Analisis dan Implementasi *Watermarking* Digital pada Dokumen PDF Menggunakan Python

Zaki Yudhistira Candra - 13522031
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: 13522031@std.stei.itb.ac.id

Penggunaan dokumen digital dalam format *Portable Document Format* (PDF) semakin meluas seiring dengan perkembangan teknologi informasi, tetapi di sisi lain, meningkatkan risiko penyalinan, modifikasi, dan penyalahgunaan dokumen tanpa izin yang dapat mengancam keamanan data dan integritas dokumen. Salah satu teknik yang dapat digunakan untuk mengatasi permasalahan tersebut adalah *digital watermarking*. Makalah ini membahas analisis dan implementasi *watermarking* digital pada dokumen PDF menggunakan bahasa pemrograman Python. Sistem yang dikembangkan berupa *command-line interface* (CLI) tools yang menerima masukan berupa *file* PDF dan gambar watermark berwarna hitam, kemudian menyisipkan *watermark* secara otomatis ke seluruh halaman dokumen. Implementasi dilakukan dengan memanfaatkan pustaka Python, yaitu pypdf dan reportlab, untuk memproses dan memodifikasi struktur dokumen PDF. Hasil pengujian menunjukkan bahwa *watermarking* digital dapat diterapkan secara efektif tanpa mengganggu keterbacaan isi dokumen, serta *watermark* tetap melekat pada dokumen hasil salinan.

Kata Kunci—*digital watermarking*, PDF, dokumen

I. INTRODUCTION (*HEADING 1*)

Perkembangan teknologi informasi telah mendorong transformasi digital yang masif di berbagai sektor vital, termasuk pendidikan, pemerintahan, dan korporasi. Dalam ekosistem ini, *Portable Document Format* (PDF) telah menjadi standar *de facto* untuk pertukaran dokumen elektronik. Popularitas ini didukung oleh data dari Adobe yang mencatat bahwa terdapat lebih dari 2,5 triliun dokumen PDF yang beredar di seluruh dunia, dengan miliaran file PDF dibuka setiap harinya melalui berbagai layanan *cloud* dan *mobile*. PDF menjadi pilihan utama karena kemampuannya menjaga integritas visual (*visual fidelity*) lintas *platform*, mulai dari Windows, macOS, hingga perangkat *mobile*.

Namun, kebederadaran dokumen PDF yang di mana-mana membawa tantangan keamanan yang serius. Meningkatnya ketergantungan pada dokumen digital berbanding lurus dengan risiko kebocoran data (*data leakage*) dan pemalsuan dokumen (*document forgery*). Berdasarkan laporan *Association of Certified Fraud Examiners (ACFE)*, manipulasi dokumen digital merupakan salah satu metode penipuan paling umum yang menyebabkan kerugian finansial signifikan bagi organisasi. Kelemahan mendasar pada PDF standar adalah sifatnya yang statis dan mudah direplikasi; tanpa perlindungan tambahan, sebuah dokumen rahasia dapat disalin (Copy-Paste), diedit menggunakan *PDF editor*, atau didistribusikan ulang tanpa meninggalkan jejak audit digital. Metode pengamanan konvensional seperti enkripsi kata sandi (*password protection*) sering kali tidak cukup efektif karena rentan terhadap serangan *brute-force* atau menjadi tidak berguna setelah kata sandi dibagikan kepada pengguna yang berwenang (masalah *insider threat*).

Oleh karena itu, diperlukan lapisan keamanan tambahan yang melekat langsung pada konten dokumen, yaitu Digital *Watermarking*. Berbeda dengan enkripsi yang melindungi akses, *watermarking* melindungi konten setelah akses diberikan. Riset dalam bidang forensik digital menunjukkan bahwa *watermarking* efektif sebagai mekanisme *copyright protection* dan *traitor tracing* (pelacakan pelaku kebocoran). Dengan menyisipkan informasi kepemilikan—baik secara *visible* (tampak) maupun *invisible* (tak tampak)—ke dalam struktur file PDF, pemilik dokumen dapat membuktikan otentikasi dokumen dan mencegah klaim kepemilikan ilegal.

