Implementasi Digital Signature dalam *Service* Platform PDF Signing Terdesentralisasi

Sebuah implementasi berbasis *web*

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

***Abstract*— Perkembangan penggunaan dokumen digital menuntut adanya mekanisme keamanan yang mampu menjamin keaslian dan integritas dokumen, khususnya pada format *Portable Document Format* (PDF) yang banyak digunakan dalam berbagai proses administratif. Penelitian ini mengimplementasikan sistem penandatanganan digital dokumen PDF berbasis *client-side*, proses pembuatan tanda tangan digital dilakukan langsung pada sisi pengguna tanpa melibatkan transmisi kunci privat melalui jaringan. Pendekatan ini meningkatkan aspek keamanan dengan memastikan bahwa kunci privat sepenuhnya berada di bawah kendali pengguna, sementara sisi server hanya berperan dalam penyimpanan kunci publik dan proses verifikasi tanda tangan. Sistem dikembangkan menggunakan *React* dan *React Router* pada sisi *front-end*, serta *Express.js* dan *MongoDB* pada sisi *back-end*. Hasil penelitian menunjukkan bahwa mekanisme penandatanganan dan verifikasi dokumen dapat berjalan dengan baik, serta mampu menjamin integritas dan autentikasi dokumen.**

Kata Kunci: Kriptografi, PDF, ECDSA

I. Latar Belakang

Perkembangan teknologi informasi telah me-*mainstream*-kan penggunaan dokumen digital sebagai media utama dalam berbagai proses, baik proses bisnis maupun administratif. Dokumen digital menawarkan kelebihan dibandingkan dengan dokumen fisik, terutama dalam hal penyimpanan, distribusi, dan integrasi dengan sistem informasi sehingga menggantikan peran dokumen fisik dalam banyak aktivitas organisasi modern. Akibatnya, reliabilitas dan keamanan dokumen digital menjadi faktor krusial dalam menjamin kelangsungan proses-proses tersebut [1].

Pemanfaatan dokumen digital melibatkan berbagai sektor, mulai dari proses bisnis perusahaan, kegiatan akademik, hingga sistem yang bersifat strategis seperti pertahanan dan keamanan. Pada sektor-sektor tersebut, dokumen digital sering memuat informasi penting dan sensitif, seperti kontrak, sertifikat akademik, atau dokumen otorisasi. Apa bila keaslian dan integritas dokumen diselewengkan, misinformasi yang terjadi dapat menimbulkan risiko serius, baik dalam bentuk kerugian finansial, penurunan kepercayaan publik, maupun ancaman terhadap keamanan organisasi [1].

Seiring meningkatnya ketergantungan terhadap dokumen digital, risiko keamanan dalam proses pertukaran dan penyimpanan dokumen juga semakin meningkat. Ancaman seperti pemalsuan dokumen, modifikasi isi tanpa izin, serta serangan dari pihak tidak bertanggung jawab (*adversary*) menuntut adanya mekanisme pengamanan yang mampu menjamin keaslian identitas pengirim dan keutuhan dokumen. Diperlukan solusi keamanan yang bersifat protektif, dan verifikatif secara kriptografis.

Bidang kriptografi telah berkembang pesat dan menghasilkan berbagai mekanisme keamanan yang terstandarisasi dan terbukti secara matematis, salah satunya adalah tanda tangan digital (*digital signature*). Tanda tangan digital memanfaatkan fungsi *hash* kriptografis dan algoritma kunci publik seperti RSA atau ECDSA untuk memastikan autentikasi, penandatangan, serta integritas dokumen. Selain itu, keberadaan *Certificate Authority* (CA) memungkinkan proses verifikasi identitas dilakukan secara terpercaya dalam ekosistem *Public Key Infrastructure* (PKI).

Implementasi tanda tangan digital pada dokumen dalam format *Portable Document Format* (PDF) menjadi relevan, berhubung PDF merupakan format dokumen yang paling umum digunakan dalam pertukaran dokumen, baik resmi maupun tidak resmi. Dengan menerapkan tanda tangan digital pada dokumen PDF, sistem dapat menyediakan mekanisme verifikasi yang mampu mengindikasikan bahwa dokumen tersebut berasal dari suatu entitas tertentu yang sah dan belum mengalami perubahan sejak proses penandatanganan dilakukan.

