**QUIC Website Analysis**

**1. What is the name of the website?**

The website requested is [**https://cloudflare-quic.com**](https://cloudflare-quic.com/).

**SNI field:** cloudflare-quic.com

**Explanation:** In QUIC, the SNI (Server Name Indication) is carried inside the TLS ClientHello embedded within the Initial packet. It indicates which website the client wants, even though the rest of the handshake is encrypted.

This was found by inspecting the first Initial packet from the client (Frame 109). Inside its CRYPTO frame, the TLS ClientHello message contains a Server Name Indication (SNI) extension, which explicitly lists the hostname cloudflare-quic.com.

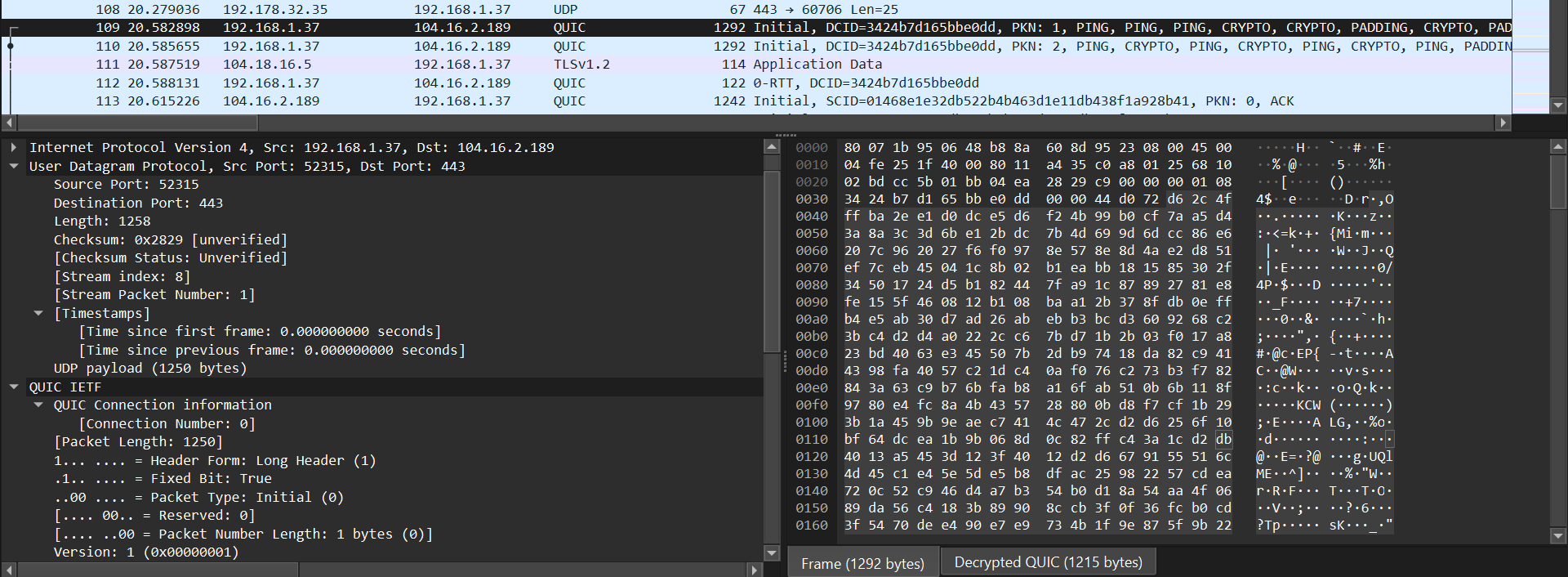
A screen shot of a computer

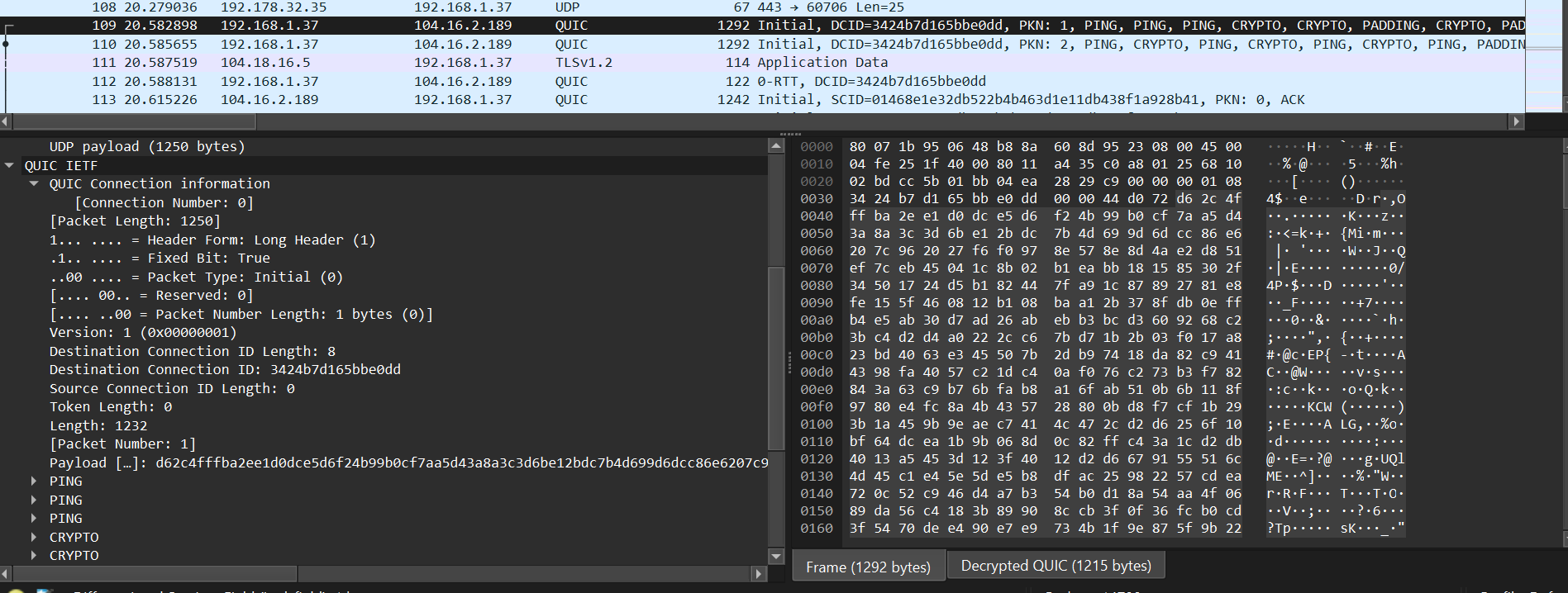
AI-generated content may be incorrect.

**2. Find the packet that contains the Initial QUIC handshake. What information is exchanged here?**

The initial QUIC handshake begins in **Frame 109**.

This is the first packet from the client (192.168.1.37) to the server (104.16.2.189). It's a QUIC Initial packet that carries the essential information to start the connection: the QUIC version, the destination and source connection IDs, and a CRYPTO frame containing the start of the TLS 1.3 handshake.

