

CryMe

PB173 Projekt

Antonín Dufka, Štěpánka Gennertová

23. května 2018

Úvod

CryMe je bezpečný IM nástroj, který:

- ▶ zprostředkovává zabezpečenou komunikaci mezi klienty přes důvěryhodný server
- ▶ jednoduchá na používání
- ▶ nenáročný grafický styl
- ▶ má oddělenou implementaci klienta a serveru
- ▶ https://github.com/dufkan/pb173_project

Registrace

- ▶ komunikace klientů se serverem je inicializovaná klientem
- ▶ klientům se po přihlášení vygeneruje/načte RSA klíč
- ▶ veřejný RSA klíč serveru je distribuován s aplikací
- ▶ přihlášení do aplikace se realizuje pomocí
./cmian pseudonym

např. Přihlášení Boba vypadá takto:

```
./cmain Bob
```

- ▶ po přihlášení se obě strany autentizují pomocí CH-R protokolu
během něhož si domluví symetrický klíč pro následnou
komunikaci mezi klientem a serverem

Navázání spojení s jiným uživatelem

- ▶ každý klient vlastní kromě páru RSA klíčů i sadu prekey pro ECDH na křivce curve25519
- ▶ prekeys po přihlášení posílá na server, kde se uloží
- ▶ pro spojení s jiným uživatelem jsou nejprve ze serveru vyžádány tyto prekeys
- ▶ sdílený klíč se dohodne s druhým klientem pomocí trojcestného Diffie-Hellmana
- ▶ výsledný klíč je 32B

Komunikace s jiným uživatelem

- ▶ samotná komunikace mezi uživateli je šifrovaná End-To-End na základě dohodnutého sdíleného klíče viz předchozí slajd
- ▶ tento klíč je následně aktualizován dle zásad Double Ratchet protokolu
- ▶ zprávy jsou šifrovány symetricky pomocí AES256
- ▶ ani server nemá přístup k otevřenému textu zpráv vyměňovaných mezi klienty
- ▶ díky Double-Ratchetu případný útočník, kterému by se povedlo odchytit jeden klíč, nemá přístup k celé komunikaci

Uživatelské rozhraní

```
c - show saved contacts  
o - get online users  
q - disconnect  
r - receive  
s - send a message to another user  
x - add contact - create a shared key  
> █
```

Použité knihovny — MbedTLS

- ▶ dostali jsme doporučení na ni, osvědčila se nám
- ▶ dobrá dokumentace (v porovnání s OpenSSL)
- ▶ obsahuje velkou škálu kryptografických primitiv
- ▶ využili jsme ji pro SHA256 šifrování, generování RSA klíčů, šifrování a podepisování, ECDH, CMAC
- ▶ pro tyto často používané části jsme si napsali jednodušší rozhraní
- ▶ pro často používané struktury jsme implementovali RAIL wrapper
- ▶ bohužel v knihovně není implementována ECDSA pro curve25519, a tak jsme k podepisování prekeys museli použít klíč RSA

Použité knihovny — asio

- ▶ populární asynchronní knihovna na práci se sítí, která je také součástí boostu
- ▶ poskytuje dobrou implementaci TCP rozhraní pro sockety OS
- ▶ implementovali jsme třídu Channel, která zaobaluje veškerou komunikaci se socketem pomocí rozhraní asio a zároveň je schopná předávaná/přijímaná data šifrovat/dešifrovat

Použitá kryptografie

- ▶ každý klient i server má vlastní pár RSA klíčů na šifrování, klienti mají navíc další RSA pár klíčů na podepisování
- ▶ všechny zprávy mezi klientem a serverem jsou šifrovány AES pomocí stálého klíče dohodnutého při CH-R
- ▶ zprávy mezi klienty jsou šifrovány AES pomocí klíče, který nám vypadne z eliptické magie $X3DH \times DR$
- ▶ využíváme implementaci X9.31 generátoru pseudonáhodných čísel, inicializován `/dev/urandom`

Specifikace kryptografických primitiv

- ▶ RSA s 4096 bitovými klíči
- ▶ AES256
- ▶ SHA256
- ▶ ECDH na křivce curve25519
- ▶ X9.31 generátor psoudonáhodných čísel

Použité zdroje

<https://signal.org/docs/specifications/x3dh/>

<https://signal.org/docs/specifications/doubleratchet/>

<https://tls.mbed.org/api/> <https://think-async.com/>

Ted' přijde praktická ukázka...

Děkujeme za pozornost.