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

Tugas 5 IF4020 Kriptografi
Semester II Tahun 2023/2024

ALS, E2EE, & DS

Deadline	: Rabu, 22 Mei 2024 23.59
Tempat pengumpulan	: https://forms.gle/WwAdizuwoahuqnQ4A
Berkas pengumpulan	: Laporan format PDF + Kode Program + Video demo
Anggota kelompok	: 2-3 orang
QnA	: https://docs.google.com/spreadsheets/d/1csV5V3yBy5a8KoUETKMduP8B0gwJEff7vt31KtqzbBk/edit?usp=sharing

Setelah dua kali di-counter oleh Ayumi Hoshida, Conan akhirnya mencari cara lagi agar dia bisa berkomunikasi bersama Ai Haibara dengan tenang. Conan tahu bahwa selama ini Ayumi Hoshida meminta bantuan Anda untuk memecahkan pesan-pesannya. Oleh sebab itu, di kesempatan ini Conan menawari Anda untuk beralih dari Ayumi Hoshida dan membantu Anda dengan imbalan anda tidak perlu mengerjakan UAS IF4020 Kriptografi. Conan menginginkan sebuah aplikasi perpesanan yang sangat-sangat aman. Selain pesannya terenkripsi (Confidentiality), ia juga ingin memastikan bahwa pesannya benar benar berasal dari Ai Haibara (Non Repudiation) serta tidak diubah-ubah oleh Ayumi Hoshida (Integrity). Conan juga tidak mempercayai jaringan yang ia pakai mengingat mereka masih satu sekolah dan sekolah mereka tidak mampu membayar untuk TLS. Oleh sebab itu, ia ingin aplikasinya juga dapat melaksanakan hal semacam TLS.

Pada tugas 5 ini anda diminta mengembangkan sebuah aplikasi perpesanan berbasis web, desktop, ataupun mobile (bebas memilih) yang memiliki fitur *application layer security* (ceritanya semacam TLS), enkripsi ujung ke ujung (*end-to-end encryption*), dan fitur tanda tangan digital. Aplikasi bersifat tersentral (terdiri atas komponen klien dan server). Application layer security akan diimplementasikan menggunakan Elliptic-curve Diffie–Hellman (ECDH) dan Block Cipher

yang sudah kalian buat. Enkripsi ujung ke ujung diimplementasikan dengan Elliptic Curve Cryptography (ECC), sedangkan penandatanganan digital pesan dilakukan dengan menggunakan skema Schnorr.

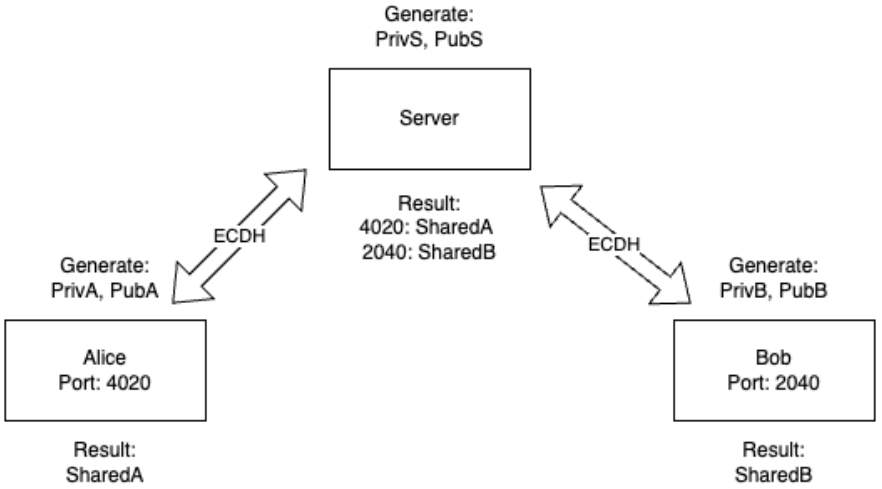
Tanda tangan digital bergantung pada isi surel dan kunci. Penyimpanan tanda tangan digital tidak harus dilakukan dengan menempelkannya (*append*) ke pesan. Namun, pengguna harus dapat melihat tanda tangan digital yang ada. Selain itu, pengguna juga dapat memverifikasi bahwa tanda tangan digital pada pesan adalah benar. Berikut adalah beberapa contoh tampilan pesan dengan tanda tangan.