Dalam praktiknya, kebutuhan akan layanan penandatanganan dokumen yang *scalable* dan terintegrasi mendorong pengembangan *platform* penandatanganan PDF berbasis arsitektur *microservice*. Pendekatan ini memungkinkan proses penandatanganan, verifikasi, dan manajemen sertifikat dilakukan secara terpusat dan tetap fleksibel untuk diintegrasikan dengan berbagai sistem lain. Apabila terjadi perubahan terhadap isi dokumen PDF setelah ditandatangani, proses verifikasi tanda tangan digital akan gagal, sehingga dokumen dinyatakan tidak valid. Dengan demikian, makalah ini akan mengeksplorasi dari implementasi tanda tangan digital dalam *service* platform PDF *signing* terpusat. Diharapkan studi implementasi *platform* tersebut mampu meningkatkan wawasan

akan keamanan, praktik, dan elemen penting lainnya dalam pengelolaan dokumen digital.

II. Dasar Teori.

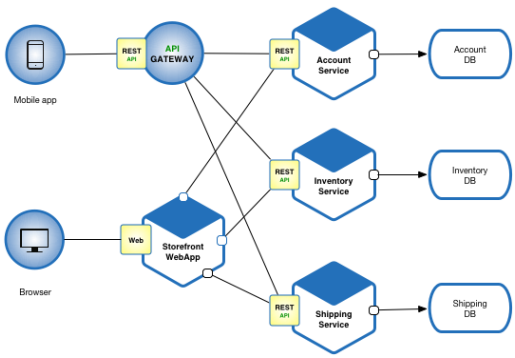
A. *Format* PDF.

Portable Document Format (PDF) merupakan format dokumen digital yang dikembangkan oleh Adobe untuk menyediakan media pertukaran dan penyajian dokumen yang konsisten serta dapat diandalkan. Format ini memungkinkan dokumen ditampilkan dengan tampilan yang sama tanpa bergantung pada perangkat lunak, perangkat keras, ataupun sistem operasi yang digunakan oleh pengguna.

Saat ini, PDF telah menjadi standar terbuka yang ditetapkan oleh *International Organization for Standardization* (ISO). Dokumen dalam format PDF mendukung berbagai fitur interaktif, seperti tautan, tombol, bidang formulir, serta integrasi media audio dan video, termasuk penerapan logika bisnis. Selain itu, dokumen PDF dapat ditandatangani secara elektronik dan diakses dengan mudah pada berbagai sistem operasi, seperti Windows dan macOS, melalui aplikasi pembaca PDF seperti Adobe Acrobat Reader.

B. Arsitektur *Microservices.*

Microservice adalah pendekatan arsitektur perangkat lunak yang mendikte bahwa sebuah sistem dibangun dari kumpulan layanan-layanan kecil yang berdiri sendiri. Setiap layanan dirancang berdasarkan fungsi atau kebutuhan bisnis tertentu dan dapat dikembangkan, dijalankan, serta diperbarui secara terpisah tanpa harus memengaruhi layanan lainnya. *Microservice* merupakan salah satu bentuk dari arsitektur *Service-Oriented Architecture* (SOA), tetapi dengan penekanan khusus pada pemisahan layanan yang jelas dan kemampuan untuk melakukan *deployment* secara mandiri.



Gambar II. 1. Contoh rancangan sistem *microservice.*
(Sumber: https://microservices.io)

Dari sisi teknis, setiap *microservice* berkomunikasi dengan *microservice* lain melalui jaringan, misalnya menggunakan protokol HTTP atau API. Selain itu, setiap *microservice* memiliki dan mengelola penyimpanan datanya sendiri, seperti basis data, yang tidak dapat diakses langsung oleh layanan lain. Dengan cara ini, data dan logika aplikasi tetap terisolasi di dalam batas layanan masing-masing, sehingga sistem menjadi lebih fleksibel, mudah dikembangkan, dan lebih tahan terhadap perubahan atau kegagalan pada salah satu layanan.

Pada implementasi ini, *microservice* hanya digunakan sebatas servis repositori tanda tangan publik dan servis *frontend*. Lebih lanjutnya terkait wujud arsitektur akan dibahas pada bagian III.

