Layanan Kriptografi API pada Punk Records-v1

Laporan Akhir ini Disusun untuk Memenuhi Ujian Akhir Semester Mata Kuliah Keamanan dan Integritas Data

Disusun oleh:

1. Ayda' Nur Salzabillah (24031554017)
2. Gresya Maranatha S (24031554027)
3. M. Zikri Widiandra (24301554088)
4. Grace Pinkkan Ladyna (24301554137)

Dosen Mata Kuliah:

Hasanuddin Al-Habib, S.Si., M.Si.

Sains Data, Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Negeri Surabaya
2025

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring berkembangnya teknologi digital, pertukaran data antar aplikasi semakin sering dilakukan melalui Application Programming Interface (API). API memudahkan komunikasi antar sistem, namun juga menimbulkan risiko keamanan ketika data yang dikirim berupa pesan atau dokumen penting. Jika tidak dilindungi dengan baik, data tersebut dapat disalahgunakan oleh pihak yang tidak bertanggung jawab. Untuk mengatasi hal tersebut, dibutuhkan sistem keamanan yang dapat memastikan autentikasi, integritas data, dan *non-repudiation*. Autentikasi berfungsi untuk memastikan bahwa pengirim data adalah pengguna yang sah, integritas data menjamin bahwa isi pesan tidak mengalami perubahan, sedangkan non-repudiation memastikan bahwa pengirim tidak dapat menyangkal pesan atau dokumen yang telah dikirim.

Salah satu cara untuk memenuhi kebutuhan tersebut adalah dengan menggunakan digital signature berbasis kriptografi kunci publik (public key). Pada sistem ini, pengguna menandatangani pesan atau dokumen menggunakan private key, sementara proses verifikasi dilakukan menggunakan public key yang tersimpan di server. Oleh karena itu, pada proyek ini dibuat layanan kriptografi berbasis API menggunakan FastAPI yang mendukung pendaftaran public key, autentikasi pengguna, pembuatan dan verifikasi tanda tangan digital, serta pengamanan pesan dan dokumen PDF agar komunikasi data menjadi lebih aman.

1.2 Tujuan

1.2.1 Mengimplementasikan sistem keamanan berbasis digital signature
1.2.2 Menerapkan multi-user signing
1.2.3 Melakukan verifikasi pesan dan dokumen PDF
1.2.4 Menguji proses relay pesan melalui API

BAB II
DESKRIPSI PROGRAM

2.1 Gambaran Umum Sistem

Pada proyek ini, sistem dikembangkan menggunakan sebuah layanan API yang dibuat untuk membantu menjaga keamanan dan keutuhan data dalam proses pertukaran informasi antar pengguna. Sistem ini berfungsi sebagai pihak ketiga yang dipercaya untuk memastikan bahwa pesan atau dokumen yang dikirim benar-benar berasal dari pengirim yang sah dan tidak mengalami perubahan selama proses pengiriman.

Pada sistem ini, setiap pengguna memiliki pasangan kunci kriptografi berupa *private key* dan *public key*. *Private key* disimpan dan digunakan oleh pengguna di sisi client untuk menandatangani pesan atau dokumen, sedangkan *public key* didaftarkan ke server. Dengan cara ini, server dapat melakukan proses verifikasi tanpa harus mengetahui *private key* pengguna, sehingga keamanan identitas tetap terjaga. Ketika pengguna mengirimkan pesan atau dokumen, data tersebut terlebih dahulu ditandatangani secara digital di sisi client. Selanjutnya, server akan memeriksa tanda tangan digital tersebut menggunakan *public key* yang tersimpan. Jika hasil verifikasi menunjukkan bahwa tanda tangan valid, maka data dianggap aman dan dapat diproses lebih lanjut, seperti diteruskan ke pengguna lain atau disimpan sebagai arsip.

