Krypterad chattapplikation

Innehåll

[Introduktion 3](#_Toc184904397)

[Syfte 3](#_Toc184904398)

[Genomförande 3](#_Toc184904399)

[Implementation 3](#_Toc184904400)

[Kryptering med Fernet 3](#_Toc184904401)

[Nyckeln genereras så här: 3](#_Toc184904402)

[Meddelandehantering med MQTT 4](#_Toc184904403)

[Kryptering: 4](#_Toc184904404)

[Dekryptering: 4](#_Toc184904405)

[Användarinmatning och multitrådning 4](#_Toc184904406)

[Defensiv programmering 4](#_Toc184904407)

[Resultat 5](#_Toc184904408)

[Diskussion och reflektion 6](#_Toc184904409)

[Framtida utveckling kan inkludera: 7](#_Toc184904410)

[Slutsats 8](#_Toc184904411)

[Referenser 9](#_Toc184904412)

[Bilagor 10](#_Toc184904413)

# Introduktion

Målet med denna uppgift var att utveckla en krypterad chattapplikation som använder MQTT-protokollet för meddelandehantering. Applikationen använder symmetrisk kryptering för att säkerställa att meddelanden mellan deltagarna skyddas från obehörig åtkomst. Projektet syftade också till att tillämpa god programmeringssed och inkluderar funktioner som defensiv programmering och tester.

## Syfte

1. Utforska användning av MQTT för meddelandehantering i realtid.

2. Implementera symmetrisk kryptering för att skydda meddelanden.

3. Tillämpa defensiv programmering och ”best practices”.

## Genomförande

Teknologier och bibliotek

- Programmeringsspråk: Python

- MQTT-klient: paho-mqtt

- Kryptering: cryptography (Fernet)

## Implementation

### Kryptering med Fernet

Applikationen använder biblioteket cryptography för att använda symmetrisk kryptering med Fernet. Vid start skapas en nyckel baserad på en lösenfras som anges av användaren. Denna nyckel används sedan för att kryptera och dekryptera meddelanden.

### Nyckeln genereras så här:

python

hashed = hashlib.sha256(passphrase.encode()).digest()

key = base64.urlsafe\_b64encode(hashed[:32])

fernet = Fernet(key)

## Meddelandehantering med MQTT

MQTT-protokollet hanterar kommunikation mellan klienter. Chattmeddelanden krypteras innan de skickas till MQTT-brokern och dekrypteras av mottagande klient.

### Kryptering:

python

encrypted\_message = self.fernet.encrypt(msg\_to\_send.encode())

self.client.publish(self.topic, encrypted\_message)

### Dekryptering:

python

decrypted\_message = self.fernet.decrypt(msg.payload).decode()

print(f"\n{decrypted\_message}")

## Användarinmatning och multitrådning

Ett separat tråd ansvarar för att hantera användarinmatning medan MQTT-klienten lyssnar efter inkommande meddelanden. Detta gör att applikationen kan bearbeta inkommande meddelanden i realtid samtidigt som användaren kan skicka nya meddelanden.

## Defensiv programmering

- Validering av användarinmatning (t.ex. att lösenfrasen inte är tom).

- Förhindrande av krasch vid felaktigt formaterade meddelanden genom undantagshantering.

## Resultat

Den utvecklade chattapplikationen mötte projektets krav och implementerade följande funktioner:

1. Kryptering och dekryptering: Meddelanden skyddas effektivt genom Fernet-kryptering.

2. Real-tidskommunikation: MQTT användes för att säkerställa snabb och tillförlitlig kommunikation mellan klienter.

3. Användarvänlighet: Enkla kommandon för att skicka meddelanden och avsluta chatten.

# Diskussion och reflektion

Projektet illustrerade vikten av att kombinera säkerhet och användbarhet. Kryptering med symmetrisk nyckel visade sig vara effektiv men krävde att användare hade samma lösenfras, vilket kan vara en svag punkt om nyckeln inte hanteras säkert.

# Framtida utveckling kan inkludera:

1. Implementering av asymmetrisk kryptering för nyckelutbyte.

2. Lägg till autentisering för användare.

3. Stöd för flera chattrum samtidigt.

# 

# Slutsats

Projektet visade hur MQTT och symmetrisk kryptering kan kombineras för att skapa en säker chattapplikation. Det är ett bra exempel på hur säkerhet och funktionalitet kan integreras i moderna applikationer.

# Referenser

Eclipse Foundation, *Paho MQTT Client Documentation*. Tillgänglig på: <https://www.eclipse.org/paho/index.php?page=clients/python/index.php> (hämtad: 12 december 2024).

Cryptography.io, *Fernet Documentation*. Tillgänglig på: <https://cryptography.io/en/latest/fernet/> (hämtad: 12 december 2024).

# Bilagor