C. *Public Key Infrastructure* (PKI)

Public Key Infrastructure (PKI) merupakan suatu infrastruktur teknologi yang mengintegrasikan kriptografi kunci publik, sertifikat digital, serta otoritas tepercaya seperti *Certification Authority* (CA) untuk mendukung proses autentikasi dan pengamanan transaksi elektronik. PKI mencakup kombinasi:

1. perangkat keras
2. perangkat lunak
3. kebijakan
4. prosedur

yang mengatur seluruh siklus hidup sertifikat digital, mulai dari penerbitan, distribusi, penggunaan, hingga pencabutan sertifikat. Berikut adalah elemen-elemen dalam sebuah PKI:

1. Sertifikat Digital
2. *Certification Authority* (CA)
3. *Registration Authority* (RA)
4. Repositori
5. Aturan/Kebijakan (*Policty*)
6. *Client*

Implementasi ini akan membuat sebuah *microservice* yang beperan sebagai CA, sebagai pintu utama verifikasi yang dapat digunakan oleh pengguna.

D. Tanda Tangan Digital

Digital signature atau tanda tangan digital merupakan sebuah konsep kriptografi yang digunakan untuk menjamin identitas pengirim dan memastikan bahwa isi pesan yang dikirim tidak mengalami perubahan. Tanda tangan berbeda bentuk dari tanda tangan basah biasa, apabila tanda tangan basah akan memiliki pola yang sama untuk setiap replika atau penggunaan, tanda tangan digital memiliki nilai yang berbeda-beda untuk setiap dokumen yang ditandatangani.

Tanda tangan digital secara umum bekerja dengan tahapan berikut:

1. Proses Penandatanganan (*Signing*): Pengirim menggunakan kunci privat miliknya untuk melakukan *hash* atau mengenkripsi pesan.
2. Proses Verifikasi (*Verifying*): Penerima menggunakan kunci publik pengirim untuk mendekripsi atau memverifikasi *hash* pesan.

Dalam implementasi teknisnya, *flow* dari proses penandatanganan digital sering menggunakan algoritme kriptografi seperti algoritme RSA, DSA, dan ECDSA.

E. Digital Signature Algorithm (DSA)

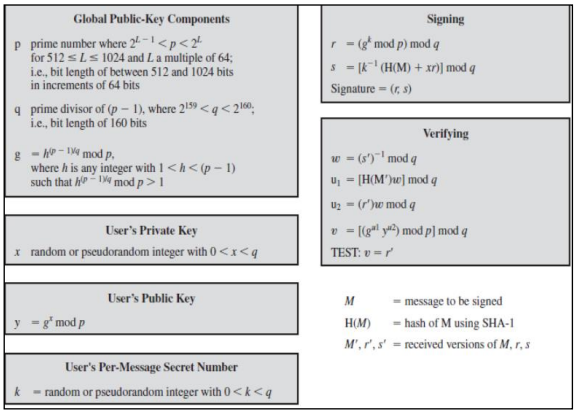
Digital Signature Algorithm atau DSA merupakan algoritme kunci-publik yang menjadi komponen utama dari *Digital Signature Standard* (DSS). DSA dirancang secara khusus hanya untuk keperluan tanda tangan digital saja, bukan untuk mengenkripsi pesan.

DS menggunakan beberapa parameter:

1. Parameter Publik (p, q, g, y). Merupakan bilangan-bilangan prima besar.
2. Parameter Privat (x). Bilangan bulat rahasia yang hanya diketahui oleh pemilik kunci.

Berikut adalah proses kerja DSA:

1. Penandatanganan: Pengirim menghitung nilai *hash* pesan, mencari bilangan acak k, dan menghasilkan sepasang nilai (r, s) yang menjadi tanda tangan digital.
2. Verifikasi: Penerima akan menggunakan kunci publik pengirim atau penanda tangan untuk menghitung sebuah nilai v. Jika nilai v sama dengan r, maka tanda tangan tersebut dinyatakan sah dan autentik.



Gambar II. 2. Ringkasan singkat dari DSA. (Sumber: Rinaldi Munir, 2025)

Apabila DSA menggunakan bilangan prima, terdapat varian DSA yaitu ECDSA yang menggunakan kriptografi kurva eliptik.

