May 2022

MECHANISM TO ENHANCE SECURITY OF THE QUIC BASED COMMUNICATION

Niranjan M M

MECHANISM TO ENHANCE SECURITY OF THE QUIC BASED COMMUNICATION

AUTHOR:
Niranjan M M

ABSTRACT

Quick UDP Internet Connections (QUIC) is based on UDP/HTTP and TLS. While TLS provides adequate security, it is not very far to crack it with the increasing compute power of the devices, as TLS is based on Public Key Infrastructure (private/public key) methods, wherein keys are derived mathematically. As we know, asymmetric encryption (PKI) methods are not immune to cryptanalysis attacks from quantum computers, whereas symmetric encryption (with sufficiently large key sizes) is immune to cryptanalysis attacks. Hence there is renewed interest in symmetric encryption methods to deploy quantum secure networks. In short, currently QUIC uses TLS based method to provide data encryption, which is based on asymmetric encryption, hence it is prone to crypto analysis attack. The techniques presented herein propose method to enhance the security of the applications and their communication built with QUIC protocol to adequately cope up with the ever increasing threats on the public networks. As per this method, use symmetric method for encryption/decryption of the QUIC payload and take advantages of UDP based QUIC to encrypt set of QUIC payloads using different Secret Keys from the pool which results in added entropy. In summary, the proposed method enhance the security of QUIC connections by encrypting QUIC payload using randomly indexed Secret Keys, instead of encrypting using TLS sessions keys.

DETAILED DESCRIPTION

Current decade is the decade of cloud and security. Wider adoption of cloud, mobility, access anywhere to anywhere and from any device to any device has completely opened up the access over public networks and internet. While these developments are welcoming, even the network threats are growing over these public networks. Hence it is essential to enhance the security of the connections of the devices involved in mission critical applications, monetary transactions and various dealings needing the privacy.

In the recent past QUIC is heard a lot and its usage is growing gradually in many websites and browsers because of its efficiency and speed. As per the data analysis, QUIC/HTTP/3 is used by 22% of all websites. More than 50% of social media traffic is QUIC/HTTP3.Also some of the content streaming sites and other services attempt QUIC first and then fallback to TCP+TLS.

QUIC is based on UDP/HTTP and TLS. While TLS provides adequate security, it is not very far to crack it with the increasing compute power of the devices, as TLS is based on Public Key Infrastructure (private/public key) methods, where-in keys are derived mathematically. As we know, asymmetric encryption (PKI) methods are not immune to cryptanalysis attacks from quantum computers, whereas symmetric encryption (with sufficiently large key sizes) is immune to cryptanalysis attacks. Hence there is renewed interest in symmetric encryption methods to deploy quantum secure networks.

Furthermore, organisations around the world are investing heavily in building quantum computers. Hence need to be prepared for the cryptanalysis attacks, in particular, the information (such as country/state secret, defence, patient health record etc.,) encrypted using RSA or ECC can be stored until quantum computers are available and later decrypted with the algorithms such as Shor's algorithm, Grover's algorithm etc., We don't know, when these things can happen, hence we need mechanism which can be implemented today to give protection for future attacks.

In short, currently QUIC uses TLS based method to provide data encryption, which is based on asymmetric encryption, hence it is prone to crypto analysis attack.

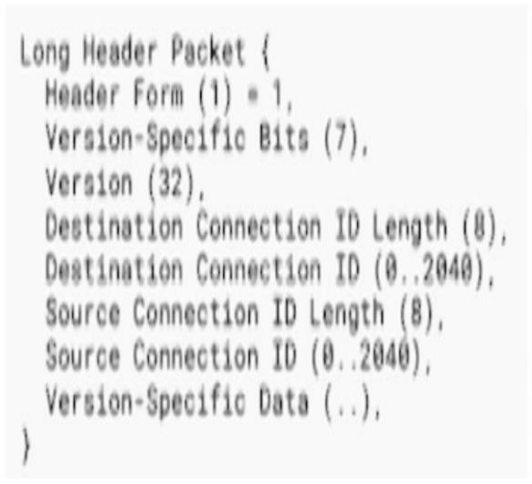
The techniques presented herein propose a method to enhance the security of the applications and their communication built with QUIC protocol to adequately cope up with the ever increasing threats on the public networks. As per this method, use symmetric method for encryption/decryption of the QUIC payload and take advantages of UDP based QUIC to encrypt set of QUIC payloads using different Secret Keys from the pool which results in added entropy. In short, the proposed method enhance the security of QUIC connections by encrypting QUIC payload using randomly indexed Secret Keys, instead of encrypting using TLS sessions keys.

As per this method, after QUIC/TLS session is established, server generates secret seed value which can be, unique per connection or common for all connections from a particular client (as per administrator configuration). Server sends this secret seed value to the client over QUIC/TLS session, so that both server and client will have common secret seed value. This secret seed is used to initialise the Function as a Service (FaaS), which will be running on both server and client. This FaaS is such that, if initialised with the same secret seed, it generates unique set of keys {SecretKey, KeyId} (e.g., {SecretKey-1,KeyId-1}, {SecretKey-2,KeyId-2}....{SecretKey-100,KeyId-100}). Now both Server and Client will have same set of {SecretKey, KeyId} pairs.