Dalam konteks implementasi teknis, penggunaan bahasa pemrograman Python menawarkan solusi yang efisien, *scalable*, dan *open source*. Pustaka Python seperti PyPDF2, ReportLab, atau pdfrw memungkinkan manipulasi struktur PDF dengan biaya komputasi yang relatif rendah dibandingkan perangkat lunak berbayar. Hal ini memungkinkan otomatisasi proses *watermarking* pada ribuan dokumen sekaligus dalam waktu singkat, hal yang sangat relevan untuk kebutuhan industri saat ini.

Berdasarkan permasalahan keamanan tersebut, makalah ini membahas analisis dan implementasi teknik digital *watermarking* pada dokumen PDF menggunakan bahasa pemrograman Python sebagai salah satu solusi untuk meningkatkan keamanan data dan menjaga integritas dokumen digital, khususnya dalam mencegah perubahan tidak sah, serta pelanggaran hak cipta.

II. LANDASAN TEORI

A. *Format* PDF.

Portable Document Format (PDF) merupakan standar dokumen digital yang dibentuk oleh Adobe untuk menjamin konsistensi visual data saat pertukaran informasi, terlepas dari perbedaan perangkat keras, perangkat lunak, ataupun sistem operasi pengguna. PDF kini telah diakui sebagai standar terbuka oleh ISO.

PDF menawarkan fleksibilitas tinggi melalui fitur interaktif seperti integrasi multimedia, formulir cerdas, dan logika bisnis. Selain itu, format ini mendukung otentikasi melalui tanda tangan elektronik dan memiliki kompatibilitas luas di berbagai platform, termasuk Windows dan macOS.

B. *Python*

Python merupakan bahasa pemrograman tingkat tinggi yang bersifat *interpreted*, *general-purpose*, dan dirancang dengan penekanan pada keterbacaan kode. Python mendukung berbagai paradigma pemrograman, termasuk pemrograman berorientasi objek, prosedural, dan fungsional. Bahasa ini menyediakan struktur data tingkat tinggi serta sistem manajemen memori otomatis sehingga memudahkan pengembang dalam menulis program yang ringkas, jelas, dan mudah dipelihara. Python juga dilengkapi dengan pustaka standar yang luas dan dapat diperluas melalui modul tambahan untuk berbagai kebutuhan aplikasi.

Pada implementasi ini, akan digunakan beberapa pustaka Python yang berperan sebagai alat manipulasi PDF:

1. *Pypdf*:

   Library open source Python yang digunakan untuk membaca, memodifikasi, dan menulis dokumen PDF. Dalam implementasi ini, pypdf digunakan untuk membuka dokumen PDF asli, mengakses setiap halaman, serta menggabungkan (*merge*) *watermark* ke seluruh halaman dokumen.

2. *reportlab*

   Library Python yang berfungsi untuk menghasilkan dokumen PDF secara terprogram. *reportlab* digunakan untuk membuat halaman *watermark* dalam bentuk PDF dari gambar *watermark* yang diberikan, sehingga dapat digabungkan dengan dokumen PDF utama.

3. io (BytesIO)

   Modul bawaan Python yang digunakan untuk mengelola data berbasis memori. Dalam implementasi *watermarking*, BytesIO digunakan untuk menyimpan sementara file PDF *watermark* sebelum digabungkan dengan dokumen utama.

4. sys

   Modul bawaan Python yang digunakan untuk mengelola argumen baris perintah *(command-line arguments*) sehingga program dapat menerima input berupa file PDF dan gambar *watermark* secara fleksibel.

C. *Digital Watermarking*

Digital watermarking merupakan teknik penyisipan informasi tertentu ke dalam suatu objek digital, seperti citra, audio, video, atau dokumen teks, dengan tujuan untuk melindungi kepemilikan, menjaga hak cipta, serta memastikan keaslian konten digital. Informasi yang disisipkan tersebut disebut *watermark*, yang dapat berupa teks, logo, gambar, data biner, atau deretan bilangan tertentu. Penyisipan watermark dilakukan sedemikian rupa sehingga tidak merusak kualitas objek digital yang dilindungi dan tetap melekat meskipun objek tersebut disalin atau didistribusikan.

Pada konteks dokumen digital, *watermarking* berperan sebagai mekanisme pengamanan tambahan yang mampu memberikan identitas kepemilikan serta membantu mendeteksi adanya perubahan atau manipulasi terhadap dokumen.

