Implementasi Algoritma Elliptic Curve Digital Signature dalam Peningkatan Kemaanan JWT

Austin Gabriel Pardosi - 13521084
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): gabrielpardosi26@gmail.com

***Abstract*—Makalah ini memberikan pendekatan untuk meningkatkan keamanan JSON Web Token (JWT) dengan mengimplementasikan Algoritma Elliptic Curve Digital Signature (ECDSA). JWT merupakan metode populer untuk otentikasi dan otorisasi dalam aplikasi web, namun memiliki beberapa kelemahan keamanan, seperti manajemen kunci yang rentan dan potensi serangan algoritma. ECDSA, dengan memanfaatkan kriptografi kurva elipik, menawarkan tanda tangan digital yang lebih aman dan efisien. Makalah ini mengeksplorasi integrasi ECDSA dalam JWT, mengatasi kelemahan yang ada, dan menunjukkan bahwa penggunaan ECDSA dapat meningkatkan perlindungan data dan integritas sistem tanpa mengorbankan kinerja.**

Keywords—JSON Web Token (JWT); Elliptic Curve Digital Signature Algorithm (ECDSA); otentikasi; otorisasi; kriptografi; manajemen kunci.

I. INTRODUCTION

Dalam era digital yang semakin maju, kemaanan data menjadi salah satu aspek terpenting dalam pengembangan aplikasi dan layanan berbasis *cloud*. Penggunaan *JSON Web Token* (JWT) telah menjadi metode populer untuk otentikasi dan otorisasi, memungkinkan transmisi informasi yang aman dan efisien antara dua pihak. JWT menyediakan format token yang ringan dan mudah digunakan, terdiri dari tiga komponen utama: *header*, *payload*, dan *signature.* Meskipun JWT menawarkan banyak keuntungan, beberapa kelemahan yang signifikan tetap ada, seperti manajemen kunci yang rentan dan potensi serangan algoritma.

Salah satu tantangan utama dalam penggunaan JWT adalah bagaimana memastikan integritas dan autentisitas token tanpa mengorbankan kinerja sistem. Manajemen kunci yang buruk dapat menyebabkan kebocoran informasi sensitive dan memungkinkan penyerang untuk memanipulasi atau membuat token palsu. Selain itu, beberapa implementasi JWT masih rentan terhadap serangan algoritma confusion, di mana penyerang dapat memanfaatkan kelemahan dalam pemilihan algoritma tanda tangan untuk menipu sistem otentikasi.

Untuk mengatasi masalah ini, Algoritma *Elliptic Curve Digital Signature* (ECDSA) menawarkan solusi yang lebih aman dan efisien. ECDSA menggunakan prinsip-prinsip kriptografi kurva eliptik untuk menghasilkan tanda tangan digital yang kuat dengan ukuran kunci yang lebih kecil dibandingkan algoritma tradisional seperti HMAC dan RSA. Keunggulan ini tidak hanya meningkatkan tingkat keamanan, tetapi juga mengurangi beban komputasi, sehingga meningkatkan kinerja keseluruhan sistem.

Makalah ini bertujuan untuk mengeksplorasi bagaimana ECDSA dapat diterapkan dalam kontek JWT untuk meningkatkan keamanan token. Kami akan membahas integrasi ECDSA dalam proses pembuatan dan verifikasi tanda tangan digitasl JWT, serta mengkaji keuntungan dan tantangan yang terkait dengan penggunaannya. Selain itu, makalah ini akan menyajikan studi kasus implementasi ECDSA dalam aplikasi nyata untuk menunjukkan bagaimana metode ini dapat diterapkan secara praktis.

Dalam proses penulisan makalah ini, akan dikaji dasar-dasar kriptografi kurva eliptik, prinsip kerja ECDSA, dan bagaimana algoritma ini dapat mengatasi kelemahan-kelemahan yang ada dalam JWT. Akan dibahas juga *best-practice* dalam manajemen kunci dan validasi tanda tangan digital, yang merupakan aspek penting dalam memastikan kemaanan sistem otentikasi dan otorisasi. Makalah ini diharapkan dapat memberikan panduan dalam mengimplementasikan ECDSA untuk meningkatkan keamanan JWT.

Dengan meningkatkan kebutuhan akan keamanan data dalam aplikasi, teknologi yang lebih aman dan efisien seperti ECDSA menjadi semakin penting. Makalah ini tidak hanya memberikan kontribusi bagi pengembang perangkat lunak, tetapi juga memberikan wawasan tambahan bagi riset dalam keamanan siber. Diharapkan bahwa hasil dari makalah ini dapat menjadi referensi dalam pengembangan sistem otentikasi dan otorisasi yang lebih aman di masa depan.

II. LANDASAN TEORI

A. JSON Web Token (JWT)

JSON Web Token (JWT) adalah standar terbuka (RFC 7519) yang mendefinisikan cara aman untuk mentransmisikan informasi antara dua pihak sebagai objek JSON. JWT digunakan secara luas dalam berbagai aplikasi dan layanan berbasis *cloud* untuk tujuan otentikasi dan otorisasi. JWT terdiri dari tiga bagian utama: *header, payload,* dan *signature.* Setiap bagian memiliki peran penting dalam menjamin integritas dan keamanan token.

