Instant Messaging

Filip Troníček
trof01@vse.cz

23. 11. 2025

Předmět: 4IZ110

Čas cvičení: středa, 12:45–14:15

Cvičící: RNDr. Radomír Palovský, CSc.

Obsah

Seznam obrázků

Úvod

V poslední dekádě se prokázala nezpochybnitelná dominance a univerzální využitelnost tzv. služeb pro okamžité zasílání zpráv, často nazývaných „instant messaging aplikace“. Ať už se jedná o využití v profesionálním kontextu pomocí aplikací jako je Microsoft Teams či Slack, nebo o využití osobní – například službu WhatsApp měsíčně využijí asi 3 miliardy lidí (tj. přes 50 % internetově-připojené populace (International Telecommunication Union, 2024)), zatímco čínský WeChat navštíví 1,4 miliardy uživatelů každý měsíc (We Are Social et al., 2025).

Užitečnost těchto služeb je v dnešní době očividná, a mnohé z nich vypadají z vnějšku velmi podobně. Proto je důležité pomoci existujícím či potenciálním uživatelům se správně rozhodnout, jaký komunikační protokol využít, pomocí dostupného shrnutí základních charakteristik a rozdílů v interním fungování protokolů různých „chatovacích“ služeb.

Cíle

Tato práce je rozdělena na sekci teoretickou a následnou sekci praktickou.

Cílem teoretické části textu bude popsat a kontrastovat tři vybrané protokoly, které stojí za aplikacemi na okamžité zprávy – Discord, Signal a Matrix, se zaměřením na doručování zpráv, šifrování a náhled do shromažďovaných metadat.

Cílem části praktické pak bude demonstrace provozuschopnosti protokolového modelu sítě Matrix. Konkrétně je jejím cílem nastavení Matrix serveru a následné otestování komunikace mezi dvěma klienty.

Metody

Primární metoda této práce je analýza oficiálních specifikací jednotlivých protokolů, jejich oficiální dokumentace a rešerše formálních publikací hodnotící jejich bezpečnostní vlastnosti.

V praktické části tyto metody doplňuje experimentace se serverem implementujícím Matrix protokol, zahrnující konfiguraci a prvotní nastavení homeserveru a sledování síťové komunikace přes vývojářské nástroje prohlížeče.

Teoretická část

Většina moderních komunikačních platforem sdílejí nějaké základní charakteristiky, které se takřka u všech aplikují. Těmi základními jsou:

- Protokoly jsou implementovány na aplikační vrstvě modelu OSI.
- Aplikace používají centralizovanou autentifikaci.
- Komunikace je šifrována (jak bude zřejmé v dalších kapitolách, úroveň šifrování se liší. I tímto způsobem se ale dnešní protokoly odlišují od protokolů 90. let jako byly IRC či ICQ[1]).

Discord

Aplikace Discord je relativně novým hráčem mezi messaging aplikacemi, ve kterých vyčnívá díky svým možnostem masové komunikace – na platformě jsou velmi populární tzv. servery, které shromažďují někdy statisícové komunity. Jak soukromé chaty, tak i servery poskytují možnost nejen zpráv, ale i hovorů skrz WebRTC (Vass, 2018).

Proprietární protokol, který platforma používá je kombinací požadavků podléhajícím REST (*Representational State Transfer*) obyčejům pro posílání zpráv a dlouhodobého připojení skrze WebSockets. Discord je založený na client-server komunikaci, která je z principu velmi centralizovaná.

Na rozdíl od ostatních popsaných protokolů je protokol používaný platformou Discord relativně triviální a velmi jednoduše čitelný (pro transport dat se využívá výhradně formátu JSON).

Formát dat

Jednoduchý požadavek pro zaslání zprávy do kanálu „12345“ vypadá následovně. Klíčovým polem je zde pole *content*, které obsahuje samotné tělo zprávy.

```
POST https://discord.com/api/v9/channels/12345/messages
```

```
{
    "mobile_network_type": "unknown",
    "content": "Ahoj svete",
    "nonce": "1439355065390333952",
    "tts": false,
    "flags": 0
}
```

Obrázek 1: Požadavek na server Discord obsahující tělo nové zprávy

Zdroj: vlastní zpracování

Po odeslání klient na druhé straně (nebo i odesílatel na jiném zařízení) dostane přes otevřený WebSocket kanál informace o nové zprávě, včetně jejího obsahu.

Pokud druhá strana aktivně neposlouchá, dostane zprávu až potom, co zavolá metodou GET na vypsání posledních N zpráv z kanálu. Odpověď na takový požadavek je jednoduchý JSON (*JavaScript Object Notation*) seznam a vypadá následovně:

[1] IRC posílá zprávy a veškerou jinou komunikaci přes nezašifrované TCP pakety (Oikarinen & Reed, 1993), zatímco ICQ má svoji UDP komunikaci (ve verzi 5) pouze obfuskovanou (Dault, 1998)

```
GET https://discord.com/api/v9/channels/12345/messages?limit=50
```

```
[
    {
        "type": 0,
        "content": "Ahoj svete",
        "mentions": [],
        "mention_roles": [],
        "attachments": [],
        "embeds": [],
        "timestamp": "2025-11-15T20:53:18.139000+00:00",
        "edited_timestamp": null,
        "flags": 0,
        "components": [],
        "id": "1439357618152800257",
        "channel_id": "12345",
        "author": {
            "id": "123456789",
            [...]
        },
        "pinned": false,
        "mention_everyone": false,
        "tts": false
    }
]
```

Obrázek 2: Odpověď serveru obsahující seznam zpráv v kanálu

Zdroj: vlastní zpracování

Jak je z předchozích výstřižků požadavků patrné, protokol neposkytuje možnost pro koncové šifrování (E2EE) a je tedy srovnatelný se základním protokolem platforem jako Telegram[2], Microsoft Teams (MSFTTracyP, 2025), či Slack, které všechny nechávají zodpovědnost šifrování na transportní vrstvě (Clark, 2024), které zajišťuje ochranu proti odposlechu zpráv od třetích stran.

