# REFLEKSJONSNOTATET

Laget av Daniel Olov Mostad

Dette prosjektet gikk ut på å lage en VPN med Python som krypterer trafikk. Jeg brukte socket hovedsakelig, med en proxy server som koblet over traffiken med SOCKS 5. Her er noen punkter å tenke på under prosjektet.

**1. Hastighet:**
Før kryptering fikk jeg 25 Mbps, men etterpå gikk det til 9 Mbps. Det er fordi AES-kryptering bruker mye CPU, og når alt må pakkes ut/pakkes inn mellom klient og server, blir det tregt.

Jeg prøvde å bruke UDP fordi det er raskere, men det ble kaos – hvis en enkelt bite ble mistet underveis, kunne ikke meldinga dekrypteres (korrupt data). Derfor brukte jeg TCP til slutt, selv om det er tregere.

**2. Key problemer:**
Jeg hadde først key-en hardkoda i koden, men skjønte fort at det er superusikkert. Prøvde å bruke cryptography-biblioteket til å generere nøkler automatisk, men fikk ikke til å synce den mellom klient og server på den tiden jeg fikk. Derfor måtte jeg gå tilbake til å bruke en hardkoda key.

**3. 4-byte header problem:**
Et stort problem var når koden skulle lese de første 4 bytesene for å finne lengda på meldinga. Noen ganger fikk den ikke hele headeren på en gang (f.eks. bare 1-3 bytes), så programmet kræsja. Løsninga var å bruke en loop som sjekka om *alle* 4 bytes var mottatt før den leste resten.

**4. Threads og CPU:**
Jeg brukte threading for å håndtere flere klienter samtidig, men hvis for mange koblet til, ble alt tregt. Tror det er fordi hver tråd bruker minne, og CPU-en ble overbelasta. Jeg fikk dessverre ikke tid til å stressteste ordentlig, men tror maks 10-15 klienter hadde gått før serveren kollapser.

**5. Andre bugs:**

- **SSL-problemer:** Noen nettsteder (som speedtest) krasja proxyen fordi de brukte HTTP/3 eller annen rar trafikk.
- **Buffer overflow:** Måtte begrense maks datamengde til 4096 bytes av gangen for å unngå at alt frøys.
- **Koblingsdrops:** Hvis serveren ikke svarte fort nok, måtte jeg legge til en "timeout" på 5 sekunder.

Hadde jeg hatt mer tid, ville jeg:

- Testet med AES-GCM (raskere enn CBC).
- Implementert UDP med feilsjekking for å unngå korrupte data.
- Lagd en grafisk hastighetstest i Python.