Kriptografija i kriptoanaliza 2022./2023.
Čavrljanje koristeći enkripciju s kraja na kraj
Druga laboratorijska vježba

1 Uvod

Cilj laboratorijskih vježbi je implementacija sigurnog klijenta za čavrljanje (*chat*) temeljenog na enkripciji s kraja na kraj (E2EE) koristeći inačicu *Double Ratchet* algoritma. Protokol *Double Ratchet* koristi se u popularnoj *chat* aplikaciji Signal, a njegove inačice koriste se u mnogim drugim aplikacijama uključujući WhatsApp i Messenger. Protokol zadovoljava mnoga snažna sigurnosna svojstva uključujući unaprijednu sigurnost (*perfect forward secrecy*) i oporavak od kompromitiranja (*brake-in recovery* ili *post-compromise security*).

Double Ratchet koriste dva sudionika kako bi razmijenili enkriptirane poruke koristeći dijeljeni ključ. Svaka poruka koja je poslana ili primljena enkriptirana je jedinstvenim simetričnim ključem koji je izveden iz dijeljenog ključa koristeći funkciju za derivaciju ključa. Ovaj dio protokola naziva se *symmetric-key ratchet* i osigurava svojstvo unaprijedne sigurnosti (*perfect forward secrecy*) — kompromitirani izvedeni ključ u budućnosti ne može se iskoristiti za dekripciju poruka iz prošlosti.

U sklopu razmijene poruka sudionici također razmijenjuju Diffie-Hellman javne ključeve pomoću kojih dolaze do novih dijeljenih ključeva. Ovaj dio protokola naziva se *Diffie-Hellman ratchet* i osigurava svojstva oporavka od kompromitiranja (*brake-in recovery* ili *post-compromise security*) — protokol može uspostaviti sigurnu komunikaciju između sudionika u budućnosti čak ako je kompromitiran trenutni dijeljeni ključ.

Prilikom implementacije koristit ćete razne kriptografske primitive i sustave s kojima ćete se upoznati na predavanjima. Neke od njih su Diffie-Hellman, digitalni potpis, autentifikacijska enkripcija i funkcija za derivaciju ključa. Iz razloga što se u praksi nikako ne preporuča implementacija vlastitih kriptografskih primitiva koristit ćete kriptografske primitive dostupne u sklopu biblioteke nekog programskog jezika (vašeg izbora).

2 Algoritam Double Ratchet

U sklopu druge laboratorijske vježbe implementirat ćete klijent za čavrljanje koji se temelji na *Double ratchet* algoritmu. Opis algoritma dostupan na adresi https://signal.org/docs/specifications/doubleratchet. Poglavlja koja trebate pročitati su 1, 2 i 3. Vaša implementacija mora na ispravan način koristiti *Double ratchet* kako je opisan u poglavlju 3 **uz sljedeće izmjene i napomene**.

- Za *Diffie-Hellman ratchet* i *symmetric-key ratchet* možete koristiti `HKDF` (key derivation function based on `HMAC` message authentication code). Pravilno korištenje opisano je u poglavlju 5.2. Ako `HKDF` nije dostupan u kriptografskoj biblioteci u programskom jeziku koji ste odabrali, potrebno je koristiti neku drugu **sigurnu funkciju za derivaciju ključa**.
- Koristite kriptosustav za simetričnu enkripciju `AES-GCM` za (de)kriptiranje poruka koristeći *sending* i *receiving* ključeve kako je opisano u poglavlju 2.4.
- Zanemarite niz bajtova `AD` prilikom poziva funkcija `RatchetEncrypt` i `RatchetDecrypt`.
- Svaki klijent će na početku generirati certifikacijski objekt koji sadrži inicijalni javni Diffie-Hellman ključ. Pretpostavit ćemo da postoji povjerljiva središnja strana (npr. centralni server naše *chat* aplikacije) koja može na siguran način primati certifikacijske objekte generirane od strane klijenata. Nakon što se uvjeri u identitet klijenta, središnja strana potpisuje certifikacijski objekt i dobiveni certifikat distribuira dalje drugim klijentima koji se sada mogu uvjeriti u valjanost javnog ključa. Dobiveni javni ključ klijenti kombiniraju sa svojim privatnim da bi došli do dijeljene tajne — inicijalnog *root* ključa.
- Nasumično generirajte novi inicializacijski vektor (IV) prilikom svake enkripcije s `AES-GCM` i proslijedite ga putem zaglavlja poruke. U slučaju da funkcija/metoda za `AES-GCM` iz biblioteke koju koristite ne prima IV kao parametar, ispravljača morate uvjeriti (putem dokumentacije ili izvornog koda) da se IV unutar metode uistinu nasumično generira.
- Klijent **ne treba** biti sposoban dekriptirati poruke koje su došle neispravnim redoslijedom (poglavlje 2.6).
- Prostorna složenost algoritma (klijenta) s obzirom na korištenje ključeva treba biti $\mathcal{O}(1)$ neovisno o broju poslanih poruka. Dakle, implementacija treba „brisati“ ključeve koji se neće koristiti.