Header biasanya mencakup tipe token, yaitu JWT, dan algoritma *hashing* yang digunakan, seperti HMAC SHA256 atau RSA. *Payload* berisi klaim, yaitu pernyataan tentang entitas (umumnya pengguna) dan data tambahan. Klai mini dapat berupa *registered claims, public claims,* atau *private claims. Signature* dibuat dengan menggabungkan *header* dan *payload*, lalu mengenkripsinya menggunakan algoritma yang ditentukan di header bersama dengan kunci rahasia atau kunci privat. Struktur JWT yang kompak memungkinkan transmisi data yang efisien dan aman antara dua pihak.

Dalam proses otentikasi, server menghasilkan JWT setelah memverifikasi kredensial pengguna. Token ini kemudian diberikan kepada *client*, yang dapat menyimpannya dalam penyimpanan local atau *cookie*. Untuk setiap permintaan ke server, klien mengirimkan token ini sebagai bagian dari *header* permintaan. Server kemudian memverifikasi token untuk memastikan bahwa permintaan tersebut berasal dari sumber yang sah dan berwenang.

Proses otorisasi menggunakan JWT juga serupa. Setelah token diverifikasi, server dapat mengevaluasi klaim di dalam token untuk menentukan Tingkat akses pengguna terhadap data yang dilindungi. Dengan menggunakan JWT, server dapat mengurangi beban pada penyimpanan sesi dan mempermudah skalabilitas sistem otentikasi dan otorisasi, terutama dalam lingkungan terdistribusi.

JWT menawarkan beberapa manfaat penting dalam konteks otentikasi dan otorisasi. Pertama, JWT memiliki struktur yang ringan dan mudah dipahami, memungkinkan integrasi yang cepat dan efisien dengan berbagai sistem dan layanan. Kedua, JWT tidak memerlukan penyimpanan sesi di server, karena semua informasi yang diperlukan untuk verifikasi dan otorisasi terkandung dalam token itu sendiri. Ini mengurangi beban penyimpanan dan meningkatkan skalabilitas.

Meskipun menawarkan banyak manfaat, JWT juga memiliki beberapa kelemahan. Salah satu kelemahan utama adalah manajemen kunci. Kunci rahasia yang digunakan untuk menandatangani token harus disimpan dengan sangat aman. Jika kunci ini bocor, seluruh sistem otentikasi dan otorisasi dapat terkompromi.

Kelemahan lainnya adalah JWT yang tidak memiliki masa kadaluwarsa yang jelas dapat disalahgunakan. Token yang lama dan tidak kadaluwarsa dapat digunakan oleh penyerang jika token tersebut bocor atau dicuri. Oleh karena itu, sangat penting untuk mengatur masa kadaluwarsa yang tepat dan menggunakan mekanisme blacklist untuk mencabut token yang tidak lagi berlaku.

JWT rentan terhadap berbagai jenis serangan. Salah satu serangan yang paling umum adalah serangan *man-in-the-middle*, di mana penyerang dapat mencegah dan memodifikasi token saat dalam transmisi. Untuk mengatasi ini, penggunaan HTTPS sangat disarankan untuk mengamankan komunikasi antara *client* dan *server*.

Serangan *brute force* adalah ancaman lain, di mana penyerang mencoba berbagai kombinasi kunci untuk menandatangani ulang token dan memvalidasi dirinya sebagai pengguna sah. Penggunaan algoritma *hashing* yang kuat dan kunci rahasia yang kompleks dapat membantu mencegah serangan semacam ini.

Serangan *algorithm confusion* adalah jenis serangan di mana penyerang dapat memanfaatkan kelemahan dalam implementasi JWT untuk mengubah algoritma tanda tangan yang digunakan. Misalnya, penyerang dapat mengubah algoritma di header JWT dari HMAC SHA256 menjadi *none*, yang mengindikasikan bahwa tidak ada tanda tangan yang diperlukan. Jika server tidak memeriksa dengan benar, token ini bisa diterima sebagai valid. Mengonfigurasi *server* untuk hanya menerima algoritma yang diizinkan dan melakukan validasi yang ketat adalah langkah penting untuk mencegah serangan ini.

B. Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography (ECC) adalah teknik enkripsi yang menggunakan struktur matematikam kurva eliptik untuk menghasilkan kunci kriptografi. ECC menjadi sangat populer karena kemampuannya untuk memberikan tingkat keamanan yang tinggi dengan ukuran kunci yang relative kecil dibandingkan dengan algoritma kriptografi tradisional seperti RSA. Keamanan ECC didasarkan pada kesulitan memecahkan masalah logaritma diskrit pada titik-titik kurva eliptik, yang secara matematika jauh lebih rumit dibandingkan dengan masalah faktorisasi bilangan besar yang digunakan dalam RSA.