Signal Protocol

Signal Protocol je koncově-šifrovacím protokolem, který ale podobně jako Discord závisí na centralizované client-server interakci pro doručování zpráv, využívaný v populárních instant messaging aplikacích, jako je WhatsApp, Signal Messenger, nebo Messenger[3].

[2] Telegram i přes svou velkou popularitu v posledních letech (We Are Social et al., 2025) v základu koncové šifrování neimplementuje, i když ho nabízí jako konfigurovatelnou možnost u přímých zpráv – tzv. „Secret chats“ (*End-to-End Encryption, Secret Chats*, b.r.).

[3] Zahrnutí Signal protokolu v aplikaci Messenger je poněkud novinkou, poněvadž až do roku 2023 koncově šifrovaný nebyl (Millican & Riley, 2023).

Double Ratchet Algoritmus

Jeho podstata je založena na tzv. Double Ratchet algoritmu (algoritmus dvojité západky)[4], který zaručuje oběma stranám dopřednou bezpečnost (*forward secrecy*). To znamená, že pokud je komunikace mezi dvěma stranami kompromitovaná (třetí strana se dozví sdílený klíč, který účastníci komunikace používají pro šifrování/dešifrování zpráv), útočník nemůže číst žádné zprávy, které byly napsány v minulosti, protože se každou novou zprávou klíč na šifrování mění (přičemž jsou nové veřejné klíče zasílány jako součást každé zprávy).

Na druhou stranu zaručuje Double Ratchet algoritmus také zabezpečení po kompromitaci (*post-compromise security*). To znamená, že díky rotaci klíčů dokáže zajistit to, že útočník, který dostane přístup k soukromému klíči jednoho z účastníků v nějaký moment konverzace nebude mít univerzální klíč ke vší budoucí komunikaci mezi stranami – komunikace se sama od sebe opět zabezpečí.

Sdílení klíčů (*key exchange*)

Pro začátek konverzace (a tím pádem i početí celé dvojité západky) se musí shodnout obě strany na nějakém prvotním sdíleném klíči pro šifrování. S tímto krokem pomáhá centralizovaná architektura Signal protokolu: když chce Alice (A) navázat konverzaci s Bobem (B), nejdříve požádá centrální server o klíčový materiál Boba, včetně jeho identity (neměnného klíče IK_B) jednorázového předklíče[5] pro nové konverzace (Bob jich předem vždy vytvoří několik, OPK_B^n), předklíče podepsaného jeho identitou (SPK_B), který Bob periodicky mění. Pro to, aby spojení mohlo navázat i když není kontaktovaná strana (v našem případě Bob) napojena na internet, nahraje všechen klíčový materiál nahrává na server předem[6] (Marlinspike & Schmidt, 2025).

Alice potom pomocí Diffie-Helmann algoritmu aplikovaného na klíčový materiál Boba a svých klíčů (klíče své identity IK_A a dočasným (jednorázovým) klíčem EK_B) vypočítá mnoho různých klíčů (typicky 4 nebo 5), které poté všechny zkombinuje pomocí funkce pro odvození klíčů (*KDF*)[7]. Výsledkem této výměny je tzv. sdílený tajný klíč (SK), který může Alice použít k zašifrování zprávy. Po vypočítání SK se Alice zbavuje svého EKA a v těle zaslané na server zasílá: svou identitu (IKA), svůj dočasný klíč (EKA), označení těch předklíčů Boba, které Alice použila, a nakonec první zprávu komunikace šifrovanou pomocí SK[8].

Tyto vlastnosti Signal protokolu poskytují možnost minimalismu s ohledem na metadata, které je nutné sdílet před navázáním konverzace. I tak jsou ale podstatné variace v implementaci koncových aplikací: analýzy aplikace WhatsApp v minulosti odhalily v tomto ohledu příležitosti ke zlepšení soukromí uživatelů (Dawson, 2025; Sohrab, 2025).

---

[4] Tento algoritmus je alespoň v referenční implementaci Signal Messengeru od října 2025 doplňován tzv. Sparse Post Quantum Ratchet mechanismem, který kombinuje Double Ratchet algoritmus s ML-KEM (*Module-Lattice-Based Key Encapsulation Mechanism*). ML-KEM je standardizovaný mechanismus pro vytváření sdílených klíčů založený na mřížkové kryptografii, která je odolná vůči útokům pomocí kvantových počítačů (Marlinspike & Schmidt, 2025).

[5] Termín "předklíč" (z anglického prekey) znamená, že se klíč vytváří ještě před tím, než se algoritmus pro sdílení klíčů spustí.

[6] Nová kvantově bezpečná verze originálního X3DH (*Extended Triple Diffie-Hellman*) KEX (*key exchange*) algoritmu zvaná PQXDH (*Post-Quantum Extended Diffie-Hellman*) používá navrch hodnoty $PQSPK_B$ a $PQOPK_B$, nicméně z většiny zůstává proces odvození klíčů podobný (Kret & Schmidt, 2023).

