Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Tugas Besar I (Tubes I) IF4020 Kriptografi Sem. I Tahun 2025/2026

# Pembuatan *Web Based Chat App* dengan Enkripsi *End-to-End*

---

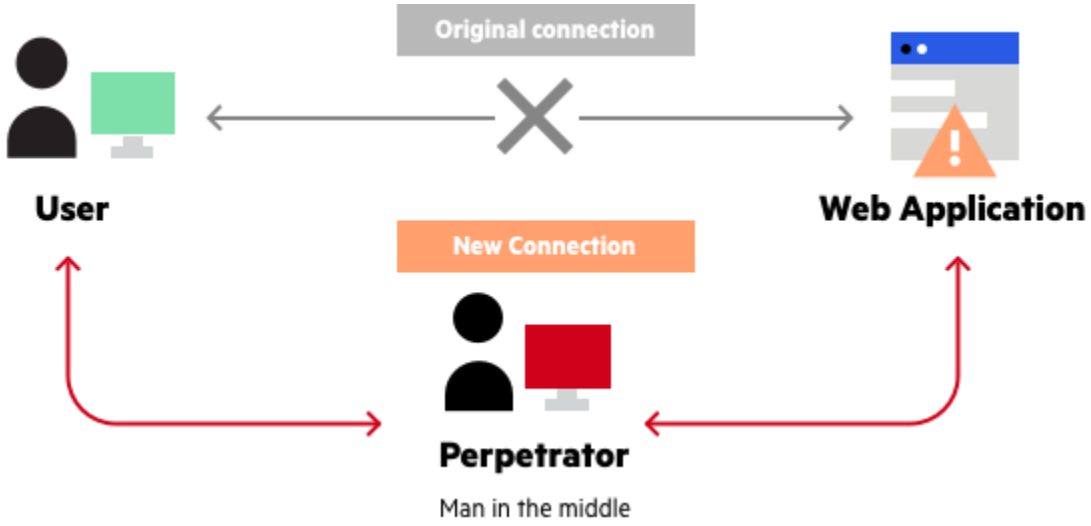
| | | |
|---|---|---|
| Batas pengumpulan | : | Senin, 1 Desember 2025, Pukul 23.59 WIB |
| Tempat pengumpulan | : | Form Pengumpulan |
| Anggota kelompok | : | 2-3 orang |
| QnA | : | Sheet QnA |

**Berkas pengumpulan :**

a. Laporan (soft copy) dengan format PDF
b. Tautan video demo singkat
c. Kode program yang bisa dijalankan, disertai README

## Latar Belakang

Berbagai *platform chat* modern telah memudahkan pertukaran miliaran pesan secara instan. Namun, laporan dari berbagai lembaga keamanan siber menunjukkan bahwa serangan terhadap sistem komunikasi modern, seperti *man-in-the-middle attack*, pemalsuan identitas pengirim, serta manipulasi konten pesan, masih marak setiap tahunnya.




**Gambar 1.** *Man-in-the-middle attack.*
Sumber: https://www.imperva.com/learn/application-security/man-in-the-middle-attack-mitm/

Enkripsi *end-to-end* (E2E) menjadi standar untuk melindungi konten pesan dari penyadapan pihak ketiga. Namun, enkripsi saja tidak cukup untuk melakukan autentikasi pengirim serta memastikan integritas pesan. Oleh karena itu, *digital signature* digunakan untuk memverifikasi keaslian pengirim serta memastikan isi pesan tidak diubah selama pengiriman. ECDSA (*Elliptic Curve*

*Digital Signature Algorithm*) dipilih karena kemampuannya untuk menghasilkan tingkat keamanan tinggi dengan kunci yang lebih kecil serta efisiensi komputasi yang lebih tinggi dibandingkan algoritma tradisional lainnya. Selain itu, SHA-3 juga digunakan sebagai fungsi *hash* kriptografis untuk menghasilkan *fingerprint* digital yang memperkuat integritas pesan.

Melalui pemanfaatan konsep kriptografi di atas, aplikasi *chat* memungkinkan pengguna untuk berkomunikasi secara *real-time* dengan jaminan privasi (enkripsi E2E), verifikasi identitas pengirim (ECDSA), serta verifikasi integritas pesan (SHA-3).

## Penjelasan Implementasi

Pada tugas ini, Anda diminta untuk mengimplementasikan sebuah aplikasi *chat* **berbasis web** yang menggunakan enkripsi ECC, ECDSA (*Elliptic Curve Digital Signature Algorithm*) dan SHA-3 *hashing* untuk menjamin kerahasiaan, keaslian, dan integritas pesan yang dikirimkan antar pengguna. Implementasi dilakukan dengan mengikuti standar keamanan kriptografi modern yang telah diajarkan di materi kuliah.

