Double ratchet algoritem

Timotej Knez

Januar 2020

Povzetek

Signal je vse bolj znan protokol za zaščito instantnih sporočil. Uporablja se v mnogih aplikacijah, kot so Signal, WhatsApp, Facebook Messenger, Skype in Google Allo. V svojem jedru vsebuje double ratchet algoritem, ki s pomočjo verig ključev poskrbi za varno pridobitev simetričnih ključev, ti pa se uporabljajo za šifriranje instantnih sporočil. V tem projektu si ogledamo delovanje double ratchet algoritma in izdelamo spletno aplikacijo, namenjeno njegovi predstavitvi. Končna aplikacija je na voljo na spletnem naslovu `https://www.doubleratchet.ga`.

1 Uvod

V računalništvu se komunikacija vse bolj premika od elektronske pošte proti instantnim sporočilom. To prinaša nove izzive, če želimo zagotoviti dovolj veliko mero varnosti in zasebnosti pri medosebni komunikaciji. Pri šifriranju instantnih sporočil želimo, da je vsako sporočilo šifrirano z drugim ključem, saj tako močno zmanjšamo možnosti, da pride do uspešnega napada. Želimo tudi, da sta sporočila zmožna odšifrirati samo oba sogovornika, ne pa tudi strežniki, ki jih uporabljata. S tem odstranimo potrebo po zaupanju ponudniku storitve.

V ta namen je bil razvit double ratchet algoritem, ki je del Signal protokola za zaščito instantnih sporočil. Algoritem na podlagi začetnih vrednosti proizvede zaporedje simetričnih ključev, ki se uporabijo za šifriranje posameznih sporočil.

Cilj tega projekta je preučiti delovanje double ratchet algoritma in izdelati interaktivno aplikacijo za predstavitev njegovega delovanja. S tem bi radi dosegli, da bi bilo razumevanje delovanja algoritma čimbolj enostavno.

Double ratchet algoritem sta leta 2013 razvila Trevor Perrin in Moxie Marlinspike [5]. Njegova najpogostejša uporaba je znotraj kriptografskega protokola Signal, ki se vse bolj uporablja v najrazličnejših aplikacijah za instantno sporočanje. Zasledili smo tudi druge primere uporabe, npr. v magisterskem delu Vehkaoja [10] je opisana uporaba double ratchet algoritma v okolju "Internet of things".

2 Zahteve algoritma

Za doseganje varne komunikacije potrebujemo protokol, ki zadošča določenim zahtevam. S temi lastnostmi poskrbimo, da je komunikacija odporna na napadalce. V primeru instantnih sporočil so varnostne zahteve nekoliko drugačne, kot jih običajno srečamo na internetu.

2.1 Zelo dolge seje

Ena od težav, ki nastopi pri komunikaciji z instantnimi sporočili, je to, da uporabnika nimata digitalnih potrdil, s katerimi bi lahko dokazala svojo identiteto. Zaradi tega je težko preprečiti napade s posrednikom (man-in-the-middle), kar nas prisili v to, da uporabnika preko drugega komunikacijskega kanala preverita, ali se ujemata njuni zgostitvi uporabljenih ključev. S tem lahko potrdita, da med vzpostavljanjem povezave ni prišlo do napada s prestrezanjem povezave, saj bi napadalec za uspešen napad moral vzpostaviti dve ločeni seji do obeh oseb, zaradi česar pa se zgostitve ključev ne bi več ujemale. Ko uporabnika enkrat preverita, da ni prišlo do napada s prestrezanjem povezave, lahko od takrat naprej povezavi zaupata. Zaradi tega od algoritma želimo, da nikoli ne izvede povsem nove inicializacije ključev, saj bi s tem izgubili to zaupanje v integriteto povezave. To pomeni, da lahko povezava med uporabnikoma ostane aktivna več let.