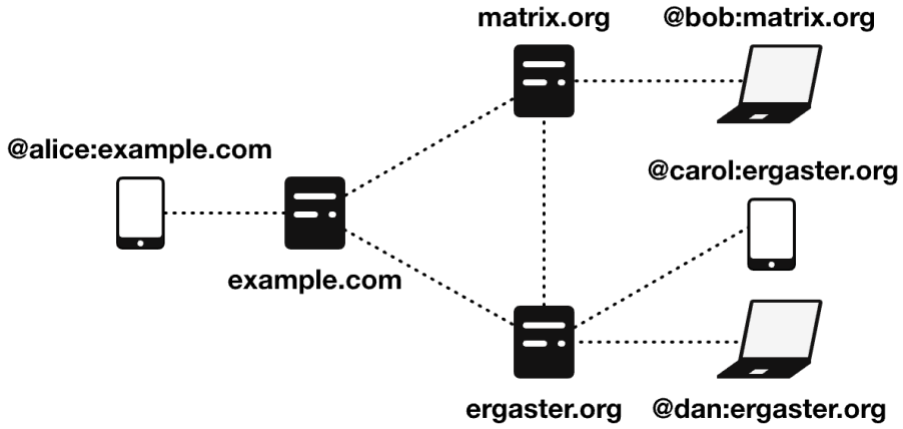
[7] V našem případě se jak pro X3DH, tak i pro PQXDH používá algoritmus HKDF (*HMAC-based Extract-and-Expand Key Derivation Function*) (Kret & Schmidt, 2023).

[8] Šifra není specifikací diktována, nicméně Signal Messenger jako šifrovací funkci používá AES-256 v režimu CBC spolu s HMAC-SHA-256 autentizací, která je ke zprávě přidána (Cohn-Gordon et al., 2019).

Matrix protokol

Protokol Matrix je otevřeným komunikačním protokolem. Od protokolů Discord a Signal se odlišuje svojí decentralizací: Matrix funguje na principu federovaných serverů, což znamená, že neexistuje žádný jeden centrální server, který by mohl ovlivňovat doručitelnost zpráv[9].

Kvůli federalizovanosti tedy Matrix funguje na dvou bázích: tou první je client-server a druhou server-server. V rámci Matrix protokolu se každý server nazývá *homeserver* a federováním dohromady vytváří síť Matrix. Federací v kontextu Matrixu se myslí to, že účastníci konverzace v jednom konverzačním kanálu nemusí být všichni pod stejným homeserverem a tím pádem má v daných ohledech podobnosti k poštovnímu protokolu SMTP (*Simple Mail Transfer Protocol*).



Obrázek 3: Schéma koncových zařízení připojených k federovaným homeserverům

Zdroj: (*Elements of Matrix*, 2023)

Ve výchozím nastavení není šifrování kanálu zapnuto a jeden z klientů ho musí spustit zasláním zprávy typu `m.room.encryption`. Tato zpráva také obsahuje šifrovací algoritmus používaný pro komunikaci (např. `m.megolm.v1.aes-sha2`). Protože mnoho informací o komunikaci jako takové musí být sdíleno mezi servery, metadata o komunikaci (jako kdo je členem jakých kanálů a kdo píše nové zprávy) jsou sdílena ve formátu JSON jako cleartext (bez šifrování) (*End-to-End Encryption Implementation Guide*, 2023).

Sdílení klíčů (*key exchange*)

Podobně jako Signal protokol používá Matrix principu západky, která je pro šifrované kanály prvotně nastavena algoritmem pro sdílení klíčů zvaným Olm. Pokud chce Alice zkontaktovat Boba, poprosí homeservery o Bobovu identitu (I_B) a jeho jednorázový předklíč (E_B), pomocí kterých v kombinaci se svou identitou (I_A) a předklíčem (E_A) vytvoří tři různé vstupy pro KDF[10]: $\mathrm{DH}_1 = ECDH(I_A, T_B)$, $DH_2 = \mathrm{ECDH}(\mathrm{E_A}, \mathrm{I_B})$ a $DH_3 = \mathrm{ECDH}(\mathrm{E_A}, \mathrm{T_B})$[11].

[9] Centralizovanost aplikací jako je Signal messenger prezentuje jednoduchou metodu opresivním politickým režimům v blokování komunikace, což je zřejmé z aplikace této praktiky v posledních letech (Bozorgmehr, 2024; Xynou & Filastò, 2021). Zároveň je ale důležité zmínit, že i blokování centralizovaných služeb není triviální - sám Signal Messenger nabízí možnost zvanou Censorship circumvention (obcházení cenzury), umožňující zasílání dat přes proxy server, zabalených tak, aby vypadala jako normální webový provoz (Whittaker, 2024).

[10] V Olm protokolu se jako funkce pro odvození klíčů používá obdobně jako u Signal protokolu HKDF-SHA-256 (van der Hoff, 2019).

[11] Matrix na rozdíl od algoritmu X3DH používaném v Signal protokolu používá pro sdílení klíčů 3DH (*Triple Diffie-Hellman*) (Ginesin & Nita-Rotaru, 2024).

Výstupem KDF je sdílený tajný klíč (S), který je poté použit jako prvotní zub západky. Olm používá stejný západkový algoritmus jako protokol Signal (*Double Ratchet*), nicméně Olm není pro samotnou komunikaci v Matrix protokolu používán (*Client-Server API*, 2023; Ginesin & Nita-Rotaru, 2024).

Šifrovaná komunikace

Pro efektivní šifrování skupinových komunikací používá Matrix algoritmus Megolm, který poskytuje možnost šifrovat zprávy pro všechny účastníky jen jednou, a tudíž obchází problém historicky problematického (Becker & Wille, 1998) sdílení klíčů ve skupinových konverzacích. Toho dosahuje šifrováním jedním klíčem, který je distribuován účastníkům přes Olm. Pro Megolm funguje zjednodušená verze západkového algoritmu, který odvozuje tajný klíč pro šifrování další zprávy hashovací funkcí aplikovanou na klíč nynější (*End-to-End Encryption Implementation Guide*, 2023) – tato vlastnost umožňuje dopřednou bezpečnost (zprávy z minulosti nelze přečíst klíčem zprávy nynější), ale znemožňuje zabezpečení po kompromitaci (van der Hoff & Hodgson, 2022).