F. Elliptic Curve DSA (ECDSA)

Elliptic Curve Digital Signature Algorithm (ECDSA) adalah varian dari algoritma DSA yang dirancang khusus untuk beroperasi menggunakan komputasi kurva eliptik. Alih-alih menggunakan bilangan prima, ECDSA menggunakan properti matematika dari titik-titik pada kurva eliptik. Hal ini menyebabkan efisiensi komputasi yang lebih tinggi daripada DSA. Berikut adalah komponen-komponen pada ECDSA:

1. Persamaan Kurva Eliptik: Menggunakan rumus

$$y^2 = x^3 + ax + b (mod\ p)$$

2. Titik Basis (G): Sebuah titik yang dipilih dari himpunan titik yang berada di dalam kurva eliptik.
3. Bilangan Bulat n: Syaratnya adalah nG = O, O adalah titik di tak hingga.
4. Pasangan Kunci:
   a. Kunci Privat (d): Merupakan sebuah bilangan bulat rahasia milik pengirim pesan.
   b. Kunci Publik (Q): Merupakan sebuah titik pada kurva yang dihasilkan dari perkalian kunci privat dengan titik basis (Q = dG).

Berikut adalah proses tanda tangan:

1. Memilih bilangan acak k dalam rentang

$$1 \le \mathrm{k} \le \mathrm{n} - 1$$

2. Menghitung titik hasil perkalian skalar kG = (x_1, y_1).
3. Menentukan nilai $r = x_1 (mod\ p)$.
4. Menghitung nilai e yang merupakan hasil fungsi hash dari pesan tersebut (misalnya menggunakan SHA-1 atau SHA-2).
5. Menghitung nilai $\mathrm{s} = \mathrm{k} - 1(\mathrm{e} + \mathrm{dr})(mod\ \mathrm{p})$ menggunakan kunci privat (d).
6. Hasil akhir tanda tangan digital adalah sepasang nilai (r, s) yang dikirimkan bersama pesan (m).

Berikut adalah proses verifikasi:

1. Memastikan nilai dan berada dalam selang valid.

$$[1, n-1]$$

2. Menghitung kembali nilai hash pesan e = fungsi_hash(m).
3. Menghitung nilai bantu w, u_1, dan u_2 menggunakan s, e, dan r.
4. Menghitung titik baru pada kurva:

$$(x_1, y_1) = u_1 G + u_2 Q$$

   Q merupakan kunci publik pengirim.

5. Tanda tangan dinyatakan valid jika hasil

$$r = x_1 (mod\ \mathrm{n})$$

   Jika tidak sama, maka tanda tangan tersebut dianggap tidak sah (*invalid*).

G. Fungsi *Hash*

Fungsi hash adalah algoritma yang mengkompresi pesan (m) berukuran sembarang menjadi string (h) yang memiliki ukuran tetap (*fixed length*). Hasil dari fungsi ini dikenal dengan istilah message-digest, nilai hash, atau *digest*.

Berikut dalah beberapa karakteristik fungsi *hash*:

1. Satu arah: Pesan yang telah diubah menjadi *message digest* tidak dapat dikembalikan lagi menjadi pesan semula.
2. Sensitivitas tinggi: Peka terhadap perubahan pada pesan yang di-*hash*.

3. *Collision Resistance*: Sangat sulit untuk menemukan dua input berbeda yang menghasilkan nilai hash yang sama.

Berikut adalah algoritma *hash* yang umum digunakan:

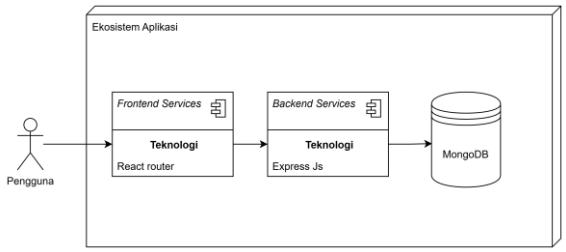
1. MD5: Menghasilkan digest 128 bit, namun saat ini sudah ditemukan kolisinya.
2. SHA-1: Menghasilkan digest 160 bit, sering digunakan dalam DSS (Digital Signature Standard.
3. SHA-2 (termasuk SHA-256): Menghasilkan digest hingga 512 bit dan dianggap masih sangat kuat serta aman.
4. SHA-3 (Keccak): Pemenang kompetisi NIST yang menggunakan konstruksi unik bernama "spons" (*sponge construction*).

III. Implementasi

Secara umum, implementasi yang dibahas pada makalah ini akan terdiri dari dua *microservices* yang dilengkapi dengan antar muka *web* untuk memudahkan *testing*.