Selain itu, sistem juga menerapkan mekanisme otentikasi berbasis token untuk mengatur sesi komunikasi. Setiap pengguna yang ingin mengakses layanan API harus memiliki token yang masih berlaku, sehingga hanya pengguna yang telah terdaftar dan terautentikasi yang dapat menggunakan sistem. Secara keseluruhan, sistem ini dirancang sebagai simulasi penerapan konsep keamanan data menggunakan kriptografi dalam sebuah layanan API sederhana. Sistem ini menekankan pada aspek keaslian data, keutuhan informasi, serta pengelolaan akses yang aman dalam proses komunikasi antar pengguna.

2.2 Arsitektur Sistem

Sistem ini dibangun dengan arsitektur *client–server*, di mana proses utama dalam pengamanan data dibagi antara sisi pengguna dan sisi server. Proses pembuatan dan penandatanganan data dilakukan di sisi client, sedangkan proses verifikasi dan pengelolaan data dilakukan di sisi server. Pembagian ini dilakukan agar data sensitif, khususnya *private key*, tetap aman dan tidak berpindah dari sisi pengguna ke server.

1. **Client (Python)**

   Client dibuat menggunakan bahasa pemrograman Python dan digunakan langsung oleh pengguna. Pada sisi ini, pengguna melakukan proses pembuatan serta penyimpanan *private key* yang digunakan untuk menandatangani pesan maupun dokumen. Seluruh proses penandatanganan dilakukan di sisi client, sehingga *private key* tidak pernah dikirimkan ke server. Selain melakukan penandatanganan, client juga bertugas menyiapkan data yang akan dikirim ke server, seperti pesan, dokumen, dan signature dalam format yang sesuai. Data tersebut kemudian dikirim ke server untuk diproses lebih lanjut.

2. **Server (FastAPI)**
   Server dikembangkan menggunakan framework FastAPI dan berperan sebagai pihak ketiga yang dipercaya dalam sistem. Server menerima permintaan dari client dan melakukan pemeriksaan terhadap keabsahan data yang diterima. Pemeriksaan ini meliputi verifikasi tanda tangan digital menggunakan *public key* pengguna yang telah terdaftar. Selain itu, server juga mengatur akses pengguna melalui mekanisme otentikasi berbasis token. Dengan cara ini, hanya pengguna yang telah terdaftar dan memiliki token yang valid yang dapat mengakses layanan yang tersedia. Server juga menangani proses penerusan pesan antar pengguna serta penyimpanan data yang diperlukan.
3. **Database**
   Database pada sistem ini menggunakan penyimpanan berbasis file JSON. Database digunakan untuk menyimpan informasi penting yang dibutuhkan oleh server seperti data pengguna (username dan identitas dasar), lokasi file *public key* pengguna, riwayat pesan yang diterima pengguna, informasi tambahan terkait integritas data. Penggunaan database berbasis file dipilih untuk menyederhanakan implementasi sistem, mengingat fokus utama project ini adalah penerapan keamanan dan integritas data, bukan optimasi sistem penyimpanan.

2.3 Algoritma Kriptografi yang Digunakan

Dalam pengembangan sistem Punk Records-v1, aspek keamanan data menjadi fokus utama. Oleh karena itu, sistem ini menerapkan kombinasi antara kriptografi kunci publik (asymmetric cryptography) dan fungsi hash untuk memastikan bahwa data yang dipertukarkan antar pengguna tetap aman, utuh, dan berasal dari pihak yang benar.