ECC menggunakan titik-titik pada kurva eliptik yang didefinisikan oleh persamaan matematis tertentu. Salah satu persamaan kurva eliptik yang umum digunakan adalah $y^2 = x^3 + ax + b$, dimana a dan b adalah konstanta yang menentukan bentuk kurva. Titik-titik pada kurva ini membentuk grup Abelian, yang berarti bahwa mereka memenuhi sifat-sifat tertentu seperti komutatif, dan memungkinkan operasi aritmatika seperti penjumlahan titik dan penggandaan titik. Dalam konteks kriptografi, operasi ini digunakan untuk menghasilkan pasangan kunci privat dan public yang kemudian digunakan untuk enkripsi dan dekripsi data.

Salah satu keunggulan utama ECC adalah efisiensi ukuran kunci. Sebagai contoh, kunci ECC dengan Panjang 256 bit menawarkan tingkat keamanan yang setara dengan kunci RSA sepanjang 3072 bit. Ini berarti bahwa ECC dapat memberikan keamanan yang sama dengan ukuran kunci yang jauh lebih kecil, yang mengurangi beban komputasi dan kebutuhan penyimpanan. Hal ini sangat penting untuk perangkat dengan sumber daya terbatas seperti *smartphone* dan perangkat *IoT*, di mana efisiensi energi dan kecepatan pemrosesan adalah faktor kritis.

Keamanan ECC berasal dari kesulitan masalah logaritma diskrit pada kurva eliptik. Untuk memahami ini, pertimbangkan bahwa menghitung hasil dari penggandaan titik pada kurva eliptik relative mudah, tetapi menemukan jumlah penggandaan yang menghasilkan titik tertentu dari titik awal sangat sulit tanpa mengetahui kunci privat. Masalah ini dikenal sebagai *Elliptic Curve Discrete Logarithm Problem* (ECDLP) dan merupakan dasar dari keamanan ECC.

C. *Elliptic Curve Digital Signature Algorithm (ECDSA)*

Elliptic Curve Digital Signature Algorithm (ECDSA) merupakan salah satu algoritma *Digital Signature Algorithm* (DSA) yang menggunakan prinsip-prinsip kriptografi kurva eliptik untuk menghasilkan tanda tangan digital. ECDSA menawarkan keamanan yang kuat dengan ukuran kunci yang lebih kecil dibandingkan algoritma tradisional seperti RSA, membuatnya sangat cocok untuk aplikasi yang membutuhkan enkripsi kuat tetapi memiliki keterbatasan dalam hal daya komputasi dan ruang penyimpanan. ECDSA telah menjadi standar dalam berbagai protokol keamanan, termasuk TLS (*Transport Layer Security*) dan aplikasi *blockchain*.

Proses pembuatan tanda tangan digital dengan ECDSA melibatkan beberapa langkah utama. Pertama, sebuah *hash* dari pesan yang akan ditandatangani dihitung menggunakan algoritma *hash* seperti SHA-256. Hash ini kemudian dikombinasikan dengan kunci privat pengguna dan parameter kurva eliptik untuk menghasilkan dua nilai, r dan s, yang bersama-sama membentuk tanda tangan digital. Nilai-nilai ini memastikan bahwa hanya pemegang kunci privat yang dapat menghasilkan tanda tangan yang valid untuk pesan tertentu, menjaga integritas dan autentisitas pesan.

Verifikasi tanda tangan digital dengan ECDSA melibatkan penggunaan kunci publik pengguna. Ketika penerima menerima pesan dan tanda tangan digital, mereka pertama-tama menghitung *hash* dari pesan yang diterima. Kemudian, menggunakan kunci publik pengirim dan nilai-nilai r dan s dari tanda tangan, penerima dapat memverifikasi apakah tanda tangan tersebut valid. Proses verifikasi ini memastikan bahwa pesan tidak diubah dan benar-benar berasal dari pengirim yang sah. Jika verifikasi gagal, pesan dianggap tidak sah dan dapat ditolak.

ECDSA memberikan beberapa keunggulan penting dalam keamanan data. Salah satu keunggulan utamanya adalah efisiensi ukuran kunci. Kunci ECDSA dengan panjang 256 bit menawarkan tingkat keamanan yang setara dengan kunci RSA sepanjang 3072 bit, mengurangi kebutuhan penyimpanan dan mempercepat proses kriptografi. Selain itu, ECDSA sangat tahan terhadap serangan kriptanalitik, berkat kesulitan masalah logarita diskrit pada kurva eliptik. Keamanan ini memastikan bahwa ECDSA dapat diandalkan dalam berbagai aplikasi kriptografi, termasuk tanda tangan digital dan enkripsi data.

ECDSA telah diterapkan secara luas dalam berbagai protokol dan aplikasi keamanan modern. Dalam protokol TLS/SSL, ECDSA digunakan untuk memastikan bahwa komunikasi antara *client* dan *server* tetap aman dan autentik. Dalam teknologi *blockchain*, ECDSA digunakan untuk menandatangani transaksi, memastikan bahwa hanya pemilik sah dari kunci privat yang dapat mengotorisasi transfer aset digital.