Každá strana komunikačního kanálu si udržuje svou vlastní západku a její původní stav, kterou každou zprávou točí dopředu. Zároveň si pomatuje původní stav tajného Megolm klíče (K_0) všech ostatních účastníků, aby mohla pomocí něho derivovat klíč pro n-tou zprávu pro arbitrárně zvolené n: $K_{i+1} = H(K_i)$, kde K je klíč a H je hashovací funkce[12].

V minulosti obdržel protokol Matrix kritiku za bezpečnostní díry (Albrecht et al., 2023), přehnané sdílení cleartextových metadat a problémy s adopcí změn, zhoršené decentralizací jeho architektury. I tak se ale v posledních letech v Matrix protokolu pracuje na mnoha klíčových zlepšení: jednou z nich je experimentace s novým skupinovým šifrovacím standardem MLS (*Messaging Layer Security*) (Hodgson & Chathi, 2023) nebo p2p (peer-to-peer) architekturou (Hodgson, 2020).

[12] Tento popis je z části zjednodušený: algoritmus na hashování je pro škálovatelnost optimalizována přes 4 záchytné body tak, aby nikdy nepotřebovala více, než 1020 hashových výpočtů (Ginesin & Nita-Rotaru, 2024).

Praktická část

Díky decentralizované architektuře Matrix protokolu je jedním z jeho cílů jednoduchá operace homeserverů, jehož nasazení se věnuje praktická část této práce.

Výběr serveru

Matrix nabízí, podobně jako pro svoje klientské aplikace, mnoho možností při výběru softwaru na hostování homeserveru. Mezi populární možnosti pro provoz Matrix serveru patří implementace protokolu jako je Synapse (referenční implementace napsaná v jazyce Python, používaná na provoz matrix.org (Edge, 2020)), Dendrite (napsaný v jazyce Go), či Conduit (napsaný v jazyce Rust, finančně podporován německou vládou (Meenzen, 2024) a jednoduchý na nastavení). Pro účely demonstrace nasazení byl zvolen server Tuwunel, který je forkem serveru conduwuit (který je sám forkem serveru Conduit).

Nasazení

Na jednoduché nastavení na virtuální Linux server byla vybrána virtualizace přes Docker a jeho rozšíření Docker Compose. Docker Compose umožňuje deklarativní nastavení Docker kontejnerů přes soubor `docker-compose.yaml`, který pro náš případ vypadá takto:

```yaml
services:
  tuwunel:
    image: jevolk/tuwunel:latest
    restart: unless-stopped
    ports:
      - "8008:6167"   # Client-Server API
      - "8448:6167"   # Server-Server API
    volumes:
      - tuwunel_data:/var/lib/tuwunel # Databáze
      - ./tuwunel.toml:/etc/tuwunel.toml:ro # Konfigurační soubor (viz níže)
    environment:
      TUWUNEL_SERVER_NAME: localhost
      TUWUNEL_DATABASE_PATH: /var/lib/tuwunel
      TUWUNEL_PORT: 6167
      TUWUNEL_MAX_REQUEST_SIZE: 20000000
      TUWUNEL_ALLOW_REGISTRATION: 'true'
      TUWUNEL_ALLOW_FEDERATION: 'true'
      TUWUNEL_ALLOW_CHECK_FOR_UPDATES: 'true'
      TUWUNEL_TRUSTED_SERVERS: '["matrix.org"]'
      TUWUNEL_ADDRESS: 0.0.0.0
      TUWUNEL_CONFIG: '/etc/tuwunel.toml'

volumes:
```

```
  tuwunel_data:
```

Obrázek 4: Konfigurační soubor docker-compose.yaml pro server Tuwunel

Zdroj: vlastní zpracování

Dále je potřeba dodat konfigurační soubor serveru, který jsme namapovali na virtuální disk kontejneru. Zde nastavíme cestu k databázovému souboru SQLite, port služby, název serveru a několik dalších polí. Důležité je zmínit pole povolující registraci novým uživatelům – `allow_registration` a `registration_token`, jehož hodnotu budeme potřebovat při registraci uživatelů.

```
[global]
server_name = "localhost"
database_path = "/var/lib/tuwunel"
port = 6167
address = "0.0.0.0"

max_request_size = 20_000_000
allow_registration = true
registration_token = "testovaci_token"
allow_federation = true
trusted_servers = ["matrix.org"]
```

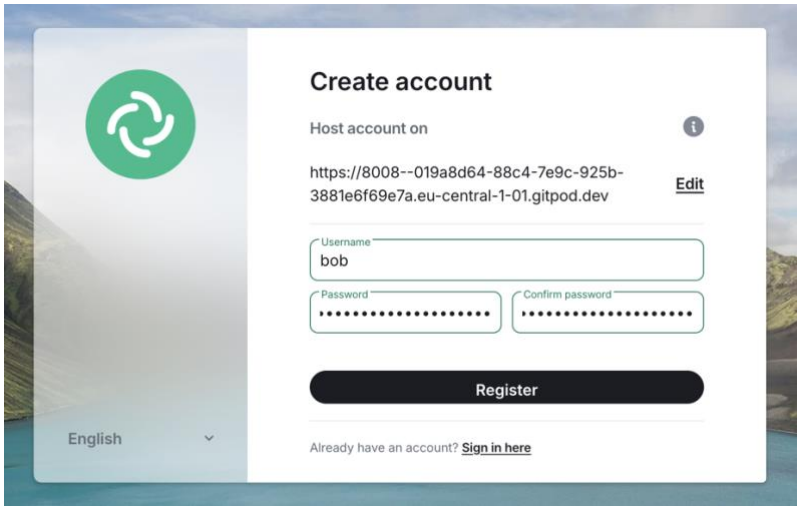
Obrázek 5: Specifický konfigurační soubor serveru Tuwunel, tuwunel.toml

Zdroj: vlastní zpracování

Po vytvoření těchto dvou souborů stačí jen zavolat Docker Compose příkazem `docker compose up -d`, který server spustí.