1. **ECDSA (Elliptic Curve Digital Signature Algorithm)**
   ECDSA berfungsi untuk membuat dan memverifikasi tanda tangan digital pada pesan maupun dokumen. Dengan memanfaatkan pasangan *private key* dan *public key*, algoritma ini memungkinkan server untuk memastikan bahwa data yang diterima benar-benar dikirim oleh pemilik *private key* yang sah. Selain itu, ECDSA juga membantu menjaga keutuhan data, karena perubahan sekecil apapun pada isi pesan akan menyebabkan tanda tangan digital menjadi tidak valid
2. **Ed25519**
   Sistem ini juga menggunakan Ed25519 sebagai algoritma tanda tangan digital alternatif. Ed25519 dipilih karena memiliki proses signing dan verifikasi yang relatif cepat serta tingkat keamanan yang tinggi. Penggunaan Ed25519 menunjukkan bahwa sistem tidak bergantung pada satu jenis algoritma saja, melainkan mendukung variasi algoritma kriptografi sesuai dengan kebutuhan pengguna.
3. **SHA-256**
   Sistem memanfaatkan SHA-256 sebagai fungsi hash. SHA-256 digunakan untuk menghasilkan nilai hash dari pesan maupun dokumen sebelum dilakukan proses penandatanganan. Nilai hash ini bersifat unik, sehingga jika terjadi perubahan data, nilai hash yang dihasilkan akan berbeda. Dengan demikian, SHA-256 berperan penting dalam mendeteksi adanya manipulasi data selama proses pertukaran informasi.

4. **JWT (JSON Web Token)**

   Selain mekanisme tanda tangan digital dan hashing, sistem juga menerapkan JWT (JSON Web Token) untuk mengatur autentikasi dan sesi komunikasi. JWT digunakan untuk memastikan bahwa setiap permintaan ke server berasal dari pengguna yang telah terautentikasi. Token ini harus disertakan pada setiap akses ke endpoint API, sehingga akses sistem menjadi lebih terkontrol dan aman.

BAB III
PENGGUNAAN PROGRAM

3.1 Menjalankan Server

Pengguna menjalankan aplikasi server menggunakan FastAPI dengan perintah:

```
PS H:\sem 3\KID\projek kid> & "H:/sem 3/KID/projek kid/.venv/Scripts/Activate.ps1"
(kid-uas) PS H:\sem 3\KID\projek kid> uvicorn api:app --reload
```

Server dapat diakses melalui browser pada alamat:

```
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
```

Swagger UI digunakan sebagai antarmuka untuk menguji seluruh endpoint sistem.

3.2 Pengecekan Kesehatan Server

Sebelum menggunakan fitur lain, pengguna dapat melakukan pengecekan status server melalui endpoint GET /health untuk memastikan layanan berjalan dengan baik.

Di Swagger UI:

- Buka GET /health
- Klik tombol Execute
- JANGAN isi apa pun
- Lihat bagian Responses

3.3 Registrasi Pengguna dan Public Key

Pengguna mendaftarkan akun dengan mengunggah public key melalui endpoint /store.

Contoh penggunaan:

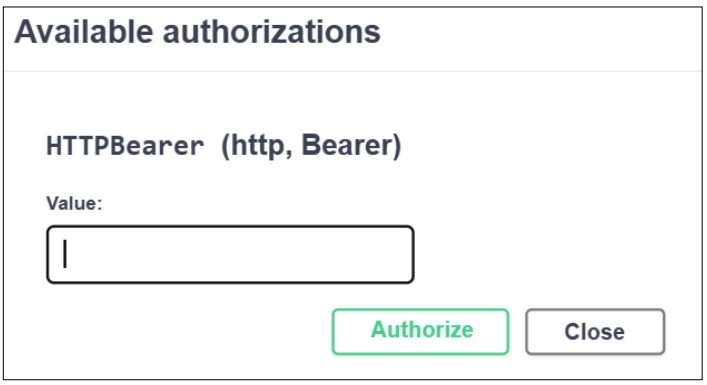
- Username: pinkkan
- Fullname: Pinkkan Ladyna
- File public key: pinkkan_public_key.pem

Jika berhasil, sistem akan menyimpan public key ke dalam database (user_db.json)

3.4 Login dan Autentikasi

Pengguna melakukan login melalui endpoint /login dengan memasukkan username yang telah terdaftar. Sistem akan mengembalikan JWT token yang digunakan untuk mengakses endpoint yang dilindungi.