A. Arsitektur Aplikasi

Berikut adalah diagram *deployment* dari aplikasi:



Gambar III. 1. Diagram *Deployment* Aplikasi. (Sumber: Penulis)

Terdapat 3 komponen utama dari arsitektur aplikasi, ketiganya saling terhubung dan saling melakukan pertukaran data antara satu sama lain:

1. *Frontend Services* (React Router)

   Frontend Services merupakan lapisan antarmuka pengguna (*user interface*) yang berfungsi sebagai titik interaksi antara pengguna dan sistem. Pada arsitektur ini, *frontend* dibangun menggunakan teknologi React dengan dukungan React Router untuk mengelola navigasi dan perpindahan antar halaman.

   Frontend bertugas menampilkan data, menerima input pengguna, serta mengirimkan permintaan (*request*) ke layanan backend melalui antarmuka API.

2. *Backend Services* (Express.js)

   Backend Services berperan sebagai inti logika aplikasi yang menangani proses bisnis berupa pemrosesan tanda tangan digital serta komunikasi antara frontend dan basis data. Pada arsitektur ini, backend diimplementasikan menggunakan Express.js, sebuah *framework* Node.js yang ringan dan fleksibel untuk membangun layanan berbasis REST API.

   Backend akan menerima masukan dari *frontend* berupa:

   1. Dokumen PDF untuk penandatanganan.
   2. Dokumen PDF untuk keperluan verifikasi.
   3. *Private Key* yang dimiliki pengguna.
   4. *Public Key* untuk repositori kunci public.

3. *Database* (MongoDB)

   MongoDB digunakan sebagai sistem manajemen basis data yang berfungsi untuk menyimpan dan mengelola data aplikasi secara persisten. Hanya pasangan nama_pengguna dan *public_key* yang akan disimpan pada basis data ini.

Sebagai tambahan juga, digunakan *docker* sebagai alat *containerization* agar menjamin ekosistem aplikasi yang konsisten dan dapat berjalan pada berbagai macam mesin.

B. Fungsionalitas *Frontend*

Berhubung *frontend* bertugas sebagai antar muka interaksi dengan pengguna, berikut adalah fungsi-fungsi yang dipenuhi oleh *frontend*:

1. Menerima *file* PDF dari pengguna.
2. Menerima input *username* dan generate *public* dan *private key pair*.
3. Mengirim data ke *backend* untuk perekapan dan pemrosesan.

C. Fungsionalitas *Backend*

Backend akan bertugas dalam menyimpan pasangan *public key* dan *username* yang telah diberikan oleh pengguna:

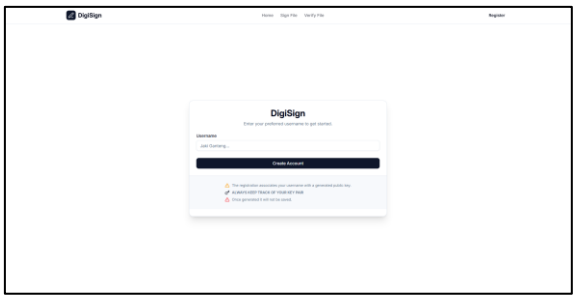
1. Menerima data *public key* pengguna.
2. Menyimpannya pada *registry* basis data.

IV. Hasil dan Pengujian

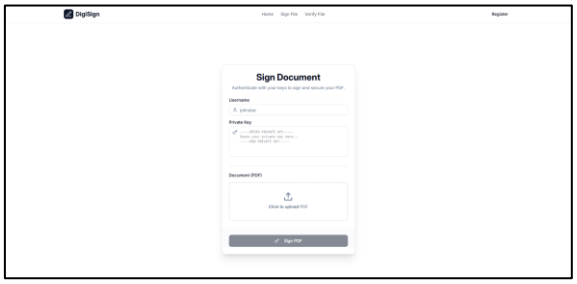
Telah diimplementasi aplikasi *signing* bernama DigiSign menggunakan *react router, express js,* dan MongoDB. Berikut adalah tahapan kerjanya:

1. Pengguna melakukan registrasi *username* dan mendapat *key pair*, pengguna bertanggung jawab untuk menyimpan *private key* yang dihasilkan oleh aplikasi.
2. Pengguna dapat melakukan *sign* dengan cara memasukkan *username* dan *private key* pada kolom yang telah disediakan pada halaman *sign file*.
3. Pengguna dapat mengunduh hasil *file* yang telah di-*sign*.
4. Pengguna dapat mengunggah *file* yang teldah di-*sign* pada halaman *verify file* dan memverifikasi apakah sebuah *file* telah di-*sign* oleh seseorang tertentu.
5. Verifikasi akan valid selama *file* pdf tidak mengalami perubahan.