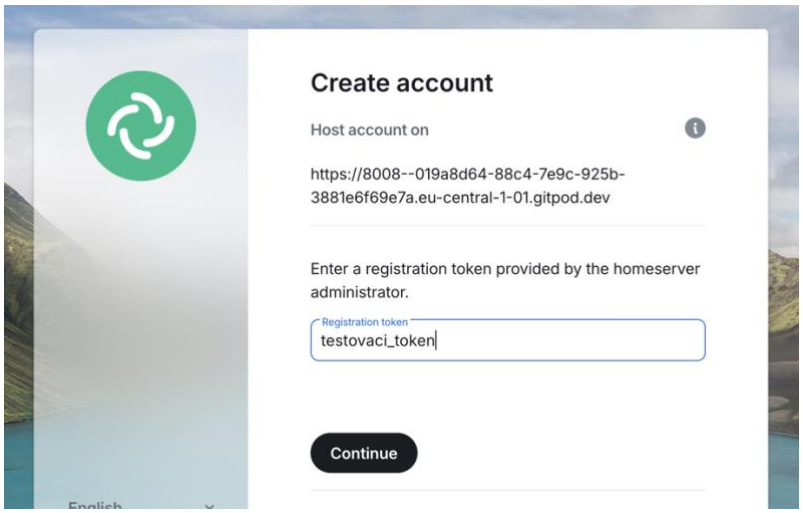
Vytváření účtů

Pro účely demonstrace vytvoříme dva účty přes webového klienta (vybrán byl otevřený webový klient Element (https://element.io)), který byl nastaven na komunikaci s naším nově spuštěným serverem. Po registraci a zadání registračního tokenu z konfiguračního souboru jsme přesměrováni na chatovací rozhraní aplikace.



Obrázek 6: Proces vytváření uživatelského účtu přes Matrix klient Element

Zdroj: vlastní zpracování

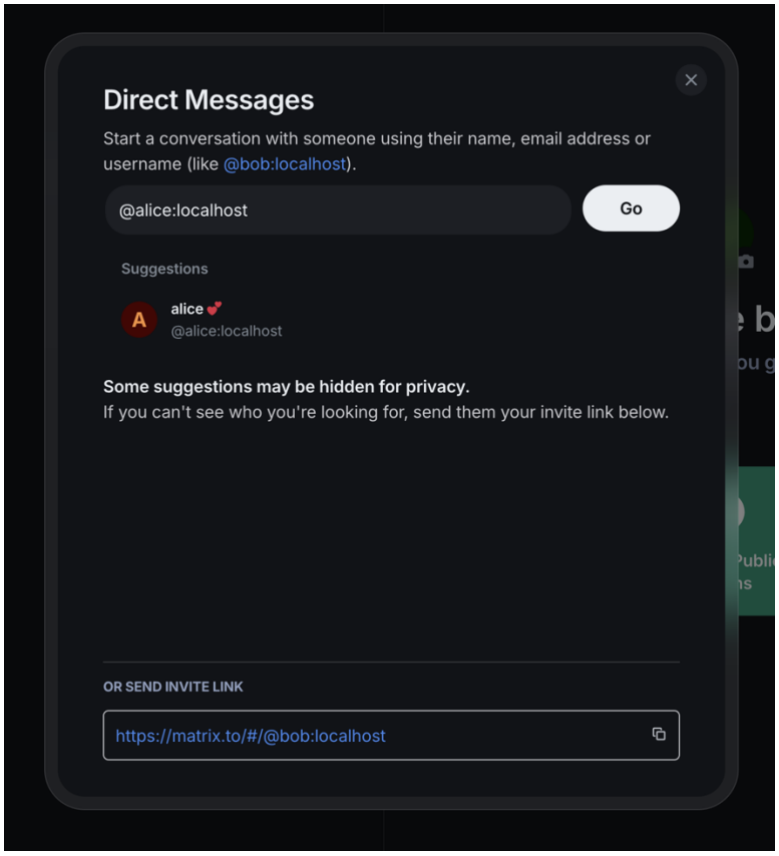


Obrázek 7: Zadávání registračního tokenu

Zdroj: vlastní zpracování

Zasílání zpráv

Po úspěšném vytvoření obou uživatelských účtů a přihlášení do účtu Boba máme přístup k prostředí klienta Element. Po zadání validního uživatelského jména se z účtu Boba najde účet Alice a po poslání zprávy se jim vytvoří společný kanál.



Obrázek 8: Dialog s výběrem uživatele pro začátek nové konverzace

Zdroj: vlastní zpracování

Bob nejdříve pošle dotaz na kryptografické klíče Alice:

`POST /_matrix/client/v3/keys/query`

Ten jí vrátí objekt s poli device_keys, master_keys a self_signing_keys:

```
{
    "device_keys": {
      "@alice:localhost": {
        "7RGxEvaoGS": {
          "keys": {
            "curve25519:7RGxEvaoGS":
"orVJVAUC/KNmNCHiw4tkXstz6t7mjR8jbJdlp7IuvSE",
            "ed25519:7RGxEvaoGS": "tvbqXGCWyFp2pT11H4uzth6EKE9A6gBI9wOc5GR/C6g"
          },
          "signatures": {...}
        }
      }
    },
    "master_keys": {...},
    "self_signing_keys": {...}
  }
```

Obrázek 9: Odpověď s persistentními klíči Alice

Zdroj: vlastní zpracování

Po vytvoření kanálu a poslání pozvánky Alici se Bob dotáže na dočasný klíč Alice, aby mohl provést proces 3DH.