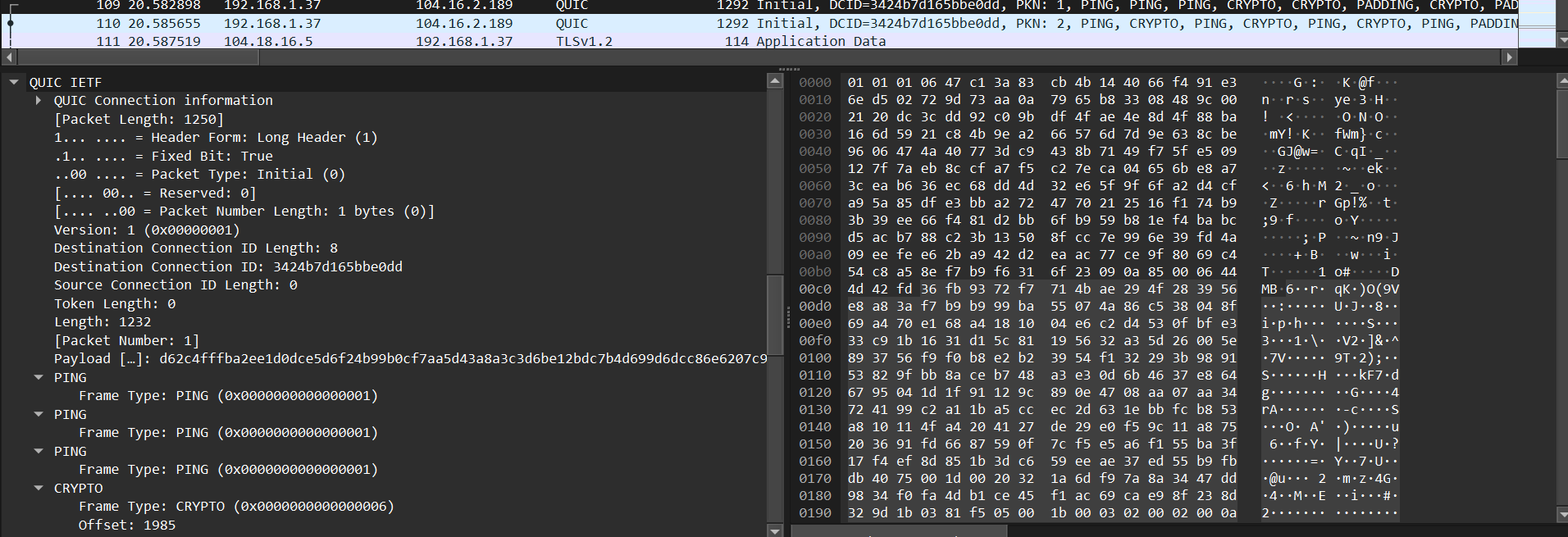




**3. Identify the QUIC packet that contains the TLS ClientHello.**

**Frame 109** is the packet that contains the TLS ClientHello.

The ClientHello message is not sent by itself; it's encapsulated