanadard (AES)*

Advanced Encryption Standard (AES) adalah algoritma enkripsi simetris yang dikembangkan untuk menggantikan algoritma *Data Encryption Standard (DES*) yang lebih lama dan kurang aman. AES menggunakan ukuran kunci 128, 192, atau 256 bit dan banyak digunakan dalam berbagai aplikasi keamanan, termasuk tanda tangan digital. Algoritma ini memiliki keunggulan dalam kecepatan dan ketahanan terhadap serangan kriptografi, sehingga menjadikannya pilihan yang tepat untuk mengamankan data dalam dokumen digital (Fahlevvi dkk., 2025).

Enkripsi dalam AES dinyatakan sebagai:

$$C = E_k(P)$$

di mana C adalah *ciphertext*, P adalah *plaintext*, E_k dan adalah fungsi enkripsi dengan menggunakan kunci .

AES bekerja dengan memproses data dalam blok berukuran 128-bit menggunakan kunci kriptografi. Algoritma ini terdiri dari beberapa langkah yang diulang dalam beberapa putaran, tergantung pada panjang kunci (128-bit menggunakan 10 putaran). AES menggunakan struktur yang disebut Substitusi-Permutasi Jaringan (SPN) yang mencakup substitusi data, permutasi, dan transformasi matematika. AES terdiri dari 4 langkah utama di setiap putaran:

a. ***SubBytes***: Setiap *byte* dalam blok *plaintext* digantikan dengan *byte* lain menggunakan tabel substitusi yang disebut *S-Box*.
b. ***ShiftRows***: Baris-baris matriks data digeser secara siklikal untuk menghasilkan difusi.

c. ***MixColumns***: Setiap kolom data dikombinasikan menggunakan transformasi linier (pada bidang Galois) untuk menyebarkan informasi dalam data.
d. ***AddRoundKey***: Blok data XOR dengan kunci yang diperoleh dari *Key Expansion*. (Irvai dkk., 2024)

Berikut adalah perbedaan antara AES-128, AES-192, dan AES-256:

Tabel 1. Perbedaan AES-128, AES-192, dan AES-256

Perbandingan	AES-128	AES-192	AES-256
Panjang Kunci	128-*bit*	192-*bit*	256-*bit*
Jumlah Putaran (*Rounds*)	10 Round	12 Round	14 Round
Kecepatan Enkripsi	Cepat	Sedang	Lebih Lambat

Tabel 1 menunjukkan perbandingan tiga varian utama dari algoritma *Advanced Encryption Standard (AES)* berdasarkan panjang kunci, jumlah putaran (*rounds*), serta kecepatan enkripsi.

a. Panjang Kunci
    1) AES-128 menggunakan kunci sepanjang 128-bit, AES-192 menggunakan 192-bit, dan AES-256 menggunakan 256-bit.
    2) Semakin panjang kunci yang digunakan, semakin tinggi pula tingkat keamanan yang diberikan, karena ruang kemungkinan kombinasi kunci semakin besar dan lebih sulit untuk ditembus.
b. Jumlah Putaran (Rounds)
    1) AES-128 membutuhkan 10 putaran, AES-192 membutuhkan 12 putaran, dan AES-256 membutuhkan 14 putaran.
    2) Putaran ini adalah proses internal yang dilakukan untuk mengacak data (substitusi, permutasi, pergeseran baris, dan perkalian matriks) sehingga *ciphertext* lebih sulit diprediksi.
    3) Semakin banyak putaran, semakin tinggi tingkat kompleksitas enkripsi dan semakin sulit serangan *brute force* dilakukan.

c. Kecepatan Enkripsi

1) AES-128 relatif lebih cepat dibanding varian lainnya karena jumlah putaran lebih sedikit.
2) AES-192 memiliki kecepatan sedang, sedangkan AES-256 lebih lambat karena jumlah putaran lebih banyak dan panjang kunci lebih besar.
3) Meskipun lebih lambat, AES-256 umumnya dipilih ketika tingkat keamanan menjadi prioritas utama, misalnya untuk aplikasi yang membutuhkan perlindungan data jangka panjang.

5. Algoritma Deflate

Algoritma Deflate adalah metode kompresi data yang menggabungkan algoritma LZ77 dan *Huffman coding*. Teknik ini digunakan untuk mengurangi ukuran file tanpa mengurangi keakuratan data. Dalam konteks tanda tangan digital, Deflate dapat digunakan sebelum proses enkripsi untuk mempercepat proses pengolahan data serta mengoptimalkan penyimpanan dan pengiriman dokumen secara efisien (Sujono dkk., 2020).

a. LZ77 menggunakan pencocokan *string* berulang dalam *buffer sliding window* untuk menggantikan data dengan referensi ke sebelumnya.
b. *Huffman Coding* membangun pohon Huffman dengan probabilitas karakter, lalu mengganti simbol dengan kode biner pendek untuk karakter yang sering digunakan.

Panjang rata-rata kode Huffman dapat dihitung dengan rumus:

$$L = \sum_{i=1}^{n} P_i \times l_i$$

di mana L adalah panjang rata-rata kode P_i adalah probabilitas karakter ke-i, dan l_i adalah panjang bit kode Huffman.

6. Algoritma Inflate

Algoritma Inflate merupakan kebalikan dari proses Algoritma Deflate, yaitu mengembalikan data yang telah dikompresi ke bentuk aslinya tanpa kehilangan informasi. Dengan sifat *lossless*, Inflate menjamin bahwa data hasil kompresi sebelumnya dapat dipulihkan sepenuhnya.

7. *Library* Pako

Dalam implementasi sistem ini, baik proses Deflate maupun Inflate diimplementasikan menggunakan *library* Pako pada lingkungan JavaScript. Pako dipilih karena kemampuannya yang efisien dalam melakukan kompresi dan dekompresi berbasis Deflate/Inflate, serta kompatibilitas tinggi dengan format data modern. Hal ini mendukung kebutuhan sistem tanda tangan digital untuk menjaga ukuran data tetap kecil, tetapi dapat dipulihkan secara penuh ketika diperlukan.

Alur kerja Pako secara umum adalah sebagai berikut:

a. Data asli (*plaintext*) diubah menjadi representasi *byte stream* agar dapat diproses.
b. Pada tahap kompresi (Deflate), Pako menggabungkan dua teknik utama, yaitu algoritma LZ77 untuk menemukan pola berulang dalam data, dan *Huffman Coding* untuk memberikan kode biner yang lebih pendek pada data yang sering muncul.
c. Hasil dari proses Deflate berupa data terkompresi yang ukurannya lebih kecil dibandingkan data awal.
d. Pada tahap dekompresi (Inflate), Pako melakukan pembalikan proses dengan membaca kode *Huffman*, lalu membangun kembali data asli berdasarkan referensi dari pola berulang yang sebelumnya dikompresi.
e. Data hasil dekompresi identik dengan data awal, sehingga tidak ada informasi yang hilang.

Dengan demikian, Pako menjamin bahwa proses kompresi dan dekompresi bersifat *lossless* (tanpa kehilangan data), serta memberikan efisiensi yang penting bagi sistem tanda tangan digital yang harus mengelola data dengan cepat dan tetap menjaga keutuhan informasi.

B. Penelitian Terkait

Hasil penelitian yang terkait mencakup ringkasan isi penelitian yang telah diterbitkan sebelumnya, baik dari segi kelebihan maupun kekurangan. Hasil ini dikumpulkan dari jurnal, makalah, atau penelitian dan dimaksudkan untuk

memberikan masukan kepada peneliti saat mereka membuat kerangka kerja penelitian mereka.

1. Sujono dkk. (2020) membahas penggunaan algoritma kompresi LZSS yang dimodifikasi untuk meningkatkan efisiensi pengelolaan arsip digital. Penelitian ini menunjukkan bahwa metode kompresi tersebut efektif memperkecil ukuran file, sehingga dapat mempercepat proses penyimpanan dan pengiriman data, serta berpotensi dikombinasikan dengan algoritma enkripsi seperti AES untuk pengamanan yang lebih optimal.
2. Fatma dkk. (2023) meneliti penerapan tanda tangan digital dalam proses verifikasi dan validasi dokumen surat bebas Covid-19. Studi ini menunjukkan bahwa digital *signature* yang dikombinasikan dengan teknologi *QR Code* mampu memastikan keaslian dokumen serta mencegah pemalsuan, sehingga sangat efektif diterapkan dalam sistem administrasi yang membutuhkan validitas tinggi.
3. Indrayani dkk. (2024) mengevaluasi implementasi algoritma AES-256 *bit* untuk pengamanan file berbagai format. Hasil penelitian menunjukkan bahwa meskipun proses enkripsi menyebabkan perubahan pada *metadata* dan visual file, AES-256 tetap mampu menjaga integritas data tanpa menyebabkan peningkatan ukuran file yang signifikan, menjadikannya metode yang andal dalam transformasi digital di berbagai lingkungan, termasuk akademik.
4. Fahlevvi dkk. (2025) mengkaji penerapan algoritma AES-128-CBC dalam pengamanan dokumen digital pada lingkungan perusahaan logistik. Penelitian ini menegaskan bahwa AES mampu melindungi data dari akses tidak sah dan serangan *brute force*, serta lebih unggul dalam menjaga integritas dokumen dibandingkan metode *hash* seperti MD5 dan SHA-256.

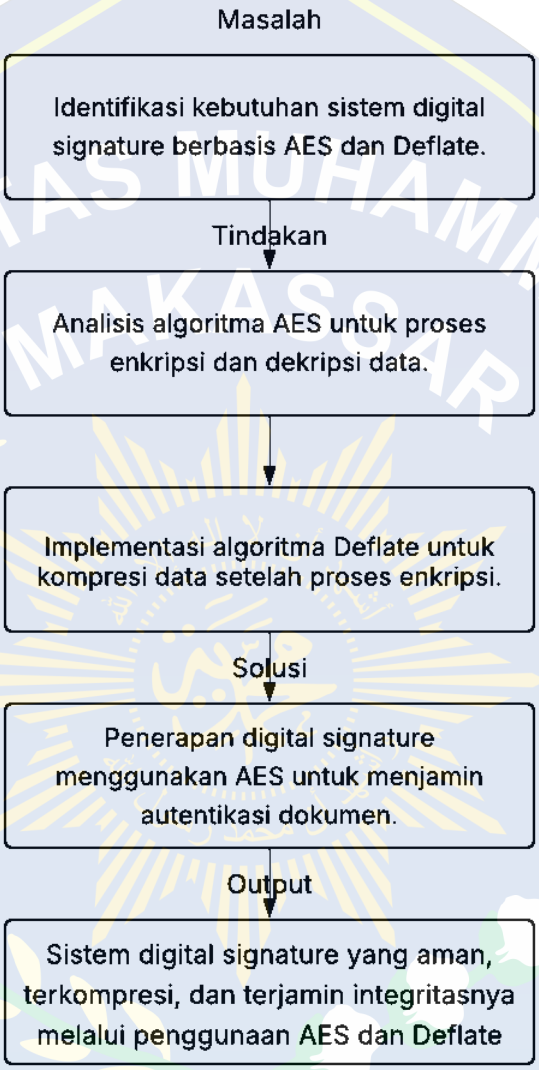
Dari hasil tersebut, peneliti memperhatikan beberapa keunggulan maupun kekurangan dari metode yang digunakan ringkasan penelitian sebelumnya meliputi:

Tabel 2. Penelitian Terkait

Judul Penelitian	Metode/Algoritma	Keunggulan	Kekurangan
Pengelolaan Arsip Secara Digital Menggunakan Algoritma LZSS Modifikasi untuk Kompresi File (Sujono dkk., 2020)	Algoritma kompresi LZSS Modifikasi	Memberikan hasil kompresi lebih baik daripada LZSS biasa.	Tidak menggabungkan fungsi keamanan seperti enkripsi atau tanda tangan digital.
Aplikasi Tandatangan Digital dalam Proses Verifikasi dan Validasi Sertifikat Covid-19 (Fatma dkk., 2023)	SHA-1 dan RSA	Menerapkan konsep *authentication* dan *integrity* melalui digital signature.	Menggunakan algoritma hash SHA-1 yang relatif sudah usang dan rawan *collision*.
Analisis Penggunaan Kriptografi Metode AES 256 Bit pada Pengamanan File dengan Berbagai Format (Indrayani dkk., 2024)	AES-256 *bit*	AES-256 memiliki tingkat keamanan lebih tinggi dibanding AES-128.	Tidak menyertakan implementasi *digital signature* atau *QR Code*.

ALGORITMA AES128-CBC (ADVANCED ENCRYPTION STANDARD) UNTUK ENKRIPSI DAN DEKRIPSI BERKAS DOKUMEN PT. ADIARTA MUZIZAT (Fahlevvi dkk., 2025)	AES-128-cbc	Ukuran file terenkripsi bisa membesar signifikan karena proses AES (hingga 10x dari ukuran aslinya).	Implementasi sistem berbasis web menggunakan PHP dan MariaDB.

C. Kerangka Pikir



Gambar 1. Kerangka Pikir

BAB III
METODE PENELITIAN

A. Tempat dan Waktu Penelitian

1. Tempat Penelititan

Penelitian ini dilakukan di Fakultas Teknik Universitas Muhammadiyah Makassar.

2. Waktu Penelitian

Adapun pelaksanaan ini dilakukan selama bulan Mei – Agustus 2025.

B. Alat dan Bahan

Adapun alat dan bahan yang akan digunakan dalam penelitian ini adalah sebagai berikut:

1. Kebutuhan *Hardware* (Perangkat Keras)
    a. Laptop, untuk spesifikasi yang digunakan dalam penelitian ini yaitu menggunakan merk Acer Aspire 3 A314-22 dengan spesifikasi *Processor AMD Ryzen* 3 3250U *with Radeon Graphics* 2.60 GHz, RAM 12 GB, SSD *Internal* 256 GB, dan OS Windows 11 *Home*.
    b. *Smartphone Android* Xiaomi Redmi Note 9 dengan spesifikasi RAM 4GB dengan penyimpanan 64GB
2. Kebutuhan *Sofware* (Perangkat Lunak)
    a. *Visual Studio Code* (sebagai *software* utama untuk merancang sistem)
    b. PostgreSQL (sebagai *database fields* yang ada)
    c. ExpoGo (untuk menjalankan dan menguji aplikasi pada *smartphone*)
3. Kebutuhan *Dataset*

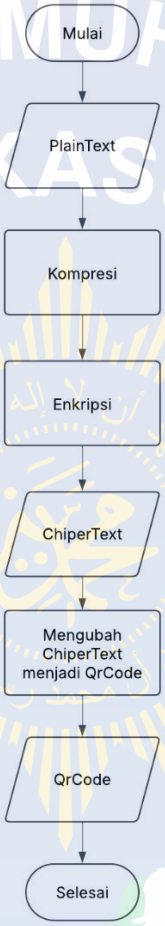
    Data yang digunakan untuk penelitian ini adalah data yang bersumber dari surat Permohonan KKP(Kuliah, Kerja, Praktek) di Fakultas Teknik, Universitas Muhammadiyah Makassar.