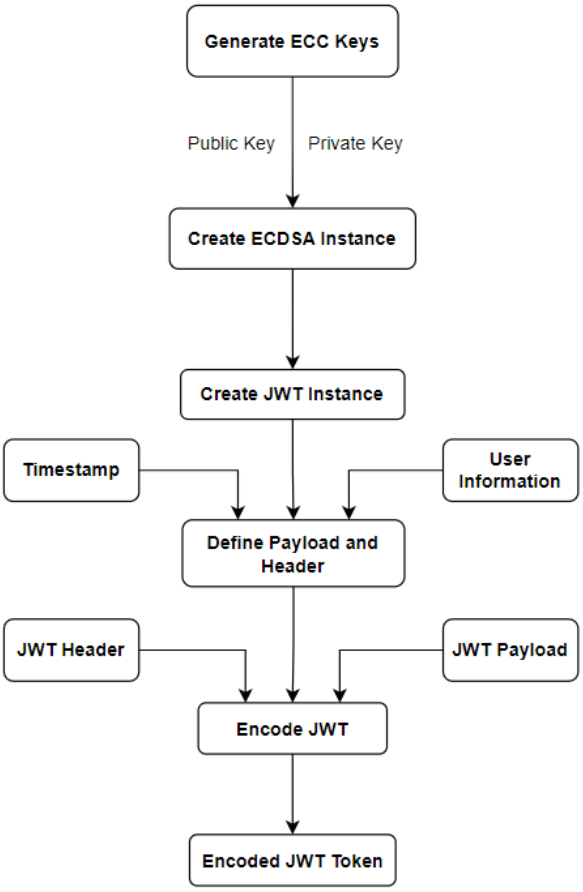
III. Rencana Penyelesaian Masalah

Rencana penyelesaian masalah yang akan diterapkan digunakan untuk meningkatkan keamanan sistem dengan menggunakan *Elliptic Curve Cryptography (ECC), Elliptic Curve Digital SIgnature Algorithm (ECDSA),* dan *JSON Web Token (JWT).* Tujuan utama dari rencana ini adlaah untuk memastikan bahwa sistem yang dibangun mampu memberikan tingkat keamanan yang tinggi dengan performa yang efisien, tanpa mengorbankan integritas data.

Dalam konteks ini, ECC akan digunakan untuk menghasilkan pasangan kunci privat dan kunci publik yang digunakan dalam proses tanda tangan digital oleh ECDSA. ECC dipilih karena kemampuannya menghasilkan kunci yang lebih kecil namun tetap aman dibandingkan dengan metode kriptografi tradisional seperti RSA. ECDSA kemudian akan menggunakan kunci privat untuk menandatangani *payload* JWT dan kunci publik untuk memverifikasi tanda tangan tersebut. JWT akan digunakan untuk menyimpan dan mengiriman informasi yang ditandatangani secara aman.

A. *Rancangan Solusi dan Arsitektur*

Berikut ini merupakan rancangan solusi yang diterapkan pada makalah ini. Penjelasan lebih mendetail mengenai pemrosesan akan dijelaskan pada bagian implementasi.



Gambar 3.1. Visualisasi rancangan solusi *encoding JWT*.

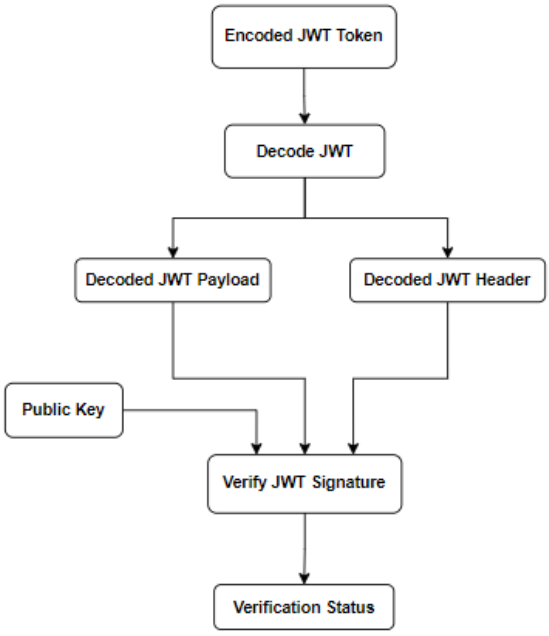
Langkah pertama dalam proses *encoding* adalah menghasilkan pasangan kunci menggunakan ECC. Proses ini melibatkan penggunaan algoritma ECC untuk menghasilkan kunci privat dan kunci publik. Kunci privat digunakan untuk menandatangani token, sementara kunci publik digunakan untuk verifikasi tanda tangan. Proses ini penting karena ECC menghasilkan kunci yang lebih kecil namun tetap aman, meningkatkan efisiensi sistem.

Setelah pasangan kunci dihasilkan, langkah berikutnya adalah membuat *instance ECDSA* menggunakan kunci privat dan publik tersebut. ECDSA adalah algoritma yang digunakan untuk menghasilkan tanda tangan digital, yang memastikan bahwa data yang ditandatangani dapat diverifikasi keasliannya oleh penerima menggunakan kunci publik. Dengan membuat *instance ECDSA*, kita dapat menggunakan kunci privat untuk menandatangani *payload* JWT.