Token dimasukkan ke menu Authorize pada Swagger UI dengan format:



3.5 Pembuatan Tanda Tangan Digital (Client)

Pengguna menjalankan program client (client.py) untuk membuat:

- Pesan dalam format Base64
- Tanda tangan digital (signature) menggunakan private key

Tanda tangan ini dihasilkan secara lokal dan tidak dibuat di server, sehingga private key tetap aman.

3.6 Verifikasi Tanda Tangan Digital

Pengguna mengirimkan:

- Username
- Message (Base64)
- Signature (Base64)

melalui endpoint /verify.

Sistem akan memverifikasi signature menggunakan public key yang tersimpan dan menampilkan status Signature VALID atau Signature INVALID

3.7 Pengiriman Pesan Antar Pengguna

Pengguna dapat mengirim pesan ke pengguna lain melalui endpoint /relay dengan menyertakan:

- Sender (pengirim)
- receiver (penerima)
- message (isi pesan)

Pesan yang dikirim akan masuk ke inbox penerima.

3.8 Unggah Dokumen PDF

Pengguna dapat mengunggah file PDF melalui endpoint /upload-pdf.

Sistem akan:

- Menghitung hash SHA-256 dari PDF
- Menyimpan file PDF ke server

3.9 Verifikasi Tanda Tangan PDF

Pengguna melakukan verifikasi tanda tangan PDF melalui endpoint /verify-pdf dengan mengirimkan:

- File PDF
- Signature PDF (Base64)
- Username pemilik public key

Sistem akan memverifikasi keaslian dan integritas dokumen PDF.

4.0 Endpoint GET /users

Endpoint GET /users digunakan untuk menampilkan data seluruh pengguna yang telah terdaftar dalam sistem. Endpoint ini hanya dapat diakses oleh pengguna yang telah login dengan token JWT yang valid. Data yang ditampilkan meliputi nama lengkap, public key, riwayat pesan (inbox), dan hash pesan terakhir, sehingga mendukung penampilan informasi pengguna secara terpusat dan aman.

BAB IV
FUNGSI-FUNGSI PROGRAM

4.1 Fungsi pada Client

Client berfungsi sebagai pihak yang digunakan oleh pengguna untuk menyiapkan data sebelum dikirim ke server. Pada sisi client, pengguna memiliki private key yang digunakan untuk membuat tanda tangan digital. Program client mendukung dua jenis algoritma kriptografi, yaitu ECDSA dan Ed25519. Dari private key tersebut, client juga menghasilkan public key yang nantinya dikirim ke server sebagai identitas pengguna.

Client digunakan untuk menandatangani pesan teks. Pesan dibuat dalam bentuk data biner, kemudian diubahke format Base64 agar mudah dikiirm ke sever. Setelah itu, pesan ditandatangani menggunakan private key sehingga dihasilkan signature digital. Signature ini digunakan untuk membuktikan bahwa pesan benar-benar berasal dari pengguna yang sah. Selain pesan teks, client juga digunakan untuk menandatangai file PDF. File PDF dibaca oleh client, kemudian dilakukan proses hashing menggunakan SHA-256. Hasil hash tersebut ditandatangani menggunakan private key untuk menghasilkan signature digital. Signature ini dikirim ke server untuk memastikan bahwa isi file PDF tidak berubah.

4.2 Fungsi pada Server

Server dibuat menggunakan FastAPI dan berfungsi sebagai layanan utama untuk sistem keamanaan. Server bertugas menyimpan public key pengguna, mengelola data pengguna, serta memveivikasi signature digital yang dikirim oleh client. Data pengguna disimpan dalam bentuk file JSON. Server juga menggunkan JSON Web Token (JWT)untuk proses autentika. Pengguna yang sudah terdaftar dapat melakukan login untuk mendapatkan token. Token ini digunakan sebagai tanda akses dan harus disertakan saat mengakses layanan server yang membutuhkan otorisasi.

