PB173 Projekt - CryMe

Aplikace **CryMe** bude sloužit k výměně zašifrovaných zpráv mezi uživateli, skrze centrální server. (Komunikaci vedeme pře centrální server, neboť při peer-to-peer spojení bz spolu nemohli komunikovat uživatelé, kteří nejsou dostupní přímo z internetu, což je v dnešní době většina.)

Hlavní rolí serveru bude správa registrovaných uživatelů, distribuce jejich veřejných klíčů a přeposílání zpráv. Zprávy půjde zasílat i když protistrana nebude připojená -- server si bude držet frontu zpráv k zaslání při dalším připojení.

Síťová komunikace bude probíhat skrze TCP spojení iniciovaná klienty. Využijeme k tomu knihovnu Asio. Samotná komunikace bude kódována vlastním jednoduchým komunikačním protokolem. K samotnému šifrování budeme využívat knihovnu mbedTLS.

Komunikace mezi klientem a serverem bude vždy probíhat šifrovaně. Iniciální část komunikace bude šifrovaná pomocí veřených klíčů (veřejný klíč serveru bude obsažen v samotné aplikaci, aby se zabránilo útoku typu MITM, a veřejný klíč uživatele bude serveru zprostředkován v první zprávě -- buď LOGIN (tedy dohledá si patřičný klíč v databázi) nebo REGISTER (rovnou přijme nový klíč)) a bude probíhat na bázi RSA4096.

Pomocí čtyčcestné challenge response pro ověření autenticity serveru i klienta dojde k ustanovení sdíleného symetrického klíče (na základně náhodných hodnot v challenge-response protokolu). Bitové reprezentace náhodných hodnot, které si vyměnili, budou zřetězeny a zahashovány SHA2-256 pro vytvoření symetrického klíče pro AES-256 se zachováním entropie z obou vstupů. Další komunikace mezi serverem a klientem bude šifrována AES-256.

Komunikace mezi klienty bude také vždy šifrovaná a to bez aktivní účasti serveru. Ten bude sloužit pouze jako prostředník pro přeposílání zašifrovaných zpráv. Klienti budou mezi sebou zprvu komunikovat pomocí veřejných klíčů (veřejný klíč protistrany si mohou vyptat od serveru), pak se ale opět přejde k symetrikcé kryptografii. Inspirováni Signal Protocolem pro zajištění forward a future secrecy využíváme Double Ratchet algoritmu k odvozování a aktualizacím sdíleného klíče.

Server si pamatuje veřejné klíče jednotlivých registrovaných uživatelů - drží databázi dvojic (pseudonym, klíč). V případě trvajícího přihlášení i TCP kanál na komunikaci s nimi (přes který se přihlašovali) a domluvený symetrický klíč. V případě LOGOUT() ukončí spojení a smaže symetrický klíč.

Komunikační protokol

Definujeme jednoduchý komunikační protokol mezi klienty a serverem.
První bajt zprávy popisuje její typ a následující bajty budou interpretovány dle typu zprávy.
Posledních 32 bajtů zprávy bude hash pro ověření integrity.

Server s nově příchozím spojení počítá, že první dvě zprávy (LOGIN nebo REGISTER a následná RESPONSE na challenge) budou zašifrovány jeho veřejným klíčem. Během této komunikace dojde k výměně informací pro dohodnutí nového symetrického klíče pro šifrování dalších zpráv daného spojení.

Druhy zpráv:

- REGISTER(pseudonym, pub_key) - iniciuje challenge response protokol pro ověření znalosti privátního klíče a následného zaregistrování klienta (uložení si pseudonymu a pub_key)
- LOGIN(pseudonym) - iniciuje challenge response protokol na oveření znalosti privátního klíče
- CHALLENGE(challenge) - zašle výzvu
- RESPONSE(response) - zašle odpověď
- SEND_TO(pseudonym, message) - zašle zprávu uživateli (od odesilatele serveru)
- RECEIVE_FROM(pseudonym, message) - zašle zprávu uživateli (od serveru příjemci)
- REQUEST_KEY(pseudonym) - odesílatel žádá o veřejný klíč uživatele
- RETURN_KEY(pseudonym, pub_key) - server zasílá veřejný klíč uživatele
- LOGOUT() - uživatel chce ukončit spojení
- ASK(pseudonym, pub_key2) - žádost od serveru příjemci (s pub_key1), že s ním chce navázat odesílatel (s puk_key2) spojení (odesílatel požádal o jeho pub_key1)

Z pohledu uživatele

- může si vygenerovat dvojici klíčů
- zaregistrovat se - poslat žádost serveru se svým pseudonymem a věřejným klíčem
- požádat server o spojení s jiným uživatelem
- posílat a přijímat zpávy, komunikovat s jiným uživatelem
- ukončit spojení se serverem
- pamatuje si uživatele, se kterými komunikoval (pseudonym, veřejný klíč)

Co chceme chránit/utajovat?

- privátní klíče uživatelů i serveru
- obsah zpráv mezi komunikujícími uživateli
- typ a obsah zprávy přenášené mezi klientem

Challenge response a symetrický klíč pro klient-server

- Klient - zná Pc (veřejný klíč), Sc (soukromý klíč), Ps
- Server - zná Ps, Ss, Pc

1. Klient vygeneruje náhodný klíč Rc pro AES a zašifruje s ním zprávu m1 obsahující jeho pseudonym a případný Pc → **Rc(m1)**, pak zašifruje **Ps(Rc)** a obojí **odešle serveru.**
2. Server dešifruje Ss(Ps(Rc))=Rc, dešifruje i zprávu m1 a tak získá pseudonym a případně i veřejný klíč (pokud jde pouze o LOGIN, tak si veřejný klíč dohledá v databázi).
3. Server vygeneruje náhodný klíč Rs pro AES a **klientovi odešle** následují zprávu, jejíž části jsou zašifrované příslušnými klíči - **Pc(Rs), Rs(Rc).**
4. Klien získá Sc(Pc(Rs))=Rs a ověří si, že Rc které mu přišlo se shoduje s tím, které na začátku posílal → **autentizace serveru**. Klient si vypočítá **nový klíč K** (zřetězením Rc a Rs a jejich následných hashováním), který bude dále sloužit k symetrickému šifrování komunikace mezi klientem a serverem. K:=sha(Rc | Rs) a odešle serveru zprávu zašifrovaou AES s využitím toho klíče K(Rs).
5. Server si také vypočítá nový klíč K:=sha(Rc | Rs) a ověří Rs, zda je totožné s původně zvoleným → **autentizace klienta**.

Komunikace Klient - Klient (případně i Klient - Server po autentizaci) bude probíhat pomocí Double Ratchet protokolu.