2.2 Varnostne lastnosti

Želimo, da komunikacija ostane varna, čeprav se ključi nikoli ne vzpostavijo na novo. Da lahko to dosežemo, moramo zagotoviti naslednje varnostne lastnosti (povzeto po [1]):

- **Pravilnost.** Zahtevamo, da prejemnik prejme sporočilo, ki ga je pošiljatelj poslal, kadar kanala ne napada noben napadalec.
- **Takojšnje odšifriranje in odpornost na izgubo sporočil.** Želimo, da za odšifriranje enega sporočila niso potrebna preostala. Hkrati želimo tudi, da je v primeru izgube enega izmed sporočil mogoče še vedno odšifrirati ostala.
- **Avtentičnost.** Če napadalec ne pozna ključev, ne more spremeniti sporočil ali pa poslati novih sporočil v imenu enega izmed pošiljateljev.
- **Zasebnost.** Če napadalec ne pozna ključev, iz prestreženih sporočil ne izve ničesar.
- **Varnost v naprej.** Vsa sporočila, ki so bila poslana pred trenutkom razkritja ključev napadalcu, ostanejo varna.
- **Varnost po razkritju ključev.** Če je napadalec pasiven, obe strani komunikacije po razkritju ključev ponovno vzpostavita varno povezavo. V

primeru, da je napadalec aktiven in uspe prestreči vse Diffie-Hellman izmenjave, ki se zgodijo po razkritju ključev, varnosti ne moremo zagotoviti, saj v tem primeru napadalec pozna vse stare ključe in vse nove skupne skrivnosti, s katerimi lahko izračuna nove ključe.

- **Odpornost na težave z naključnostjo.** Vse prejšnje lastnosti razen varnosti po razkritju ključev veljajo tudi, če ima napadalec popoln nadzor nad naključnimi vrednostmi, ki jih generirata udeleženca v komunikaciji.

3 Delovanje algoritma

3.1 Uporabljene kriptografske funkcije

Double ratchet algoritem uporablja več kriptografskih konceptov. Za lažje razumevanje je spodaj povzeto njihovo delovanje.

3.1.1 Diffie-Hellman algoritem

Diffie-Hellman (DH) dogovor o ključu je algoritem, ki omogoča vzpostavitev skupne skrivnosti med dvema napravama, kljub temu da lahko komunikaciji napadalci prisluškujejo. Uporablja matematične operacije v multiplikativni grupi praštevilskega obsega. Vsak uporabnik ima svoj javni in svoj zasebni ključ. Zasebni ključ (S_i) se določi naključno, javni ključ pa se izračuna po formuli $P_i = \alpha^{S_i} \pmod p$ $i \in \{a, b\}$. Pri tem mora biti α generator ciklične grupe $\mathbb{Z}_p^*$. Skupno skrivnost obe strani pridobita tako, da uporabita javni ključ druge osebe in svoj zasebni ključ. $Skupni\ ključ = P_b^{S_a} = \alpha^{S_a * S_b} = P_a^{S_b}$ [3].

3.1.2 Funkcija za izpeljavo ključa

Namen funkcije za izpeljavo ključev je, da na podlagi trenutnega ključa pridobi nov ključ. Algoritem uporablja dve vrsti funkcij za izpeljavo ključa. Obe kot vhod sprejmeta prejšnji ključ verige ter vrneta naslednji ključ verige in pa izhodni ključ. Razlika med funkcijama je v tem, da ena sprejme tudi dodatno vrednost, ki jo uporabi pri izpeljavi ključev. Funkcije za izpeljavo ključa temeljijo na močnih enosmernih zgoščevalnih funkcijah (priporoča se uporaba HMAC in HKDF algoritmov z zgoščevalnimi funkcijami SHA-256 ali SHA-512) [7].

3.1.3 Zgoščevalna funkcija

Namen zgoščevalnih funkcij je, da na podlagi vhoda poljubne dolžine ustvarijo izhod konstantne dolžine. Da je zgoščevalna funkcija uporabna za kriptografske namene, mora imeti tudi naslednje lastnosti [8]:

- **Odpornost praslik.** Če poznamo izhod funkcije, mora biti računsko težko v doglednem času najti vhod, ki bi nam dal enak izhod.
- **Odpornost drugih praslik.** Za dani vhod mora biti računsko težko v doglednem času najti drugi vhod, ki bi dal enak izhod kot dani vhod.

- **Odpornost na kolizije.** Računsko težko mora biti v doglednem času najti dva različna vhoda, ki bi jima pripadali enaki zgostitvi.