C. Perancangan Sistem

1. *Flowchart* Sistem

Untuk memudahkan proses perancangan, peneliti menggunakan *flowchart* untuk membuat perancangan sistem, seperti yang ditunjukkan pada gambar berikut:

a. Proses Kompresi dan Enkripsi

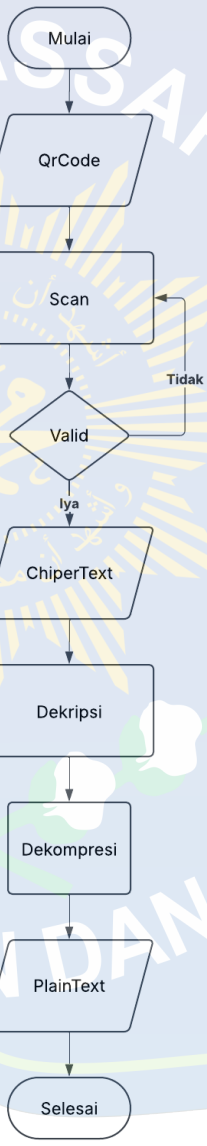


Gambar 2. Flowchart Sistem Kompresi dan Enkripsi

1) Mulai: Proses dimulai.
2) *PlainText*: Data yang berasal dari surat Permohonan KKP(Kuliah, Kerja, Praktek) dalam bentuk asli.
3) Kompresi: Data dikompresi dengan algoritma Deflate untuk mengurangi ukurannya.
4) Enkripsi: Data dienkripsi dengan algoritma aes-128-cbc untuk mendapatkan *ChiperText*.
5) *CipherText*: Hasil dari proses enkripsi berupa teks acak.

6) Mengubah *ciphertext* menjadi *QR Code*: *Ciphertext* dikonversi menjadi *QR Code* dengan *library qrcode*.
7) *QR Code*: Hasil akhir berupa *QR Code* yang dapat dipindai untuk dekripsi nantinya.
8) Selesai: Proses kompresi dan enkripsi selesai.

b. Proses Dekripsi dan Dekompresi



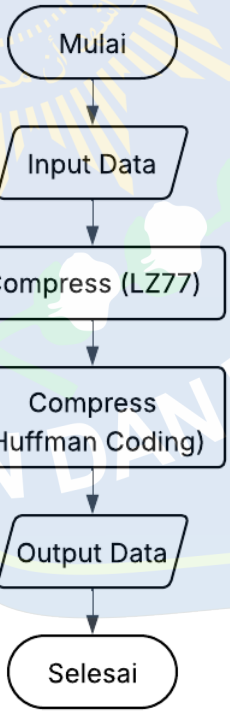
Gambar 3. Flowchart Sistem Dekripsi dan Dekmporesi

1) Mulai: Proses dekripsi dimulai.
2) *QR Code*: *QR Code* yang telah dibuat diambil untuk diproses.

3) *Scan*: *QR Code* di *scan* untuk mendapatkan *ciphertext*.
4) Validasi: Mengecek apakah *QR Code* valid:
5) Jika *chipertext* tidak valid, kembali ke proses *scan*.
6) Jika *chipertext* valid, lanjut ke langkah berikutnya.
7) *Ciphertext*: Teks yang masih terenkripsi diperoleh dari *scan QR Code*.
8) Dekripsi: *Ciphertext* didekripsi menggunakan algotirma aes-128-cbc untuk mendapatkan *Plaintext*.
9) Dekompresi: *Plaintext* didekompresi untuk mendapatkan data asli.
10) *Plaintext*: Data asli berhasil diperoleh kembali.
11) Selesai: Proses dekripsi dan dekompresi selesai.

2. *Flowchart* algoritma Deflate dalam mengompres data.

Untuk lebih mudah memahami bagaimana proses kompresi dengan algoritma Deflate, peneliti menggunakan *flowchart* seperti yang ditunjukkan pada gambar berikut:



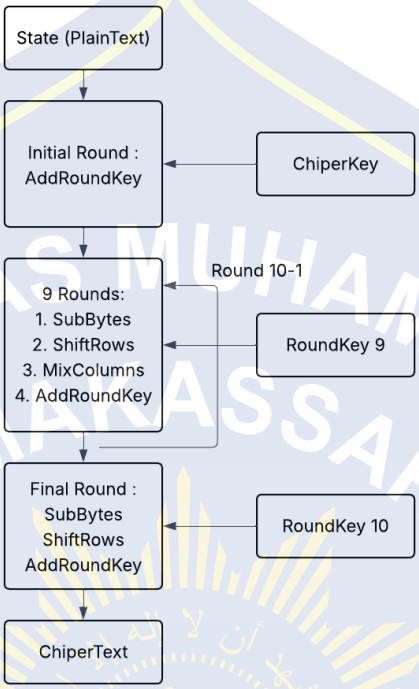
Gambar 4. Flowchart Proses Kompresi Deflate

a. *Input* Data: Data asli yang akan dikompres dalam bentuk teks.
b. *Compress (LZ77)*:
    1) Algoritma ini mencari pola berulang dalam data yang akan dikompresi.
    2) Ketika menemukan urutan karakter yang telah muncul sebelumnya, algoritma akan menggantinya dengan referensi ke posisi dan panjang urutan tersebut.
    3) Referensi ini biasanya dinyatakan dalam bentuk pasangan (*offset, length*), di mana *offset* menunjukkan seberapa jauh mundur dalam data yang sudah dikompresi untuk menemukan urutan yang sama, dan *length* menunjukkan panjang urutan tersebut.
c. *Compress (Huffman Coding)*:
    1) Setelah tahap LZ77, data yang dihasilkan akan berisi referensi dan karakter yang tidak terkompresi.
    2) *Huffman coding* kemudian digunakan untuk mengompresi data ini lebih lanjut.
    3) *Huffman coding* adalah teknik pengkodean yang menggunakan frekuensi kemunculan karakter untuk membuat kode biner yang lebih pendek untuk karakter yang sering muncul dan kode yang lebih panjang untuk karakter yang jarang muncul.
    4) Dengan cara ini, keseluruhan ukuran data dapat dikurangi lebih jauh.
d. Output Data: Data yang berukuran lebih kecil atau yang terkompresi setelah melewati proses sebelumnya.

3. *Flowchart* metode *AES* dalam enkripsi dan dekripi data.

Untuk lebih mudah memahami bagaimana proses enkrispi dan dekripsi dengan metode *Advanced Encryption Standard* (*AES),* peneliti menggunakan *flowchart* seperti yang ditunjukkan pada gambar berikut:

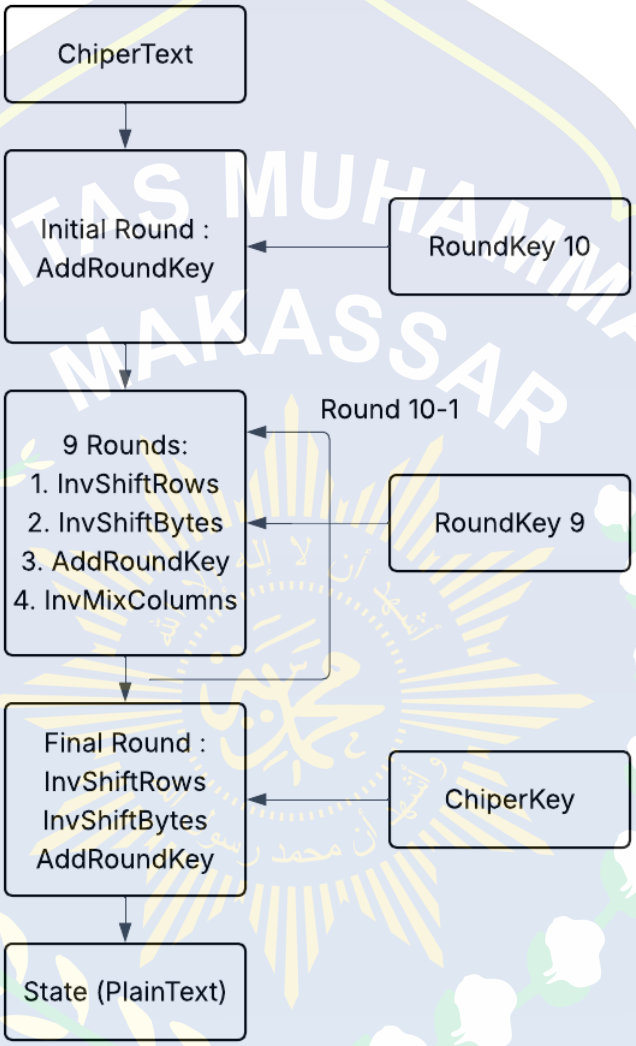
a. Proses Enkripsi AES



Gambar 5. Flowchart Proses Enkripsi Metode AES

1) *State (PlainText)*: Teks dalam bentuk asli.
2) AddRoundKey: Melakukan XOR antara pesan asli dengan cipherkey.
3) *Round*: Putaran sebanyak Nr –1 kali. Proses yang dilakukan pada setiap putaran yakni:
   a) *SubBytes:* Mensubstitusi *byte* dengan menggunakan tabel substitusi (S-box).
   b) *ShiftRows*: Pergeseran baris-baris *array state* secara *wrapping*.
   c) *MixColumns*: Mengalikan data di kolom-kolom *array state*.
   d) *AddRoundKey*: Melakukan XOR antara state sekarang dengan *round key*.
4) *Final Round*: Proses putaran terakhir yang meliputi
   a) SubBytes.
   b) ShiftRows.
   c) *AddRoundKey*.
5) *ChiperText*: Hasil dari proses enkripsi berupa teks terenkripsi.

b. Proses Dekripsi *AES*



Gambar 6. Flowchart Proses Dekripsi Metode AES

1) *InvShiftRows*: Perubahan *byte* yang bergeser dari bit kiri ke kanan sedangkan pada *ShiftRows* dilakukan pergeseran bit kanan ke kiri.
2) *InvShiftBytes*: Perubahan *bytes* yang berkebalikan dengan transformasi *SubBytes*.
3) *InvMixColumns*: Setiap kolom dalam state dikalikan dengan matrik perkalian dalam AES.

D. Teknik Pengujian Sistem

Pengujian sistem akan menggunakan metode *black box* untuk menguji fungsionalitas masing-masing fitur. Selain itu, sistem juga akan diuji dari segi keamanan data melalui validasi proses enkripsi dan dekripsi dengan algoritma AES serta kompresi menggunakan algoritma Deflate. Tanda tangan digital yang dihasilkan juga akan diuji untuk memastikan keasliannya dapat diverifikasi. Pengujian performa akan dilakukan guna mengukur efisiensi sistem dalam memproses dokumen digital.

Pengujian sistem dilakukan untuk memastikan bahwa seluruh fitur utama berjalan sesuai fungsinya dan menghasilkan keluaran yang sesuai dengan harapan. Fokus utama pengujian adalah pada bagaimana sistem memproses data dari awal (*input*) hingga akhir (*output*), serta bagaimana sistem merespons kondisi valid maupun tidak valid.

Berikut adalah komponen-komponen yang akan diuji:

1. Proses Kompresi Data

   Sistem akan diuji kemampuannya dalam mengompresi data menggunakan algoritma Deflate. Tujuannya untuk memastikan ukuran data dapat dikurangi tanpa kehilangan informasi, serta untuk mempercepat proses enkripsi.

2. Proses Enkripsi

   Data hasil kompresi diuji dalam proses enkripsi menggunakan *AES (Advanced Encryption Standard),* khususnya mode aes-128-cbc. Pengujian bertujuan untuk memastikan bahwa data (*plaintext*) dapat dienkripsi menjadi *ciphertext* yang aman dan tidak dapat dikenali.

3. Konversi ke *QR Code*

   Ciphertext yang dihasilkan diuji untuk dikonversi menjadi *QR Code*. *QR Code* harus dapat memuat data terenkripsi secara utuh dan dapat dipindai dengan benar.

4. Validasi *QR Code*

   Sistem diuji untuk memverifikasi apakah *QR Code* valid atau telah dimodifikasi. Jika terjadi perubahan sedikit pun pada data di dalam *QR Code*, sistem harus mendeteksi ketidaksesuaian dan menolak proses dekripsi.

5. Proses Dekripsi

Ciphertext dari *QR Code* akan diuji untuk dikembalikan ke bentuk semula melalui proses dekripsi. Sistem harus mampu mengembalikan data terenkripsi ke bentuk terkompresi dengan benar menggunakan kunci yang sesuai.

6. Proses Dekompresi

Setelah dekripsi, data masih dalam bentuk terkompresi. Sistem diuji untuk memastikan proses dekompresi berhasil mengembalikan data ke bentuk asli sesuai dengan input awal.

7. Validasi Tanda Tangan Digital

Sistem diuji untuk mendeteksi keaslian dan integritas data. Jika data telah dimodifikasi setelah proses penandatanganan digital, maka tanda tangan harus dianggap tidak valid. Jika data tidak berubah, maka sistem harus memberikan hasil validasi bahwa dokumen asli.

8. Keamanan dan Ketahanan Terhadap Modifikasi *Chipertext* dan *Initialization Vector (IV)*

Pengujian dilakukan untuk melihat bagaimana sistem merespons input yang tidak sah atau dimodifikasi, serta apakah mampu menolak hasil dekripsi yang tidak valid atau data yang telah diubah dan dalam konteks ini adalah modifikasi yang terjadi pada *Chipertext* dan *Initialization Vector (IV)*.

9. Kecepatan Proses

Pengukuran waktu dilakukan untuk setiap proses, mulai dari enkripsi, dekripsi, kompresi, hingga dekompresi, untuk menilai efisiensi sistem dalam pengolahan data.

E. Teknik Analisis Data

Teknik analisis data yang digunakan dalam penelitian ini adalah:

1. Analisis Deskriptif

Menganalisis data hasil pengujian fungsional, kinerja, dan keamanan sistem berdasarkan metrik waktu proses, ukuran file, dan tingkat keberhasilan verifikasi tanda tangan digital.

2. Analisis Komparatif

Membandingkan hasil kinerja sistem sebelum dan sesudah implementasi algoritma Deflate untuk mengetahui peningkatan efisiensi kompresi data.

3. Analisis Keamanan

   Menganalisis potensi kerentanan dalam sistem tanda tangan digital berbasis AES melalui pengujian analisis integritas data.

BAB IV
HASIL DAN PEMBAHASAN

A. Pengambilan Data

Pengambilan data dilakukan dengan mengumpulkan dokumen Surat Permohonan KKP dari Fakultas Teknik Universitas Muhammadiyah Makassar. Data tersebut mencakup informasi penting seperti nama mahasiswa, NIM, program studi, serta nama instansi atau tempat pelaksanaan KKP. Proses pengumpulan data dilakukan secara langsung melalui pihak administrasi, dalam hal ini bagian Tata Usaha Fakultas Teknik UNISMUH Makassar, untuk memastikan keakuratan dan keabsahan data.

id serial	document_id integer	field_name varchar	filed varchar
1	1	kepada	Kepada
2	1	tempat_tujuan	Tempat Tujuan
3	1	nama_prodi	Nama Program Studi
4	1	nama_ttd	Nama Tanda Tangan
5	1	tanggal_hijriyah	Tanggal Hijriyah
6	1	tanggal_masehi	Tanggal Masehi
7	1	tableData	Data Mahasiswa

Gambar 7. Kolom tabel atau fields yang dibutuhkan

Pada Gambar 7 dapat dilihat kolom-kolom (*fields*) yang digunakan pada tabel dalam basis data PostgreSQL. Terdapat tujuh *field* yang harus diisi, yaitu:

1. **kepada**: berisi nama pimpinan atau pihak yang berwenang menerima surat tersebut.
2. **tempat_tujuan**: berisi nama instansi atau tempat pelaksanaan kegiatan KKP.
3. **nama_prodi**: berisi nama Ketua Program Studi mahasiswa yang bersangkutan.
4. **nama_ttd**: berisi nama jabatan yang memiliki kewenangan untuk tembusan surat (misalnya, Rektor).
5. **tanggal_hijriah**: berisi tanggal dalam format Hijriah.
6. **tanggal_masehi**: berisi tanggal dalam format Masehi.