Secara garis besar, sistem ini memiliki tiga tahap besar, yaitu pembuatan kunci, pengiriman pesan dengan proteksi digital, dan verifikasi pesan yang diterima.

### Tahap 1: Registrasi Pengguna

Ketika pengguna baru mendaftar, pengguna akan mengisi **Username** dan **Password** pada aplikasi web. Password yang pengguna daftarkan akan digunakan sebagai *seed random number generator* (atau *key derivation function*) untuk pembuatan pasangan kunci ECC (bukan *best practices* di dunia nyata, anggap pengguna menggunakan *password* yang sangat aman). Setiap pengguna yang mendaftarkan diri pada aplikasi akan secara otomatis menghasilkan sepasang kunci ECDSA, yaitu:

- ***Private Key*** : Digunakan untuk menandatangani dan mengenkripsi pesan.
- ***Public Key*** : Digunakan oleh penerima untuk mendekripsi dan memverifikasi tanda tangan pesan.

Kunci-kunci ini dibuat menggunakan kurva eliptik standar (seperti secp256r1 atau sejenisnya, dibebaskan). Sebagai contoh, kunci privat dan publik dapat direpresentasikan dalam format heksadesimal berikut:

| Tipe Kunci | Contoh |
|---|---|
| *Private Key* | 0x1f8a2b...a0e9 |
| *Public Key* | (0x9fc4...a6f, 0x08bc...4e5) |

Sepasang kunci ini akan disimpan dengan mekanisme sebagai berikut.

1. *Private key* disimpan di *local storage client* (bukan dikirim ke *server* untuk menjaga keamanan),

2. *Public key* dikirim ke *server* bersama *username*.

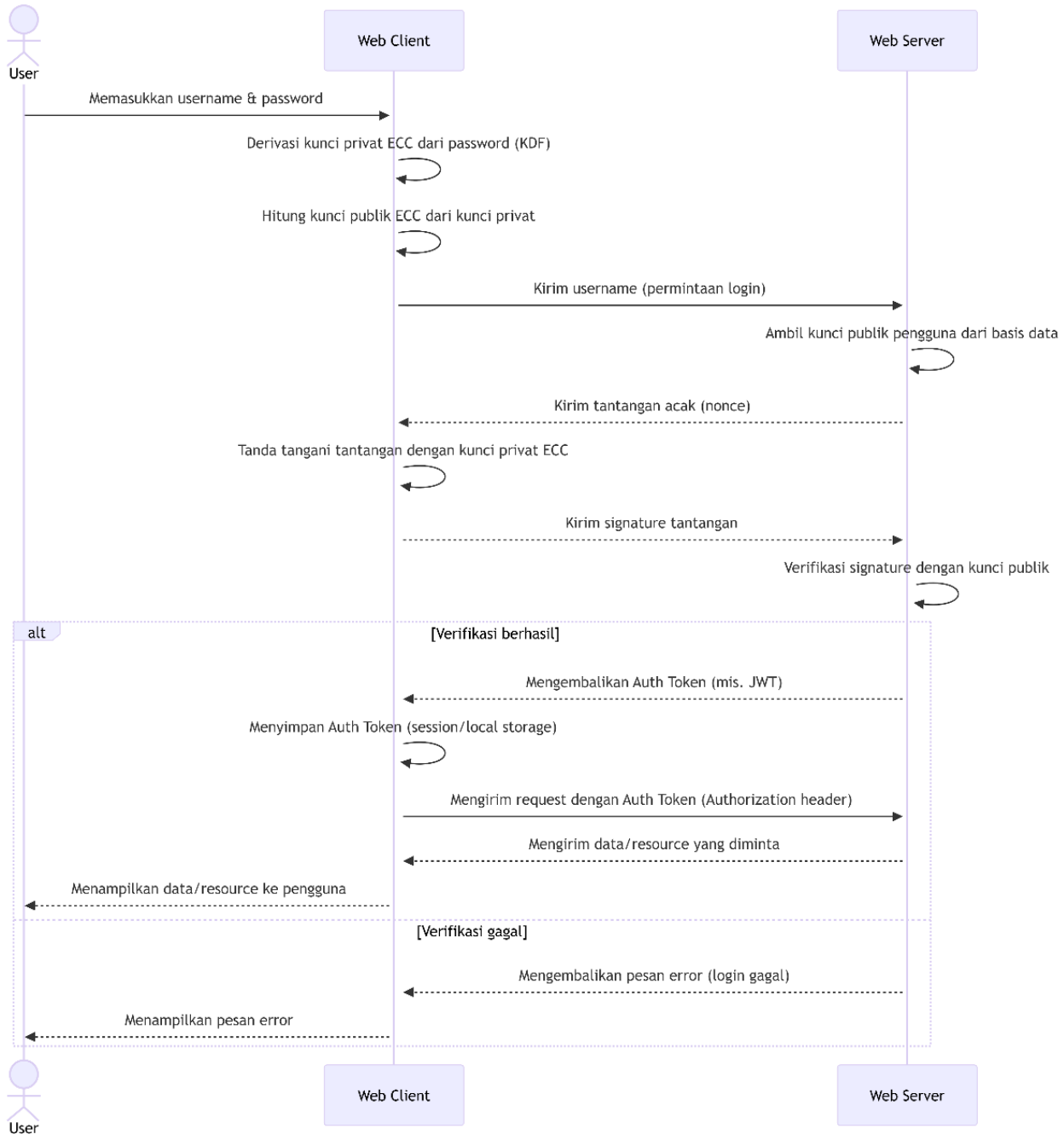
*Web server* akan menyimpan *username*, dan *public key* ke *storage* (mekanisme penyimpanan bebas, disarankan menggunakan *database* SQL/NoSQL).