3.2 Predpriprava

Preden double ratchet algoritem začne delovati, mora naprava, ki začenja komunikacijo, pridobiti javni Diffie-Hellman ključ prejemnikove naprave ter začetni ključ glavne verige. Ti podatki se pridobijo z drugim algoritmom. V ta namen se običajno uporablja algoritem X3DH, ki je bil tako kot double ratchet algoritem razvit za potrebe signal komunikacijskega protokola [6].

3.3 Pregled delovanja

Double ratchet algoritem uporablja verige ključev. Vsebuje tri verige ključev, med katerimi je ena namenjena izdelavi ključev za šifriranje poslanih sporočil, druga izdelavi ključev za odšifriranje prejetih sporočil, tretja pa izdelavi začetnih ključev ostalih dveh verig. Poleg tega v algoritmu nastopa tudi Diffie-Hellman veriga, ki je sestavljena iz zaporedja DH izmenjav. Ta veriga proizvaja skupne skrivnosti med obema stranema komunikacije, ki se uporabijo v glavni verigi za izpeljavo novih ključev.

3.4 Veriga ključev

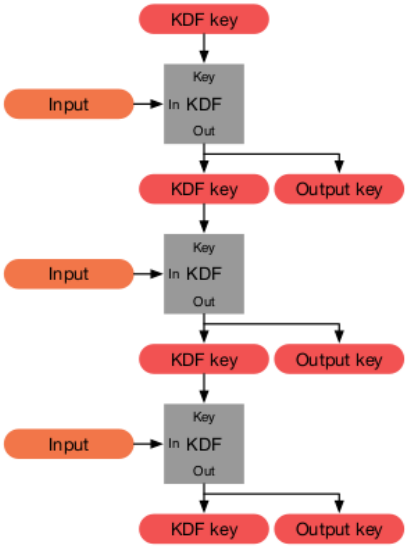
Vsaka veriga ključev ima trenutno vrednost verige ter funkcijo za izpeljavo nove vrednosti. Funkcija za izpeljavo ključev sprejme trenutni ključ verige in vrne novi ključ verige ter izhodni ključ. V glavni verigi funkcija za izpeljavo ključev sprejme tudi vrednost, pridobljeno z Diffie-Hellman izmenjavo, tako da izpeljana ključa nista odvisna samo od prejšnjih ključev (glej sliko 1). Ker je funkcija za izpeljavo ključev enosmerna, napadalec ne more na podlagi trenutnih ključev pridobiti ključev, ki so bili uporabljeni za šifriranje prejšnjih sporočil.

3.5 Diffie-Hellman veriga

Algoritem poskuša poleg prej omenjene varnosti starih sporočil, zagotoviti tudi to, da se lahko po razkritju trenutnih ključev varnost ponovno vzpostavi. To doseže tako, da ima vsak izmed uporabnikov tudi Diffie-Hellman (DH) verigo ključev, ki z DH algoritmom pridobi vrednosti za izračun ključev.

Pošiljatelj ima na začetku algoritma prejemnikov javni ključ. Generira še svoj par ključev in tako izračuna skupno skrivnost. Z njo nato izračuna ključ za pošiljanje in pošlje prvo sporočilo, ki mu priloži še svoj javni ključ. S tem lahko prejemnik sporočilo odšifrira.

Ko drugi uporabnik pošlje svoje sporočilo, generira svoj novi par ključev in ustvari novo skupno skrivnost. Na podlagi te skrivnosti začne svojo verigo za pošiljanje in svoj novi javni ključ priloži poslanim sporočilom.



Slika 1: Diagram delovanja verige ključev, pridobljen iz dokumentacije algoritma [7].

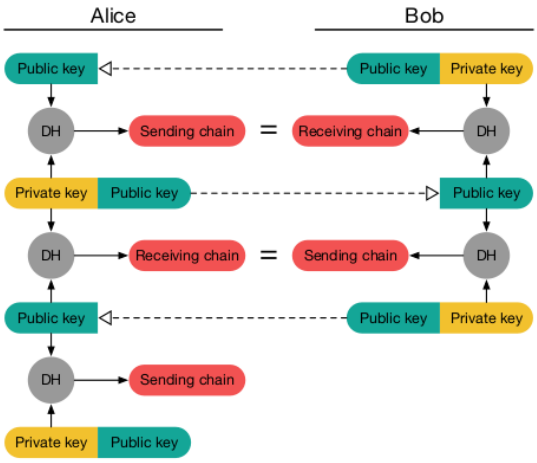
Tako si uporabnika izmenično generirata nove pare ključev in pri vsaki izmenjavi obnovita verigi za pošiljanje in prejemanje (glej sliko 2). S tem algoritem doseže, da se lahko tudi po razkritju ključev ponovno vzpostavi varnost.