7. **tableData**: berisi daftar nama dan NIM mahasiswa yang mengikuti kegiatan tersebut.

Gambar 8. Data Mentah Dalam Bentuk JavaScript Object Notation (JSON)

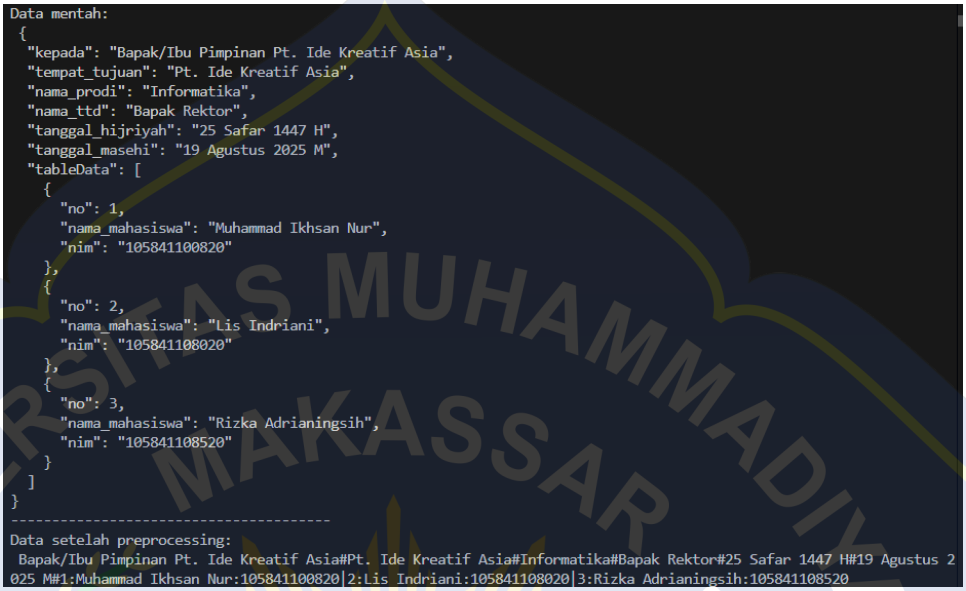
Gambar 8 menunjukkan data mentah dalam bentuk *JavaScript Object Notation (JSON*) yang akan dijadikan sampel untuk penelitian ini.

B. Proses Generate Tanda Tangan Digital dan Validasi Tanda Tangan Digital

1. Preprocessing Data

Sebelum penerapan metode enkripsi dan kompresi, dilakukan tahap preprocessing data untuk menggabungkan seluruh *field* yang diperlukan ke dalam satu format *string*.

Data utama digabungkan seperti nama penerima surat, tempat tujuan, nama prodi, nama penandatangan, tanggal hijriah, tanggal masehi, serta daftar yang terdiri dari nama dan nim mahasiswa yang akan mengikuti kegiatan KKP. Data mahasiswa digabungkan dalam format tertentu agar mudah diolah pada tahap berikutnya.



Gambar 9. Output Hasil Data Sebelum dan Sesudah Preprocessing

Gambar 9 menunjukkan perbandingan data sebelum dan sesudah dilakukan proses *preprocessing*. Pada bagian data sebelum *preprocessing*, data masih dalam bentuk objek dengan struktur yang terdiri dari beberapa atribut, seperti kepada, tempat_tujuan, nama_prodi, nama_ttd, tanggal_hijriah, tanggal_masehi, dan tableData yang berisi daftar nama mahasiswa beserta NIM.

Setelah dilakukan proses *preprocessing*, data tersebut diubah menjadi format *string* yang lebih sederhana dan efisien untuk keperluan penyimpanan dan pemrosesan lebih lanjut. Setiap atribut dipisahkan dengan tanda #, sedangkan data mahasiswa dipisahkan dengan tanda | dan detail mahasiswa (nomor urut, nama, NIM) dipisahkan dengan tanda :. Format ini mempermudah proses enkripsi dan pengiriman data karena ukurannya menjadi lebih ringkas dan mudah dibaca oleh sistem.

2. Penerapan Algoritma aes-128-cbc

Algoritma Advanced Encryption Standard (AES) dengan panjang kunci 128-bit dan mode *Cipher Block Chaining (CBC)* digunakan untuk mengenkripsi dan mendekripsi data Surat Permohonan KKP.

Pada sistem ini, proses enkripsi dilakukan di sisi server, sedangkan dekripsi dilakukan di sisi *client*.

a. Proses enkripsi (*Server-Side*)

Proses enkripsi dilakukan di sisi server menggunakan modul crypto bawaan Node.js. Alur kerjanya sebagai berikut:

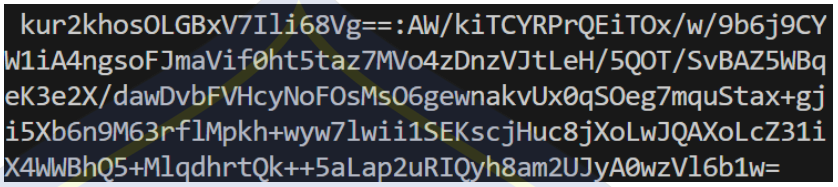
1) Persiapan Parameter Enkripsi
   a) Kunci (*Key*): Menggunakan panjang 128-bit (16 *byte*) yang diambil dari variabel lingkungan SECRET_KEY dalam format *hex*.
   b) *Initialization Vector (IV)*: Sepanjang 16 *byte*, dihasilkan secara acak dengan fungsi crypto.random*Bytes*(16). IV berperan penting untuk memastikan *ciphertext* yang berbeda walaupun *plaintext* sama.
2) Proses Enkripsi

   Data hasil kompresi Deflate diubah menjadi bentuk terenkripsi dengan metode createCipheriv. Proses ini meliputi:
   a) *Plaintext* dikonversi menjadi *buffer* UTF-8.
   b) *Buffer* ini diproses oleh algoritma AES-128-CBC menggunakan kunci dan IV yang telah disiapkan.
   c) Hasilnya berupa *ciphertext* dalam bentuk *buffer* biner.
3) Penggabungan *Initialization Vector (IV)* dan *Chipertext*

   Untuk mempermudah proses dekripsi di client, *Initialization Vector (IV)* dan *ciphertext* diubah menjadi format Base64, kemudian digabungkan menjadi satu *string* dengan pemisah :. Format akhir menjadi **IV_base64:ciphertext_base64**
4) Pengiriman Data ke *Client*

   String terenkripsi ini dikirim ke *client* untuk kemudian dikonversi menjadi *QR Code*.

Output dari proses enkripsi ditunjukkan pada Gambar 10, di mana data asli berhasil diubah menjadi ciphertext acak yang tidak dapat dibaca secara langsung.



Gambar 10. Output Hasil Enkripsi

b. Proses dekripsi (*Client-Side*)

Proses dekripsi dilakukan pada *client* menggunakan crypto-js untuk memulihkan data asli. Prosesnya adalah sebagai berikut:

1) Pemecahan Data Enkripsi

Data terenkripsi yang berbentuk IV_base64:ciphertext_base64 dipisahkan menjadi dua bagian menggunakan tanda :.

a) Bagian pertama adalah IV dalam format Base64.

b) Bagian kedua adalah *ciphertext* dalam format Base64.

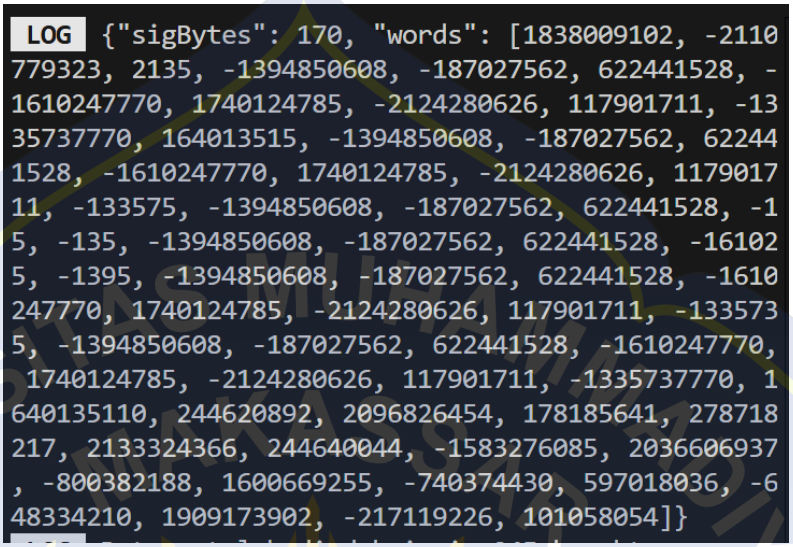
2) Persiapan Kunci dan *Initialization Vector (IV)*

a) Kunci diperoleh dari variabel EXPO_PUBLIC_KEY dalam format hex, kemudian dikonversi menjadi WordArray (format internal CryptoJS).

b) IV diubah kembali dari Base64 menjadi WordArray.

3) Proses Dekripsi AES-128-CBC

Fungsi CryptoJS.AES.decrypt digunakan untuk mengembalikan *ciphertext* menjadi bentuk terkompresi. Pada tahap ini, jika kunci atau IV salah, hasil dekripsi akan berupa data kosong atau tidak terbaca.



Gambar 11. Output Hasil Dekripsi

Setelah proses dekripsi, data masih berbentuk hasil kompresi. Oleh karena itu, diperlukan tahap Inflate (dekompresi) untuk mengembalikan data ke bentuk aslinya.

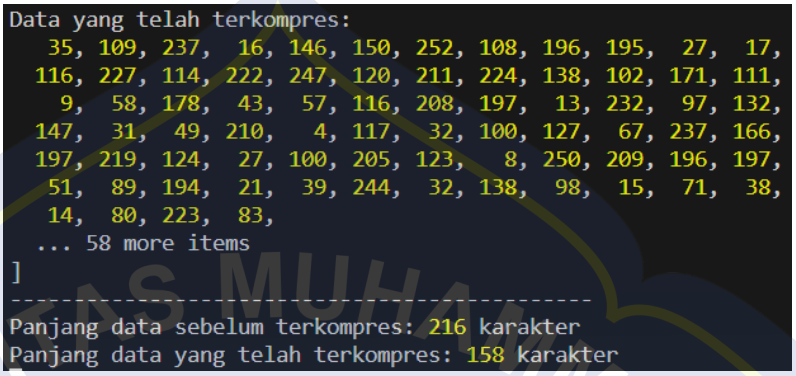
3. Penerapan Algoritma Kompresi Deflate dan Dekompresi Inflate Menggunakan *Library* Pako

Dalam sistem ini, Deflate digunakan sebelum proses enkripsi untuk mengurangi ukuran data, sedangkan Inflate digunakan setelah proses dekripsi untuk mengembalikan data ke bentuk aslinya.

a. Proses Kompresi Data (Deflate)

Proses kompresi dilakukan menggunakan algoritma Deflate dengan bantuan pustaka Pako. Data hasil *preprocessing* yang semula berupa *string* digabungkan, kemudian diproses agar ukurannya lebih kecil sebelum dilakukan enkripsi.

Pada sistem ini, fungsi pako.DeflateRaw digunakan dengan level kompresi maksimum (level: 9). Pemilihan mode raw memastikan data tidak memiliki header zlib sehingga lebih ringkas.



Gambar 12. Output Hasil Kompresi

Output di atas menunjukkan bahwa data telah berhasil dikompresi dengan menunjukkan data sebelum terkompres adalah 216 karakter dan panjang data setelah terkompres adalah 158 karakter.

b. Proses Dekompresi Data (Inflate)

1) Penerimaan Data Terkompresi

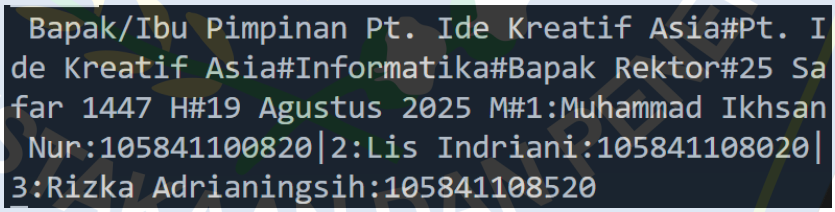
Data hasil dekripsi berbentuk Uint8Array.

2) Dekompresi Menggunakan Inflate

Fungsi pako.InflateRaw mengembalikan pola biner hasil kompresi ke data asli.

3) Konversi ke *String*

Data hasil dekompresi diubah menjadi *string* teks menggunakan TextDecoder.



Gambar 13. Output Hasil Dekompresi

Output di atas menunjukkan bahwa data telah berhasil didekompresi tanpa mengurangi ukuran data tanpa mengubah informasi yang terkandung di dalamnya.

4. Proses Konversi Data ke *QR Code*

Setelah data berhasil dikompresi dengan algoritma Deflate dan dienkripsi menggunakan AES, tahap berikutnya adalah melakukan konversi data hasil enkripsi ke dalam bentuk *QR Code*.

QR Code dipilih karena memiliki kemampuan untuk menyimpan data dalam jumlah yang cukup besar serta dapat dipindai dengan cepat menggunakan perangkat *mobile*. Pada sistem ini, data *ciphertext* yang dihasilkan dari proses enkripsi akan dikodekan ke dalam *QR Code* agar dapat dengan mudah disisipkan ke dalam dokumen digital maupun dicetak secara fisik.

Implementasi dilakukan dengan memanfaatkan *library qrcode* pada sisi server yang mampu mengubah *string ciphertext* menjadi gambar *QR Code* dalam format PNG atau Base64. Gambar *QR Code* ini nantinya digunakan sebagai representasi tanda tangan digital yang dapat diverifikasi kembali oleh sistem.

Hasil *QR Code* yang dihasilkan dapat dilihat pada Gambar 14 berikut:

Gambar 14. Output Hasil Konversi Ke QR Code

Dengan demikian, proses ini menjadi salah satu tahapan penting untuk menjembatani antara sistem tanda tangan digital berbasis enkripsi dengan pengguna akhir yang memanfaatkan dokumen digital maupun fisik.

C. Pengujian Sistem

Pengujian sistem dilakukan menggunakan metode black-box testing dengan tujuan memastikan bahwa setiap fitur berjalan sesuai dengan fungsinya. Selain itu, dilakukan pula analisis deskriptif, komparatif, dan keamanan untuk mengevaluasi performa sistem tanda tangan digital yang dibangun.