Selanjutnya, instance JWT dibuat menggunakan *instance ECDSA* yang telah disiapkan. JWT adalah standar yang digunakan untuk membuat token akses yang aman dan ringkas, yang dapat digunakan untuk otentikasi dan pertukaran informasi yang aman. *Instance JWT* ini bertanggung jawab untuk meng-*encode* dan menandatangani *payload* menggunakaan ECDSA, memastikan bahwa token yang dihasilkan aman dan tidak dapat diubah tanpa terdeteksi.

Pada tahap selanjutnya, *payload* dan *header JWT* didefinsikan. *Payload* berisi klaim atau data yang ingin disertakan dalam token, seperti informasi pengguna dan *timestamp*. *Header* berisi informasi tentang algoritma yang digunakan untuk menandatangani token. Kedua komponen ini digabungkan untuk membentuk token yang lengkap. Dengan mendefinisikan *payload* dan *header* dengan benar, kita dapat memastikan bahwa informasi yang dikandung dalam token sesuai dengan kebutuhan aplikasi.

Langkah terakhir dalam proses *encoding* adalah meng-*encode JWT* menggunakan *payload*, *header*, dan *instance JWT* yang telah dibuat. Proses ini melibatkan penggabungan *header* dan *payload*, serta penandatanganan token menggunakan kunci privat melalui *instance ECDSA*. Hasilnya adalah token JWT yang telah di-*encode* dan siap digunakan dalam aplikasi. Token ini berisi semua informasi yang diperlukan dan ditandatangani secara digital, memastikan keamanan dan integritas data.



Gambar 3.2. Visualisasi rancangan solusi *decoding JWT*.

Langkah pertama dalam proses *decoding* adalah menerima token JWT yang telah di-*encode*. Token ini dikirimkan dari *client* ke *server* sebagai bagian dari mekanisme otentikasi atau otorisasi. Token ini mengandung semua informasi yang diperlukan untuk verifikasi identitas dan integritas data.

Setelah token JWT yang telah di-*encode* diterima, langkah selanjutnya adlaha men-*decode* token tersebut. Proses *decoding* ini melibatkan pemisahan token menjadi tiga bagian utama: *header*, *payload*, dan *signature*. Dalam konteks JWT, token biasanya terdiri dari tiga bagian yang dipisahkan oleh titik (.), yaitu *header*, *payload*, dan *signature*.

- **Decoded JWT Payload:** Bagian payload berisi klaim atau data yang dimasukkan saat token dibuat, seperti informasi pengguna, waktu pembuatan, dan waktu kadaluwarsa.
- **Decoded JWT Header:** Bagian header berisi metadata tentang token, seperti algoritma yang digunakan untuk menandatangani token dan tipe token.

Setelah *payload* dan *header* berhasil di-*decode*, langkah berikutnya adalh memverifikasi tanda tangan (*signature*) JWT. Proses ini memastikan bahwa token tidak diubah sejak ditandatangani dan diterbitkan oleh entitas yang terpercaya. Verifikasi tanda tangan dilakukan menggunakan kunci publik yang sesuai.

Langkah terakhir adalah menghasilkan status verifikasi. Status ini menunjukkan apakah verifikasi berhasil atau gagal. Jika verifikasi berhasil, *payload* JWT dianggap *valid* dan dapat digunakan untuk mengidentifikasi pengguna atau memverifikasi hak akses. Jika verifikasi gagal, token dianggap tidak valid dan ditolak oleh sistem.

B. Implementasi Algoritma Elliptic Curve Cryptography

Seperti yang dijelaskan sebelumnya, implementasi *Elliptic Curve Cryptography* dalam dalam bahasa Python menggunakan pustaka *cryptography* adalah untuk menghasilkan dan mengelola pasangan kunci publik dan privat yang aman. ECC dipilih karena efisiensinya dalam menghasilkan kunci yang lebih kecil namun tetap memberikan tingkat keamanan yang tinggi. Implementasi ini melibatkan beberapa fungsi utama untuk menghasilkan, menyerialisasi, dan mendeserialisasi kunci.

Proses pembuatan kunci dimulai dengan fungsi *generate_keys*. Fungsi ini menghasilkan sepasang kunci privat dan publik menggunakan kurva eliptik SECP256R1. Langkah pertama dalam fungsi ini adalah memanggil *ec.generate_private_key* untuk menghasilkan kunci privat. Dari kunci privat tersebut, kunci publik dapat diperoleh dengan memanggil metode *public_key*. Fungsi ini mengembalikan pasangan kunci privat dan publik yang dapat digunakan untuk proses kriptografi lebih lanjut.