Prinsip utama digital *watermarking* adalah bahwa *watermark* harus:

1. Terintegrasi ke dalam objek digital dan tidak berdiri sebagai elemen terpisah.
2. Tidak merusak kualitas visual atau isi dari objek digital.
3. Dapat dideteksi atau diekstraksi kembali sebagai bukti kepemilikan, hak cipta, atau adanya modifikasi terhadap konten.

Berbeda dengan metode konvensional seperti penempelan label atau tanda hak cipta secara visual, watermarking digital lebih sulit dihilangkan karena watermark menjadi bagian dari struktur data objek digital itu sendiri.

Berikut adalah praktik aplikasi dari *digital watermarking* yang lazim dilakukan:

1. Identifikasi dan pembuktian kepemilikan *(ownership identification* dan *proof of ownership*)
2. Autentikasi konten dan deteksi perubahan (*tamper proofing*)
3. Pelacakan distribusi dokumen (*transaction tracking*)
4. Perlindungan terhadap pembajakan dan penggandaan ilegal
5. *Monitoring* penyiaran konten digital

D. Jenis *Digital Watermarking*

Digital watermarking dapat diklasifikasikan ke dalam beberapa kategori:

1. *Visible Watermarking*

   Visible watermarking merupakan teknik *watermarking* dengan *watermark* yang dapat terlihat secara langsung / tak kasat mata oleh pengguna, misalnya berupa logo atau teks transparan yang ditampilkan di atas dokumen. Jenis *watermark* ini bertujuan untuk memberikan peringatan visual terkait kepemilikan dan mencegah penggunaan tanpa izin.

2. *Invisible Watermarking*

Invisible watermarking adalah teknik penyisipan watermark yang tidak terlihat secara kasat mata, tetapi dapat dideteksi atau diekstraksi menggunakan metode tertentu. *Invisible watermarking* umumnya digunakan untuk pembuktian kepemilikan, autentikasi, dan pelacakan distribusi konten digital.

Invisible watermarking selanjutnya dibagi menjadi:

a. *Fragile watermarking*, yang *watermark*-nya akan rusak jika terjadi manipulasi sehingga cocok untuk menjaga integritas dan keaslian dokumen.

b. *Robust watermarking,* yang *watermark*-nya tetap bertahan meskipun dokumen mengalami proses seperti kompresi, pemotongan, atau pengeditan sehingga cocok untuk perlindungan hak cipta.

III. Implementasi

Implementasi dari kakas akan berbentuk *command line interface* atau CLI.

A. Gambaran Umum Sistem

Sistem yang dirancang merupakan sebuah CLI *tools* berbasis Python yang berfungsi untuk melakukan *watermarking* digital pada dokumen PDF. Aplikasi ini menerima masukan berupa:

1. *File* dokumen PDF yang akan diberi *watermark*
2. *File* gambar *watermark* berwarna hitam

Watermark akan disisipkan ke seluruh halaman dokumen PDF secara otomatis, kemudian menghasilkan file PDF baru yang telah ter-*watermark*.

B. Spesifikasi

1. Input:
    a. Dokumen PDF
    b. Gambar *Watermark*
2. Output
    a. PDF yang sudah ter-*watermark*.

Input diberikan melalui argumen baris perintah (*command-line arguments*).

```
python watermark_cli.py input.pdf
watermark.png output.pdf
```

C. Flow Aplikasi

Alur kerja sistem dapat dijelaskan sebagai berikut:

1. Pengguna menjalankan program melalui CLI dengan memberikan argumen file PDF dan gambar *watermark*.
2. Sistem melakukan validasi terhadap format dan keberadaan *file input*.
3. Sistem membuat halaman *watermark* dalam format PDF dari gambar *watermark* hitam.
4. *Watermark* disisipkan ke setiap halaman dokumen PDF secara merata.
5. Sistem menghasilkan dokumen PDF baru sebagai hasil *watermarking*.

D. Arsitektur Implementasi

Implementasi sistem memanfaatkan beberapa library Python sebagai berikut:

1. sys / *argparse*
   Digunakan untuk membaca dan memproses argumen baris perintah.
2. *reportlab*
   Digunakan untuk mengonversi gambar *watermark* menjadi halaman PDF.
3. *pypdf*
   Digunakan untuk membaca dokumen PDF asli dan menggabungkan watermark ke seluruh halaman.