within a CRYPTO frame inside this first QUIC Initial packet. This is a core feature of QUIC's design, merging the transport and cryptographic handshakes.

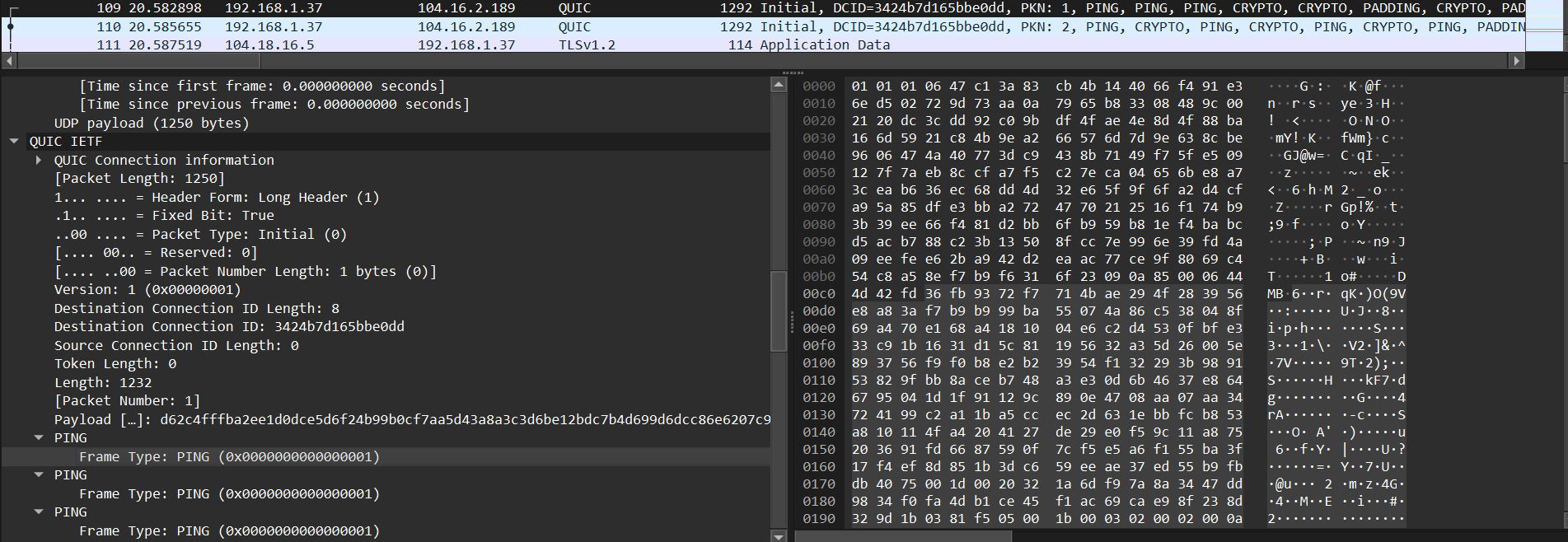
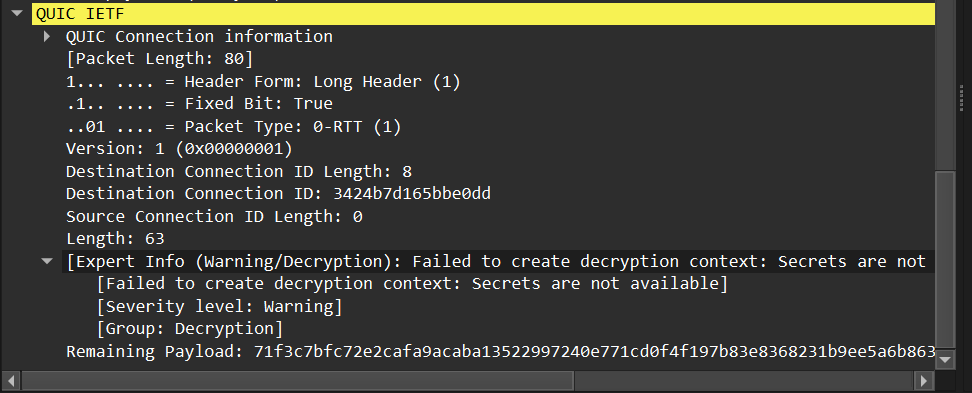