Setelah kunci privat dihasilkan, langkah berikutnya adalah menyerialisasi kunci tersebut menjadi format PEM yang dapat disimpan atau ditransmisikan. Fungsi *serialize_private_key* menggunakan metode *private_bytes* dari objek kunci privat, dengan parameter *encoding serialization.Encoding.PEM*,

format *serialization.PrivateFormat.PKCS8*, dan tanpa enkripsi *serialization.NoEncryption*. Hasil dari proses ini adalah string PEM yang dapat disimpan dalam bentuk teks.

Mirip dengan kunci privat, kunci publik juga perlu diserialisasi menjadi format PEM untuk penyimpanan atau transmisi. Fungsi *serialize_public_key* melakukan ini dengan menggunakan metode *public_bytes* dari objek kunci publik. Format yang digunakan adalah *serialization.PublicFormat.SubjectPublicKeyInfo* untuk memastikan kompatibilitas dengan berbagai sistem yang menggunakan kunci publik.

Fungsi *deserialize_private_key* digunakan untuk mengubah string PEM kembali menjadi objek kunci privat. Ini dilakukan dengan memanggil *serialization.load_pem_private_key* yang mengkonversi string PEM menjadi objek *EllipticCurvePrivateKey*. Fungsi ini menerima string PEM dan parameter password yang diatur ke *None* karena kunci privat tidak dienkripsi.

Untuk mengembalikan string PEM menjadi objek kunci publik, digunakan fungsi *deserialize_public_key*. Fungsi ini memanggil *serialization.load_pem_public_key* yang mengkonversi string PEM menjadi objek *EllipticCurvePublicKey*. Ini memungkinkan kunci publik yang telah diserialisasi dapat dengan mudah digunakan kembali dalam aplikasi.

C. Implementasi Elliptic Curve Digital Signature Algorithm

Algoritma ECDSA adalah algoritma yang digunakan untuk menghasilkan tanda tangan digital, memastikan integritas dan keaslian pesan. Implementasi ECDSA dalam kode ini menggunakan pustaka *cryptography* dari Python untuk menangani proses tanda tangan dan verifikasi tanda tangan dengan menggunakan kunci privat dan publik dari *Elliptic Curve Cryptography*.

Kelas ECDSA diinisialisasi dengan menerima kunci privat dan publik sebagai parameter. Kedua kunci ini digunakan untuk proses penandatanganan dan verifikasi tanda tangan. Kunci privat digunakan untuk menandatangani data, sementara kunci publik digunakan untuk memverifikasi tanda tangan tersebut.

Fungsi *sign* digunakan untuk menandatangani data menggunakan kunci privat. Data yang akan ditandatangani diterima sebagai parameter dalam bentuk *bytes*. Proses penandatanganan menggunakan algoritma ECDSA dengan fungsi *hash* SHA-256. Fungsi sign memanggil metode *sign* dari objek kunci privat, yang menghasilkan tanda tangan digital dari data yang diberikan. Tanda tangan ini kemudian dikembalikan dalam bentuk *bytes*.

Fungsi *verify* digunakan untuk memverifikasi tanda tangan digital yang dihasilkan oleh fungsi *sign*. Fungsi ini menerima data asli dan tanda tangan sebagai parameter dalam bentuk *bytes*. Proses verifikasi menggunakan kunci publik dan algoritma ECDSA dengan fungsi *hash* SHA-256. Fungsi *verify* memanggil metode *verify* dari objek kunci publik, yang membandingkan tanda tangan dengan data asli. Jika verifikasi berhasil, fungsi ini mengembalikan *True*; jika tidak, mengembalikan *False*.

IV. HASIL DAN PEMBAHASAN

Pada bagian ini, dilakukan pengujian terhadap implementasi beserta analisis terhadap hasil pengujian yang didapatkan.

Pengujian dengan 3 Kasus Uji

1. Kasus 1: Payload Kecil

header	{
	'alg': 'ES256',
	'typ': 'JWT'
	}

payload	{
	'sub': '13521084',
	'name': 'Austin Gabriel Pardosi'
	}

Hasil :

Private Key	MIGHAgEAMBMGByqGSM49AgEGCCqGSM49AwEHBG0wawIBAQQgsbl363BDCuQXgfzOL9dZyQYhLeeTefhbUUyY8h3Mm3ChRANCAATbMQbpFipR9kqALu0UUhesgVDqbWnxs25NY55GzTS+XIREMnzLTHZ0O9Y2bE82nlNnv5IWwEEVYYZlB+TyOXb5
Public Key	MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE2zEG6RYqUfZKgC7tFFIXrIFQ6m1p8bNuTWOeRs00vlyERDJ8y0x2dDvWNmxPNp5TZ7+SFsBBFWGGZQfk8jl2+Q==
JWT Token	eyJhbGciOiAiRVMyNTYiLCAidHlwIjogIkpXVCJ9.eyJzdWIiOiAiMTM1MjEwODQiLCAibmFtZSI6ICJBdXN0aW4gR2FicmllbCBQYXJkb3NpIn0.MEUCIQDbWE3RVhwt0Z-QYWQ4c3GhooZJWx255kXzrkmQmU6scwIgfj-TCbJjEPcADRTZ_jQdbVu2wCoMo8O9pkGaJTN9Cnw