E. Karakteristik *Watermark*

Watermark yang diterapkan dalam sistem memiliki karakteristik:

1. Berbentuk gambar berwarna hitam
2. Bersifat *visible watermarking*
3. Disisipkan ke seluruh halaman dokumen
4. Tidak mengubah isi utama dokumen secara signifikan
5. Tetap melekat pada dokumen hasil salinan

Pemilihan *watermark* berwarna hitam bertujuan untuk menjaga keterbacaan serta konsistensi tampilan pada berbagai jenis dokumen PDF.

F. Flow Aplikasi

Alur kerja sistem dapat dijelaskan sebagai berikut:

1. Pengguna menjalankan program melalui CLI dengan memberikan argumen file PDF dan gambar *watermark*.
2. Sistem melakukan validasi terhadap format dan keberadaan *file input*.
3. Sistem membuat halaman *watermark* dalam format PDF dari gambar *watermark* hitam.
4. *Watermark* disisipkan ke setiap halaman dokumen PDF secara merata.
5. Sistem menghasilkan dokumen PDF baru sebagai hasil *watermarking*.

IV. Hasil dan Pengujian

Program yang dikembangkan merupakan sebuah *CLI tools* berbasis Python yang berfungsi untuk menambahkan *watermark* berupa gambar ke seluruh halaman dokumen PDF. Secara umum, program ini menerima tiga argumen masukan, yaitu *file* PDF *asli*, *file* gambar *watermark*, dan nama file PDF keluaran.

Proses *watermarking* dilakukan melalui beberapa tahapan utama yang direpresentasikan dalam fungsi-fungsi terpisah agar struktur program bersifat modular dan mudah dipahami.

A. Struktur Umum Program

Program terdiri dari tiga bagian utama:

1. Fungsi *create_watermark_pdf*() untuk membuat halaman *watermark* dalam format PDF.
2. Fungsi *add_watermark_to_pdf*() untuk menyisipkan *watermark* ke seluruh halaman PDF.
3. Fungsi *main*() sebagai pengendali alur utama program (CLI).

Pendekatan modular ini memisahkan antara proses pembuatan watermark dan proses penyisipannya ke dokumen PDF.

B. Fungsi *create_watermark_pdf()*

Fungsi ini bertanggung jawab untuk mengonversi gambar *watermark* menjadi satu halaman PDF yang nantinya akan digabungkan ke setiap halaman dokumen PDF asli.

Alur kerja fungsi ini adalah sebagai berikut:

1. Program menginisialisasi objek BytesIO sebagai buffer di memori untuk menyimpan PDF *watermark* sementara.
2. Canvas PDF dibuat menggunakan *library* reportlab dengan ukuran halaman letter.
3. Gambar *watermark* dibaca menggunakan *ImageReader*, kemudian ukuran aslinya diambil.
4. Ukuran *watermark* dihitung ulang agar menyesuaikan ukuran halaman PDF (maksimal 70% dari ukuran halaman) dengan tetap menjaga rasio aspek gambar.
5. *Watermark* diposisikan di tengah halaman dan diberikan tingkat transparansi menggunakan *setFillAlpha*(0.3) agar tidak mengganggu keterbacaan dokumen.
6. Gambar *watermark* digambar ke dalam *canvas*, lalu *canvas* disimpan.
7. *Buffer* dikembalikan ke awal posisi dan dikembalikan sebagai keluaran fungsi.

Fungsi ini menghasilkan PDF *watermark* di dalam memori, sehingga tidak diperlukan pembuatan *file* sementara di *disk*.

C. Fungsi *add_watermark_to_pdf()*

Fungsi ini merupakan inti dari proses *watermarking*, yaitu menyisipkan *watermark* ke seluruh halaman PDF.

Tahapan yang dilakukan dalam fungsi ini meliputi:

1. Program memvalidasi keberadaan *file* PDF input dan file gambar *watermark* menggunakan modul os.
2. Fungsi *create_watermark_pdf*() dipanggil untuk menghasilkan halaman *watermark* dalam format PDF.
3. Dokumen PDF asli dibuka menggunakan PdfReader dari *library* PyPDF2.
4. PDF *watermark* dibaca dari *buffer* memori, dan halaman *watermark* diambil sebagai satu halaman referensi.
5. Program menginisialisasi objek PdfWriter untuk menulis dokumen PDF hasil *watermarking*.
6. Program melakukan iterasi terhadap seluruh halaman PDF asli. Pada setiap iterasi:
    1. Halaman PDF diambil
    2. *Watermark* digabungkan ke halaman menggunakan metode *merge_page*()
    3. Halaman hasil penggabungan ditambahkan ke objek *PdfWriter*
7. Setelah seluruh halaman diproses, dokumen PDF hasil *watermarking* disimpan ke file keluaran.

