Encryption in Python using One-Time Pad and QKD

Deependra Singh Shekhawat

Introduction:

This paper explores the concept of one-time pad (OTP) encryption and its implementation using a simplified Quantum Key Distribution (QKD) simulation. The provided code includes two functions: encrypt and decrypt, which demonstrate encryption in python by means of a one time pad encryption, with the key material being derived from a quantum key exchange.

One-Time Pad (Vernam Cipher) - A one-time pad is a type of stream cipher. Stream ciphers encrypt messages by combining them with a pseudo-random key stream, bit by bit. In the context of the provided code snippets:

- The key generated through the QKD process acts as the key stream.
- The XOR operation between the message bits and the key bits serves as the stream cipher mechanism.

Considerations and Challenges

While the code simulates the core idea of OTP encryption, several crucial aspects were considered during development:

- Key Length: The key length needs to match the message length (in bits) for the XOR operation to function correctly. The code assumes this requirement is met.
- Key Randomness: Ideally, the key should be a truly random sequence of 0s and 1s. The provided code incorporates a simplified QKD process to generate the key, but real-world QKD implementations involve error correction and authentication mechanisms for enhanced security, which are not included here.
- Key Reuse: A fundamental principle of OTP is that the key must be used only once. Reusing the same key weakens security. The current implementation doesn't explicitly prevent key reuse.

Code Snippets:

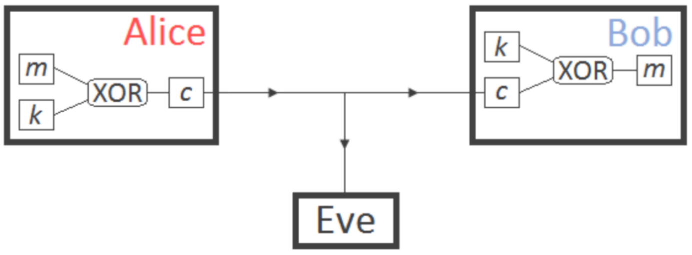
```
unencrypted_string = "I. atom in cosmos, cosmos in atom."
key = QKD(1024) # generates random key of same length(264) as of message(in binary)
```

```
def encrypt(message, key):
    # Convert the message to a binary string
    binary_message = ''.join(format(ord(char), '08b') for char in
unencrypted_string)
    binary_message_list = [int(char) for char in binary_message]

    # Truncate the message list to match the key length
    message_truncated = binary_message_list[:len(key)]
    encrypted_message = []
    for i in range(len(message_truncated)):
        encrypted_message.append(message_truncated[i] ^ key[i])
    return ''.join(str(x) for x in encrypted_message)
```

```
def decrypt(encrypted_message, key):
    decrypted_message = []
    for i in range(len(encrypted_message)):
        decrypted_message.append(int(encrypted_message[i]) ^ key[i])
    # Convert list to binary string
    binary_decrypted = ''.join(str(x) for x in decrypted_message)
    decrypted_string = ''.join(chr(int(binary_decrypted[i:i+8], 2)) for i in
range(0, len(binary_decrypted), 8))
    return decrypted_string
```

Analysis:



If an eavesdropper were to intercept the communication, they would face significant challenges in deciphering the encrypted data, assuming the key is securely generated and shared using QKD. Since OTP encryption provides perfect secrecy when used with a truly random key of equal length to the message and is never reused, an eavesdropper would gain no information about the plaintext from the intercepted ciphertext. However, if the key is compromised or reused, the security of the encryption is compromised, potentially allowing the eavesdropper to decrypt the data.

Conclusion:

The development of quantum encryption, particularly through methods like Quantum Key Distribution (QKD), holds immense importance for the future of secure communication. Traditional encryption methods rely on the complexity of mathematical algorithms, which could potentially be broken by advancements in quantum computing power or novel cryptographic attacks. Quantum encryption, on the other hand, leverages the fundamental principles of quantum mechanics to provide unconditional security, making it theoretically impossible for an eavesdropper to intercept or tamper with the communication without detection. As cyber threats continue to evolve, the adoption of quantum encryption technologies and the development of Post-Quantum Cryptography (PQC) is crucial. Despite quantum computers not yet having the processing power to break widely used cryptographic algorithms as of now(April 2024), the field is rapidly evolving. Cryptographers are already preparing for ***"Q-Day"***, the day when existing algorithms will become vulnerable to quantum attacks. As we progress in the digital age, the role of quantum encryption and PQC in securing communication continues to grow, underscoring the need to safeguard our digital communication infrastructure against future threats.

References:

- https://en.wikipedia.org/wiki/One-time_pad
- https://en.wikipedia.org/wiki/Post-quantum_cryptography
- Using quantum key distribution for cryptographic purposes: A survey

  Crossref DOI link: https://doi.org/10.1016/j.tcs.2014.09.018