2. Kasus 2: Payload Sedang

header	{
	'alg': 'ES256',
	'typ': 'JWT'
	}

payload	{
	'sub': '13521084',
	'name': 'Austin Gabriel Pardosi',
	'email': 'gabrielpardosi26@gmail.com',
	'iat': datetime.utcnow().isoformat(),
	'exp': (datetime.utcnow() + timedelta(minutes=30)).isoformat()
	}

Hasil :

Private	MIGHAgEAMBMGByqGSM49AgEGCCq

Key	GSM49AwEHBG0wawIBAQQgbVnV6lgR YnBJe18rg46vC0+QB1B8nawgaN40kGw2 m7ahRANCAAQVXvx5wRLqqpRyNbMC LXg7AGwo9/MAFhOttDDW8J01oiJNrBW rogacILYD2R7KU8uWZXyGll2eA 0mUM0TIp6dp
Public Key	MFkwEwYHKoZIzj0CAQYIKoZIz j0DAQcDQgAEFV78ecES6qqUcjW zAi14OwBsKPfzABYTrbQw1vCdN aIiTawVq6IGnCC2A9keylPLlmV8h pZdngNJlDNEyKenaQ==
JWT Token	eyJhbGciOiAiRVMyNTYiLCAidHl wIjogIkpXVCJ9.eyJzdWIiOiAiMTM 1MjEwODQiLCAibmFtZSI6ICJBdX N0aW4gR2FicmllbCBQYXJkb3NpIi wgImVtYWlsIjogImdhYnJpZWxwY XJkb3NpMjZAZ21haWwuY29tIiwgI mlhdCI6ICIyMDI0LTA2LTEyVDEz OjQxOjIyLjgyMjc0OSIsICJleHAiOi AiMjAyNC0wNi0xMlQxNDoxMToy Mi44MjI3NDkifQ.MEYCIQC7QopV lFJVpOOEoxmcSzoV8z_UM0FQVIT XPp1x6Icx7wIhALddAx_A0hsIT_pj Ye8tF_YCTY80VKhT4rSQmn1G2EPf

3. Kasus 3: Payload Besar

header	{
	'alg': 'ES256',
	'typ': 'JWT'
	}

payload	{
	'sub': '13521084',
	'name': 'Austin Gabriel Pardosi',
	'email': 'gabrielpardosi26@gmail.com',
	'address': 'Jl. Ganesa No.10, Lb. Siliwangi, Kecamatan Coblong, Kota Bandung, Jawa Barat 40132',
	'phone': '+6281313131313',
	'roles': ['admin', 'user', 'guest'],
	'permissions': {
	'read': True,
	'write': True,
	'delete': False
	},
	'iat': datetime.utcnow().isoformat(),
	'exp': (datetime.utcnow() + timedelta(hours=1)).isoformat(),
	'additional_info': 'This is a very large payload to test the limits of the JWT implementation. ' * 10
	}

Hasil :

Private Key	MIGHAgEAMBMGByqGSM49AgE GCCqGSM49AwEHBG0wawIBAQ Qgm2kEsc6AXrQ0+LXwcJZgvc89B Y+o6fb6/lP2NnRfZsmhRANCAAS7 rGU9Cq5ldIH9Vjig6JxRe9SZm0q1Q kB/pa9l4Shi76gUmf7+MbneIH33Vf 1pRtZF4i4RG6UKWqAjnimFvFED
Public Key	MFkwEwYHKoZIzj0CAQYIKoZIzj0 DAQcDQgAEu6xlPQquZXSB/VY4o OicUXvUmZtKtUJAf6WvZeEoYu+o FJn+/jG53iB991X9aUbWReIuERulC lqgI54phbxRAw==
JWT Token	eyJhbGciOiAiRVMyNTYiLCAidHl wIjogIkpXVCJ9.eyJzdWIiOiAiMTM 1MjEwODQiLCAibmFtZSI6ICJBdX N0aW4gR2FicmllbCBQYXJkb3NpI iwgImVtYWlsIjogImdhYnJpZWxwY XJkb3NpMjZAZ21haWwuY29tIiwg ImFkZHJlc3MiOiAiSmwuIEdhbmV zYSBOby4xMCwgTGIuIFNpbGl3Y W5naSwgS2VjYW1hdGFuIENvYm xvbmcsIEtvdGEgQmFuZHVuZywg SmF3YSBCYXJhdCA0MDEzMiIsI CJwaG9uZSI6ICIrNjI4MTMxMzEz MTMxMyIsICJyb2xlcyI6IFsiYWRta W4iLCAidXNlciIsICJndWVzdCJdL CAicGVybWlzc2lvbnMiOiB7InJlY WQiOiB0cnVlLCAid3JpdGUiOiB0c nVlLCAiZGVsZXRlIjogZmFsc2V9 LCAiaWF0IjogIjIwMjQtMDYtMT JUMTM6NTA6NDQuOTI1NTE3Ii wgImV4cCI6ICIyMDI0LTA2LTEy VDE0OjUwOjQ0LjkyNTUxNyIsIC JhZGRpdGlvbmFsX2luZm8iOiAiV GhpcyBpcyBhIHZlcnkgbGFyZ2Ugc GF5bG9hZCB0byB0ZXN0IHRoZS BsaW1pdHMgb2YgdGhlIEpXVCB pbXBsZW1lbnRhdGlvbi4gVGhpcy BpcyBhIHZlcnkgbGFyZ2UgcGF5b G9hZCB0byB0ZXN0IHRoZSBsaW1 pdHMgb2YgdGhlIEpXVCBpbXBsZ W1lbnRhdGlvbi4gVGhpcyBpcyBhI HZlcnkgbGFyZ2UgcGF5bG9hZCB 0byB0ZXN0IHRoZSBsaW1pdHMg b2YgdGhlIEpXVCBpbXBsZW1lbnR hdGlvbi4gVGhpcyBpcyBhIHZlcnk gbGFyZ2UgcGF5bG9hZCB0byB0ZXN0IHRoZSBsaW1pdHMgb2Yg dGhlIEpXVCBpbXBsZW1lbnRh dGlvbi4gVGhpcyBpcyBhIHZlcnk gbGFyZ2UgcGF5bG9hZCB0byB0ZXN0IHRoZSBsaW1pdHMgb2Yg dGhlIEpXVCBpbXBsZW1lbnRhd Glvbi4gVGhpcyBpcyBhIHZlcnk gbGFyZ2UgcGF5bG9hZCB0byB0ZXN0IHRoZSBsaW1pdHMgb2 YgdGhlIEpXVCBpbXBsZW1lbn