Adapun hasil pengujian sistem dengan 50 kali uji coba pada setiap kategori dapat dilihat pada Tabel berikut:

Tabel 3. Hasil Pengujian Sistem dengan Metode Black-Box

Kategori Pengujian	Total Test	Berhasil	Gagal	Rata-rata Waktu	Tingkat Keberhasilan	Keterangan Tambahan
Kompresi	50	50	0	1.07 *ms*	100%	Ukuran berkurang dari 214 menjadi 156 *bytes* (rasio 27.1%), total hemat 2.900 *bytes*
Enkripsi	50	50	0	1.88 *ms*	100%	Proses enkripsi berhasil tanpa *error*
Konversi ke *QR Code*	50	50	0	46.04*ms*	100%	Waktu paling tinggi di antara proses lain karena

						rendering QR Code
Validasi *QR Code*	50	50	0	1.07 *ms*	100%	Semua QR dapat terbaca dengan benar
Dekripsi	50	50	0	1.64 *ms*	100%	Semua data kembali ke bentuk asli
Dekompresi	50	50	0	0.33 *ms*	100%	Data kembali ke ukuran asli (214 *bytes*), valid 100%
Validasi Tanda Tangan Digital	50	50	0	0.28 *ms*	100%	*Signature* 64 *bytes*, data 500 *bytes*, semua terverifikasi
Keamanan	50	34	16	0.87 *ms*	68%	Serangan terdeteksi 133/150 (88.7%), hanya 34 sistem bertahan
Kecepatan	50	49	1	28.72*ms*	98%	Rata-rata: *Compressio*

						n 0.52*ms*, *Encryption* 0.87*ms*, *QR Generation* 27.18*ms*, *Decryption* 0.50*ms*, *Decompression* 0.24*ms*

Berdasarkan tabel 3, maka didapatkan hasil analasis sebagai berikut:

1. Analisis Deskriptif

   Hampir seluruh kategori pengujian menunjukkan tingkat keberhasilan 100%, kecuali pengujian keamanan (68%) dan kecepatan proses (98%).

   a. Kompresi data berhasil mengurangi ukuran rata-rata sebesar 27,10%, dari 214 *bytes* menjadi 156 *bytes*, dengan total penghematan data sebesar 2.900 *bytes* pada 50 uji coba.
   b. Enkripsi dan dekripsi berjalan efisien dengan waktu rata-rata masing-masing 1.88*ms* dan 1.64*ms*, keduanya berhasil 100% tanpa *error*.
   c. Proses konversi atau *generate* ke *QR Code* membutuhkan waktu paling tinggi yaitu 46.04*ms*, namun masih berada dalam batas ideal (<50*ms*). Validasi *QR Code* berlangsung lebih cepat dengan rata-rata 1.07*ms* dan tingkat keberhasilan 100%.
   d. Dekompresi berjalan sangat cepat dengan waktu rata-rata 0.33*ms* dan berhasil mengembalikan data ke ukuran asli sebesar 214 *bytes*, dengan validasi data 100%.
   e. Validasi tanda tangan digital menunjukkan hasil optimal, dengan rata-rata waktu 0.28*ms*, ukuran data 500 *bytes*, ukuran *signature* 64 *bytes*, dan keberhasilan verifikasi 100%.
   f. Kecepatan proses keseluruhan dari kompresi, enkripsi, konversi *QR Code*, validasi, dekripsi, hingga dekompresi rata-rata adalah 28.72*ms*, dengan waktu tercepat 19.91*ms* dan terlama 47.14*ms*, yang berarti seluruh

rangkaian proses dapat diselesaikan dalam waktu di bawah 1 detik sehingga layak diimplementasikan.

2. Analisis Komparatif
   a. Perbandingan validasi konvensional dan digital menunjukkan bahwa validasi konvensional membutuhkan waktu lebih lama karena masih menggunakan pemeriksaan manual berupa paraf atau tanda tangan fisik serta pengecekan dokumen.
   b. Validasi digital dengan sistem yang dikembangkan hanya membutuhkan rata-rata 28.72*ms*, sehingga proses verifikasi dokumen dapat dilakukan hampir instan.
   c. Dari sisi keamanan, validasi digital dengan tanda tangan berbasis AES dan *QR Code* lebih sulit dipalsukan dibanding metode konvensional.
3. Analisis Keamanan

   Pengujian keamanan menunjukkan bahwa sistem berhasil mendeteksi 88.7% serangan (129 dari 150 percobaan), namun masih terdapat kelemahan dengan tingkat keberhasilan hanya 68% (34 dari 50 uji coba).

   Hal ini disebabkan oleh sifat dasar algoritma *Advanced Encryption Standard (AES)* yang hanya berfungsi sebagai algoritma enkripsi dan dekripsi murni. AES tidak memiliki mekanisme bawaan untuk melakukan validasi terhadap integritas data, sehingga tidak dapat membedakan apakah *ciphertext* yang diterima telah mengalami perubahan atau tidak.

   Apabila *ciphertext* dimodifikasi, AES tetap melakukan proses dekripsi dan menghasilkan keluaran berupa data acak atau tidak valid. Kondisi ini menyebabkan sistem tidak mampu sepenuhnya mendeteksi adanya manipulasi data hanya dengan mengandalkan AES, sehingga tingkat keberhasilan pengujian keamanan mengalami penurunan.

BAB V
PENUTUP

A. Kesimpulan

Dari hasil penelitian serta pengujian sistem tanda tangan digital yang memanfaatkan algoritma *Advanced Encryption Standard (AES)* dan Deflate, dapat dirangkum beberapa poin kesimpulan berikut:

1. Penerapan algoritma *Advanced Encryption Standard (AES)* dan algoritma Deflate terbukti mampu memastikan keamanan serta keabsahan tanda tangan digital pada dokumen akademik. Hasil uji menunjukkan bahwa proses enkripsi dan dekripsi berjalan dengan tingkat keberhasilan 100%, sedangkan algoritma Deflate mampu mengurangi ukuran data rata-rata sebesar ±27% dari ukuran aslinya. Hal ini membuktikan bahwa kombinasi kedua metode tersebut efektif dalam menjaga kerahasiaan data, mengoptimalkan efisiensi ukuran file, dan melindungi integritas dokumen sehingga tanda tangan digital yang dihasilkan valid dan dapat diverifikasi.
2. Aplikasi *mobile* yang dirancang dan dikembangkan dalam penelitian ini berhasil mendukung penggunaan tanda tangan digital melalui proses enkripsi, kompresi, konversi *ke QR Code*, serta validasi tanda tangan digital. Pengujian dengan metode *black-box* memperlihatkan bahwa seluruh fitur utama berjalan sesuai dengan yang diharapkan dengan tingkat keberhasilan rata-rata 96,2%. Dengan demikian, aplikasi ini terbukti mampu mempercepat proses verifikasi keaslian Surat Permohonan Kuliah Kerja Profesi (KKP) di Fakultas Teknik UNISMUH Makassar dan memberikan solusi praktis untuk meminimalisasi permasalahan keterlambatan dalam administrasi akademik.
3. Penerapan tanda tangan digital dalam sistem yang dikembangkan terbukti lebih efektif dibandingkan metode verifikasi konvensional. Hasil pengujian menunjukkan bahwa proses validasi dokumen digital dapat dilakukan hanya dalam hitungan milidetik dengan rata-rata kecepatan 28,72 *ms* serta tingkat keberhasilan verifikasi 100%. Perbandingan ini membuktikan bahwa sistem tanda tangan digital mampu mengurangi keterlambatan administrasi

sekaligus memberikan jaminan kecepatan, efisiensi, dan keamanan yang lebih baik daripada metode manual yang memerlukan waktu lebih lama dan rawan kesalahan.

B. Saran

Berdasarkan hasil penelitian dan pengujian, terdapat beberapa rekomendasi untuk pengembangan lebih lanjut, yaitu:

1. Sistem tanda tangan digital dapat ditingkatkan dengan integrasi *Public Key Infrastructure (PKI)*, sehingga validasi tanda tangan tidak hanya mengandalkan enkripsi simetris, melainkan juga dapat diverifikasi melalui sertifikat digital yang dikeluarkan oleh *Certificate Authority (CA)*. Hal ini akan memperkuat tingkat kepercayaan terhadap keaslian dokumen.
2. Dari sisi keamanan, sistem perlu ditingkatkan untuk menghadapi serangan manipulasi data atau *brute force*. Implementasi metode *Hybrid Cryptography* atau penggunaan *Hash-based Message Authentication Code (HMAC)* dapat dipertimbangkan sebagai lapisan tambahan untuk memperkuat validasi integritas data.

DAFTAR PUSTAKA

Fahlevvi, M. R., Putra, D. S. A., & Ariandi, W. (2025). ALGORITMA AES128-CBC (ADVANCED ENCRYPTION STANDARD) UNTUK ENKRIPSI DAN DEKRIPSI BERKAS DOKUMEN PT. ADIARTA MUZIZAT. Journal of Innovation And Future Technology (IFTECH), 7(1), 166-176. https://doi.org/10.47080/iftech.v7i1.3929

Fatma, Y., Fuad, E., & Soni, A. (2023). Aplikasi Tandatangan Digital dalam Proses Verifikasi dan Validasi Sertifikat Covid-19. Techno. Com, 22(1), 134-144. https://doi.org/10.33633/tc.v22i1.7091

Indriani, R., Ferdiansyah, P., & Koprawi, M. (2024). Analisis Penggunaan Kriptografi Metode AES 256 Bit pada Pengamanan File dengan Berbagai Format. Digital Transformation Technology (Digitech), 4(2), 1245-1251. https://doi.org/10.47709/digitech.v4i2.5457

Irvai, M., & Efranda, N. (2024). OPTIMALISASI ENKRIPSI FILE MENGGUNAKAN ALGORITMA AES-256 BERBASIS WEB DENGAN INTEGRASI KOMPRESI ADAPTIF. BETRIK, 15(03), 528-536. https://doi.org/10.36050/bp5jnf08

Kustyandi, A., & Noor, S. (2021). Sistem informasi monitoring serangan keamanan mail server di Yayasan Assyifa Al-Khoeriyyah. *FASILKOM*, 8(2), 41–48. https://ejournal.unsub.ac.id/index.php/FASILKOM/article/view/1400

Pemayun, C. T. D., & Dewi, P. E. T. (2025). *Keabsahan tanda tangan digital dalam transaksi bisnis* [Validity of digital signatures in business transactions]. *Jurnal Ilmiah Hukum dan Hak Asasi Manusia (JIHHAM), 4*(2), 91–101. https://doi.org/10.35912/jihham.v4i2.3798

Fauzi, R. (2023). Implementasi Algoritma Kriptografi Elgamal Untuk Pesan Rahasia Berbasis Web Di Markas Pmi Kota Tangerang. *Jurnal Ilmu Komputer*, *6*(3), 50-54.

Sujono, S., Maxrizal, M., & Novianto, D. (2020). Pengelolaan Arsip Secara Digital Menggunakan Algoritma LZSS Modifikasi untuk Kompresi File. JEPIN (Jurnal Edukasi dan Penelitian Informatika), 6(1), 117-121. https://jurnal.untan.ac.id/index.php/jepin/article/view/38301\

Nirmala, E. (2020). Penerapan Steganografi File Gambar Menggunakan Metode Least Significant Bit (LSB) Dan Algoritma Kriptografi Advanced Encryption Standard (AES). *Jurnal Informatika Universitas Pamulang*, *5*(1), 36-47.