Dalam proses verivikasi pesan, server menerima pesan dan signature dari client. server akan mengambil public key pengguna, kemudian melakukan verifikasi signature sesuai dengan algoritma yang digunakan. Jika signature valid, server aka menyatakan bahwa pesan tersebut asli dan tidak mengalami perubahan. server juga menyediakan fitur pengiriman pesan antar pengguna tujuan. Selain itu, server mendukung upload file PDF dan verifikasi signature PDF. Server mengitung hash dari file PDF dan mencocokkannya dengan signature digital untuk memastikan keaslian dokumen.

4.3 Mekanisme Verifikasi Signature

Mekanisme verivikasi signature dimulai dari client yang menandatangani pesan atau file menggunakan private key. Pesan dan signature yang dihasilkan kemudian dikirim ke server. Server memverifikasi signature menggunakan public key pengguna yang tersimpan. Untuk algoritma ECDSA, proses verivikasi dilakukkan dengan bantuan hash-256 sedangkan untuk algoritma Ed25519 verivikasi dilakukan secara langsung. Jika hasil verifikasi menunjukkan siganture valid, maka data dinyatakan aman dan tidak diidifikasi selama pengiriman. Dengan adanya mekanisme tanda tangan digital ini, sistem dapat

menjamin keaslian pengirim, menjaga keutuhan data, serta meningkatkan keamanan komunikasi antara client dan server.

BAB V
HASIL PENGUJIAN

5.1 Pengujian Kesehatan Server (/health)

Tujuan:

Untuk memastikan server API berjalan dengan baik dan dapat menerima permintaan dari client.

Langkah Pengujian:

Pengujian dilakukan dengan mengakses endpoint GET /health melalui Swagger UI tanpa menggunakan parameter apa pun.

Hasil Pengujian:

Server memberikan respons berupa status layanan dan waktu akses sistem.

Contoh Output:

Response body


```
{
  "status": "Security Service is running",
  "timestamp": "2025-12-18T00:33:37.264843"
}
```


Download

Status: Berhasil

5.2 Pengujian Penyimpanan Public Key (/store)

Tujuan:

Menguji proses registrasi pengguna dan penyimpanan public key.

Skenario Uji:

- Username: pinkkan
- Fullname: Pinkkan Ladyna
- File: pinkkan_public_key.pem

Hasil:

Public key berhasil disimpan dan data pengguna tercatat di users_db.json.


```
"pinkkan": {
    "fullname": "Pinkkan Ladyna",
    "public_key": "pinkkan_public_key.pem",
    "inbox": [],
    "last_message_hash": ""
}
```


Output:

Response body


```
{
  "message": "Public key stored",
  "user": "pinkkan"
}
```


Download

Status: Berhasil

5.3 Pengujian Login dan Token JWT (/login)

Tujuan:

Menguji proses autentikasi pengguna.

Hasil:

Sistem berhasil menghasilkan token JWT untuk pengguna terdaftar.

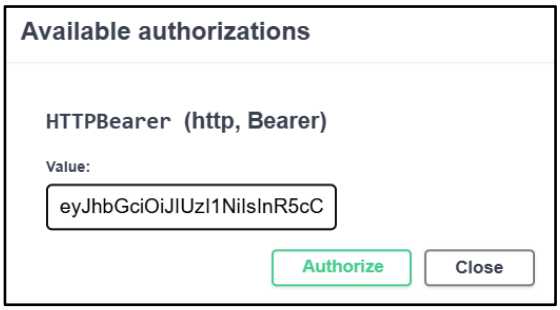
Output:

