Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Tugas 3 II4021 Kriptografi – Semester II tahun 2024/2025
Pembuatan *Secure Message Chat App* dengan ECDSA dan SHA3

Batas pengumpulan	**:** Minggu, 18 Mei 2025, Pukul 23.59 WIB
Tempat pengumpulan	**:** Form Pengumpulan
Anggota kelompok	**:** 2-3 orang
QnA	**:** QnA Tugas II4021 Kriptografi

Berkas pengumpulan :

a. Laporan (soft copy) dengan format PDF
b. Kode program yang bisa dijalankan, disertai README

Latar Belakang

Di era digital ini, keamanan komunikasi menjadi aspek yang sangat penting. Perkembangan teknologi informasi membuka peluang bagi berbagai bentuk komunikasi yang cepat dan praktis, seperti aplikasi pesan singkat. Namun, kemudahan ini juga disertai dengan ancaman seperti penyadapan, pemalsuan identitas, dan manipulasi pesan. Metode keamanan tradisional seperti penggunaan *username* dan *password* sering kali tidak cukup untuk memastikan keaslian pengirim dan integritas pesan yang diterima.

Gambar 1. Aplikasi pengiriman pesan yang rentan terhadap keamanan.
Sumber: https://shift.com/blog/apps-hub/most-secure-messaging-apps/

Untuk menjawab tantangan tersebut, *digital signature* menjadi solusi yang efektif. Dengan menandatangani pesan secara kriptografis, penerima dapat memverifikasi keaslian

pengirim sekaligus memastikan isi pesan tidak diubah selama pengiriman. Salah satu algoritma yang banyak digunakan adalah ECDSA (*Elliptic Curve Digital Signature Algorithm*), yang menawarkan keamanan tinggi dengan ukuran kunci lebih kecil dan efisiensi lebih baik dibandingkan algoritma tradisional. Fungsi *hash* seperti SHA-3 juga digunakan untuk menghasilkan sidik jari digital unik dari setiap pesan, memperkuat jaminan integritas data.

Dalam Tugas 3 ini, Anda diminta untuk mengimplementasikan sistem pengiriman pesan aman (*Secure Message Chat App*) berbasis *digital signature*. Sistem ini akan memungkinkan pengguna untuk berkomunikasi secara *real-time,* dengan setiap pesan yang dikirim disertai tanda tangan digital berbasis ECDSA dan hash SHA-3, sehingga pesan dapat diverifikasi keaslian pengirimnya dan dijamin keutuhannya sepanjang jalur komunikasi.

Penjelasan Implementasi

Pada tugas ini, Anda diminta untuk mengimplementasikan sebuah aplikasi *chat* **berbasis web** yang menggunakan ECDSA (*Elliptic Curve Digital Signature Algorithm*) dan SHA-3 *hashing* untuk menjamin 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. Setiap pengguna yang mendaftarkan diri pada aplikasi akan secara otomatis menghasilkan sepasang kunci ECDSA, yaitu:

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

Kunci-kunci ini dibuat menggunakan kurva eliptik standar (seperti secp256r1 atau sejenisnya). 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* dan *hashed password*.

Aplikasi web akan menyimpan *username*, *hashed password*, 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**
2. Aplikasi web akan memverifikasi *password* (bcrypt/argon2 *comparison*),
3. Jika valid, pengguna bisa masuk dan menggunakan aplikasinya.

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

Tahap 3: Daftar Kontak

Setelah melalui tahap login, pengguna dapat memilih lawan komunikasi dengan melihat daftar kontak. Pada fitur ini, tampilkan semua **username pengguna** yang pernah mendaftarkan diri pada aplikasi kecuali diri sendiri dan pengguna dapat memilih salah satunya 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 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. **Pengemasan**
   Data yang dikirim melalui *socket* minimal mencakup
   a. Username pengirim,
   b. Username penerima,
   c. Isi pesan (plainteks),
   d. *Hash* pesan (hasil SHA-3),
   e. *Signature* (hasil signing ECDSA)
4. **Penanganan**
   Server akan menerima pesan dan:
   a. Menyimpan pesan dalam *database*/*message queue*,
   b. (**Bonus**) Mengirimkan notifikasi ke penerima secara *real-time* (WebSocket).

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

```
{
  "sender_username": "michael",
  "receiver_username": "leon",
  "plaintext_message": "Halo Leon!",
  "message_hash": "abc123...",
  "signature": {
    "r": "abc...",
    "s": "def..."
  },
  "timestamp": "2025-04-27T12:30:00Z"
}
```

Tahap 5: Pengiriman dan Verifikasi Pesan

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

1. Pesan plainteks di-*hash* ulang menggunakan SHA-3,
2. Ambil *public key* pengirim dari *storage* (yang di-*fetch* dari server),
3. 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.

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 Minggu, 18 Mei 2025 sebelum pukul 23.59**. Pengumpulan tugas yang terlambat **tidak akan memperoleh nilai** dan nilai Tugas 3 menjadi 0.
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 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).

Isi Laporan

Berikut adalah prosedur pengerjaan dari tugas ini.

1. *Cover* laporan ada foto anggota kelompok (foto bertiga). Foto ini menggantikan logo "gajah" ganesha.
2. Teori singkat (*digital signature*, ECDSA, Hash, SHA-3, dan lainnya).
3. Perancangan dan implementasi.
4. Pengujian program dan hasil analisis dengan kasus minimal mencakup beberapa hal berikut.
    a. 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.
    b. Gunakan *interceptor* seperti Burp Suite atau OWASP Zap untuk:
        - Memproxy request yang keluar masuk *client* dan *server*.
        - Intercept *request* pesan, lalu ubah isi pesan atau komponen lainnya dalam *payload*, lalu biarkan pesan sampai ke tujuan. Di sisi penerima, tunjukkan bahwa verifikasi pesan gagal.
5. Tautan kode sumber program dalam sebuah repositori Github dengan README yang berisi minimal tata cara menjalankan program dan identitas pembuat.
6. Lampiran yang berisi antarmuka program.
7. Daftar Pustaka.

"Selamat Mengerjakan!"