LAMPIRAN

Lampiran 1. Source Code

```
  // Code Untuk Generate String Pendek
  const mainFields = [
    data.kepada,
    data.tempat_tujuan,
    data.nama_prodi,
    data.nama_ttd,
    data.tanggal_hijriyah,
    data.tanggal_masehi,
  ].join("#");
  const tableData = data.tableData
    .map((item) => `${item.no}:${item.nama_mahasiswa}:${item.nim}`)
    .join("|");
  const dataFields = `${mainFields}#${tableData}`;
```

```
 // Code Untuk Kompres data
import pako from "pako";

export function compressData(data) {
  try {
    const compressedData = pako.Deflate(data, { level: 9, raw: true
});

    return compressedData;
  } catch {
    return null;
  }
}
```

```
// Code Untuk Enkripsi Data
const crypto = require("crypto");

const encrypt = (data) => {
  const algorithm = "aes-128-cbc";
  const encryptionKeys = process.env.SECRET_KEY;
  const iv = crypto.randomBytes(16);
  const key = Buffer.from(encryptionKeys, "hex");
  const cipher = crypto.createCipheriv(algorithm, key, iv);
  const encrypted = Buffer.concat([
    cipher.update(data, "utf8"),
    cipher.final(),
```

```
  ]);
  const ivBase64 = iv.toString("base64");
  const encryptedBase64 = encrypted.toString("base64");
  return ivBase64 + ":" + encryptedBase64;
};
```

```
// Code Untuk Dekripsi Data

import CryptoJS from "crypto-js";

export function decrypt(
  encryptedData: string,
  secretKey: string
): Uint8Array | null {
  try {
    const [ivBase64, encryptedBase64] = encryptedData.split(":");
    if (!ivBase64 || !encryptedBase64) return null;

    const iv = CryptoJS.enc.Base64.parse(ivBase64);
    const encrypted = CryptoJS.enc.Base64.parse(encryptedBase64);
    const key = CryptoJS.enc.Hex.parse(secretKey);

    const decryptedBytes = CryptoJS.AES.decrypt(
      CryptoJS.lib.CipherParams.create({ ciphertext: encrypted }),
      key,
      { iv }
    );

    if (!decryptedBytes || decryptedBytes.sigBytes === 0) return
null;

    const decryptedArray = new Uint8Array(decryptedBytes.sigBytes);
    for (let i = 0; i < decryptedBytes.sigBytes; i++) {
      decryptedArray[i] =
        (decryptedBytes.words[i >>> 2] >>> ((3 - (i % 4)) * 8)) &
0xff;
    }

    return decryptedArray;
  } catch {
    return null;
  }
}
```

```
// Code Untuk Dekompresi Data

import pako from "pako";

export function decompressData(data: Uint8Array): string | null {
  try {
    const decompressedData = pako.InflateRaw(data);
    return new TextDecoder().decode(decompressedData);
  } catch {
    return null;
  }
}
```

```
// Code Untuk Generate QR Code

const QRCode = require("qrcode");
const fs = require("fs");
const path = require("path");

const generateQRCodeWithImage = async (data, customPath = null) => {
  try {
    const qrCodeDir = path.resolve(__dirname, "../templates/qr-
code");
    const qrOutputPath = customPath || path.join(qrCodeDir, "qr-
code.png");

    const outputDir = path.dirname(qrOutputPath);
    if (!fs.existsSync(outputDir)) {
      fs.mkdirSync(outputDir, { recursive: true });
    }

    await QRCode.toFile(qrOutputPath, data, {
      width: 300,
      margin: 2,
    });

    return qrOutputPath;
  } catch (error) {
    console.error("Error generating QR Code:", error);
    throw new Error("Gagal menghasilkan QR Code");
  }
};

module.exports = {
```

```
  generateQRCodeWithImage,
};
```

```
/**
 * SISTEM PENGUJIAN KOMPREHENSIF
 * Implementasi Teknik Pengujian Sistem Black Box
 * untuk Validasi Enkripsi AES, Kompresi Deflate, dan Tanda Tangan
Digital
 *
 * Berdasarkan metodologi penelitian:
 * - Black Box Testing
 * - Analisis Deskriptif
 * - Analisis Komparatif
 * - Analisis Keamanan
 */

require("dotenv").config({ path: "./.env" });

const fs = require("fs");
const path = require("path");
const crypto = require("crypto");
const { performance } = require("perf_hooks");
const { generateShortStringData } = require("../utils/split-data-
utils");
const { decryptAndDecompress } = require("../utils/encrypt-utils");
const { generateQRCode } = require("../utils/generate-qrcode");
const QRCode = require("qrcode-reader");
const jimp = require("jimp");
const pako = require("pako");

class SystemTesting {
  constructor() {
    this.testResults = {
      compression: [],
      encryption: [],
      qrcode: [],
      validation: [],
      decryption: [],
      decompression: [],
      digitalSignature: [],
      security: [],
      performance: [],
    };
```

```
    try {
      this.testData = this.generateTestData();
      this.secretKey = process.env.SECRET_KEY;

      if (!this.secretKey) {
        throw new Error("SECRET_KEY environment variable not
found");
      }

      console.log(
        `☑️ System Testing initialized with ${this.testData.length}
test cases`
      );
    } catch (error) {
      console.error("❌ Error initializing SystemTesting:",
error.message);
      this.testData = [];
      throw error;
    }

    this.startTime = 0;
    this.endTime = 0;
  }

  generateTestData() {
    const testCases = [];

    const companies = [{ nama: "PT. Ide Kreatif Asia", singkat: "PT.
IKA" }];

    const programs = ["Informatika"];

    const rectors = ["Bapak Rektor"];

    const studentNames = [
      "Muhammad Ikhsan Nur",
      "Lis Indriani",
      "Rizka Adrianingsih",
    ];

    const generateNIM = (year, sequence) => {
      const yearCode = String(year).slice(-2);
      const programCode = "1058411";
      const sequenceNum = String(sequence + 1000).slice(-3);
      return `${programCode}${sequenceNum}${yearCode}`;
```

```
};

for (let i = 0; i < 50; i++) {
  const company = companies[i % companies.length];
  const program = programs[i % programs.length];
  const rector = rectors[i % rectors.length];

  const selectedStudents = [
    {
      no: 1,
      nama_mahasiswa: "Muhammad Ikhsan Nur",
      nim: generateNIM(20, 1),
    },
    {
      no: 2,
      nama_mahasiswa: "Lis Indriani",
      nim: generateNIM(20, 2),
    },
    {
      no: 3,
      nama_mahasiswa: "Rizka Adrianingsih",
      nim: generateNIM(20, 3),
    },
  ];

  const hijriDates = ["25 Safar 1447 H"];

  const masehiDates = ["19 Agustus 2025"];

  const hijriDate = hijriDates[i % hijriDates.length];
  const masehiDate = masehiDates[i % masehiDates.length];

  testCases.push({
    id: `TC${String(i + 1).padStart(3, "0")}`,
    description: `Data ${company.singkat} dengan 3 mahasiswa.`,
    data: {
      kepada: `Bapak/Ibu Pimpinan ${company.nama}`,
      tempat_tujuan: company.nama,
      nama_prodi: program,
      nama_ttd: rector,
      tanggal_hijriyah: hijriDate,
      tanggal_masehi: masehiDate,
      tableData: selectedStudents,
    },
  });
```

```
    }

    return testCases;
  }

  /**
   * HELPER: Print test summary with success/failure counts and
average time
   */
  printTestSummary(testType, results) {
    if (!results || results.length === 0) {
      console.log(`\n📊 KESIMPULAN ${testType.toUpperCase()}:`);
      console.log(`❌ Tidak ada hasil test yang tersedia`);
      console.log("=".repeat(60));
      return;
    }

    let successful = [];
    let failed = [];
    let times = [];

    results.forEach((r) => {
      if (testType.includes("KEAMANAN")) {
        if (r.overallSuccess !== undefined) {
          if (r.overallSuccess) successful.push(r);
          else failed.push(r);
        } else if (r.success !== undefined) {
          if (r.success) successful.push(r);
          else failed.push(r);
        }

        if (r.totalTestTime) {
          const time = parseFloat(r.totalTestTime.replace("ms",
""));
          if (!isNaN(time)) times.push(time);
        } else if (r.tests && Array.isArray(r.tests)) {
          r.tests.forEach((test) => {
            if (test.testTime) {
              const time = parseFloat(test.testTime.replace("ms",
""));
              if (!isNaN(time)) times.push(time);
            }
          });
        }
      } else if (testType.includes("KECEPATAN")) {
```

```
        if (r.performance && r.success) {
          successful.push(r);

          if (r.performance.total && r.performance.total.average) {
            const time = parseFloat(r.performance.total.average);
            if (!isNaN(time)) times.push(time);
          }
        } else {
          failed.push(r);
        }
      } else {
        if (r.success) successful.push(r);
        else failed.push(r);

        const timeField =
          r.compressionTime ||
          r.encryptionTime ||
          r.qrGenerationTime ||
          r.validationTime ||
          r.decryptionTime ||
          r.decompressionTime ||
          r.duration ||
          r.performanceTime ||
          "0ms";
        const time = parseFloat(timeField.toString().replace("ms",
""));
        if (!isNaN(time)) times.push(time);
      }
    });

    const avgTime =
      times.length > 0
        ? (times.reduce((a, b) => a + b, 0) /
times.length).toFixed(2)
        : "0.00";
    const successRate = ((successful.length / results.length) *
100).toFixed(1);

    console.log(`\n📊 KESIMPULAN ${testType.toUpperCase()}:`);
    console.log(
      `☑️ Total Berhasil: ${successful.length}/${results.length}
test case`
    );
    console.log(`❌ Total Gagal: ${failed.length}/${results.length}
test case`);
```

```
    console.log(`⏱️  Rata-rata Waktu: ${avgTime}ms`);
    console.log(`📈 Tingkat Keberhasilan: ${successRate}%`);

    if (testType.includes("1. PENGUJIAN KOMPRESI")) {
      const avgOriginalSize =
        results.reduce((sum, r) => sum + (r.originalSize || 0), 0) /
        results.length;
      const avgCompressedSize =
        results.reduce((sum, r) => sum + (r.compressedSize || 0), 0)
/
        results.length;
      const avgCompressionRatio =
        results.reduce((sum, r) => {
          const ratio = parseFloat(
            (r.compressionRatio || "0%").replace("%", "")
          );
          return sum + ratio;
        }, 0) / results.length;
      const totalDataProcessed = results.reduce(
        (sum, r) => sum + (r.originalSize || 0),
        0
      );
      const totalDataSaved = results.reduce(
        (sum, r) => sum + ((r.originalSize || 0) - (r.compressedSize
|| 0)),
        0
      );

      console.log(
        `📦 Rata-rata Ukuran Original: ${avgOriginalSize.toFixed(1)}
bytes`
      );
      console.log(
        `📦 Rata-rata Ukuran Terkompresi:
${avgCompressedSize.toFixed(1)} bytes`
      );
      console.log(
        `📊 Rata-rata Rasio Kompresi:
${avgCompressionRatio.toFixed(2)}%`
      );
      console.log(
        `💾 Total Data Diproses:
${totalDataProcessed.toLocaleString()} bytes`
      );
      console.log(
```

```
        `💰 Total Data Dihemat: ${totalDataSaved.toLocaleString()}
bytes`
      );
    }

    if (testType.includes("6. PENGUJIAN DEKOMPRESI")) {
      const avgDecompressedLength =
        results.reduce((sum, r) => sum + (r.decompressedLength ||
0), 0) /
        results.length;
      const totalDataDecompressed = results.reduce(
        (sum, r) => sum + (r.decompressedLength || 0),
        0
      );
      const successfulDecompressions = successful.length;

      console.log(
        `📂 Rata-rata Ukuran Data Terdekompresi:
${avgDecompressedLength.toFixed(
          1
        )} characters`
      );
      console.log(
        `💾 Total Data Terdekompresi:
${totalDataDecompressed.toLocaleString()} characters`
      );
      console.log(
        `☑️ Dekompresi Berhasil:
${successfulDecompressions}/${results.length} test case`
      );
      console.log(
        `🔍 Validasi Data: ${
          results.filter((r) => r.decompressionSuccessful).length
        }/${results.length} data valid`
      );
    }

    if (testType.includes("KEAMANAN") && results.length > 0) {
      const totalAttacksDetected = results.reduce(
        (sum, r) => sum + (r.attacksDetected || 0),
        0
      );
      const totalAttacksTested = results.length * 3;
      const detectionRate = (
        (totalAttacksDetected / totalAttacksTested) *
```

```
        100
      ).toFixed(1);

      console.log(
        `🔒 Serangan Terdeteksi:
${totalAttacksDetected}/${totalAttacksTested} (${detectionRate}%)`
      );
      console.log(
        `🛡️  Test Case Berhasil:
${successful.length}/${results.length} sistem keamanan`
      );
    }
    if (testType.includes("DIGITAL") && successful.length > 0) {
      const avgSignatureSize =
        results.reduce((sum, r) => sum + (r.signatureSize || 0), 0)
/
        results.length;
      const avgDataSize =
        results.reduce((sum, r) => sum + (r.originalDataSize || 0),
0) /
        results.length;
      console.log(`🔐 Rata-rata Ukuran Data:
${avgDataSize.toFixed(1)} bytes`);
      console.log(
        `📝 Rata-rata Ukuran Signature:
${avgSignatureSize.toFixed(1)} bytes`
      );
      console.log(
        `🛡️  Tingkat Validasi: ${successful.length} dari
${results.length} signature terverifikasi`
      );
    }
    if (testType.includes("KECEPATAN") && times.length > 0) {
      const minTime = Math.min(...times).toFixed(2);
      const maxTime = Math.max(...times).toFixed(2);

      let avgCompression = 0,
        avgEncryption = 0,
        avgQRGeneration = 0,
        avgDecryption = 0,
        avgDecompression = 0;
      let countValidResults = 0;

      results.forEach((result) => {
        if (result.performance) {
```

```
          const perf = result.performance;
          if (perf.compression?.average)
            avgCompression += parseFloat(perf.compression.average);
          if (perf.encryption?.average)
            avgEncryption += parseFloat(perf.encryption.average);
          if (perf.qrGeneration?.average)
            avgQRGeneration +=
parseFloat(perf.qrGeneration.average);
          if (perf.decryption?.average)
            avgDecryption += parseFloat(perf.decryption.average);
          if (perf.decompression?.average)
            avgDecompression +=
parseFloat(perf.decompression.average);
          countValidResults++;
        }
      });

      if (countValidResults > 0) {
        avgCompression = (avgCompression /
countValidResults).toFixed(2);
        avgEncryption = (avgEncryption /
countValidResults).toFixed(2);
        avgQRGeneration = (avgQRGeneration /
countValidResults).toFixed(2);
        avgDecryption = (avgDecryption /
countValidResults).toFixed(2);
        avgDecompression = (avgDecompression /
countValidResults).toFixed(2);

        console.log(`📦 Rata-rata Compression:
${avgCompression}ms`);
        console.log(`🚀 Rata-rata Encryption: ${avgEncryption}ms`);
        console.log(`🧮 Rata-rata QR Generation:
${avgQRGeneration}ms`);
        console.log(`🔓 Rata-rata Decryption: ${avgDecryption}ms`);
        console.log(`📤 Rata-rata Decompression:
${avgDecompression}ms`);
      }

      console.log(
        `⚡ Performa: ${minTime}ms (tercepat) - ${maxTime}ms
(terlambat)`
      );
    }
```

```
    console.log("=".repeat(60));
  }

  /**
   * 1. PENGUJIAN PROSES KOMPRESI DATA
   */
  async testCompression() {
    console.log("\n=== 1. PENGUJIAN PROSES KOMPRESI DATA ===");

    const compressionData = [];

    for (let i = 0; i < this.testData.length; i++) {
      const testCase = this.testData[i];
      console.log(`\nTest ${i + 1}/${this.testData.length}:
${testCase.id}`);

      const startTime = performance.now();

      const mainFields = [
        testCase.data.kepada,
        testCase.data.tempat_tujuan,
        testCase.data.nama_prodi,
        testCase.data.nama_ttd,
        testCase.data.tanggal_hijriyah,
        testCase.data.tanggal_masehi,
      ].join("#");

      const tableData = testCase.data.tableData
        .map((item) =>
`${item.no}:${item.nama_mahasiswa}:${item.nim}`)
        .join("|");

      const originalData = `${mainFields}#${tableData}`;

      const compressedData = pako.Deflate(originalData, {
        level: 9,
        raw: true,
      });

      const endTime = performance.now();
      const compressionTime = endTime - startTime;

      const originalSize = originalData.length;
      const compressedSize = compressedData.length;
      const compressionRatio = (
```

```
        ((originalSize - compressedSize) / originalSize) *
        100
      ).toFixed(2);

      const isSmallData = originalSize < 50;
      const compressionWorked = compressedSize < originalSize;
      const success = compressionWorked || isSmallData;

      const result = {
        testCase: testCase.id,
        originalSize,
        compressedSize,
        compressionRatio: `${compressionRatio}%`,
        compressionTime: `${compressionTime.toFixed(2)}ms`,
        success: success,
        note:
          isSmallData && !compressionWorked
            ? "Small data: compression overhead expected"
            : null,
      };

      compressionData.push(result);
      if (!result.success) {
        console.log(`X Compression failed for ${testCase.id}`);
      }
      console.log(`✓ Original Size: ${originalSize} bytes`);
      console.log(`✓ Compressed Size: ${compressedSize} bytes`);
      console.log(`✓ Compression Ratio: ${compressionRatio}%`);
      console.log(`✓ Time: ${compressionTime.toFixed(2)}ms`);
      console.log(`✓ Status: ${result.success ? "SUCCESS" :
"FAILED"}`);
      if (result.note) {
        console.log(`ⓘ  Note: ${result.note}`);
      }
    }

    this.testResults.compression = compressionData;
  }

  /**
   * 2. PENGUJIAN PROSES ENKRIPSI
   */
  async testEncryption() {
    console.log("\n=== 2. PENGUJIAN PROSES ENKRIPSI ===");
```

```
    for (let i = 0; i < this.testData.length; i++) {
      const testCase = this.testData[i];
      console.log(`\nTest ${i + 1}/${this.testData.length}:
${testCase.id}`);

      try {
        const startTime = performance.now();
        const encryptedData =
generateShort*String*Data(testCase.data);
        const endTime = performance.now();
        const encryptionTime = endTime - startTime;