Dengan pendekatan ini, *watermark* diterapkan secara konsisten pada setiap halaman dokumen.

D. Fungsi *main()*

Fungsi *main*() berperan sebagai pengendali alur utama program dan antarmuka CLI.

Alur kerja fungsi ini adalah.

1. Program menampilkan informasi awal aplikasi.
2. Program memeriksa jumlah argumen baris perintah. Jika argumen tidak sesuai, program menampilkan panduan penggunaan dan dihentikan.
3. Argumen baris perintah dipetakan menjadi variabel *input* PDF, *watermark*, dan *output* PDF.
4. Program menampilkan ringkasan parameter yang digunakan.
5. Fungsi *add_watermark_to_pdf*() dipanggil untuk menjalankan proses *watermarking*.
6. Setelah proses selesai, program menampilkan pesan keberhasilan dan berakhir.

Berikut adalah *command* yang dijalankan:

```
python watermark_cli.py input.pdf
watermark.png output.pdf
```

E. Alur Akhir Program

Secara ringkas, alur eksekusi program adalah sebagai berikut:

1. Program dijalankan melalui CLI
2. Argumen input divalidasi
3. PDF *watermark* dibuat dari gambar
4. Dokumen PDF dibaca
5. *Watermark* digabungkan ke setiap halaman
6. Dokumen PDF hasil disimpan
7. Program selesai

F. Jenis *Watermarking*

Implementasi ini merepresentasikan *visible digital watermarking*, *watermark* berupa gambar dapat terlihat secara langsung oleh pengguna. *Watermark* disisipkan secara terintegrasi pada setiap halaman dokumen PDF sehingga tetap terbawa pada hasil penggandaan dokumen dan dapat berfungsi

sebagai penanda kepemilikan serta mekanisme perlindungan dokumen digital.

G. Spesifikasi Pengujian

Akan diberikan beberapa kasus uji sebanyak 2 PDF dengan isi yang berbeda-beda. Berikut adalah *detail* dari spesifikasi tiap PDF:

1. File 1:
    1. Nama: 13522031.pdf
    2. Keterangan: PDF dari makalah berjudul "*Enhancing Security in a Password Manager Using RSA Encryption*"
2. File 2:
    1. Nama: pages.pdf
    2. Keterangan: Tangkapan layar dari *web* Gymale, sebuah *web company profile* Gym yang berlokasi di Surabaya, Jawa Timur.

Berikut adalah gambar *watermark* yang akan digunakan:

Gambar IV. 1. Logo *Watermark.* (Sumber: Apple)

H. Metode Pengujian

Pengujian akan dilakukan dengan beberapa langkah berikut:

1. Akan ditunjukkan *file* sebelum proses *watermarking*.
2. *Watermarking* dilakukan terhadap *file tersebut*.
3. Ditunjukkan tangkapan layar *file* setelah proses *watermarking*.

I. Hasil Pengujian:

1. Kasus Uji 1:

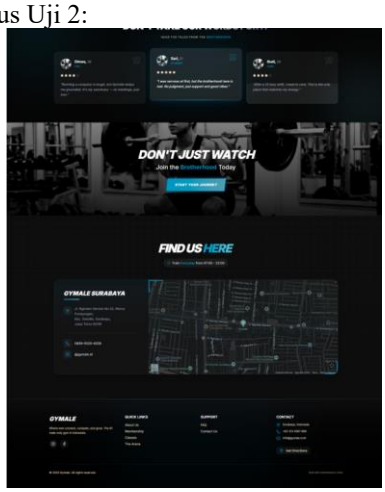
Enhancing Security in a Password Manager Using RSA Encryption

Gambar IV. 2. Dokumen sebelum proses *watermarking* (Sumber: Penulis)