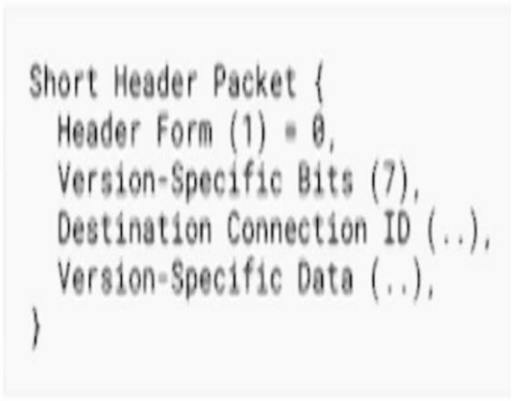
Whenever server want to send QUIC payload to the client, it randomly select key pair from the locally generated list and encrypts the QUIC payload. Server will also add KeyId (which is just an index to the key pair of the list) as part of QUIC header and send it to the client. On receiving this QUIC packet, the client uses KeyId present in the header to fetch the corresponding SecretKey and decrypts the QUIC payload. Same thing holds good for client sending QUIC payload to the server.

In other words, server can select any SecretKey to encrypt the QUIC payload and client uses the corresponding SecretKey by mapping with the KeyId sent in the QUIC header. This mechanism ensures that, only Key Identifiers (aka Key Index) are exchanged as part of the communication.

The KeyId is sent in the version-specific header of the QUIC protocol header as below:

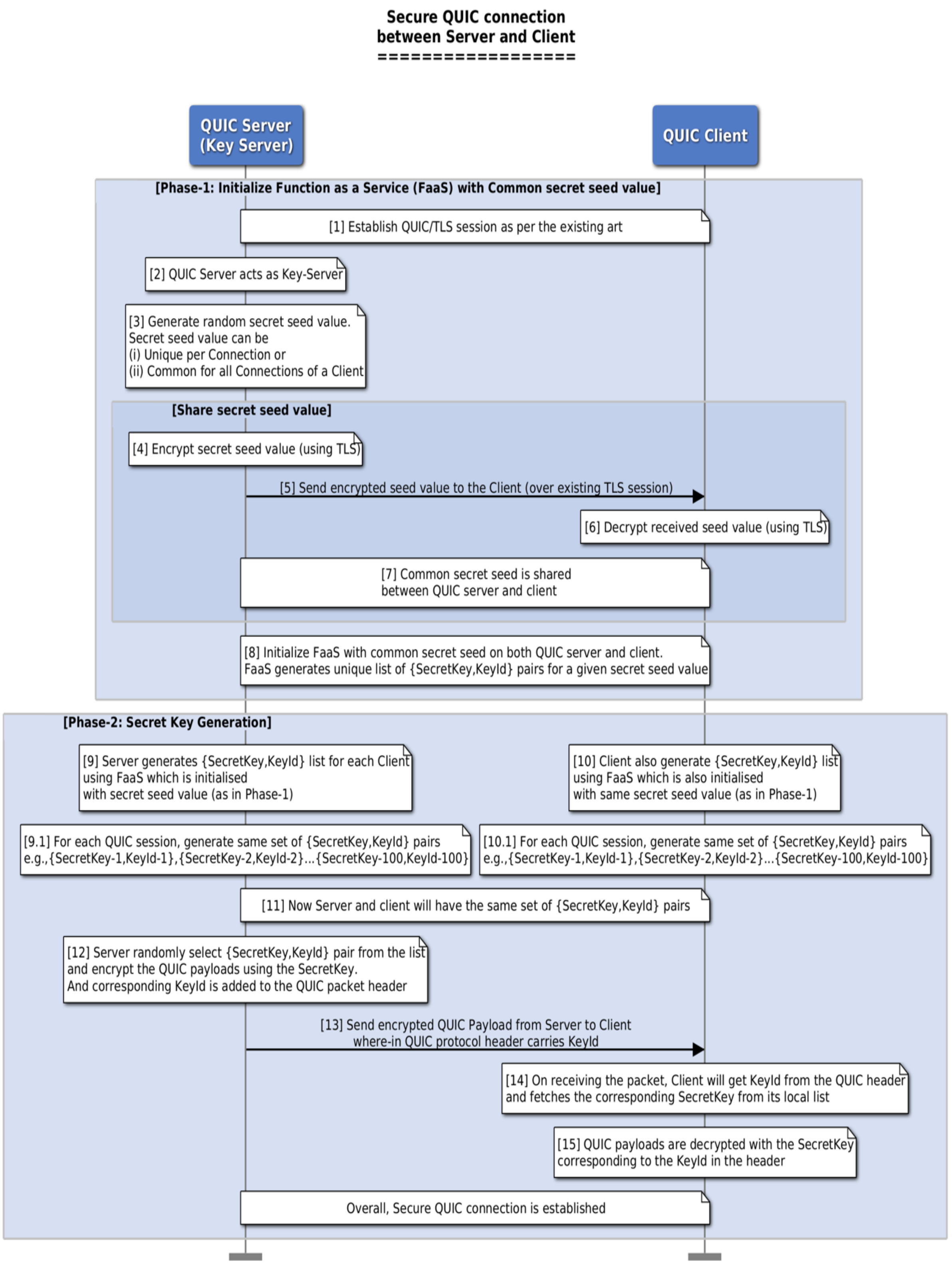


```
Long Header Packet {
  Header Form (1) = 1,
  Version-Specific Bits (7),
  Version (32),
  Destination Connection ID Length (8),
  Destination Connection ID (0..2040),
  Source Connection ID Length (8),
  Source Connection ID (0..2040),
  Version-Specific Data (..),
}
```



```
Short Header Packet {
  Header Form (1) = 0,
  Version-Specific Bits (7),
  Destination Connection ID (..),
  Version-Specific Data (..),
}
```