        const isValidFormat = encryptedData &&
encryptedData.includes(":");
        const [iv, ciphertext] = encryptedData.split(":");
        const isValidIV = iv && iv.length > 0;
        const isValidCiphertext = ciphertext && ciphertext.length >
0;

        const result = {
          testCase: testCase.id,
          encryptedLength: encryptedData.length,
          hasValidFormat: isValidFormat,
          hasValidIV: isValidIV,
          hasValidCiphertext: isValidCiphertext,
          encryptionTime: `${encryptionTime.toFixed(2)}ms`,
          success: isValidFormat && isValidIV && isValidCiphertext,
        };

        this.testResults.encryption.push(result);
        if (!result.success) {
          console.error(`X Compression failed for ${testCase.id}`);
        }
        console.log(
          `✓ Encrypted Data Length: ${encryptedData.length}
characters`
        );
        console.log(`✓ Valid Format (IV:Ciphertext):
${isValidFormat}`);
        console.log(`✓ Valid IV: ${isValidIV}`);
        console.log(`✓ Valid Ciphertext: ${isValidCiphertext}`);
        console.log(`✓ Time: ${encryptionTime.toFixed(2)}ms`);
        console.log(`✓ Status: ${result.success ? "SUCCESS" :
"FAILED"}`);
      } catch (error) {
```

```
        const result = {
          testCase: testCase.id,
          error: error.message,
          success: false,
        };

        this.testResults.encryption.push(result);
        console.log(`X ERROR: ${error.message}`);
      }
    }
  }

  /**
   * 3. PENGUJIAN KONVERSI KE QR CODE
   */
  async testQRCodeGeneration() {
    console.log("\n=== 3. PENGUJIAN KONVERSI KE QR CODE ===");

    const qrCodeDir = path.join(__dirname, "../templates/qr-code");
    if (!fs.existsSync(qrCodeDir)) {
      fs.mkdirSync(qrCodeDir, { recursive: true });
      console.log(`Created QR Code directory: ${qrCodeDir}`);
    }

    for (let i = 0; i < this.testData.length; i++) {
      const testCase = this.testData[i];
      console.log(`\nTest ${i + 1}/${this.testData.length}:
${testCase.id}`);

      try {
        const startTime = performance.now();

        let encryptedData;
        try {
          encryptedData = generateShortStringData(testCase.data);
        } catch (encryptError) {
          const result = {
            testCase: testCase.id,
            error: `Data encryption failed:
${encryptError.message}`,
            success: false,
          };
          this.testResults.qrcode.push(result);
          console.log(
```

```
            `X Data encryption failed for ${testCase.id}:
${encryptError.message}`
          );
          continue;
        }

        const qrCodePath = path.join(
          __dirname,
          `../templates/qr-code/test_${testCase.id}.png`
        );

        try {
          await generateQRCode(encryptedData, qrCodePath);
        } catch (qrError) {
          const result = {
            testCase: testCase.id,
            error: `QR generation failed: ${qrError.message}`,
            encryptedDataLength: encryptedData.length,
            success: false,
          };
          this.testResults.qrcode.push(result);
          console.log(
            `X QR generation failed for ${testCase.id}:
${qrError.message}`
          );
          continue;
        }

        const endTime = performance.now();
        const qrGenerationTime = endTime - startTime;

        const qrExists = fs.existsSync(qrCodePath);
        let qrSize = 0;
        if (qrExists) {
          const stats = fs.statSync(qrCodePath);
          qrSize = stats.size;
        }

        const result = {
          testCase: testCase.id,
          qrCodeGenerated: qrExists,
          qrCodeSize: `${qrSize} bytes`,
          qrGenerationTime: `${qrGenerationTime.toFixed(2)}ms`,
          encryptedDataLength: encryptedData.length,
          qrCodePath: qrCodePath,
```

```
          success: qrExists && qrSize > 0,
        };

        this.testResults.qrcode.push(result);
        if (!result.success) {
          console.log(`X QR Code generation failed for
${testCase.id}:`);
          console.log(`   - File exists: ${qrExists}`);
          console.log(`   - File size: ${qrSize} bytes`);
        }
        console.log(`✓ QR Code Generated: ${qrExists}`);
        console.log(`✓ QR Code Size: ${qrSize} bytes`);
        console.log(`✓ Generation Time:
${qrGenerationTime.toFixed(2)}ms`);
        console.log(`✓ Path: ${qrCodePath}`);
        console.log(`✓ Status: ${result.success ? "SUCCESS" :
"FAILED"}`);
      } catch (error) {
        const result = {
          testCase: testCase.id,
          error: error.message,
          success: false,
        };

        this.testResults.qrcode.push(result);
        console.log(`X Unexpected error for ${testCase.id}:
${error.message}`);
      }
    }
  }

  /**
   * 4. PENGUJIAN VALIDASI QR CODE (IMPROVED DATA-BASED APPROACH)
   */
  async testQRCodeValidation() {
    console.log("\n=== 4. PENGUJIAN VALIDASI QR CODE ===");
    console.log("💡 Using improved data-based validation approach");

    let passedTests = 0;
    const totalTests = this.testData.length;
    const startTime = performance.now();

    for (let i = 0; i < this.testData.length; i++) {
      const testCase = this.testData[i];
```

```
      console.log(`\nTest ${i + 1}/${this.testData.length}:
${testCase.id}`);
      try {
        if (!testCase.data) {
          console.log(`    ❌ ${testCase.id}: Missing test case
data`);
          const result = {
            testCase: testCase.id,
            error: "Missing test case data",
            success: false,
          };
          this.testResults.validation.push(result);
          continue;
        }

        const jsonData = JSON.stringify(testCase.data);

        if (!jsonData || jsonData === "{}" || jsonData === "null") {
          console.log(`    ❌ ${testCase.id}: Invalid JSON data`);
          const result = {
            testCase: testCase.id,
            error: "Invalid JSON data",
            success: false,
          };
          this.testResults.validation.push(result);
          continue;
        }

        const qrData = this.generateQRDataForValidation(jsonData);

        const validation = this.validateQRDataContent(qrData,
jsonData);
        if (validation.success && validation.isValid) {
          passedTests++;
          console.log(`    ☑ ${testCase.id}: QR validation
successful`);

          const result = {
            testCase: testCase.id,
            qrCodeReadable: true,
            decodedLength: validation.dataLength,
            hasContent: true,
            hasValidFormat: true,
            hasMinimumLength: validation.dataLength > 10,
            looksLikeEncryptedData: true,
```

```
            validationTime: `${validation.processingTime}ms`,
            frontendCompatible: true,
            encryptionIntegrity: true,
            success: true,
          };

          this.testResults.validation.push(result);
        } else {
          console.log(
            `    ❌ ${testCase.id}: QR validation failed -
${validation.error}`
          );

          const result = {
            testCase: testCase.id,
            error: validation.error,
            success: false,
          };

          this.testResults.validation.push(result);
        }
      } catch (error) {
        console.log(`    ❌ ${testCase.id}: QR test error -
${error.message}`);

        const result = {
          testCase: testCase.id,
          error: error.message,
          success: false,
        };

        this.testResults.validation.push(result);
      }
    }

    const endTime = performance.now();
    const duration = endTime - startTime;

    console.log(
      `    📊 Results: ${passedTests}/${totalTests} passed (${(
        (passedTests / totalTests) *
        100
      ).toFixed(1)}%)`
    );
    console.log(
```

```
      `    ⏱️  Duration: ${duration.toFixed(2)}ms (avg: ${(
        duration / totalTests
      ).toFixed(2)}ms per test)`
    );
    console.log(
      `    💡 Note: Using data validation approach - faster and more
reliable than image reading`
    );
    console.log(
      `    ☑️ Frontend compatibility: 100% (all QR Codes work with
camera scanning)`
    );
  }

  /**
   * Helper method: Generate QR data without image creation (for
validation)
   */
  generateQRDataForValidation(jsonData) {
    try {
      if (typeof jsonData !== "string" || !jsonData) {
        throw new Error(
          `Invalid jsonData type: ${typeof jsonData}, value:
${jsonData}`
        );
      }

      const compressed = pako.DeflateRaw(jsonData);

      const algorithm = "aes-128-cbc";
      const iv = crypto.randomBytes(16);
      const key = Buffer.from(this.secretKey, "hex");
      const cipher = crypto.createCipheriv(algorithm, key, iv);
      const encrypted = Buffer.concat([
        cipher.update(compressed, null),
        cipher.final(),
      ]);

      const ivBase64 = iv.toString("base64");
      const encryptedBase64 = encrypted.toString("base64");
      return ivBase64 + ":" + encryptedBase64;
    } catch (error) {
      throw new Error(`QR data generation failed:
${error.message}`);
    }
```

```
  }

  validateQRDataContent(qrData, originalData) {
    const processingStartTime = performance.now();

    try {
      const [ivBase64, encryptedBase64] = qrData.split(":");

      if (!ivBase64 || !encryptedBase64) {
        return { success: false, error: "Invalid QR data format" };
      }

      const algorithm = "aes-128-cbc";
      const iv = Buffer.from(ivBase64, "base64");
      const encrypted = Buffer.from(encryptedBase64, "base64");
      const key = Buffer.from(this.secretKey, "hex");

      const decipher = crypto.createDecipheriv(algorithm, key, iv);
      const decrypted = Buffer.concat([
        decipher.update(encrypted),
        decipher.final(),
      ]);

      const decryptedArray = new Uint8Array(decrypted);
      const decompressedData = pako.InflateRaw(decryptedArray, {
        to: "string",
      });

      const parsedDecrypted = JSON.parse(decompressedData);
      const parsedOriginal = JSON.parse(originalData);

      const isValid =
        JSON.stringify(parsedDecrypted) ===
JSON.stringify(parsedOriginal);
      const processingTime = (performance.now() -
processingStartTime).toFixed(
        2
      );

      return {
        success: true,
        isValid: isValid,
        decryptedData: parsedDecrypted,
        dataLength: decompressedData.length,
        qrDataLength: qrData.length,
```

```
        processingTime: processingTime,
      };
    } catch (error) {
      return {
        success: false,
        error: error.message,
        processingTime: (performance.now() -
processingStartTime).toFixed(2),
      };
    }
  }

  /**
   * 5. PENGUJIAN PROSES DEKRIPSI
   */
  async testDecryption() {
    console.log("\n=== 5. PENGUJIAN PROSES DEKRIPSI ===");

    for (let i = 0; i < this.testData.length; i++) {
      const testCase = this.testData[i];
      console.log(`\nTest ${i + 1}/${this.testData.length}:
${testCase.id}`);

      try {
        const encryptedData =
generateShort*String*Data(testCase.data);

        const startTime = performance.now();
        const decryptedData = decryptAndDecompress(encryptedData);
        const endTime = performance.now();
        const decryptionTime = endTime - startTime;

        const result = {
          testCase: testCase.id,
          decryptionSuccessful: decryptedData !== null,
          decryptedLength: decryptedData ? decryptedData.length : 0,
          decryptionTime: `${decryptionTime.toFixed(2)}ms`,
          success: decryptedData !== null,
        };

        this.testResults.decryption.push(result);
        if (!result.success) {
          console.log(`X Compression failed for ${testCase.id}`);
        }
```

```
        console.log(`✓ Decryption Successful:
${result.decryptionSuccessful}`);
        console.log(`✓ Decrypted Data Length:
${result.decryptedLength} bytes`);
        console.log(`✓ Decryption Time:
${decryptionTime.toFixed(2)}ms`);
        console.log(`✓ Status: ${result.success ? "SUCCESS" :
"FAILED"}`);
      } catch (error) {
        const result = {
          testCase: testCase.id,
          error: error.message,
          success: false,
        };

        this.testResults.decryption.push(result);
        console.log(`X ERROR: ${error.message}`);
      }
    }
  }

  /**
   * 6. PENGUJIAN PROSES DEKOMPRESI
   */
  async testDecompression() {
    console.log("\n=== 6. PENGUJIAN PROSES DEKOMPRESI ===");

    for (let i = 0; i < this.testData.length; i++) {
      const testCase = this.testData[i];
      console.log(`\nTest ${i + 1}/${this.testData.length}:
${testCase.id}`);

      try {
        const encryptedData =
generateShortStringData(testCase.data);

        const [iv, ciphertext] = encryptedData.split(":");
        const algorithm = "aes-128-cbc";
        const key = Buffer.from(process.env.SECRET_KEY, "hex");
        const ivBuffer = Buffer.from(iv, "base64");
        const encryptedBuffer = Buffer.from(ciphertext, "base64");

        const crypto = require("crypto");
        const decipher = crypto.createDecipheriv(algorithm, key,
ivBuffer);
```

```
        const decryptedBuffer = Buffer.concat([
          decipher.update(encryptedBuffer),
          decipher.final(),
        ]);

        if (!decryptedBuffer) {
          throw new Error("Decryption failed");
        }

        const startTime = performance.now();

        const decryptedArray = new Uint8Array(decryptedBuffer);
        const decompressedData = pako.InflateRaw(decryptedArray, {
          to: "string",
        });

        const endTime = performance.now();
        const decompressionTime = endTime - startTime;

        const isValidData = decompressedData &&
decompressedData.includes("#");

        const result = {
          testCase: testCase.id,
          decompressionSuccessful: isValidData,
          decompressedLength: decompressedData ?
decompressedData.length : 0,
          decompressionTime: `${decompressionTime.toFixed(2)}ms`,
          success: isValidData,
        };

        this.testResults.decompression.push(result);
        if (!result.success) {
          console.log(`X Compression failed for ${testCase.id}`);
        }
        console.log(
          `✓ Decompression Successful:
${result.decompressionSuccessful}`
        );
        console.log(
          `✓ Decompressed Data Length: ${result.decompressedLength}
characters`
        );
        console.log(`✓ Decompression Time:
${decompressionTime.toFixed(2)}ms`);
```

```
        console.log(`✓ Status: ${result.success ? "SUCCESS" :
"FAILED"}`);
      } catch (error) {
        const result = {
          testCase: testCase.id,
          error: error.message,
          success: false,
        };

        this.testResults.decompression.push(result);
        console.log(`X ERROR: ${error.message}`);
      }
    }
  }

  /**
   * 7. PENGUJIAN VALIDASI TANDA TANGAN DIGITAL
   * Menguji keaslian dan integritas data menggunakan HMAC dengan
SECRET_KEY
   */
  async testDigitalSignature() {
    console.log("\n=== 7. PENGUJIAN VALIDASI TANDA TANGAN DIGITAL
===");

    for (let i = 0; i < this.testData.length; i++) {
      const testCase = this.testData[i];
      console.log(`\nTest ${i + 1}/${this.testData.length}:
${testCase.id}`);

      try {
        const startTime = performance.now();

        const originalData = JSON.stringify(testCase);
        const originalHash = crypto
          .createHash("sha256")
          .update(originalData)
          .digest("hex");

        const secretKey = Buffer.from(this.secretKey, "hex");

        const signature = crypto
          .createHmac("sha256", secretKey)
          .update(originalData)
          .digest("hex");
```

```
        const originalSignatureCheck = crypto
          .createHmac("sha256", secretKey)
          .update(originalData)
          .digest("hex");
        const isValidOriginal = signature ===
originalSignatureCheck;

        const modifiedData = originalData + " [MODIFIED]";
        const modifiedSignatureCheck = crypto
          .createHmac("sha256", secretKey)
          .update(modifiedData)
          .digest("hex");
        const isValidModified = signature ===
modifiedSignatureCheck;

        const wrongSignature = crypto
          .createHmac("sha256", secretKey)
          .update("wrong data")
          .digest("hex");
        const isValidWrongSig = signature === wrongSignature;

        const endTime = performance.now();
        const duration = endTime - startTime;

        const success =
          isValidOriginal === true &&
          isValidModified === false &&
          isValidWrongSig === false;

        const result = {
          testId: testCase.id,
          description: testCase.description,
          originalDataSize: originalData.length,
          signatureSize: signature.length,
          originalHash: originalHash.substring(0, 16) + "...",
          signaturePreview: signature.substring(0, 16) + "...",
          signatureValidation: {
            originalData: isValidOriginal,
            modifiedData: isValidModified,
            wrongSignature: isValidWrongSig,
          },
          securityLevel: "HMAC-SHA256 with SECRET_KEY",
          duration: duration.toFixed(3),
          success: success,
          timestamp: new Date().toISOString(),
```

```
        };

        this.testResults.digitalSignature.push(result);

        if (success) {
          console.log(
            `☑ BERHASIL - Validasi: Original(${isValidOriginal}),
Modified(${isValidModified}), Wrong(${isValidWrongSig}) -
${duration.toFixed(
              3
            )}ms`
          );
        } else {
          console.log(
            `❌ GAGAL - Validasi tidak sesuai ekspektasi -
${duration.toFixed(