Spesifikasi Wajib Aplikasi:

a. Spesifikasi Umum

1. Komponen-komponen aplikasi dapat dibuat sendiri atau **memodifikasi** proyek *open source.*
2. Bahasa dan kakas yang digunakan bebas (Java, C#, C++, Python, Java, Javascript, Golang, dll).
3. Klien memiliki fitur-fitur dasar aplikasi perpesanan seperti menulis pesan, mengirim pesan, dsb.
4. Aplikasi tidak diwajibkan memiliki fitur kirim file.
5. Tugas dibuat berkelompok max 3 orang.
6. Fungsi hash dibebaskan dan boleh memakai *library* ataupun *built-in.*

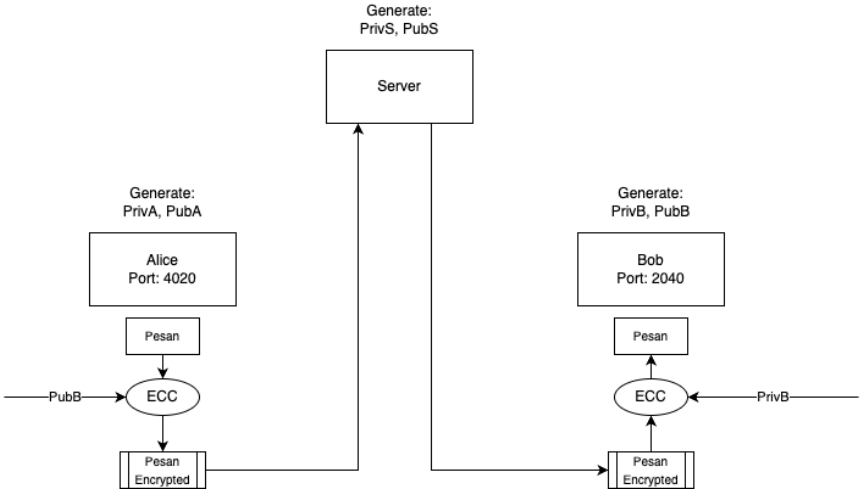
b. Spesifikasi ALS



1. Saat pertama kali klien berkomunikasi dengan server, klien dan server akan melakukan *handshake* untuk melakukan pertukaran kunci.
2. Mekanisme *handshake* dibebaskan, tidak wajib meniru seperti mekanisme *handshake* pada TLS.
3. Pertukaran kunci dilakukan menggunakan Elliptic-curve Diffie–Hellman (ECDH)

4. Setelah didapatkan *shared key* antara klien dan server, setiap klien akan mengirimkan *body request* atau server mengirimkan *body request*, *body request* akan dienkripsi menggunakan Block Cipher yang sudah kalian buat dengan key menggunakan *shared key*
5. Mode yang digunakan (seperti CBC, OFB, dst) dibebaskan
6. Tidak perlu melakukan handshake setiap kali melakukan komunikasi. Server dapat menyimpan shared key di database atau penyimpanan lain seperti stuktur data map. Di sisi lain, klien dapat menyimpan shared key di local storage. Diasumsikan bahwa antar klien menggunakan port komunikasi yang berbeda-beda, yang kelak akan digunakan sebagai *identifier*
7. Misalkan ada klien A dan server B. Klien A akan mengenkripsi body requestnya menggunakan Blok Cipher yang sudah kalian buat menggunakan shared key, dan Server B akan mendekripnya menggunakan shared key. Begitu pula sebaliknya.
8. Format body request dibebaskan, tidak ada ketentuan khusus
9. Asumsikan shared key valid sampai waktu tertentu. Jika sudah invalid, akan melakukan *handshake* ulang.

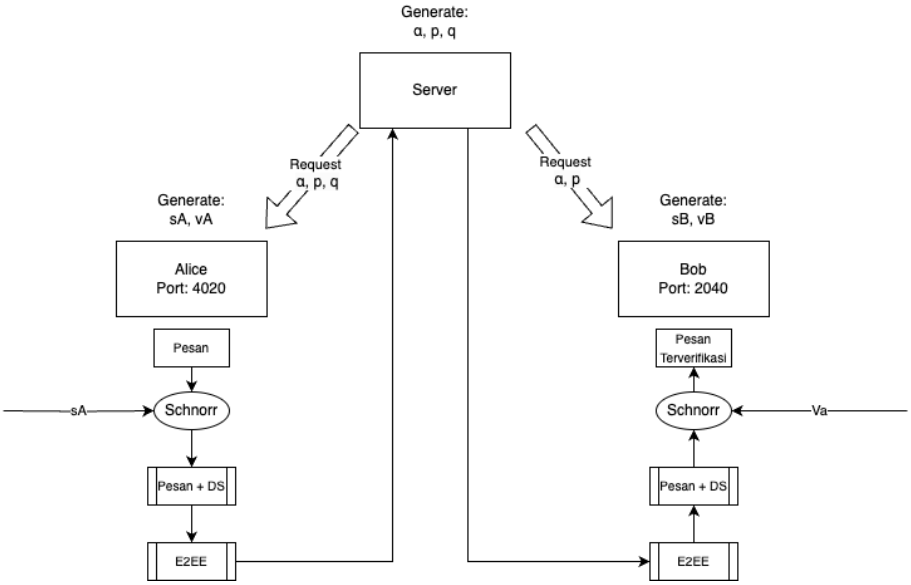
c. Spesifikasi Enkripsi Ujung ke Ujung (E2EE)