3.6 Izgubljena sporočila

Ena izmed lastnosti, ki jih algoritem zagotavlja, je tudi odpornost na izgubljena sporočila in sporočila, prejeta v napačnem vrstnem redu. Pri tem si algoritem pomaga tako, da v glavi vsakega sporočila poleg javnega DH ključa, ki je bil uporabljen, pošlje tudi zaporedno številko sporočila. Ko prejemnik tako prejme sporočilo, ki nosi višjo zaporedno številko od pričakovane, ustvari vse ključe v verigi do ključa, ki ga potrebuje za odšifriranje sporočila, in vmesne ključe shrani. Tako lahko, v primeru da vmesno sporočilo prispe pozneje, to sporočilo odšifrira z enim od shranjenih ključev.

4 Izdelava programa

Glavni cilj projekta je bil izdelati interaktivno predstavitev delovanja double ratchet algoritma, s katero lahko uporabnik lažje razume njegovo delovanje.



Slika 2: Diagram delovanja DH izmenjav (iz dokumentacije algoritma [7]).

4.1 Opis aplikacije

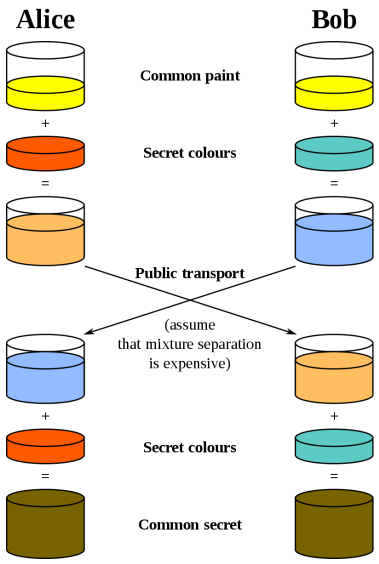
Želeli smo, da bi bila aplikacija dostopna čim večjemu številu ljudi, zato smo se odločili, da smo jo izdelali v obliki preproste spletne strani. Stran prikazuje diagram izpeljave ključev na strani Ane in Boba med njuno komunikacijo. Prikazuje tudi vrsto sporočil, ki čakajo na prejem na Anini ter Bobovi strani. Uporabniku omogočamo, da pošlje poljubno sporočilo, izbriše sporočilo v vrsti za prejem, s tem simulira izgubo sporočila, ter da sporočilo iz vrste za prejem prejme. Ob vseh teh akcijah se prikazano stanje ključev ustrezno posodablja.

4.2 Prikaz delovanja DH algoritmoa

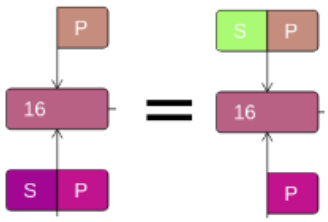
Pri izdelavi aplikacije sem naletel na izziv, kako na vizualen način predstaviti Diffie-Hellman algoritem. V ta namen sem uporabil prikaz z mešanjem barv, ki je uporabljen v drugi literaturi (glej sliko 3). Poleg barvnih vrednosti aplikacija računa tudi prave DH vrednosti v majhni modulski grupi in jih prikazuje uporabniku (slika 4).

4.3 Izračun vrednosti ključev

Aplikacija računa in prikazuje tudi vse ključe verig in ključe za šifriranje sporočil. Zato mora imeti definirano funkcijo za izpeljavo ključev. Zanjo smo uporabili algoritem SHA-256 iz knjižnice [9]. Za ključe smo uporabili po pet znakov iz štiriinšestdesetiškega kodiranja zgoščene vrednosti. S tem so bili ključi dovolj kratki, da smo jih lahko v aplikaciji prikazali uporabniku. Tako lahko uporabnik vidi celotno stanje sistema in lažje razume njegovo delovaje (glej sliko 5).