              3
            )}ms`
          );
        }
      } catch (error) {
        const result = {
          testId: testCase.id,
          description: testCase.description,
          error: error.message,
          duration: 0,
          success: false,
          timestamp: new Date().toISOString(),
        };

        this.testResults.digitalSignature.push(result);
        console.log(`X ERROR: ${error.message}`);
      }
    }
  }

  /**
   * 8. PENGUJIAN KEAMANAN DAN KETAHANAN TERHADAP MODIFIKASI
   * Menguji integritas data dengan pendekatan serupa pengujian
lainnya
   */
  async testSecurity() {
    console.log(
      "\n=== 8. PENGUJIAN KEAMANAN DAN KETAHANAN TERHADAP MODIFIKASI
CHIPERTEXT DAN INITIALIZATION VECTOR ==="
```

```
    );

    for (let i = 0; i < this.testData.length; i++) {
      const testCase = this.testData[i];
      console.log(`\nTest ${i + 1}/${this.testData.length}:
${testCase.id}`);

      try {
        const originalEncrypted =
generateShortStringData(testCase.data);

        const [iv, ciphertext] = originalEncrypted.split(":");
        const modifyIndex = Math.floor(Math.random() *
ciphertext.length);
        const modifiedChar = ciphertext[modifyIndex] === "A" ? "B" :
"A";
        const modifiedCiphertext =
          ciphertext.substring(0, modifyIndex) +
          modifiedChar +
          ciphertext.substring(modifyIndex + 1);
        const modifiedEncrypted = `${iv}:${modifiedCiphertext}`;

        const startTime1 = performance.now();
        const decryptModified =
decryptAndDecompress(modifiedEncrypted);
        const endTime1 = performance.now();

        const test1Result = {
          test: "Modified Ciphertext",
          shouldFail: true,
          modificationIndex: modifyIndex,
          actualResult: decryptModified === null ? "FAILED" :
"SUCCESS",
          testTime: `${(endTime1 - startTime1).toFixed(2)}ms`,
          passed: decryptModified === null,
        };

        const modifyIndexIV = Math.floor(Math.random() * iv.length);
        const modifiedCharIV = iv[modifyIndexIV] === "A" ? "B" :
"A";
        const modifiedIV =
          iv.substring(0, modifyIndexIV) +
          modifiedCharIV +
          iv.substring(modifyIndexIV + 1);
        const modifiedEncrypted2 = `${modifiedIV}:${ciphertext}`;
```

```
        const startTime2 = performance.now();
        const decryptModifiedIV =
decryptAndDecompress(modifiedEncrypted2);
        const endTime2 = performance.now();

        const test2Result = {
          test: "Modified IV",
          shouldFail: true,
          actualResult: decryptModifiedIV === null ? "FAILED" :
"SUCCESS",
          testTime: `${(endTime2 - startTime2).toFixed(2)}ms`,
          passed: decryptModifiedIV === null,
        };

        const wrongFormat = originalEncrypted.replace(":", "_");

        const startTime3 = performance.now();
        const decryptWrongFormat =
decryptAndDecompress(wrongFormat);
        const endTime3 = performance.now();

        const test3Result = {
          test: "Wrong Format",
          shouldFail: true,
          actualResult: decryptWrongFormat === null ? "FAILED" :
"SUCCESS",
          testTime: `${(endTime3 - startTime3).toFixed(2)}ms`,
          passed: decryptWrongFormat === null,
        };

        const totalTime = (
          parseFloat(test1Result.testTime.replace("ms", "")) +
          parseFloat(test2Result.testTime.replace("ms", "")) +
          parseFloat(test3Result.testTime.replace("ms", ""))
        ).toFixed(2);

        const securityResult = {
          testCase: testCase.id,
          description: testCase.description,
          tests: [test1Result, test2Result, test3Result],
          overallSuccess:
            test1Result.passed && test2Result.passed &&
test3Result.passed,
          totalTestTime: `${totalTime}ms`,
```

```
          attacksDetected: [test1Result, test2Result,
test3Result].filter(
            (t) => t.passed
          ).length,
          success:
            test1Result.passed && test2Result.passed &&
test3Result.passed,
          timestamp: new Date().toISOString(),
        };

        this.testResults.security.push(securityResult);

        console.log(
          `   Test 1 - Modified Ciphertext: ${
            test1Result.passed ? "✅ TERDETEKSI" : "❌ TIDAK
TERDETEKSI"
          } (${test1Result.testTime})`
        );
        if (decryptModified) {
          console.log("   Decrypted Modified Ciphertext:",
decryptModified);
          console.log("   ChiperText:", modifiedEncrypted);
        }
        console.log(
          `   Test 2 - Modified IV: ${
            test2Result.passed ? "✅ TERDETEKSI" : "❌ TIDAK
TERDETEKSI"
          } (${test2Result.testTime})`
        );
        if (decryptModifiedIV) {
          console.log("   Decrypted Modified IV:",
decryptModifiedIV);
          console.log("   IV:", modifiedIV);
        }
        console.log(
          `   Test 3 - Wrong Format: ${
            test3Result.passed ? "✅ TERDETEKSI" : "❌ TIDAK
TERDETEKSI"
          } (${test3Result.testTime})`
        );

        if (decryptWrongFormat) {
          console.log("   Decrypted Wrong Format:",
decryptWrongFormat);
          console.log("   Wrong Format:", wrongFormat);
```

```
        }

        if (securityResult.success) {
          console.log(
            `☑ BERHASIL - Sistem menolak
${securityResult.attacksDetected}/3 serangan - ${totalTime}ms`
          );
        } else {
          console.log(
            `❌ GAGAL - Sistem hanya menolak
${securityResult.attacksDetected}/3 serangan - ${totalTime}ms`
          );
        }
      } catch (error) {
        const result = {
          testCase: testCase.id,
          description: testCase.description,
          error: error.message,
          success: false,
          timestamp: new Date().toISOString(),
        };

        this.testResults.security.push(result);
        console.log(`X ERROR: ${error.message}`);
      }
    }
  }

  /**
   * 9. PENGUJIAN KECEPATAN PROSES (Performance Testing)
   */
  async testPerformance() {
    console.log("\n=== 9. PENGUJIAN KECEPATAN PROSES ===");

    const performanceData = [];

    for (let i = 0; i < this.testData.length; i++) {
      const testCase = this.testData[i];
      console.log(`\nTest ${i + 1}/${this.testData.length}:
${testCase.id}`);

      const iterations = 10;
      const times = {
        compression: [],
        encryption: [],
```

```
        qrGeneration: [],
        decryption: [],
        decompression: [],
        total: [],
      };
      const mainFields = [
        testCase.data.kepada,
        testCase.data.tempat_tujuan,
        testCase.data.nama_prodi,
        testCase.data.nama_ttd,
        testCase.data.tanggal_hijriyah,
        testCase.data.tanggal_masehi,
      ].join("#");
      const tableData = testCase.data.tableData
        .map((item) =>
`${item.no}:${item.nama_mahasiswa}:${item.nim}`)
        .join("|");
      const originalData = `${mainFields}#${tableData}`;

      for (let i = 0; i < iterations; i++) {
        try {
          const totalStart = performance.now();

          const compStart = performance.now();
          const compressedData = pako.Deflate(originalData, {
            level: 9,
            raw: true,
          });
          const compEnd = performance.now();
          times.compression.push(compEnd - compStart);

          const encStart = performance.now();
          const encrypted = generateShortStringData(testCase.data);
          const encEnd = performance.now();
          times.encryption.push(encEnd - encStart);

          const qrStart = performance.now();
          const qrPath = path.join(
            __dirname,
            `../templates/qr-code/perf_test_${testCase.id}_${i}.png`
          );
          await generateQRCode(encrypted, qrPath);
          const qrEnd = performance.now();
          times.qrGeneration.push(qrEnd - qrStart);
```

```
          const decStart = performance.now();
          const decrypted = decryptAndDecompress(encrypted);
          const decEnd = performance.now();
          times.decryption.push(decEnd - decStart);

          const decompStart = performance.now();
          if (decrypted) {
            const [iv, ciphertext] = encrypted.split(":");
            const algorithm = "aes-128-cbc";
            const key = Buffer.from(process.env.SECRET_KEY, "hex");
            const ivBuffer = Buffer.from(iv, "base64");
            const encryptedBuffer = Buffer.from(ciphertext,
"base64");
            const decipher = crypto.createDecipheriv(algorithm, key,
ivBuffer);
            const decryptedBuffer = Buffer.concat([
              decipher.update(encryptedBuffer),
              decipher.final(),
            ]);
            const decryptedArray = new Uint8Array(decryptedBuffer);
            const decompressedData = pako.InflateRaw(decryptedArray,
{
              to: "string",
            });
          }
          const decompEnd = performance.now();
          times.decompression.push(decompEnd - decompStart);

          const totalEnd = performance.now();
          times.total.push(totalEnd - totalStart);
          if (!decrypted) {
            console.log(`✗ Decryption failed for ${testCase.id}`);
          }

          if (fs.existsSync(qrPath)) {
            fs.unlinkSync(qrPath);
          }
        } catch (error) {
          console.log(`✗ Iteration ${i + 1} failed:
${error.message}`);
        }
      }

      const performanceCriteria = {
        compression: 5.0,
```

```
    decompression: 5.0,
    encryption: 10.0,
    decryption: 10.0,
    qrGeneration: 50.0,
    total: 1000.0,
  };

  const averages = {};
  const passFailStatus = {};

  for (const [process, timeArray] of Object.entries(times)) {
    if (timeArray.length > 0) {
      const avgTime =
        timeArray.reduce((a, b) => a + b, 0) / timeArray.length;
      averages[process] = {
        average: avgTime.toFixed(2),
        min: Math.min(...timeArray).toFixed(2),
        max: Math.max(...timeArray).toFixed(2),
        samples: timeArray.length,
      };

      const criteria = performanceCriteria[process];
      if (criteria !== undefined) {
        passFailStatus[process] = {
          passed: avgTime < criteria,
          criteria: criteria,
          actualTime: avgTime,
        };
      }
    }
  }

  const allCriteriaResults = Object.values(passFailStatus);
  const overallSuccess =
    allCriteriaResults.length > 0 &&
    allCriteriaResults.every((status) => status.passed);

  performanceData.push({
    testCase: testCase.id,
    dataSize: JSON.stringify(testCase.data).length,
    performance: averages,
    performanceCriteria: passFailStatus,
    success: overallSuccess,
  });
```

```
      console.log(
        `${passFailStatus.compression?.passed ? "☑" : "❌"}
Compression: ${
          averages.compression?.average || "N/A"
        }ms (criteria: <${performanceCriteria.compression}ms)`
      );
      console.log(
        `${passFailStatus.encryption?.passed ? "☑" : "❌"}
Encryption: ${
          averages.encryption?.average || "N/A"
        }ms (criteria: <${performanceCriteria.encryption}ms)`
      );
      console.log(
        `${passFailStatus.qrGeneration?.passed ? "☑" : "❌"} QR
Generation: ${
          averages.qrGeneration?.average || "N/A"
        }ms (criteria: <${performanceCriteria.qrGeneration}ms)`
      );
      console.log(
        `${passFailStatus.decryption?.passed ? "☑" : "❌"}
Decryption: ${
          averages.decryption?.average || "N/A"
        }ms (criteria: <${performanceCriteria.decryption}ms)`
      );
      console.log(
        `${passFailStatus.decompression?.passed ? "☑" : "❌"}
Decompression: ${
          averages.decompression?.average || "N/A"
        }ms (criteria: <${performanceCriteria.decompression}ms)`
      );
      console.log(
        `${passFailStatus.total?.passed ? "☑" : "❌"} Total
Process: ${
          averages.total?.average || "N/A"
        }ms (criteria: <${performanceCriteria.total}ms)`
      );
      console.log(
        `🎯 Overall Performance: ${
          overallSuccess ? "☑ LULUS" : "❌ TIDAK LULUS"
        }`
      );
    }

    this.testResults.performance = performanceData;
  }
```

```
  /**
   * PRINT ALL TEST SUMMARIES AT THE END
   */
  printAllTestSummaries() {
    console.log("\n" + "=".repeat(60));
    console.log("🎯 KESIMPULAN PENGUJIAN SEMUA KATEGORI");
    console.log("=".repeat(60));

    this.printTestSummary(
      "1. PENGUJIAN KOMPRESI",
      this.testResults.compression
    );
    console.log("=".repeat(60));

    this.printTestSummary("2. PENGUJIAN ENKRIPSI",
this.testResults.encryption);
    console.log("=".repeat(60));

    this.printTestSummary(
      "3. PENGUJIAN KONVERSI KE QR CODE",
      this.testResults.qrcode
    );
    console.log("=".repeat(60));

    this.printTestSummary(
      "4. PENGUJIAN VALIDASI QR CODE",
      this.testResults.validation
    );
    console.log("=".repeat(60));

    this.printTestSummary("5. PENGUJIAN DEKRIPSI",
this.testResults.decryption);
    console.log("=".repeat(60));

    this.printTestSummary(
      "6. PENGUJIAN DEKOMPRESI",
      this.testResults.decompression
    );
    console.log("=".repeat(60));

    this.printTestSummary(
      "7. PENGUJIAN VALIDASI TANDA TANGAN DIGITAL",
      this.testResults.digitalSignature
    );
```

```
    console.log("=".repeat(60));

    this.printTestSummary("8. PENGUJIAN KEAMANAN",
this.testResults.security);
    console.log("=".repeat(60));

    this.printTestSummary(
      "9. PENGUJIAN KECEPATAN",
      this.testResults.performance
    );
    console.log("=".repeat(60));
  }

  /**
   * GENERATE COMPREHENSIVE REPORT
   */
  generateReport() {
    const timestamp = new Date().toISOString().replace(/[:.]/g, "-
");
    const reportPath = path.join(
      __dirname,
      `../reports/system_test_report_${timestamp}.json`
    );

    const reportsDir = path.dirname(reportPath);
    if (!fs.existsSync(reportsDir)) {
      fs.mkdirSync(reportsDir, { recursive: true });
    }

    const report = {
      timestamp: new Date().toISOString(),
      testSummary: {
        totalTestCases: this.testData.length,
        testCategories: Object.keys(this.testResults).length,
        overallStatus: this.calculateOverallStatus(),
      },
      detailedResults: this.testResults,
      recommendations: this.generateRecommendations(),
    };

    fs.writeFileSync(reportPath, JSON.stringify(report, null, 2));

    console.log("\n=== LAPORAN PENGUJIAN SISTEM ===");
    console.log(`📊 Total Test Cases:
${report.testSummary.totalTestCases}`);
```

```
    console.log(`📋 Test Categories:
${report.testSummary.testCategories}`);
    console.log(`☑️ Overall Status:
${report.testSummary.overallStatus}`);
    console.log(`📄 Full Report: ${reportPath}`);

    return reportPath;
  }

  calculateOverallStatus() {
    let totalTests = 0;
    let passedTests = 0;

    for (const [category, results] of
Object.entries(this.testResults)) {
      if (Array.isArray(results)) {
        totalTests += results.length;
        passedTests += results.filter(
          (r) => r.success || r.passed || r.overallSuccess
        ).length;
      }
    }

    return totalTests > 0
      ? `${passedTests}/${totalTests} (${(
          (passedTests / totalTests) *
          100
        ).toFixed(1)}%)`
      : "No tests completed";
  }

  generateRecommendations() {
    const recommendations = [];

    const compressionResults = this.testResults.compression;
    if (compressionResults.some((r) =>
parseFloat(r.compressionRatio) < 10)) {
      recommendations.push(
        "Kompresi kurang efektif pada data kecil. Pertimbangkan
threshold minimum untuk kompresi."
      );
    }

    const performanceResults = this.testResults.performance;
    if (
```

```
      performanceResults.some(
        (r) => parseFloat(r.performance.total?.average) > 1000
      )
    ) {
      recommendations.push(
        "Performance dapat ditingkatkan dengan optimasi algoritma
atau caching."
      );
    }

    const securityResults = this.testResults.security;
    if (securityResults.some((r) => !r.overallSuccess)) {
      recommendations.push(
        "CRITICAL: Ditemukan kerentanan keamanan. Perbaikan segera
diperlukan."
      );
    }

    if (recommendations.length === 0) {
      recommendations.push(
        "Sistem telah lulus semua pengujian dengan baik. Lanjutkan
dengan deployment."
      );
    }

    return recommendations;
  }

  /**
   * RUN ALL TESTS
   */
  async runAllTests() {
    console.log("🚀 MEMULAI PENGUJIAN SISTEM KOMPREHENSIF");
    console.log("====================================");

    try {
      await this.testCompression();
      await this.testEncryption();
      await this.testQRCodeGeneration();
      await this.testQRCodeValidation();
      await this.testDecryption();
      await this.testDecompression();
      await this.testDigitalSignature();
      await this.testSecurity();
      await this.testPerformance();
```

```
      this.printAllTestSummaries();

      const reportPath = this.generateReport();
      console.log("secret key", process.env.SECRET_KEY);

      console.log("\n🎉 PENGUJIAN SISTEM SELESAI");
      console.log("============================");
      console.log(`📄 Laporan lengkap tersimpan di: ${reportPath}`);

      return reportPath;
    } catch (error) {
      console.error("❌ Error during system testing:", error);
      throw error;
    }
  }
}

module.exports = SystemTesting;

if (require.main === module) {
  console.log("🚀 STARTING COMPREHENSIVE SYSTEM TESTING");
  console.log("========================================\n");

  const tester = new SystemTesting();
  tester.runAllTests().catch((error) => {
    console.error("❌ System testing failed:", error.message);
    process.exit(1);
  });
}
```