## Tahap 2: Login

Pada saat melakukan *login*, alur yang terjadi adalah sebagai berikut.

1. Pengguna memasukkan **Username** dan **Password** pada aplikasi web (Web Client).
2. Web Client menurunkan kunci privat ECC dari Password (misalnya melalui KDF), menghitung kunci publik, lalu mengirim permintaan login ke Web Server dengan username untuk mendapatkan tantangan acak (nonce).
3. Web Client menandatangani tantangan acak tersebut menggunakan kunci privat ECC dan mengirimkan signature kembali ke Web Server.
4. Web Server memverifikasi signature menggunakan kunci publik yang tersimpan; jika verifikasi berhasil, server mengembalikan Auth Token (misalnya JWT) yang kemudian disimpan dan digunakan oleh Web Client untuk mengakses resource aplikasi.

Perlu untuk diingat bahwa *private key* dari ECC dan ECDSA **tetap** berada di sisi *client*.




**Gambar 2.** *Sequence diagram* login.

## Tahap 3: Daftar Kontak

Setelah melalui tahap Login, pengguna dapat memilih lawan komunikasi dengan melihat daftar kontak. Pada fitur ini, pengguna akan **menambahkan *username*** lawan bicaranya ke daftar kontak. Setelah itu, pengguna akan memilih salah satu dari daftar kontaknya untuk memulai percakapan (*chat*) dengan membuat sebuah sesi (implementasi dan antarmuka dibebaskan, dapat menggunakan WebSocket).

### Tahap 4: Pengiriman Pesan (*Signing* and *Sending*)

Saat pengguna mengetik pesan dan menekan tombol "Send", terjadi proses sebagai berikut:

1. ***Hashing***
   **Isi pesan, *timestamp, username* pengirim, serta *username* penerima** akan di-*hash* menggunakan algoritma SHA-3 (variasi SHA3-256) menghasilkan output *fixed-length hash*.
2. **Penandatanganan (*Signing*)**
   *Hash* dari pesan kemudian ditandatangani menggunakan *private key* pengguna dengan algoritma ECDSA yang disimpan pada *local storage*. Proses ini menghasilkan tanda tangan digital (*digital signature*).
3. **Enkripsi**
   Isi pesan akan dienkripsi dengan metode ECC sehingga menghasilkan pesan yang hanya didekripsi oleh kedua pihak dalam percakapan.
4. **Pengemasan**
   Data yang dikirim melalui *socket* minimal mencakup
   a. *Username* pengirim,
   b. *Username* penerima,
   c. Pesan terenkripsi (hasil ECC),
   d. *Hash* pesan (hasil SHA-3),
   e. *Signature* (hasil signing ECDSA)
5. **Penanganan**
   *Server* akan menerima pesan dan:
   a. **(Opsional)** Menyimpan pesan terenkripsi dalam *database*/*message queue*,
   b. Mengirimkan pesan ke penerima secara *real-time* (*Short Polling* atau WebSocket).

Format data dapat berupa JSON dengan struktur seperti berikut (hanya referensi, Anda dibebaskan untuk menyusun *payload* yang relevan).

```
{
 "sender_username": "poirot",
 "receiver_username": "agatha",
 "encrypted_message": "3bdc7620e7cf8e2ded0d0e24bbd...",
 "message_hash": "abc123...",
 "signature": {
  "r": "abc...",
  "s": "def..."
 },
 "timestamp": "2025-11-10T12:30:00Z"
}
```

### Tahap 5: Pengiriman dan Verifikasi Pesan

Pada sisi penerima, aplikasi akan melakukan proses **verifikasi pesan** sebagai berikut.