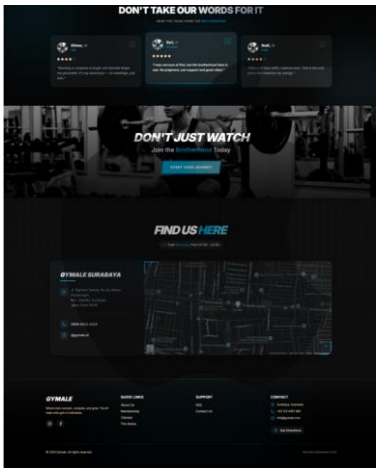
Enhancing Security in a Password Manager Using RSA Encryption

Gambar IV. 3. Dokumen setelah proses *watermarking* (Sumber: Penulis)

2. Kasus Uji 2:



Gambar IV. 4. Dokumen sebelum proses *watermarking* (Sumber: Penulis)



Gambar IV. 5. Dokumen setelah proses *watermarking*
(Sumber: Penulis)

V. KESIMPULAN

Berdasarkan hasil analisis dan implementasi yang telah dilakukan, dapat disimpulkan bahwa aplikasi berbasis CLI *watermarking* digital pada dokumen PDF menggunakan bahasa pemrograman Python berhasil diimplementasikan dan diuji dengan baik. Pengujian menunjukkan bahwa *watermark* berupa logo berwarna hitam dapat disisipkan ke seluruh halaman dokumen PDF secara konsisten tanpa mengganggu keterbacaan isi dokumen.

Meskipun demikian, implementasi yang dikembangkan masih memiliki beberapa keterbatasan. Aplikasi belum menyediakan skema penyesuaian ukuran dan distribusi *watermark* terhadap berbagai macam dimensi halaman PDF. *Watermark* hanya ditempatkan satu kali pada setiap halaman sehingga belum mampu menutupi seluruh area halaman secara merata. Oleh karena itu, pada pengembangan selanjutnya dapat diterapkan teknik pengulangan (*tiling*) *watermark* agar logo dapat menutupi seluruh bagian halaman dokumen PDF.

Selain itu, sistem yang dikembangkan saat ini masih berbasis *command-line interface* (CLI) dan ditujukan untuk pengguna *tech-savvy* atau teknis. Untuk meningkatkan kemudahan penggunaan oleh pengguna umum, sistem ini dapat dikembangkan lebih lanjut dengan menambahkan antarmuka pengguna grafis (*graphical user interface*).

VI. LAMPIRAN

Berikut dilampirkan tautan repositori program:

ACKNOWLEDGMENT

Terima kasih terbesar saya berikan terhadap Bapak Rinaldi Munir selaku dosen pengajar mata kuliah Kriptografi IF. Melalui pengajaran beliau, penulis telah mendapatkan berbagai ilmu baru, khususnya dalam bidang kriptografi. Ilmu ini telah membantu penulis dalam meningkatkan aspek kualitas *security* pada aplikasi yang tengah penulis kembangkan.

Terima kasih juga penulis berikan kepada teman penulis Vanson Kurnialim yang telah memberikan tahu gejrot ketika penulis sedang menyusun makalah ini.

REFERENCES

[1] Adobe. (2020). *Adobe unveils ambitious multi-year vision for PDF: Introduces Liquid Mode*. Adobe Blog.

[2] Smallpdf. (2024). *PDF Statistics & Usage: Tools and Trends Shaping Digital Documents*.

[3] Association of Certified Fraud Examiners (ACFE). (2022). *Occupational Fraud 2022: A Report to the Nations*.

[4] Begum, M., & Uddin, M. S. (2020). *Digital Image Watermarking Techniques: A Review*. *Information*, 11(2), 110.

[5] Adobe. (2 B.C.E.). *Semua yang perlu Anda ketahui tentang PDF*. https://www.adobe.com/id_id/acrobat/about-adobe-pdf.html

[6] Python Software Foundation. (2025). *Python 3 documentation*. Retrieved December 26, 2025, from https://docs.python.org/3/

[7] Munir, R. (2025). *Digital Watermark*. Materi kuliah kriptografi. Institut Teknologi Bandung.

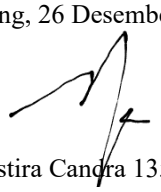
[8] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[9] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740-741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 26 Desember 2025



Zaki Yudhistira Candra 13522031