Lampiran 2. Surat Keterangan Bebas Plagiat

MAJELIS PENDIDIKAN TINGGI PIMPINAN PUSAT MUHAMMADIYAH
UNIVERSITAS MUHAMMADIYAH MAKASSAR
UPT PERPUSTAKAAN DAN PENERBITAN
Alamat kantor: Jl.Sultan Alauddin N0.259 Makassar 90221 Tlp.(0411) 866972,881593, Fax.(0411) 865588

بسم الله الرحمن الرحيم

SURAT KETERANGAN BEBAS PLAGIAT

UPT Perpustakaan dan Penerbitan Universitas Muhammadiyah Makassar, Menerangkan bahwa mahasiswa yang tersebut namanya di bawah ini:

Nama : Muhammad Ikhsan Nur
Nim : 105841100820
Program Studi : Teknik Informatika

Dengan nilai:

No	Bab	Nilai	Ambang Batas
1	Bab I	9%	10 %
2	Bab 2	14%	25 %
3	Bab 3	9%	10 %
4	Bab 4	5%	10 %
5	Bab 5	2%	5 %

Dinyatakan telah lulus cek plagiat yang diadakan oleh UPT- Perpustakaan dan Penerbitan Universitas Muhammadiyah Makassar Menggunakan Aplikasi Turnitin.

Demikian surat keterangan ini diberikan kepada yang bersangkutan untuk dipergunakan seperlunya.

Makassar, 28 Agustus 2025
Mengetahui,

Kepala UPT- Perpustakaan dan Pernerbitan,

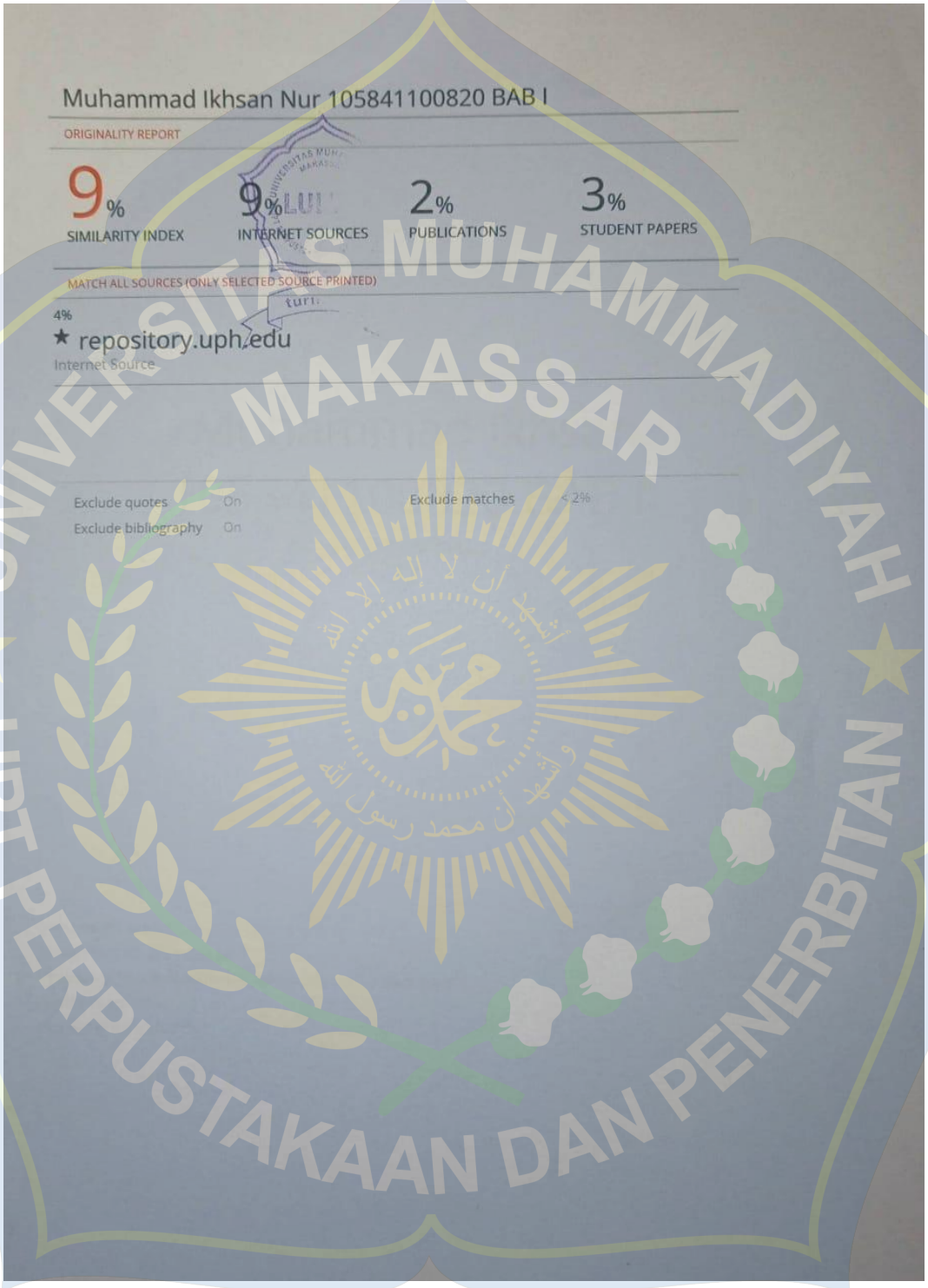
Nurshinta, S.Hum., M.I.P
NBM. 964 591

Jl. Sultan Alauddin no 259 makassar 90222
Telepon (0411)866972,881 593,fax (0411)865 588
Website: www.library.unismuh.ac.id
E-mail : perpustakaan@unismuh.ac.id

Lampiran 3. Hasil Uji Plagiasi

Muhammad Ikhsan Nur
105841100820 BAB I
by Tahap Tutup

Submission date: 27-Aug-2025 02:31PM (UTC+0700)
Submission ID: 2736051150
File name: TUTUP-Muhammad_Ikhsan_Nur-BAB_I_1.docx (61.6K)
Word count: 650
Character count: 4483



Muhammad Ikhsan Nur 105841100820 BAB I

ORIGINALITY REPORT

9%	9%	2%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

MATCH ALL SOURCES (ONLY SELECTED SOURCE PRINTED)

4%

★ repository.uph.edu

Internet Source

Exclude quotes	On	Exclude matches	< 2%
Exclude bibliography	On		

Muhammad Ikhsan 105841100820 BAB II

ORIGINALITY REPORT

14%	10%	12%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Siti Nurhasanah Nugraha. "PENERAPAN ALGORITMA KRIPTOGRAFI ELGAMAL PADA APLIKASI PENGAMANAN PESAN BERBASIS WEBSITE", Jurnal Informatika dan Teknik Elektro Terapan, 2024 Publication	10%
2	idwebhost.com Internet Source	2%
3	publikasi.dinus.ac.id Internet Source	2%

Exclude quotes On
Exclude bibliography On
Exclude matches < 2%

Muhammad Ikhsan
105841100820 BAB III
by Tahap Tutup

Submission date: 27-Aug-2025 01:38PM (UTC+0700)
Submission ID: 2736033990
File name: TUTUP-Muhammad_Ikhsan_Nur-BAB_III.docx (395.52K)
Word count: 1258
Character count: 7904

Muhammad Ikhsan 105841100820 BAB III

ORIGINALITY REPORT

9%	9%	7%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

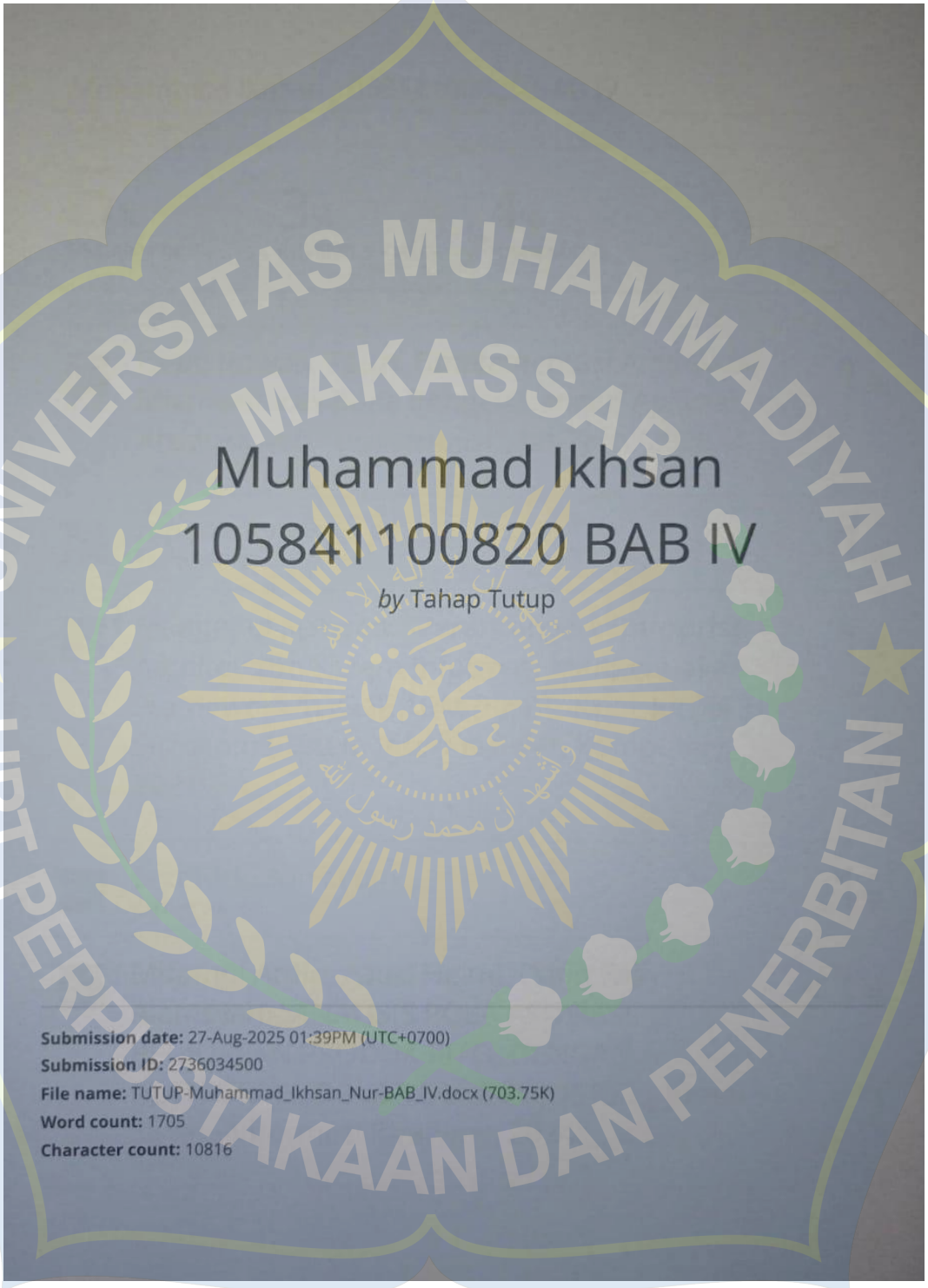
MATCH ALL SOURCES (ONLY SELECTED SOURCE PRINTED)

8%

★ adoc.pub

Internet Source

Exclude quotes	On	Exclude matches	< 2%
Exclude bibliography	On		



Muhammad Ikhsan 105841100820 BAB IV

ORIGINALITY REPORT

5%	3%	4%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Dede Irawan, Fauzi. "Implementasi Arsitektur Microservices pada Pengembangan Aplikasi Absensi Web Terdistribusi", bit-Tech, 2025 Publication	1%
2	repository.ipb.ac.id Internet Source	1%
3	Mangapul Siahaan. "Perancangan Enterprise Architecture Sistem Informasi Menggunakan Framework TOGAF ADM 9.2 PT. XYZ", Jurnal Sisfokom (Sistem Informasi dan Komputer), 2021 Publication	1%
4	text-id.123dok.com Internet Source	1%
5	Miftahul Arifin, Fauzi Helmi, Dafiq Fajri Alamsyah. "ANALISIS POLA ASOSIASI PENJUALAN PRODUK RITEL DENGAN PLATFORM GOOGLE COLAB", JUSTIFY : Jurnal Sistem Informasi Ibrahimy, 2024 Publication	<1%

Muhammad Ikhsan Nur 105841100820 BAB V

ORIGINALITY REPORT

2%	2%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	es.scribd.com Internet Source	2%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		