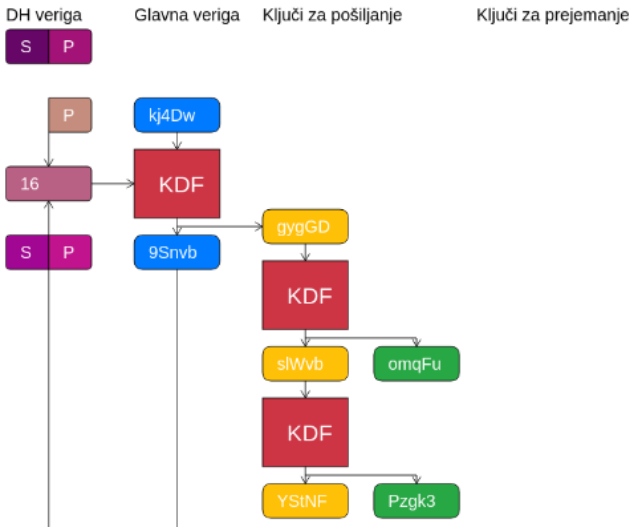
Slika 3: Prikaz delovanja Diffie-Hellman alogirtma [2].



Slika 4: Prikaz delovanja Diffie-Hellman alogirtma v aplikaciji.

5 Zaključek

S pregledom delovanja double ratchet algoritma smo ugotovili, da zagotavlja visoko mero varnosti celo pod zelo zahtevnimi pogoji. V primeru razkritja ključa algoritem z enosmerno funkcijo zagotovi, da napadalec ne more pridobiti ključev za stara sporočila. Prav tako z Diffie-Hellman verigo zagotovi, da pasivni napadalec ne more pridobiti ključev za prihodna sporočila. V primeru, da je napadalec aktiven in izvaja napad s prestrezanjem sporočil, mora prestreči vsa sporočila od trenutka razkritja ključev, saj se sicer že ob prvem sporočilu ponovno vzpostavi varno stanje. Opazili smo šibkost algoritma v primeru, da aktivni napadalec prestreza celotno povezavo že od začetka komunikacije. To lahko preprečimo samo z ročnim preverjanjem ujemanja uporabljenih ključev na obeh straneh komunikacije.



Slika 5: Primer prikaza Aninega stanja v aplikaciji.

Uspešno smo izdelali tudi spletno aplikacijo, s katero lahko uporabnik vidi delovanje algoritma (dostopno na spletnem naslovu [4]). S tem ko razumemo delovanje algoritma, lažje vidimo njegove prednosti in šibkosti. Glede na naučeno pri projektu, bi uporabo obdelanega algoritma v vsakodnevni komunikaciji močno priporočal.

Literatura

[1] J. Alwen, S. Coretti in Y. Dodis. The double ratchet: security notions, proofs, and modularization for the signal protocol. V *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, str. 129–158. Springer, 2019.

[2] Diffie–Hellman key exchange. Diffie–Hellman key exchange - Wikipedia. 2020 [Online; Uporabljeno 30.1.2020], Dostopno na: `https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange`.

[3] M. Just. Diffie–Hellman Key Agreement. V *Encyclopedia of Cryptography and Security*, str. 154–154. Springer US, 2005.

[4] T. Knez. Double ratchet. 2020 [Online; Uporabljeno 30.1.2020], Dostopno na: `https://www.doubleratchet.ga`.

[5] M. Marlinspike. Advanced cryptographic ratcheting. Whisper System Blog, 2013.

[6] M. Marlinspike in T. Perrin. The x3dh key agreement protocol. Open Whisper Systems, 2016.

[7] T. Perrin in M. Marlinspike. The double ratchet algorithm. Open Whisper Systems, 2016.

[8] P. Rogaway in T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. V *International workshop on fast software encryption*, str. 371–388. Springer, 2004.

[9] E. Stark, M. Hamburg in D. Boneh. Symmetric cryptography in javascript. V *2009 Annual Computer Security Applications Conference*, str. 373–381. IEEE, 2009.

[10] V. Vehkaoja. End-to-end encryption protocolfor internet of things devices. Magisterska naloga, University of Oulu, 2017.