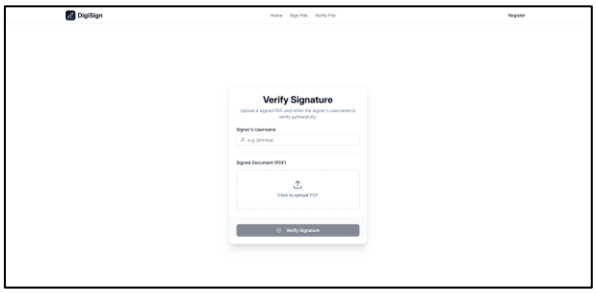
A. Hasil Implementasi Antar Muka



Gambar IV. 1. Laman registrasi pengguna. (Sumber: Penulis)



Gambar IV. 2. Laman penandatanganan dokumen PDF. (Sumber: Penulis)



Gambar IV. 3. Laman verifikasi dokumen. (Sumber: Penulis)

B. Pengujian Kasus Uji

Terdapat beberaa kasus uji yang akan digunakan. Utamanya, variable yang akan selalu berubah-ubah adalah *file* pdf yang akan ditandangani dan diverifikasi.

Akan digunakan *file* pdf makalah matematika diskrit penulis berjudul: *Enhancing Security in a Password Manager Using RSA Encryption.* Berikut adalah spesifikasi *file* lebih lanjut.

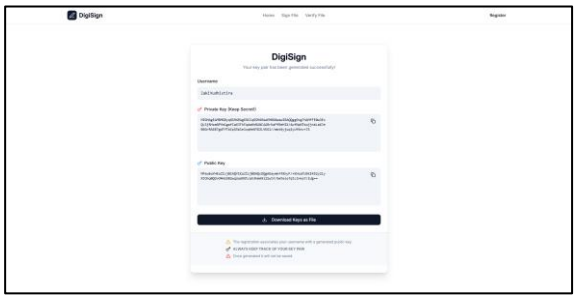
Nama *file*: 13522031.pdf

Extension: .pdf

Ukuran *file*: 402KB

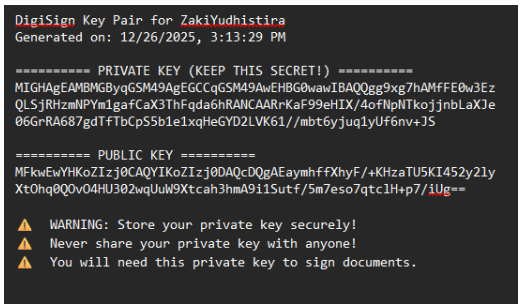
Berikut adalah langkah-langkah pengujian yang dilakukan terhadap *file* 13522031.pdf:

1. Dilakukan registrasi pengguna terlebih dahulu dengan *username* ZakiYudhistira.



Gambar IV. 4. Hasil dari registrasi (Sumber: Penulis)

Dari gambar di atas, terlihat bahwa telah dihasilkan sebuah *private key* dan *public key* untuk pengguna ZakiYudhistira.



Gambar IV. 5. Hasil unduhan *key pair* (Sumber: Penulis)

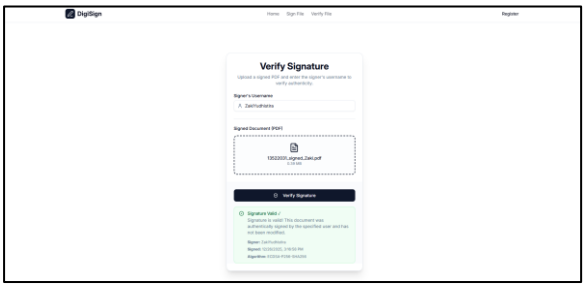
Pengguna dapat mengunduh dalam format *file txt* untuk pasangan kunci yang dihasilkan.

2. Dilakukan *signing* terhadap *file* pdf dengan cara mengunggahnya pada aplikasi.

Gambar IV. 6. Hasil penandatanganan *file* pdf (Sumber: Penulis)

Dari gambar di atas, telah berhasil dilakukan penandatanganan untuk *file* 13522031.pdf, sesuai dengan *file* yang dijelaskan pada kasus uji.

