# Maturitní program

*Filip Ježek*

<https://bitbucket.org/filipjezek/chat/src/master/>

## Stručný popis

Jedná se o anonymní webovou chatovací aplikaci. Uživatel se připojí na server, zvolí si jednorázovou přezdívku a může buď

* vytvořit novou místnost
* připojit se k existující místnosti

K místnosti se připojuje na základě jejího id (při zakládání místnosti lze id specifikovat). Seznam existujících místností není nikde viditelný.

Anonymita je nadále zajištěna:

* duplicitní přezdívky uživatelů se rozlišují pouze v měřítku jednotlivých místností
* místnost lze uzamknout heslem
* připojení k serveru je zabezpečené (HTTPS, HSTS)
* po 20 minutách, kdy je místnost opuštěná, se sama smaže
* rovněž uživatelé se po určité době neaktivity mažou
* chat je E2E šifrovaný pomocí Signal protokolu. To má dvě výhody:
  + zprávy nejsou pro útočníky dostupné ani když je server kompromitovaný
  + zprávy se šifrují na míru konkrétním účastníkům. Když se připojí někdo nový, staré zprávy neuvidí. Zprávy se šifrují i pro uživatele, kteří se z konverzace sami odpojili (nebyli vykopnuti) pro případ, že by se připojili nazpátek
* z místnosti může její zakladatel vykopnout jiné uživatele. To vždy vyústí ve změnu zámku místnosti
* v místnosti je možné “šeptat” jednotlivým uživatelům
* jednotlivé taby jsou od sebe odděleny - nemůžete mít stejného uživatele ve dvou tabech naráz
* uživatel si dále může místnost lokálně přizpůsobit:
  + lze nastavit jméno tabu
  + lze vypnout/zapnout zvuky
  + lze vypnout/zapnout scrollování dolů při nové zprávě
  + lze nastavit z několika předvoleb ikonku
  + lze vypnout/zapnout blikání ikonky při nové zprávě (pouze pro výchozí ikonku)
  + nastavení se ukládá zvlášť pro každou místnost

Nevýhody:

* vzhledem k naprosté anonymitě uživatelů nelze nijak ověřit, s kým uživatel chatuje
* při kompromitaci serveru sice nelze dešifrovat již poslané zprávy (ani pokračující konverzace), ale kdyby útočník nahradil soubory frontendu, není jak se bránit

Různé:

* kromě klasických zpráv se v chatu objevují i notifikace (připojení člena, změna zámku…)
* každý člen dostane přidělenou barvu vygenerovanou ze svého nicku
* u zpráv i notifikací lze zjistit přesný čas odeslání - buď kliknutím na zprávu, nebo umístěním kurzoru nad notifikaci

## Technologie

* frontend:
  + Angular 9
  + NgRx 9
  + Socket.IO
* backend:
  + Node.js
  + Express.js
  + Socket.IO
* encryption:
  + Signal protocol

## Existující alternativy

Pochopitelně nápad zabezpečeného chatu není nic nového. Každá z následujících služeb vyvinula vlastní šifrovací protokol, který je opensource.

### [Telegram](https://telegram.org/)

Chatovací aplikace ukládající šifrovaná data do cloudu. Podporuje posílání fotek, souborů, hovory, online status, sledování zobrazení zpráv i skupinové konverzace. K identifikaci uživatele se používá telefonnní číslo. Běžný chat je šifrovaný symetricky, ale je zde možnost secret-chatu, který využívá e2e encryption a umí nastavit autodestrukci zpráv po určité době. Skupinové zprávy e2e šifrovat Telegram neumí. Celkově aplikace funguje jako (jak sami říkají) náhrada jak SMS, tak emailu. Je dělaná na “nekonečné” konverzace typu Messenger.

### [Signal](https://signal.org/)

Signal je hodně podobný Telegramu, pár rozdílů se ale najde. Data do cloudu neukládají, zůstávají v zařízeních klientů. Skupinové konverzace jsou e2e šifrované vždy, stejně jako standardní chat. [Signal protokol](https://signal.org/docs/) je v tomto ohledu asi největším průkopníkem v oboru. Už ho implementoval i WhatsApp a Messenger (do soukromých chatů), je považován za velmi bezpečný.