`POST /_matrix/client/v3/keys/claim`


```
{
    "one_time_keys": {
        "@alice:localhost": {
            "7RGxEvaoGS": "signed_curve25519"
        }
    },
    "timeout": 10000
}
```


Obrázek 10: Požadavek na jednorázový klíč Alice

Zdroj: vlastní zpracování

Na což dostane odpověď s klíčem a jeho podpisem:

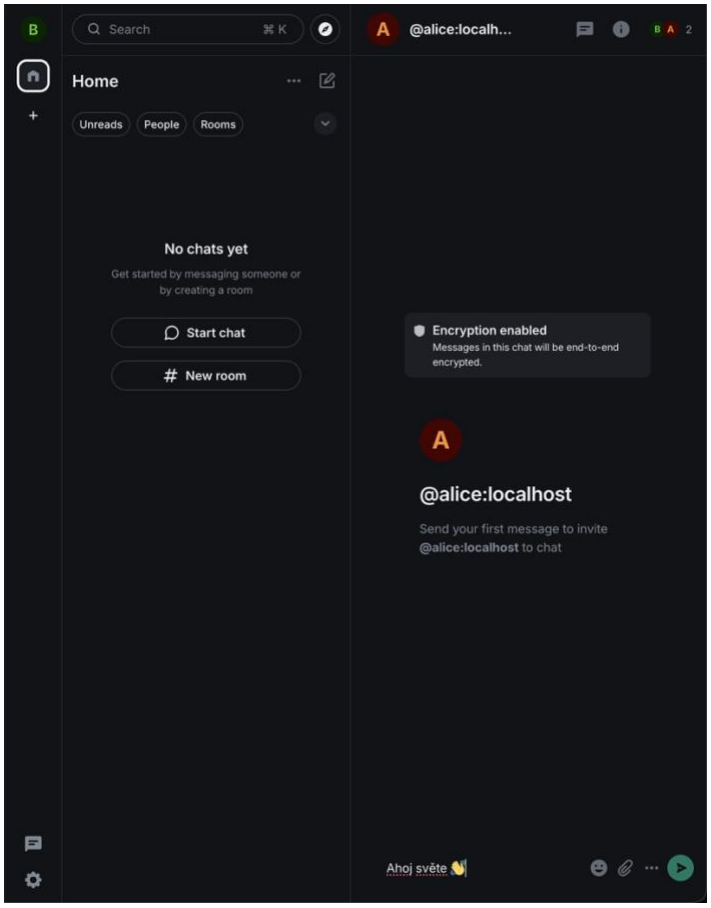

```
{
    "failures": {},
    "one_time_keys": {
        "@alice:localhost": {
            "7RGxEvaoGS": {
                "signed_curve25519:AAAAAAAAAA0": {
                    "key": "r67e2NhWDx7/kuaGITBYyMzlUOVB1zcJglMgoS3BX0Q",
                    "signatures": {
                        "@alice:localhost": {
                            "ed25519:7RGxEvaoGS":
"dgGLUsRAOnCD18JAS3sGD4Z7zgwgSwI+LM2kTg6JH0AuyH8BmQpQiahCBX1+/8R/Np6sLWQYlACwB5Lrj
Xp8AQ"
                        }
                    }
                }
            }
        }
    }
}
```


Obrázek 11: Odpověď s jednorázovým klíčem Alice

Zdroj: vlastní zpracování

Bob poté zavolá na API metodu pro vytváření nových kanálů (POST `/_matrix/client/v3/createRoom`), která vrátí identifikátor kanálu ve formátu JSON.


```
{
    "room_id": "!7QLmzhj6xnOfHwDkCF:localhost"
}
```



Obrázek 12: Textové okno s připravenou zprávou pro Alici

Zdroj: vlastní zpracování

Po vytvoření sdíleného tajného Olm klíče pro kanál ho použije k zašifrování nově náhodně vygenerovaného klíče místnosti (*room key*), který Alici pošle zavoláním na `/_matrix/client/v3/sendToDevice/m.room.encrypted/{ID kanálu}`. Zprávu teď může zašifrovat a odeslat:

`PUT /_matrix/client/v3/sendToDevice/m.room.encrypted/{ID kanálu}`

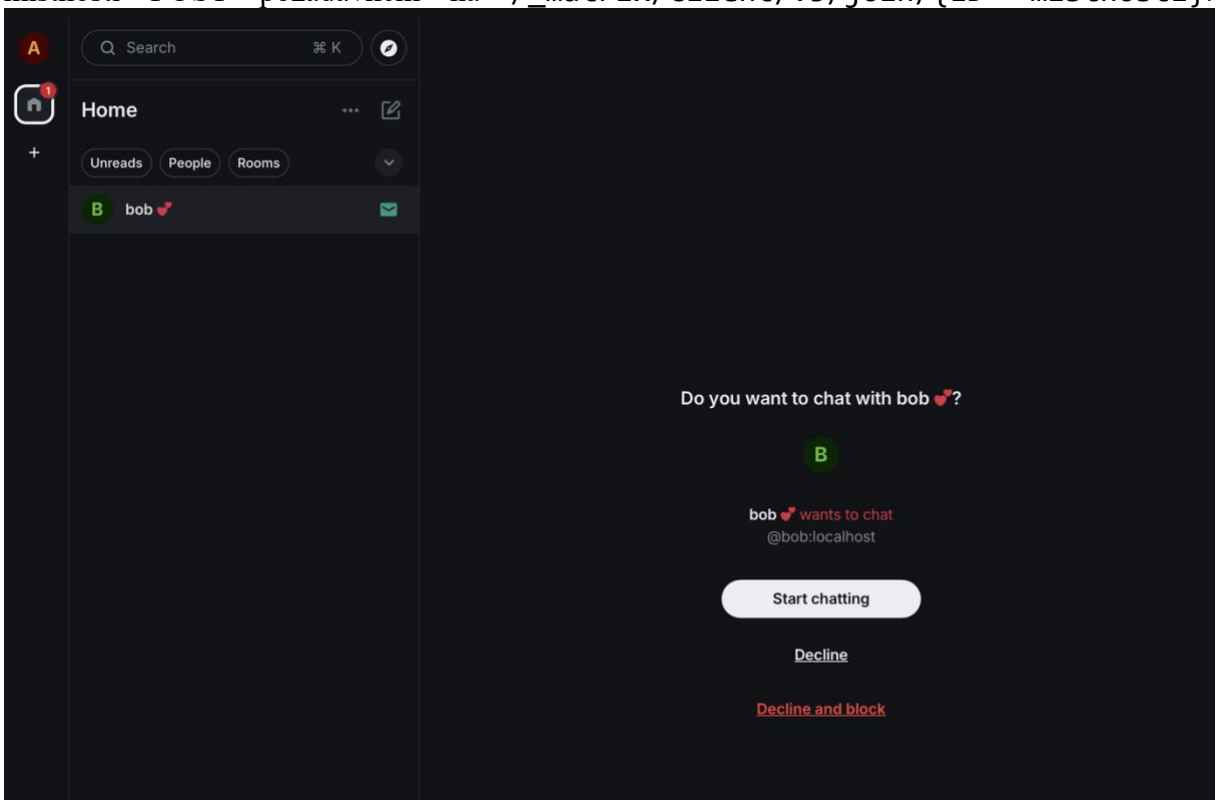
```
{
    "messages": {
        "@alice:localhost": {
            "7RGxEvaoGS": {
                "algorithm": "m.olm.v1.curve25519-aes-sha2",
                "ciphertext": {
                    "orVJVAUC/KNmNCHiw4tkXstz6t7mjR8jbJdlp7IuvSE": {
                        "body": "Awogr67e2NhWDx….[zkráceno pro ilustraci]",
                        "type": 0
                    }
                },
                "org.matrix.msgid": "a6a6e3b22f9948a484159abb25360746",
                "sender_key": "LkcI03ej/4MnPEcdgUGTGNa6WpXOxGTz9ES8ZVi+w20"
            }
        }
```

```
    }
}
```

Obrázek 13: Požadavek s prvotní zašifrovanou zprávou na server Matrixu

Zdroj: vlastní zpracování

Na druhé straně (z pohledu Alice) musí Alice nejdříve přijmout pozvánku do Bobem-vytvořené místnosti POST požadavkem na `/_matrix/client/v3/join/{ID místnosti}`.



Obrázek 14: Domovská stránka Matrix klienta Alice

Zdroj: vlastní zpracování

Alice se obdobně jako Bob homeserveru Boba zeptá na jeho šifrovací klíče, které použije k ověření identity odesílatele. Homeserveru se také zeptá na zprávy v místnosti požadavkem GET na `/_matrix/client/v3/rooms/{ID kanálu}/messages`: v odpovědi vidíme nejen naši zprávu, ale i původní pozvánku do místnosti od Boba a jeho požadavek na zapnutí šifrování na komunikaci.


```
{
    "start": "571",
    "end": "380",
    "chunk": [
        {
            "content": {
                "algorithm": "m.megolm.v1.aes-sha2",
                "ciphertext":
"AwgAEpABTFud7FBifiZE5PMtOUpOAta/yqB+BQ+KeTwnSiPPNU...",
                "device_id": "VIfJMazsIp",
                "sender_key": "LkcI03ej/4MnPEcdgUGTGNa6WpXOxGTz9ES8ZVi+w20",
```

```
                "session_id": "13n4jU9uwX7I2gLpm+2OwXL1oRThMx+njXPBQfCNLvE"
            },
            "event_id": "$s121XnnSE2nfyD-v3lA0NkkOqiKQOuq_RFUCmHEBcmc",
            "sender": "@bob:localhost",
            "type": "m.room.encrypted"
        },
        {
            "content": {
                "algorithm": "m.megolm.v1.aes-sha2"
            },
            "sender": "@bob:localhost",
            "type": "m.room.encryption"
        },
        {
            "content": {
                "join_rule": "invite"
            },
            "sender": "@bob:localhost",
            "type": "m.room.join_rules"
        }
    ]
}
```


Obrázek 15: Požadavek na Matrix server žádající o zapnutí šifrování místnosti

Zdroj: vlastní zpracování

Závěr

Teoretická část práce poskytla informativní přehled o všech probíraných komunikačních protokolech, včetně formátů dat, podle kterých jsou data přenášena přes síťové rozhraní. V rámci této části se mi nepovedlo zachytit síťovou komunikaci protokolu Signal, převážně protože aplikaci Signal messenger, podle kterého jsem Signal protokol popisoval, chybí webové rozhraní a nativní aplikace pro macOS zabraňuje přístup k vývojářským nástrojům.

Praktická část nabídla možnost ověřit teoretické poznatky v reálném sandboxovém prostředí. Úspěšným nasazením Matrix-kompatibilního serveru Tuwunel pomocí Docker virtualizace se podařilo demonstrovat, že zprovoznění federované a koncově šifrované komunikační platformy je proveditelné i s minimálními nároky na infrastrukturu.