	RhdGlvbi4gVGhpcyBpcyBhIH ZlcnkgbGFyZ2UgcGF5bG9hZ CB0byB0ZXN0IHRoZSBsaW 1pdHMgb2YgdGhlIEpXVCBpb XBsZW1lbnRhdGlvbi4gVGhpc yBpcyBhIHZlcnkgbGFyZ2Ugc GF5bG9hZCB0byB0ZXN0IHR oZSBsaW1pdHMgb2YgdGhlIE pXVCBpbXBsZW1lbnRhdGlvb i4gVGhpcyBpcyBhIHZlcnkgbG FyZ2UgcGF5bG9hZ CB0byB0ZXN0IHRoZSBsaW1 pdHMgb2YgdGhlIEpXVCBpb XBsZW1lbnRhdGlvbi4gVGhpc yBpcyBhIHZlcnkgbGFyZ2Ugc GF5bG9hZCB0byB0ZXN0IHR oZSBsaW1pdHMgb2YgdGhlIE pXVCBpbXBsZW1lbnRhdGlvb i4gIn0.MEQCICSouJcXmWR3 F0VRZD3My9EM8_Y5Km8H iDEPk5dkDQhHAiBjUcwwGV g9TdRXslj_vyQus3LPQPPcuK HSLMCvL_OpRw

V. KESIMPULAN

Dari makalah ini, dapat disimpulkan bahwa implementasi algoritma *Elliptic Curve Cryptography* (ECC), *Elliptic Curve Digital Signature Algorithm* (ECDSA), dan *JSON Web Token (JWT)* dalam sistem keamanan berbasis Python berhasil memberikan tingkat keamanan yang tinggi dengan performa yang efisien. Pengujian yang dilakukan dengan berbagai ukuran payload menunjukkan bahwa sistem mampu menghasilkan dan memverifikasi token dengan cepat dan menggunakan sumber daya yang minimal. Implementasi ini menunjukkan bahwa ECC dan ECDSA adalah pilihan yang tepat untuk aplikasi yang membutuhkan otentikasi dan pertukaran data yang aman, sekaligus memberikan fleksibilitas dan efisiensi yang diperlukan untuk menangani data dalam skala yang lebih besar.

UCAPAN TERIMA KASIH

Penulis mengucapkan puji dan syukur yang sebesar-besarnya kepada Tuhan Yang Maha Esa atas berkat dan Rahmat-Nya sehingga makalah yang berjudul "Implementasi Algoritma Elliptic Curve Digital Signature dalam Peningkatan Kemaanan JWT" dapat diselesaikan dengan baik. Penulis juga menyampaikan terima kasih kepada dosen pengajar IF4020 Kriptografi, Dr. Rinaldi Munir, S.T., M.T., yang telah memberikan bimbingan dan ilmu terkait materi kriptografi, khususnya dalam penerapan algoritma tanda tangan digital. Ucapan terima kasih yang sebesar-besarnya juga penulis sampaikan kepada para penulis sumber referensi yang digunakan dalam makalah ini, yang telah memberikan wawasan dan pengetahuan yang diperlukan untuk menyelesaikan tugas ini.

PRANALA KODE PROGRAM IMPLEMENTASI

Kode program implementasi dapat diakses pada pranala berikut.

https://github.com/AustinPardosi/Enhanced-JWT-Security-with-ECDSA

DAFTAR PUSTAKA

[1] Munir, Rinaldi. 2024. Slide Kuliah Kriptografi (Bandung: 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, 12 Juni 2024

Austin Gabriel Pardosi
13521084