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 by 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. Ksamotnému 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.

Oboustranným challenge response protokolem (popsán níže) dojde k ověření autenticity serveru a klienta a k dohodnutí symetrického klíče pro AES-256, kterým bude šifrována další komunikace. Tímto vznikne šifrovaný kanál mezi klienty a serverem, skrze nějž se budou zasílat všechny zprávy.

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 komunikovat pomocí Double Ratchet protokolu, pro zajištění Perfect Forward Secrecy. Iniciální klíče budou ustanoveny protokolem X3DH.

Server si ukládá do souboru pojmenovaného unikátním pseudonymem uživatele jeho veřejný klíč, počet prekey, a pak následují dvojice id_prekey, prekey. Když uživatel přejde do režimu online, server si uloží do struktury jeho pseudonym, veřejný klíč, symetrická klíč dané komunikace. 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 spojením očekává autentizaci a dohodnutí symetrického klíče pomocí oboustranného challenge response protokolu. V případě nového uživatele může dojít zároveň i k zaregistrování (předání veřejného klíče).

Druhy zpráv:

- 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
- REQUEST_PREKEY(pseudonym) - odesílatel žádá o prekey uživatele
- RETURN_PREKEY(prekey) - server zasílá uřivateli prekey, o který si požádal
- ASK_PREKEY() - server hlásí uživateli, že mu dochází prekeys
- UPLOAD_PREKEY(prekey) - u6ivatel doplňuje na server zásobu jeho prekeys
- 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)

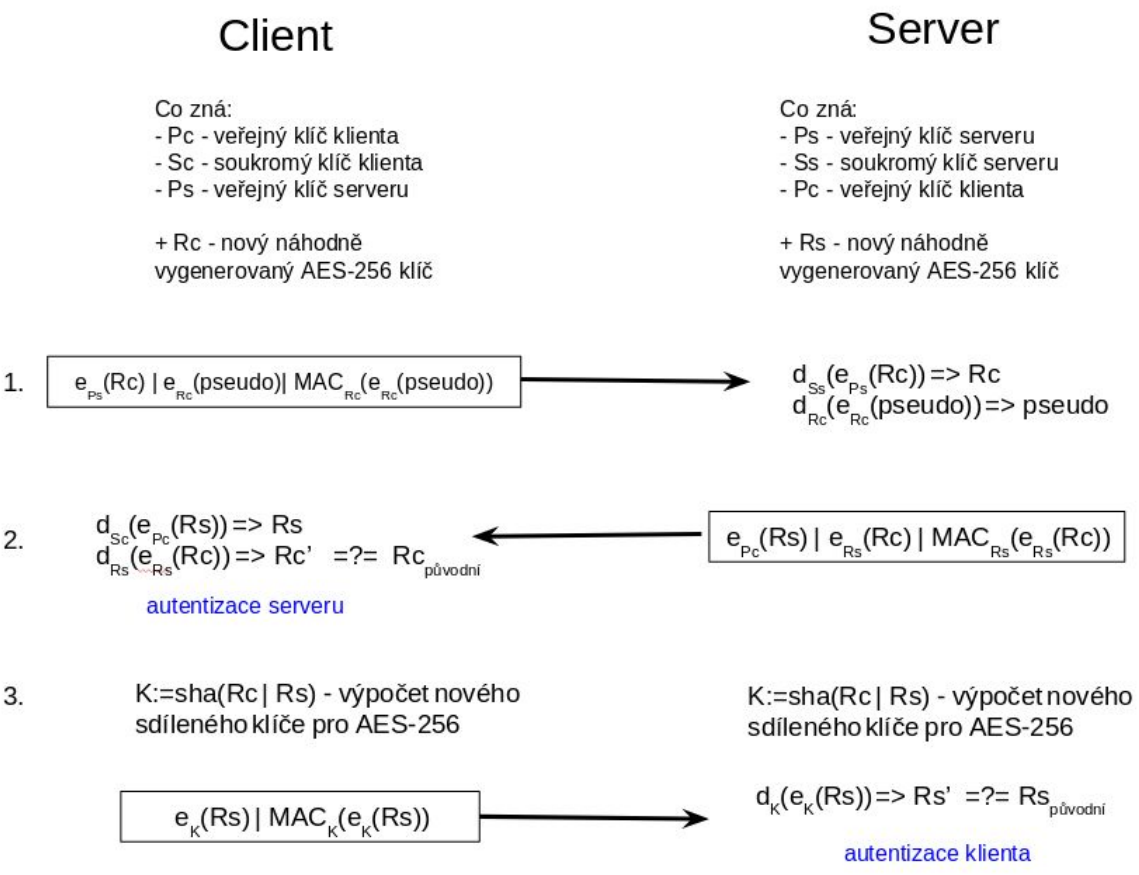
Co chceme chránit/utajovat?

- 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**.



Útoky a hrozby:

- **zcizení privátního klíče klienta/serveru**
- **nabourání se do databáze zaregistrovaných klientů**
- **Man-in-the-middle attack** - veřejný klíč serveru je přímo uložený v aplikaci, od začátku se posílají zašifrovaná data konkrétními klíči (při autentizaci a domlouvání si klíče pro symetrickou krzptografii asymetrickými), Man-in-the-middle nemá jak vzniknout
- **Replay attack** - Možný mezi klientem-serverem - Lze vyřešit zavedením počítadla přijatých zpráv a každá zpráva bude obsahovat své pořadí. Nicméně možná rizika jsou téměř zanedbatelná (neměl by to k čemu využít).
- Padding Oracle - snažíme se proti němu bránit používáním paddingu u RSA PKCS#1 v2.1
-