Analýza síťového provozu při zaslání zprávy následně potvrdila popis protokolu (včetně sdílení klíčů a zasílání zpráv) z teoretické části. Práce tak splnila svůj cíl komparace architektury vybraných protokolů a praktického ověření decentralizovaného modelu sítě Matrix. Výsledky ukazují, že výběr komunikačních platforem není jednoznačný a neexistuje jedna „správná“ a univerzálně aplikovatelná možnost pro všechny uživatele.

Použitá literatura

Albrecht, M. R., Celi, S., Dowling, B., & Jones, D. (2023). Practically-exploitable Cryptographic Vulnerabilities in Matrix. *2023 IEEE Symposium on Security and Privacy (SP)*, 164–181. https://doi.org/10.1109/SP46215.2023.10351027

Becker, K., & Wille, U. (1998). Communication complexity of group key distribution. *Proceedings of the 5th ACM conference on Computer and communications security*, 1–6. https://doi.org/10.1145/288090.288094

Bozorgmehr, N. (2024, prosinec 24). Iran lifts ban on WhatsApp and Google Play. *Financial Times*. https://www.ft.com/content/bb68c686-89d2-4305-bd1e-94452795809c

Clark, K. (2024, říjen 30). *Security at Slack: How Slack Protects Your Data*. Slack. https://slack.com/blog/collaboration/security-at-slack-how-slack-protects-your-data

Client-Server API. (2023, říjen 3). Matrix Specification. https://spec.matrix.org/v1.16/client-server-api/

Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., & Stebila, D. (2019). *A Formal Security Analysis of the Signal Messaging Protocol*. 7.

Dault, S. (1998, červenec 21). *ENCRYPTION and CHECKCODE of the ICQ Protocol V5*. https://web.archive.org/web/20030213020108/http://www.algonet.se/~henisak/icq/encrypt-v5.txt

Dawson, S. (2025, říjen 31). *Whatsapp App Review 2025: Privacy, Pros and Cons, Personal Data*. Mozilla Foundation. https://www.mozillafoundation.org/en/nothing-personal/whatsapp-privacy-review/

Edge, J. (2020, listopad 4). A Matrix overview. *LWN.Net*. https://lwn.net/Articles/835880/

Elements of Matrix. (2023, únor 8). Matrix. https://matrix.org/docs/matrix-concepts/elements-of-matrix/

End-to-End Encryption implementation guide. (2023, únor 8). https://matrix.org/docs/matrix-concepts/end-to-end-encryption/

End-to-End Encryption, Secret Chats. (b.r.). Telegram. Získáno 15. listopad 2025, z https://core.telegram.org/api/end-to-end

Ginesin, J., & Nita-Rotaru, C. (2024). *The Matrix Reloaded: A Mechanized Formal Analysis of the Matrix Cryptographic Suite* (No. arXiv:2408.12743). arXiv. https://doi.org/10.48550/arXiv.2408.12743

Hodgson, M. (2020, červen 2). *Introducing P2P Matrix*. Matrix. https://matrix.org/blog/2020/06/02/introducing-p2p-matrix/

Hodgson, M., & Chathi, H. (2023, červenec 18). *A giant leap forwards for encryption with MLS*. Matrix. https://matrix.org/blog/2023/07/a-giant-leap-with-mls/

International Telecommunication Union. (2024). Measuring digital development—Facts and Figures 2024. *ITU*. https://www.itu.int/hub/publication/d-ind-ict_mdd-2024-4/

Kret, E., & Schmidt, R. (2023). *The PQXDH Key Agreement Protocol*.

Marlinspike, M., & Schmidt, R. (2025, listopad 4). *The Double Ratchet Algorithm*. Signal Messenger. https://signal.org/docs/specifications/doubleratchet/

Meenzen, S. (2024, březen 13). *Introduction—Conduit*. https://docs.conduit.rs/

Millican, J., & Riley, R. (2023, prosinec 7). Building end-to-end security for Messenger. *Engineering at Meta*. https://engineering.fb.com/2023/12/06/security/building-end-to-end-security-for-messenger/

MSFTTracyP. (2025, červenec 22). *Security guide for Microsoft Teams overview—Microsoft Teams*. Microsoft Learn. https://learn.microsoft.com/en-us/microsoftteams/teams-security-guide

Oikarinen, J., & Reed, D. (1993). *Internet Relay Chat Protocol* (Request for Comments No. RFC 1459; s. 57). Internet Engineering Task Force. https://doi.org/10.17487/RFC1459

Sohrab, K. (2025). Privacy in the Age of Digital Surveillance: Analyzing WhatsApp's Policy and Cybersecurity Implications. *Journal of Information Systems Engineering and Management*, *10*, 937–965. https://doi.org/10.52783/jisem.v10i40s.7543

van der Hoff, R. (2019, listopad 8). *Olm: A Cryptographic Ratchet*. GitLab. https://gitlab.matrix.org/matrix-org/olm/-/blob/master/docs/olm.md

van der Hoff, R., & Hodgson, M. (2022, září 1). *Megolm group ratchet*. GitLab. https://gitlab.matrix.org/matrix-org/olm/-/blob/master/docs/megolm.md

Vass, J. (2018, září 10). *How Discord Handles Two and Half Million Concurrent Voice Users using WebRTC*. Discord Blog. https://discord.com/blog/how-discord-handles-two-and-half-million-concurrent-voice-users-using-webrtc

We Are Social, DataReportal, & Meltwater. (2025, říjen 15). *Most popular global mobile messenger apps as of October 2025, based on number of monthly active users (in millions)*. Statista. https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/

Whittaker, M. (2024, srpen 9). *Proxy Please: Help People Connect to Signal*. Signal Messenger. https://signal.org/blog/proxy-please/

Xynou, M., & Filastò, A. (2021, říjen 21). *How countries attempt to block Signal Private Messenger App around the world*. https://ooni.org/post/2021-how-signal-private-messenger-blocked-around-the-world/