**Project Report : Secure Chat Application​ using MQTT and Encryption symmetric**

**Introduction**

The project aims to develop of a secure chat applications that use MQTT (Message Queuing Telemetry Transport) protocol for messaging in real time and IMPLEMENTING encryption symmetric for A ensure confidentiality and security communications . The application allows uSERS to join​ chat rooms , to MAIL MESSAGES ENCRYPTED and saddle receive MESSAGES decrypted , respecting the may good programming practices .​

**objectives project**

1. **Communication Secured** : Implementation encryption symmetric for protecting posts against access unauthorized .
2. **Messaging in Real Time** : Using MQTT for transmission efficiency and easy messaging .​
3. **interaction To the User** : Providing of a simple command - line interfaces for interaction users .
4. **Best Practices** : Incorporation defensive programming , management ERRORS and testing functionalities .

**Presentation TECHNICAL**

**Encryption**

* **Encryption Symmetrical** : All participants USE same key , derivative from a 32- character password , both for encryption , how much and for decryption .
* **Fernet Encryption** : The cryptography library is use for Fernet encryption , thanks to simplicity and its security features .

**MQTT integration**

* **Broker** : The application connects to a public MQTT broker (broker.hivemq.com ) for distribution messages .
* **Configuring Client** : An MQTT client is initialized with an ID unique , connects to the broker and subscribes to a topic specific to each chat rooms .

**Interface User**

* **Initialization Chat** : Users enter a username , choose a chat room , and​ enter a password for encryption .
* **administrator Messages** : Users can send messages , which are encrypted before transmission , and can receive​ messages , which are decrypted upon reception .

**Implementation Details​**

**Chat Class**

* **Initialization** : Class is initialized with a username , a chat room , and a password . Configure MQTT client , encryption key and the necessary callbacks .
* **Configuring Encryption** : The password is validity for length , and a Fernet object is creator for encryption / decryption operations .​
* **MQTT Connection** : Class MANAGEMENT connection to the MQTT broker , subscribing to the topic chat room and handling messages .
* **administrator Messages** : Messages are encrypted before sending​ and decrypted upon receipt , with management A ERRORS for decryption failures .​

**function Main (main)**

* **The input User** : Request uSERS information REQUIRED for initialization the chat application .
* **administrator Errors** : Includes checks for length PASSWORD and MANAGEMENT exceptions in time initialization and RUNNING application .

**Security Considerations**

1. **administrator Keys** : Encryption key​ is CONSEQUENTIAL from a password provided by the user , which must managed or stock in safety in a real scenario .
2. **administrator Errors** : A management robust A ERRORS ensure that the application does not continue in an unstable state , protecting against security vulnerabilities .​

**Best Practices**

* **Defensive Programming** : The code includes management ERRORS for network problems , encryption / decryption failures​​ and input validation​ user .
* **Quality Code** : Compliance PEP 8 style guide , use of docstrings and a structure modulation for legibility and maintenance .
* **Testing** : Unit tests are recommended for encryption / decryption functions , MQTT connection and​ administrator posts for A ensure reliability .

**Structure project**

* **Chat Class** : Manage MQTT client , encryption and handling messages .
* **function Main** : Handles input**​**​ USERS for initialization and runs the chat app .

**GitHub repository**

* **README.md** : Contains description project , configuration instructions , user guide​​ and NAMEs contributors .
* **Code Source** : Organization in a Python package with a structure clear files​ for legibility and maintenance .

**Conclusion**

This design demonstrate integration communication practices​ secured with IoT messaging protocols , providing an example practical about how encryption can be applied in applications REAL for protecting dATES users . Using Python, MQTT and the cryptography library highlights a combination of techniques modern programming​ and security considerations , making this an educational software​ and functional . Improvements future It could include key management systems , storage​​ secure passwords​ and an interface graphics for an experience may good user experience .