1. Klien dilengkapi dengan fungsi untuk membangkitkan kunci publik dan kunci privat berdasarkan *Elliptic Curve Cryptography.*
2. Kunci publik dan kunci privat klien dapat disimpan ke dalam file terpisah (contoh format file: .ecpub & .ecprv). Perlu diingat bahwa kunci ini digunakan untuk mengenkripsi pesan.
3. History chat dapat disimpan di database. Akan tetapi, pesan yang disimpan adalah dalam bentuk yang terenkripsi menggunakan ECC.
4. Saat ingin memulai perpesanan dengan seorang lawan bicara. Supaya enkripsi ujung ke ujung dapat digunakan, pengguna wajib diminta memasukkan **kunci private dirinya** dan **kunci publik lawan bicaranya** untuk kepentingan enkripsi

dan dekripsi**.** Perlu diingat bahwa kunci yang dimaksud adalah yang berdasarkan ECC

d. Spesifikasi Tanda Tangan Digital



1. Server memiliki fungsi untuk membangkitkan nilai α, p, dan q yang merupakan *global public key* dan menyimpannya selama server berjalan. Nilai-nilai dari ***global public key*** **dapat di-*request* oleh klien untuk dipakai dalam skema *Schnorr***.
2. Klien dilengkapi dengan fungsi untuk membangkitkan kunci publik dan kunci privat berdasarkan skema *Schnorr.*
3. Kunci publik dan kunci privat klien dapat disimpan ke dalam lokal file terpisah (contoh format file: .scpub & .scprv). Perlu diingat bahwa kunci ini digunakan untuk tanda tangan digital.
4. Pengguna dapat memilih apakah pesan dapat ditandatangani atau tidak (ada tombol kirim khusus). Jika klien memilih menandatangani pesan, maka aplikasi meminta kunci privat pengguna (yang berdasarkan skema Schnorr). Lalu aplikasi klien membangkitkan tanda tangan.
5. Pada sisi penerima, aplikasi klien dapat membedakan apakah pesan yang diterima memiliki tanda tangan atau tidak. Jika pesan memiliki tanda tangan, pengguna dapat melihat dan memverifikasi tanda tangan. Saat ingin memverifikasi tanda tangan, pengguna diminta memasukkan kunci publik (Schnorr) lawan bicaranya. Lalu klien memberitahu pengguna apakah pesan dan tanda tangan cocok.

Berikut adalah ilustrasi penggunaan ALS dan tanpa ALS pada *body request* dari sudut pandang penyadap. Ini adalah contoh dan diperbolehkan untuk dimodifikasi.

No.	Tanpa ALS	Dengan ALS

1.	{
	“username”: “Dimas”,
	“password”: “Dimas_G4nt3ng_53k4l1”
	}
		{
		“encrypted:
		“encrypted_with_your_block_cipher”
		}

2.	{
	“sender”: “Michael”,
	“receiver”: “Herman”,
	“message”: “encrypted_with_ECC”
	}
		{
		“encrypted:
		“encrypted_with_your_block_cipher”
		}

Spesifikasi Bonus Aplikasi:

a. Sha-3 (Keccak)

1. Fungsi hash menggunakan algoritma Keccak yang dibuat sendiri tanpa menggunakan library.
2. Fungsi hash ini akan digunakan pada proses tanda tangan digital

b. Cryptographically Secure Pseudorandom Generator (CSPRNG)

1. Untuk melakukan pembangkitan bilangan acak, gunakan algoritma Blum Blum Shub (BSS).
2. Algoritma Blum Blum Shub wajib dibuat sendiri tanpa menggunakan library.
3. Pembangkitan ini dapat digunakan misalnya pada saat ingin membentuk *random key*. Buatlah algoritmanya sehingga memenuhi berbagai macam kasus, misalkan ketika bilangan acak yang dibuat diharapkan merupakan bilangan prima.

Isi laporan:

1. Pranala github yang berisi kode program
2. Link google drive menuju video demo yang berisi:
   - Uji enkripsi pesan dengan memasukkan:
     1. Private key yang salah, perlihatkan bahwa pesan masih dalam bentuk acak
     2. Private key yang benar, perlihatkan bahwa pesan berhasil didekripsi.
   - Uji tanda tangan digital, perlihatkan verifikasi tanda tangan berhasil
   - Gunakan Burp Suite atau OWASP Zap untuk:
     1. Memproxy request yang keluar masuk klien dan server. Perlihatkan contoh komunikasi klien server yang terenkripsi oleh ALS.
     2. Intercept request pesan yang memiliki tanda tangan. Dekripsi body dari request (anda diperbolehkan mengeprint shared key), lalu ubah isi pesan, lalu enkripsi lagi dan biarkan pesan sampai ke tujuan. Di sisi penerima, tunjukkan bahwa verifikasi tanda tangan gagal.
3. Lampiran yang berisi:
   - Antarmuka program;

Referensi:

1. https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/37-Pembangkit-bilangan-acak-2023.pdf
2. https://sematext.com/glossary/ssl-tls-handshake/
3. https://atsign.com/resources/articles/what-is-end-to-end-encryption-e2ee/
4. https://www.cloudflare.com/learning/privacy/what-is-end-to-end-encryption/