3 Programsko okruženje

Unutar `kik-lab-2.zip` datoteke nalaze se dva direktorija: `python_template` i `java_template`. Direktoriji sadrže početni kôd `messengerClient` i *unit* testove

`messengerClientTest` pisane u Python i Java programskim jezicima. U sklopu laboratorijske vježbe možete koristiti jedan od navedenih predložaka. Početni kôd sadrži specificirane metode koje trebate implementirati. **Implementacija metoda i sve pomoćne funkcije trebaju se nalaziti samo u datoteci** `messengerClient`. Molimo ne pisati kôd u drugim datotekama.

U svrhu korištenja kriptografskih primitiva morate koristiti pouzdanu kriptografsku biblioteku prema Vašem izboru. Za programski jezik Python predlažemo biblioteku `cryptography` (https://cryptography.io/en/latest/) koju možete instalirati pomoću naredbe `pip install cryptography`. Za programski jezik Java preporučamo biblioteku `Google Tink` (https://developers.google.com/tink) koja ima jednostavnije sučelje u usporedi sa standardnim *Java Cryptography Extension* okvirom te, dodatno, podržava `HKDF`. Navedenu biblioteku možete instalirati kao *Maven project dependency* koristeći konfiguraciju iz dostupnog predloška.

Aplikaciju ne morate nužno implementirati u programskom jeziku Python ili Java. U slučaju da se odlučite za neki drugi programski jezik zahtjevamo da sami napišete *unit* testove analogne postojećima. Grafičko korisničko sučelje (GUI) nije nužno implementirati i neće se dodatno bodovati.

4 Predaja i obrana

Rješenje laboratorijske vježbe `messengerClient` zajedno s dobivenim testovima `messengerClientTest` zapakirajte u `.zip` datoteku tako da naziv bude u obliku *ime-prezime-jmbag.zip*.

Laboratorijsku vježbu potrebno je **predati i zaključati** u sustavu FERKO do **10.1.2023 u 23:59**. Vježbu predajete pod „Komponente kolegija / Domaće zadaće / Double ratchet / Predaja rješenja“.

Obrana laboratorijske vježbe odvijat će se uživo i sastojat će se od pokretanja postojećih testova i moguće nekih novih, ispitivanja razumijevanja programskog kôda i ispitivanja gradiva vezanog za laboratorijsku vježbu. Zbog velikog broja studenata **molimo da unaprijed pripremite okolinu za izvršavanje kôda**.

Pitanja za laboratorijsku vježbu moguća su putem elektroničke pošte `kik@fer.hr` ili konzultacija uz prethodni dogovor putem elektroničke pošte.

Važno: Dozvoljeno je i poželjno diskutiranje mogućih pristupa rješavanju vježbe između studenata. Međutim, samu laboratorijsku vježbu studenti moraju raditi samostalno. Nastavno osoblje će provesti provjere sličnosti predanih rješenja, a ponašanje koje nije u skladu s Kodeksom ponašanja studenata FER-a ćemo prijaviti Povjerenstvu za stegovnu odgovornost studenatate odrediti dodatne sankcije u sklopu predmeta. U slučaju problema ili nedoumica prilikom izrade vježbe molimo da pravovremeno kontaktirate nastavno osoblje.