**4. Which QUIC version is used in your trace?**

The trace uses **QUIC version 1**, identified by the hex value 0x00000001.

This version is negotiated in the first Initial packet (Frame 109) and is the standardized version of QUIC defined by the IETF in RFC 9000.

| **Field** | **Value / Notes** |
| --- | --- |
| QUIC Version | 0x00000001 |
| Protocol | IETF QUIC v1 |

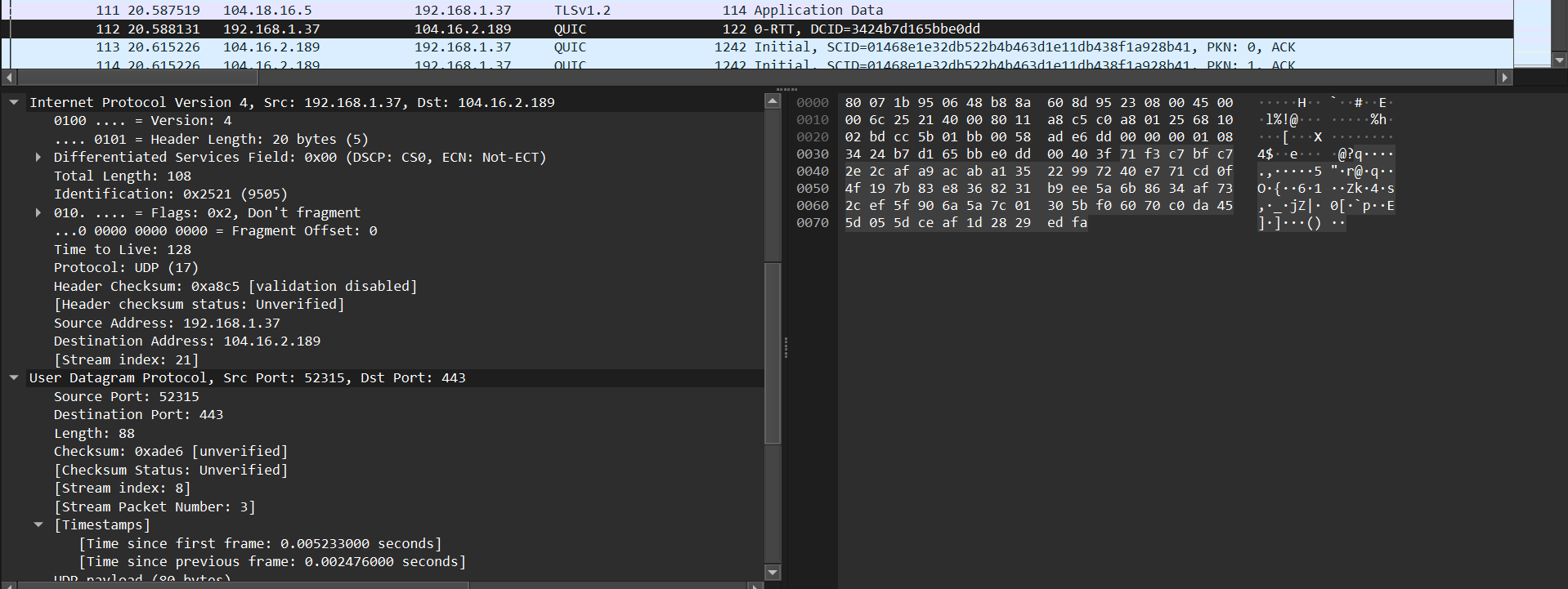
 

**5. Locate the packet where 0-RTT or 1-RTT keys are first used.**

The first packet to use 0-RTT (Zero Round-Trip Time) keys is **Frame 112**.