3. Dilakukan verifikasi tanda tangan dengan memasukkan *file* dan nama pengguna yang menandatangani.



Gambar IV. 7. Hasil verifikasi *file* pdf (Sumber: Penulis)

Dari gambar di atas, telah berhasil dilakukan verifikasi untuk *file* 13522031.pdf, sesuai dengan *file* yang dijelaskan pada kasus uji. Ini pertanda bahwa belum dilakukan modifikasi apapun terhadap *file* yang telah ditandatangani.

4. Dilakukan *merging file* pdf dengan *file* lain untuk merubah isi PDF.

   Digunakan laman bernama *IlovePDF* untuk melakukan penggabungan *file* pdf.

5. Dilakukan verifikasi ulang.

Gambar IV. 7. Hasil verifikasi *file* pdf (Sumber: Penulis)

Dari gambar di atas, dapat dilihat bahwa verifikasi gagal dilakukan karena terjadi perubahan pada *file*.

C. Catatan Lain.

Telah dilakukan kasus uji lainnya dengan cara mengirimkan *file* yang telah ditandatangani melalui saluran komunikasi yang lazim digunakan seperti Whatsapp Messenger, hasil menunjukkan bahwa *file* tetap dapat diverifikasi dan menunjukkan hasil yang valid.

V. KESIMPULAN

Penelitian ini berhasil mengimplementasikan mekanisme penandatanganan dan verifikasi dokumen secara *client-side*, proses pembuatan tanda tangan digital dilakukan langsung pada sisi pengguna. Pendekatan ini memastikan bahwa kunci privat sepenuhnya berada di bawah kendali pengguna dan tidak pernah ditransmisikan melalui jaringan ataupun disimpan di sisi server sehingga risiko kebocoran kunci dapat diminimalkan. Pada sisi server, sistem hanya menyimpan kunci publik pengguna yang digunakan dalam proses verifikasi tanda tangan digital. Sistem ini diimplementasikan menggunakan *React* dan *React Router* pada sisi *front-end*, serta *Express.js* dan *MongoDB* pada sisi *back-end* untuk mendukung pengelolaan layanan dan penyimpanan data.

Sebagai pengembangan lanjutan, aplikasi ini masih dapat ditingkatkan menjadi sistem yang lebih komprehensif dengan mengintegrasikan *Certification Authority* (CA) serta infrastruktur *Public Key Infrastructure* (PKI) yang lebih memadai.

LAMPIRAN

1. Tautan repositori aplikasi

https://github.com/ZakiYudhistira/Kripto-digisign

UCAPAN TERIMAKASIH

Terimakasih banyak disampaikan kepada Bapak Rinaldi Munir selaku pengampu mata kuliah kriptografi IF untuk tahun ini, banyak ilmu dan ide-ide dari beliau yang telah menjadi inspirasi selama penyusunan makalah ini.

Tak lupa, terima kasih juga penulis ucapkan kepada teman penulis Edbert Eddyson Gunawan karena telah memberikan banyak makanan kepada penulis ketika lapar.

REFERENSI

[1] Kaupa, S. & Chisa, K. (2020). Adoption of the Electronic Document Records Management System within the Public Sector in Namibia: Exploring the Challenges and Opportunities. *International Journal of Operations Management, 1(1),* 7-18. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68-73.

[2] Newman, S. (2019). *Monolith to Microservices: Evolutionary patterns to transform your Monolith*.

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

[4] Munir, R. (2025). Public Key Infrastructure. Materi kuliah kriptografi. Institut Teknologi Bandung.

[5] Munir, R. (2025). Tanda tangan digital. Materi kuliah kriptografi. Institut Teknologi Bandung.

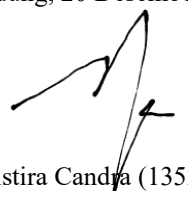
[6] Munir, R. (2025). ElGamal signature. Materi kuliah kriptografi. Institut Teknologi Bandung.

[7] Munir, R. (2025). Digital Signature Algorithm (DSA) dan Elliptic Curve Digital Signature Algorithm (ECDSA). Materi kuliah kriptografi. Institut Teknologi Bandung.

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, 20 Desember 2025



Zaki Yudhistira Candra (13522031)