### [Cyph](https://www.cyph.com/)

Cyph není zdaleka tolik známá platforma jako předchozí dvě. Umožňuje však “burner” chat bez nutnosti registrace uživatelů. Protokol, který používá, [Castle](https://www.cyph.com/castle), se Signal protokolem do značné míry inspiroval. Je do něj zakomponována ochrana před kvantovými počítači (dosud poněkud experimentální oblast kryptografie), ale kvůli ukládání dat do cloudu byla zavržena forward secrecy (to znamená, že stačí prolomit jeden klíč a útočník je schopen přečíst celou konverzaci - Signal během konverzace derivuje klíče neustále nové pro každou zprávu) (neplatí pro burner chat). Zároveň Cyph vytvořil unikátní technologii, kterou se snaží zabránit útočníkovi s přístupem k serveru přepsat skripty frontendu - na frontendu běží service worker, který zkontroluje digitální podpis souborů, než je stáhne. Podpis vzniká pomocí speciálního hardware odděleného od sítě. Podle [reportu Cure 53](https://cure53.de/pentest-report_cyph.pdf) je ovšem bezpečnost tohoto přístupu sporná. Další zajímavou věcí je autentifikace uživatelů (aby měl uživatel jistotu, že si píše se správným člověkem, většinou implementované pomocí telefonních čísel a QR kódů). Při registraci nového uživatele (neplatí pro burner chat) se vygeneruje dlouhodobý certifikát, který musí schválit osobně operátor serveru (zase je na to hardware). To sice trvá docela dlouho, ale stačí to udělat jednou a odpadá pak nutnost scházet se s adresátem a porovnávat QR kódy ve společné konverzaci (což je mimochodem krok, který spousta lidí přeskakuje, nicméně vystavují se tak určitému riziku). Cyph si nechal svůj protokol Castle, kontrolu skriptů WebSign a spoustu dalších věcí patentovat, což se zas tak často nevidí.

## Implementace

### Architektura

Frontend je napsaný v Angularu. Člení se do komponent (např. textbox, dialog, header…), každá komponenta má své vlastní html, css a javascript. Globální state aplikace je zpracováván pomocí NgRx. Znamená to, že informace, které jsou důležité na víc místech, se ukládají do globálního readonly *storu*, ke kterému má přístup kterákoli komponenta chce. Pokud je potřeba informace změnit, vyšle se z komponenty *action*. Tu zachytí globální *reducer* (to je funkce, která na základě obsahu a typu akce upravuje *store*), který *store* upraví. Komponenty jsou o změně informací notifikovány. Zajímavé je taky routování - web je tzv. Single Page Application, takže přechody mezi jednotlivými částmi webu se obejdou bez viditelného načítání.

Komunikace s jednoduchým serverem probíhá přes websockety - to je dvousměrný komunikační kanál, což umožňuje serveru třeba notifikovat frontend o přijaté zprávě.

### Stručný náhled šifrování

Šifruje se end-to-end. Příjemce i odesílatel se potřebují domluvit na šifrovacím klíči, musí to ale udělat tak, aby útočník, který jejich komunikaci sleduje, klíč nezískal. Lze to udělat pomocí poměrně zajímavé matematiky - úplně nejjednodušší je [Diffie-Hellman](https://en.wikipedia.org/wiki/Diffie–Hellman_key_exchange). [Signal protokol](https://signal.org/docs/) je nicméně podstatně složitější, kromě iniciálního klíče totiž postupně vytváří spoustu dalších a dosahuje tím forward secrecy (jeden prolomený klíč dešifruje jen jednu zprávu) a deniability (zpětně lze do konverzace vložit další platné zprávy (v průběhu ne!), tím se vyvrací jakékoli soudní stíhání za obsah případných dešifrovaných zpráv). Také umožňuje výměnu zpráv v asynchronním prostředí, kde se zprávy kvůli síti mohou zpozdit nebo ztratit.

Skupinový chat funguje tak, že se zpráva symetricky zašifruje (tzn. není to e2e). Klíč k dekódování se pak zašifruje asymetricky (e2e) pro každého příjemce zvlášť. Tím se dosáhne určitého zefektivnění, protože stačí e2e šifrovat (drahá operace) při každé zprávě jen relativně krátký string. Trochu víc k tématu je [tady](https://signal.org/blog/private-groups/).

## Spuštění

Pro spuštění je nutné mít nainstalovaný [node.js](https://nodejs.org/en/).

node server/main.js

Poznámka: Bez HTTPS server vůbec nebude fungovat. Bohužel se ale nedá vytvořit certifikát platný po celé síti (předem nemůžu vědět, na jakém stroji kdo bude chtít server spustit), krom toho jsem si ho podepsal sám. Proto ignorujte křičící browser a pokračujte na web (bude to nějaká volba advanced > proceed to the site (unsafe)). Ač prohlížeč vidí, že certifikát není autentifikovaný, na jeho šifrovací funkčnosti to nic nemění.