Figure-1 depicts secure QUIC connection between server and client.



Secure QUIC connection
between Server and Client
==================

QUIC Server
(Key Server)

QUIC Client

[Phase-1: Initialize Function as a Service (FaaS) with Common secret seed value]

[1] Establish QUIC/TLS session as per the existing art

[2] QUIC Server acts as Key-Server

[3] Generate random secret seed value.
Secret seed value can be
(i) Unique per Connection or
(ii) Common for all Connections of a Client

[Share secret seed value]

[4] Encrypt secret seed value (using TLS)

[5] Send encrypted seed value to the Client (over existing TLS session)

[6] Decrypt received seed value (using TLS)

[7] Common secret seed is shared
between QUIC server and client

[8] Initialize FaaS with common secret seed on both QUIC server and client.
FaaS generates unique list of {SecretKey,KeyId} pairs for a given secret seed value

[Phase-2: Secret Key Generation]

[9] Server generates {SecretKey,KeyId} list for each Client
using FaaS which is initialised
with secret seed value (as in Phase-1)

[10] Client also generate {SecretKey,KeyId} list
using FaaS which is also initialised
with same secret seed value (as in Phase-1)

[9.1] For each QUIC session, generate same set of {SecretKey,KeyId} pairs
e.g.,{SecretKey-1,KeyId-1},{SecretKey-2,KeyId-2}...{SecretKey-100,KeyId-100}

[10.1] For each QUIC session, generate same set of {SecretKey,KeyId} pairs
e.g.,{SecretKey-1,KeyId-1},{SecretKey-2,KeyId-2}...{SecretKey-100,KeyId-100}

[11] Now Server and client will have the same set of {SecretKey,KeyId} pairs

[12] Server randomly select {SecretKey,KeyId} pair from the list
and encrypt the QUIC payloads using the SecretKey.
And corresponding KeyId is added to the QUIC packet header

[13] Send encrypted QUIC Payload from Server to Client
where-in QUIC protocol header carries KeyId

[14] On receiving the packet, Client will get KeyId from the QUIC header
and fetches the corresponding SecretKey from its local list

[15] QUIC payloads are decrypted with the SecretKey
corresponding to the KeyId in the header

Overall, Secure QUIC connection is established

Figure-1

The technique presented herein is explained in two phases:

Phase-1: Initialise Function as a Service (FaaS) with common secret seed value:

- QUIC uses classical secure TLS handshake to establish secure connection between client and server.
- QUIC server generates random secret seed value. This could be generated "one for each QUIC connection" or "common for all QUIC connections of a client".
- QUIC server sends the secret seed value over the QUIC/TLS session.
- Upon receiving the QUIC packet, client decrypts payload and get the secret seed value.
- Now both QUIC server and client have a common secret seed value.

Phase-2: Secret Key Generation

- QUIC server and client runs Function as a Service (FaaS), such that, if initialised with the same secret seed value, it generates unique set of {SecretKey, KeyId} pairs.
- QUIC server and client initialise the FaaS with common secret seed value exchanged in Phase-1.
- Now the FaaS generates unique pairs of {SecretKey, KeyId} on both QUIC server and client.
- As we know, symmetric encryption methods are immune to cryptanalysis attacks, unlike asymmetric methods (PKI) which are mathematically derived and can be prone to cryptanalysis attacks.
- Server randomly select {SecretKey, KeyId} pair from the list and encrypt the QUIC payloads using the SecretKey. And corresponding KeyId is added to the QUIC packet header.
- On receiving the packet, client will get KeyId from the QUIC header and fetches the corresponding SecretKey from its local list.
- Client decrypts QUIC payload using this SecretKey.
- Same mechanism is followed while client sending QUIC payload to the Server.
- This will make QUIC protocol both resistant from cryptanalysis attacks as well as having robust security.
- Note: The {Secretkey, KeyId} list can be refreshed by generating the new secret seed value periodically and then exchange among server and client over the existing QUIC/TLS connection. This will further strengthen the security of the overall mechanism.

In summary, the technique presented herein will enhance the security aspect of QUIC connections. In this method, random generation of the SecretKey list and selection of the SecretKey will increase the entropy of the overall mechanism. Moreover, this method inherently uses symmetric encryption which is resistant from cryptanalysis attacks. Additionally, this method allows frequent key refresh with no administrative cost.