Response body

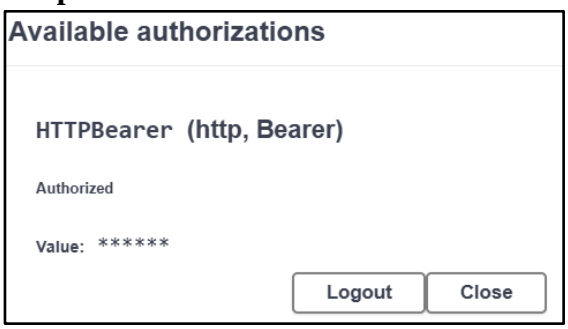

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJwaW5ra2FuIiwiZXhwIjo
xNzY1OTk1MTAyfQ.XOXPPcyxq-cW4qj0sFyrcmD31RrTh1NhmFpEXOtyQ9g"
}
```


Download

Selanjutnya bisa memasukan token ke menu Authorize pada Swagger UI agar dapat mengakses endpoint yang dilindungi.



Output:



Status: Berhasil

5.4 Pengujian Verifikasi Signature Pesan (/verify)

Tujuan:

Menguji validasi tanda tangan digital menggunakan public key.

Skenario Uji:

- Username
- Message: Base64 hasil dari client.py
- Signature: Base64 hasil tanda tangan private key

Hasil:

Sistem berhasil memverifikasi signature dan menghasilkan hash pesan.

Output (Valid):

Response body


```
{
  "message": "Signature VALID",
  "hash": "b94d27b9934d3e08a52e52d7da7dabfac484efe37a5380ee9088f7ace
}
```


Download

Output (Invalid):

Response body


```
{
  "message": "Signature INVALID"
}
```


Download

Status: Berhasil

5.5 Pengujian Relay Message (/relay)

Tujuan:

Menguji pengiriman pesan aman antar dua pengguna, serta memastikan hanya pengirim yang sah yang dapat mengirim pesan menggunakan JWT.

Skenario Uji:

- Sender: pinkkan
- Receiver : grace
- Message: hallo grace, apa kabar?

Hasil:

Sistem berhasil melakukan relay pesan dan tersimpan pada database (users_db.json)

Output:

Response body


```
{
  "message": "Message relayed",
  "to": "grace"
}
```


Download

Output pada database:


```
    "grace": {
        "fullname": "Grace Gyna",
        "public_key": "grace_public_key.pem",
        "inbox": [
            {
                "from": "pinkkan",
                "message": "hallo grace, apa kabar?",
                "timestamp": "2025-12-18T17:02:33.455440"
            }
        ],
        "last_message_hash": ""
    }
}
```


Status: Berhasil

5.6 Pengujian Upload Dokumen PDF (/upload-pdf)

Tujuan:

Menguji sistem untuk menerima dokumen dalam bentuk PDF, menghitung nilai hash dokumen, serta memastikan proses upload hanya dapat dilakukan oleh pengguna yang telah terautentikasi.

Skenario Uji:

File: Unggah Dokumen PDF (contoh: Dummy.pdf)

Hasil:

Sistem berhasil mengunggah dokumen PDF dan menghasilkan hash dokumen.

Output:

Response body


```
{
  "message": "PDF uploaded",
  "hash": "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
  "uploaded_by": "pinkkan"
}
```


Download

Status: Berhasil

5.7 Pengujian Verifikasi Tanda Tangan PDF (/verify-pdf)

Tujuan:

Menguji sistem dalam memverifikasi tanda tangan digital pada dokumen PDF yang diunggah pengguna. Pengujian ini bertujuan memastikan bahwa tanda tangan digital yang dikirimkan benar-benar sesuai dengan file PDF dan public key pengguna yang tersimpan di server.

Langkah pengujian:

- File : Dummy.pdf
- Input : File PDF, signature digital (Base64), dan username pengguna yang telah terautentiksi

Hasil:

Sistem berhasil memverifikasi tanda tangan digital pada file yang dikirim oleh pengguna. Jika file dan signature sesuai, maka hsil verivikasi dinytakan valid. Apabila signature tidak ssuai maka sistem akan mendeteksi bahwa signature tidka cocok dengan isi dokumen, sehingga hasil verivikasi dinyatakan tidak valid.

Output Valid:

Response body