This packet is explicitly typed as a 0-RTT Protected packet. It is sent by the client immediately after its initial packets to transmit encrypted "early data" without waiting for the server's full handshake response, dramatically reducing connection latency.

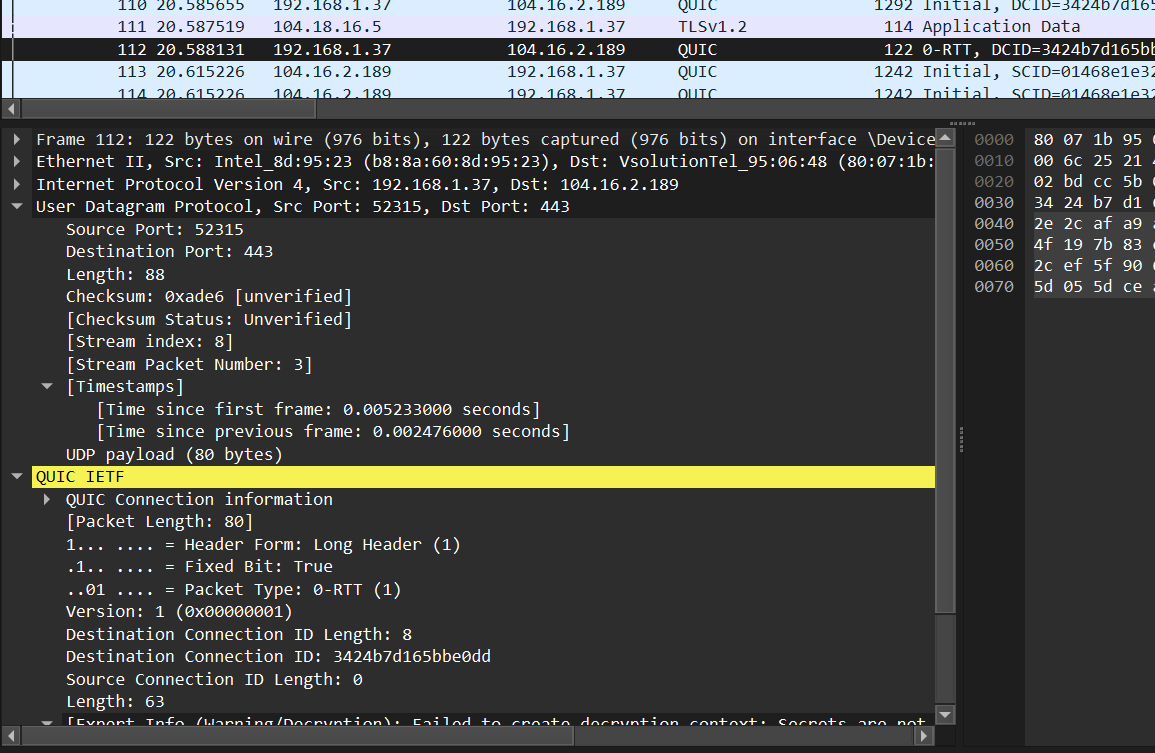
| **Field** | **Value** |
| --- | --- |
| Frame | 112 |
| Packet Type | 0-RTT Protected |
| Direction | Client → Server |
| Notes | The client attempted to send early application data using 0-RTT keys immediately after Initial. If the server accepts, it can reduce latency by avoiding a full handshake. |



**6. Find the first packet that carries application data (HTTP/3). How does this differ from HTTP over TCP?**

The first packet carrying application data (an HTTP/3 request) is **Frame 112**.

This packet uses 0-RTT encryption to send a STREAM frame containing an HTTP/3 HEADERS frame (the GET / request for the website).



This approach is fundamentally different from and faster than HTTP over TCP:

* **QUIC (with 0-RTT):** The client sends its first HTTP/3 request in the very first flight of packets (Frame 112). No round trips are wasted on handshakes before application data can be sent.
* **TCP + TLS:** A client must first complete the TCP handshake (1 RTT) and then the TLS handshake (1-2 RTTs). Only after these handshakes are fully finished can the first HTTP request be sent. This multi-step process introduces significant initial latency that QUIC's 0-RTT feature eliminates.