1. Pesan terenkripsi akan didekripsi kembali untuk mendapat plainteks,
2. Plainteks akan di-*hash* ulang menggunakan SHA-3,
3. Ambil *public key* pengirim dari *storage* (yang di-*fetch* dari server),
4. Verifikasi *signature* menggunakan *public key* dan *hash*. *Hash* hasil komputasi terhadap plainteks dibandingkan dengan *message hash* yang dikirim. Jika berbeda, pesan dianggap **korup** dan tidak diproses.
    a. Jika verifikasi berhasil, pesan ditampilkan dengan label "✅ **Verified**",
    b. Jika gagal, ditampilkan dengan label "❌ **Unverified**".

Semua proses verifikasi ini dilakukan secara otomatis, sehingga pengguna dapat dengan cepat mengetahui apakah sebuah pesan benar-benar berasal dari pengirim yang valid. Cara penyajian label pada antarmuka dibebaskan.

## Bonus

1. *Deployment*
   Aplikasi web harus dideploy sehingga dapat diakses melalui URL publik. URL tersebut harus dicantumkan di laporan/dokumentasi sebagai bukti bahwa aplikasi sudah berjalan secara online.

## Prosedur Pengerjaan

Berikut adalah prosedur pengerjaan dari tugas ini.

1. Tugas dikerjakan berkelompok dengan anggota **minimal** 2 orang dan maksimal 3 orang, dilarang *gabut*. Cantumkan pembagian tugas dengan jelas antara anggota kelompok.
2. Waktu pengumpulan tugas **paling lambat Senin, 1 Desember 2025 sebelum pukul 23.59**. Pengumpulan tugas yang terlambat **akan mendapatkan pengurangan nilai** secara progresif.
3. Implementasi aplikasi dilakukan berbasis **web**, kakas yang digunakan untuk *client*, *server*, dan *storage*/basis data **dibebaskan**.
4. Program harus mengandung komentar yang jelas serta mudah dibaca.
5. Anda dilarang menggunakan kode program yang didapatkan dari internet (alasan menggunakan kakas seperti GitHub Copilot tidak diterima). Anda harus membuat program sendiri, diperbolehkan untuk belajar dari program yang sudah ada maupun menggunakan kakas AI (tetapi Anda harus tetap memahami apa yang dikerjakan, bila menggunakan kakas AI maka lampirkan tangkapan layar atau tautan penggunaannya di laporan).
6. Fungsi hash dibebaskan dan boleh memakai *library* ataupun *built-in*.

7. Program memiliki antarmuka yang *user-friendly*. Anda juga dapat menambahkan fitur lain untuk menunjang program yang Anda buat (unsur kreativitas).

## Berkas Pengumpulan

1. **Repositori perangkat lunak** berisi
    a. kode sumber (*source code*),
    b. sebuah folder berisi berkas uji,
    c. berkas README, minimal berisi
        i. nama dan deskripsi program,
        ii. kumpulan teknologi yang digunakan (*tech stack*),
        iii. dependensi,
        iv. tata cara menjalankan program.

2. **Video demo** berdurasi maksimal **5 menit**, berisi
    a. deskripsi singkat program,
    b. teknologi yang digunakan (*tech stack*),
    c. penjelasan singkat rancangan, terutama bagian yang dibebaskan atau sesuai kreatifitas dan bonus,
    d. demo kasus uji (yang utama, sesuai dengan ketentuan kasus uji pada 3.d.).

3. **Berkas laporan** dengan ketentuan nama **NIM1_NIM2_NIM3_Tubes1_IF4020.pdf**, berisi
    a. *Cover* laporan ada foto anggota kelompok (foto bertiga). Foto ini menggantikan logo "gajah" ganesha.
    b. Teori singkat (*digital signature*, ECC, ECDSA, Hash, SHA-3, dan lainnya).
    c. Perancangan dan implementasi (**tip**: cukup tampilkan diagram database, diagram sistem, serta perancangan dan implementasi hal-hal yang menarik/dibebaskan/bonus).
    d. Pengujian program dan hasil analisis dengan kasus minimal mencakup beberapa hal berikut (**tip**: cukup *screenshot* dari video demo.
        i. Uji *digital signature* dengan menggunakan:
            - *Private key* yang salah, perlihatkan bahwa pesan masih dalam bentuk acak
            - *Private key* yang benar, perlihatkan bahwa pesan berhasil didekripsi.
        ii. Gunakan *interceptor* seperti Burp Suite atau OWASP Zap untuk:
            - Memproxy request yang keluar masuk *client* dan *server*.
            - Intercept *request* pesan, lalu ubah teks terenkripsi, *hash*, *timestamp*, atau komponen lainnya dalam *payload*, lalu biarkan pesan sampai ke tujuan. Di sisi penerima, tunjukkan bahwa verifikasi pesan gagal.

e. Tautan kode sumber program dalam sebuah repositori Github dengan **README** yang berisi minimal daftar fitur, tata cara menjalankan program, dan identitas pembuat.
f. Lampiran yang berisi antarmuka program.
g. Daftar Pustaka.