```
{
  "message": "PDF signature VALID"
}
```


Output Invalid:

Response body


```
{
  "message": "PDF signature INVALID"
}
```

Status: Berhasil

5.8 Pengujian Endpoint GET /users

Tujuan:

Pengujian ini bertujuan untuk memastikan bahwa endpoint GET /users dapat menampilkan data pengguna dengan benar dan hanya dapat diakses oleh pengguna yang telah terautentikasi menggunakan token JWT yang valid.

Langkah pengujian:

Pengujian dilakukan dengan mengakses endpoint GET /users melalui Swagger UI tanpa menggunakan parameter apa pun.

Hasil Pengujian:

Berdasarkan pengujian yang dilakukan, endpoint GET /users berhasil menampilkan seluruh data pengguna yang terdaftar ketika token JWT valid digunakan. Apabila token tidak disertakan atau tidak valid, sistem menolak akses dengan pesan kesalahan, sehingga aspek keamanan data tetap terjaga.

Contoh Output:

Response body


```
{
  "grace": {
    "fullname": "Grace Gyna",
    "public_key": "grace_public_key.pem",
    "inbox": [
      {
        "from": "pinkkan",
        "message": "hallo grace, apa kabar?",
        "timestamp": "2025-12-18T15:04:33.985339"
      }
    ],
    "last_message_hash": "7e9ff4c3b861d1c4a5466a7504c949bb5186cdb807096a09aee0d03924bc80a4"
  },
  "pinkkan": {
    "fullname": "Pinkkan Ladyna",
    "public_key": "pinkkan_public_key.pem",
    "inbox": [
      {
        "from": "grace",
        "message": "hallo pikkan, luar biasa kabar aku baik",
        "timestamp": "2025-12-18T16:17:31.289476"
      }
    ],
    "last_message_hash": "b94d27b9934d3e08a52e52d7da7dabfac484efe37a5380ee9088f7ace2efcde9"
  }
}
```


Download

Status: Berhasil

BAB VI
KESIMPULAN

Berdasarkan seluruh proses perancangan hingga pengujian yang telah dilakukan, dapat disimpulkan bahwa layanan kriptografi API pada Punk Records-v1 telah berhasil diimplementasikan dengan optimal. Sistem ini mampu menjamin keamanan data melalui mekanisme *digital signature*, yang memastikan bahwa setiap pesan dan dokumen PDF yang dipertukarkan terjamin keasliannya (*authentication*), tidak mengalami perubahan (*integrity*), dan tidak dapat disangkal oleh pengirimnya (*non-repudiation*). Penggunaan arsitektur *client-server* yang memungkinkan proses penandatanganan dilakukan langsung di sisi pengguna. Dengan cara ini, *private key* tetap berada di bawah kendali user dan tidak pernah disimpan di server, sehingga keamanan lebih terjaga. Kombinasi algoritma ECDSA, Ed25519, SHA-256, serta penggunaan JWT untuk autentikasi terbukti sangat efektif dalam memperkuat sistem. Secara keseluruhan, pengujian menunjukkan bahwa semua fitur berjalan sesuai rencana dan sistem ini siap menjadi fondasi yang kuat bagi pengembangan keamanan data di masa depan.

REFERENSI

Adeshina, A. A. A. (2024, May 8). *Securing FastAPI with JWT token-based authentication*. TestDriven.io. https://testdriven.io/blog/fastapi-jwt-auth/

Al-Zubaidie, M., Zhang. Z., Zhang., J (2019). *Efficient and secure ECDSA algorithm and its applications: A survey* (SSRN Scholarly Paper No. 5280338). SSRN Vol. 11, No. 1. https://papers.ssrn.com/abstract=5280338

FastAPI Team. (n.d.). *OAuth2 with Password (and hashing), Bearer with JWT tokens*. FastAPI. https://fastapi.tiangolo.com/tutorial/security/oauth2-jwt/

Muraya, D. (2025, October 20). *How to handle file uploads in FastAPI*. DavidMuraya.com. https://davidmuraya.com/blog/fastapi-file-uploads/

MojoAuth. (n.d.). *Generate keypair using ECDSA with FastAPI*. MojoAuth. https://mojoauth.com/keypair-generation/generate-keypair-using-ecdsa-with-fastapi/

LAMPIRAN

Link Github: