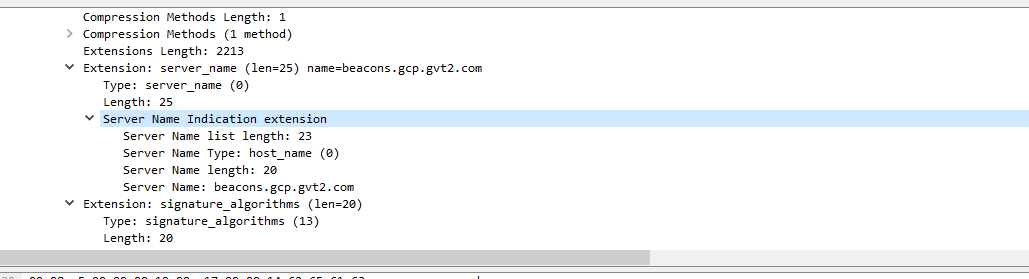
QUIC TRACES WIRSHARK

WEBSITE(YOUTUBE)

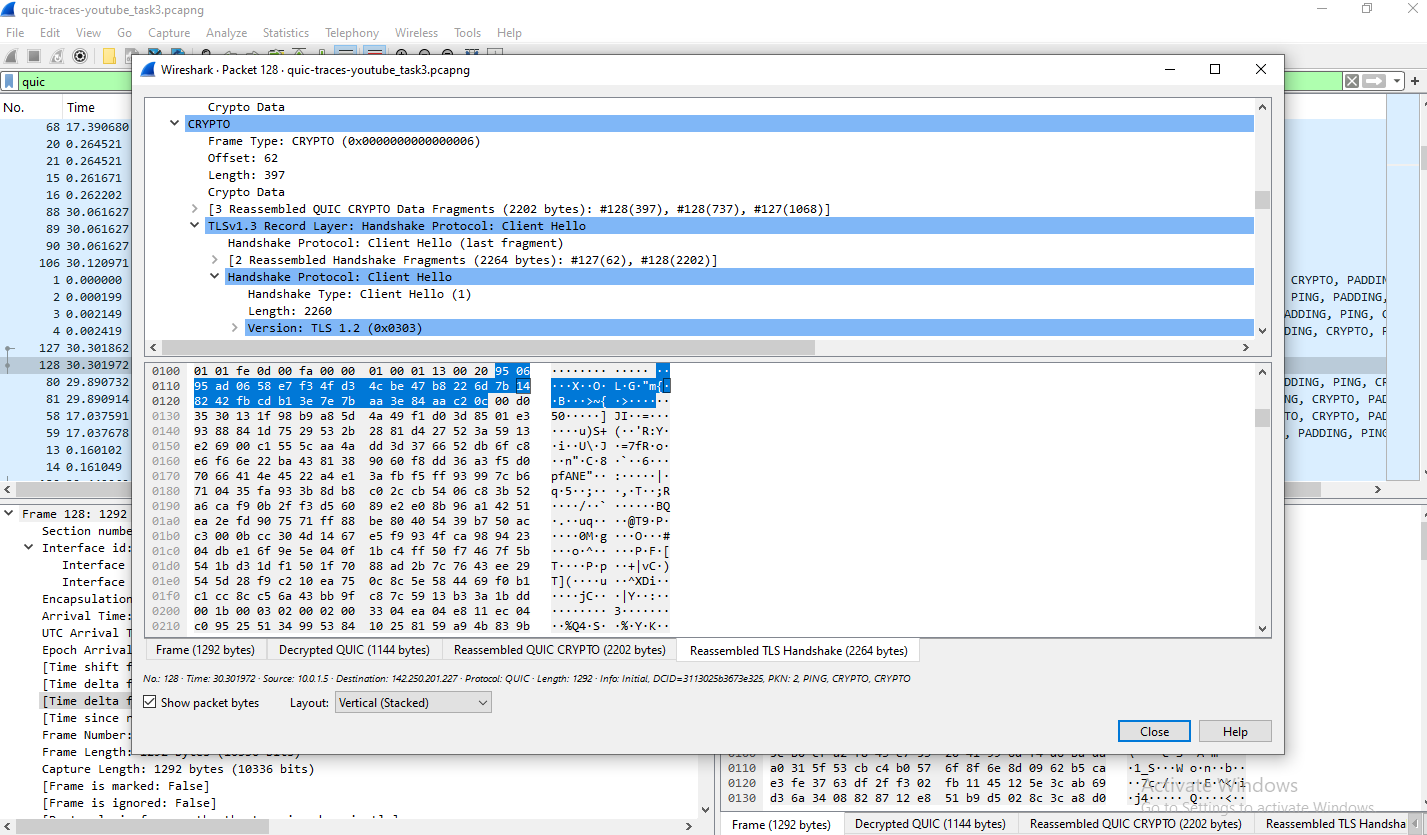
1. What is the name of website?

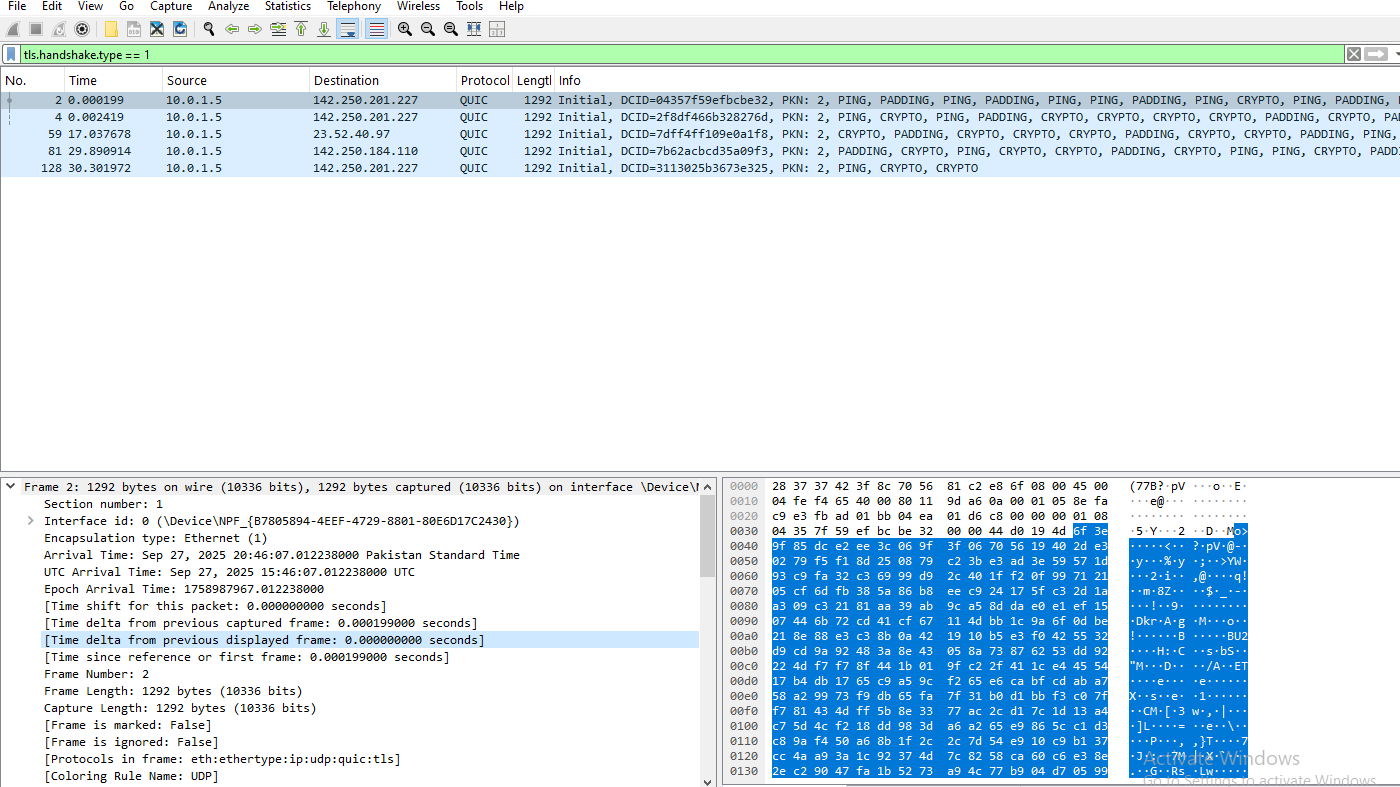
Youtube

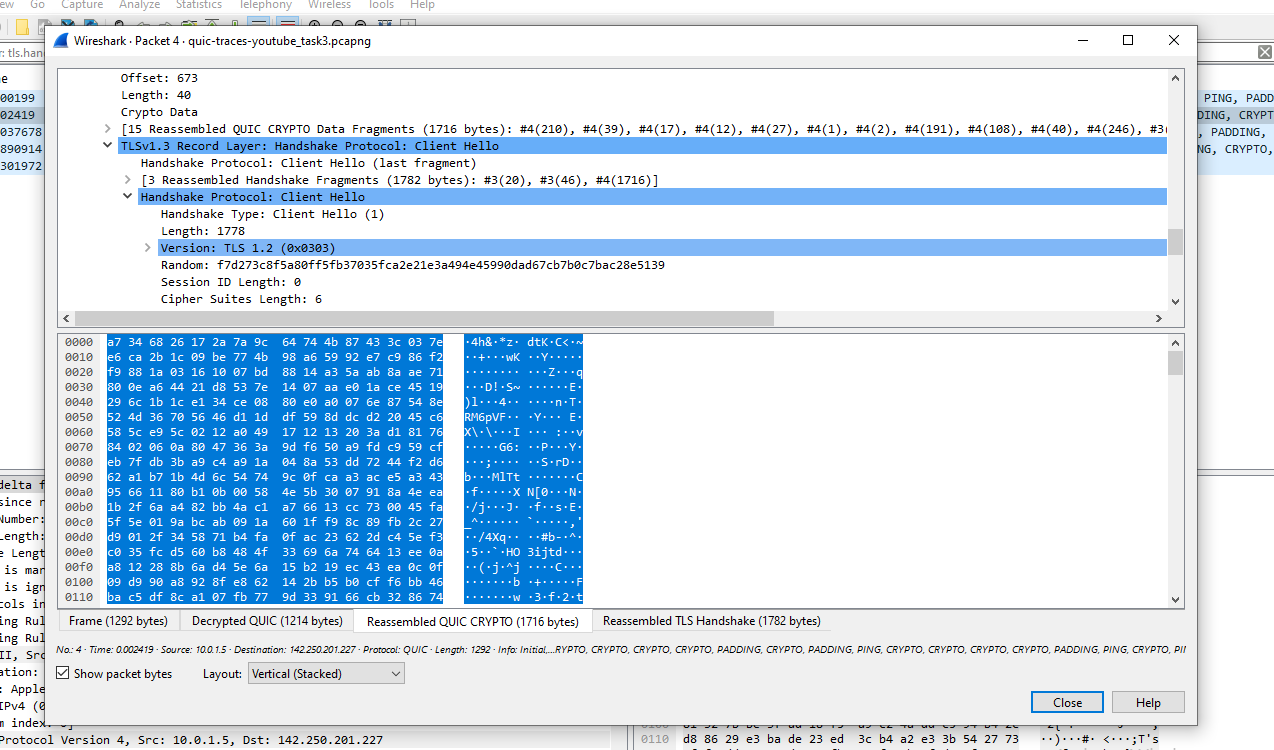




2. Find the packet that contains the Initial QUIC handshake. What information is exchanged here?



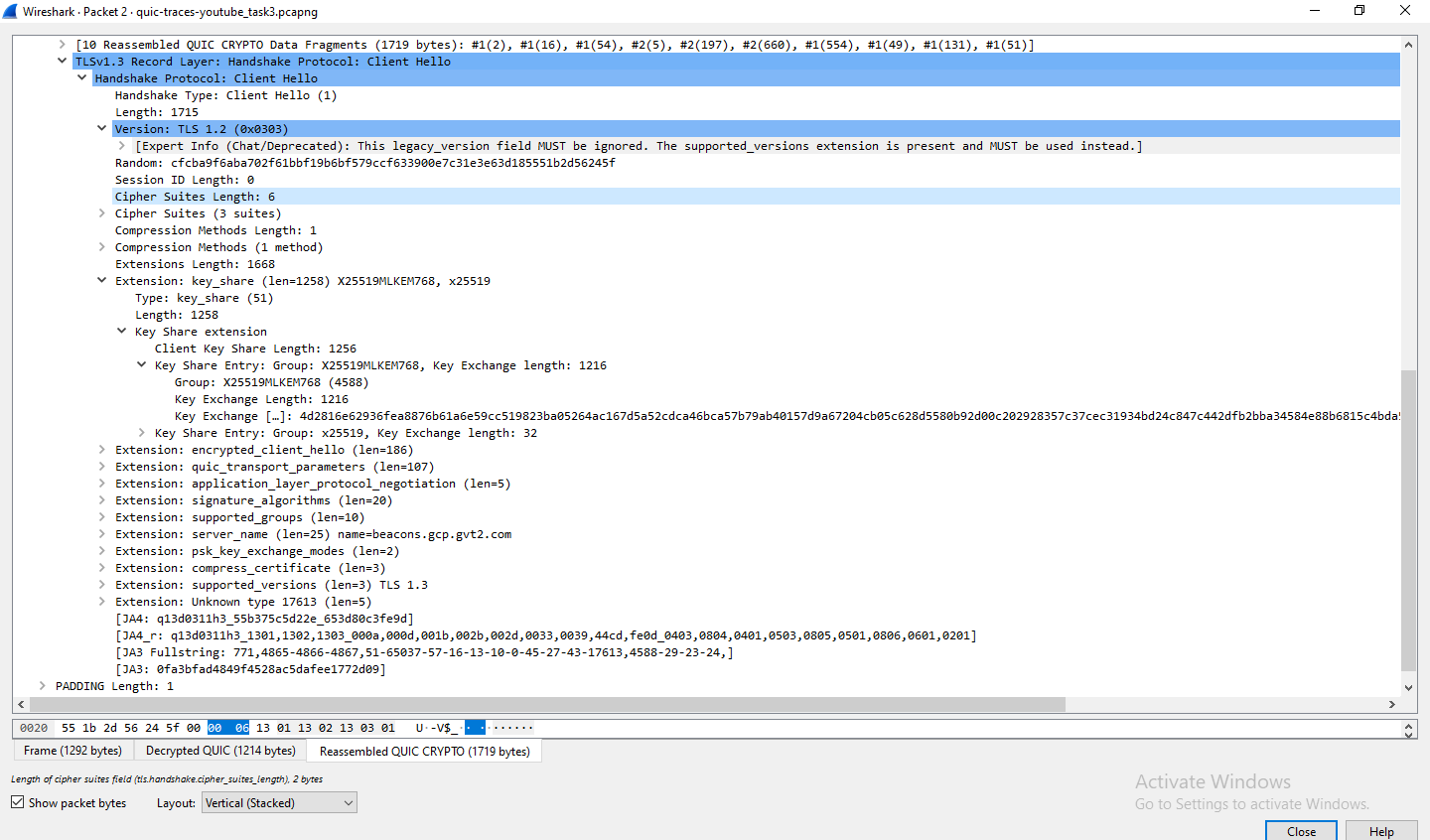


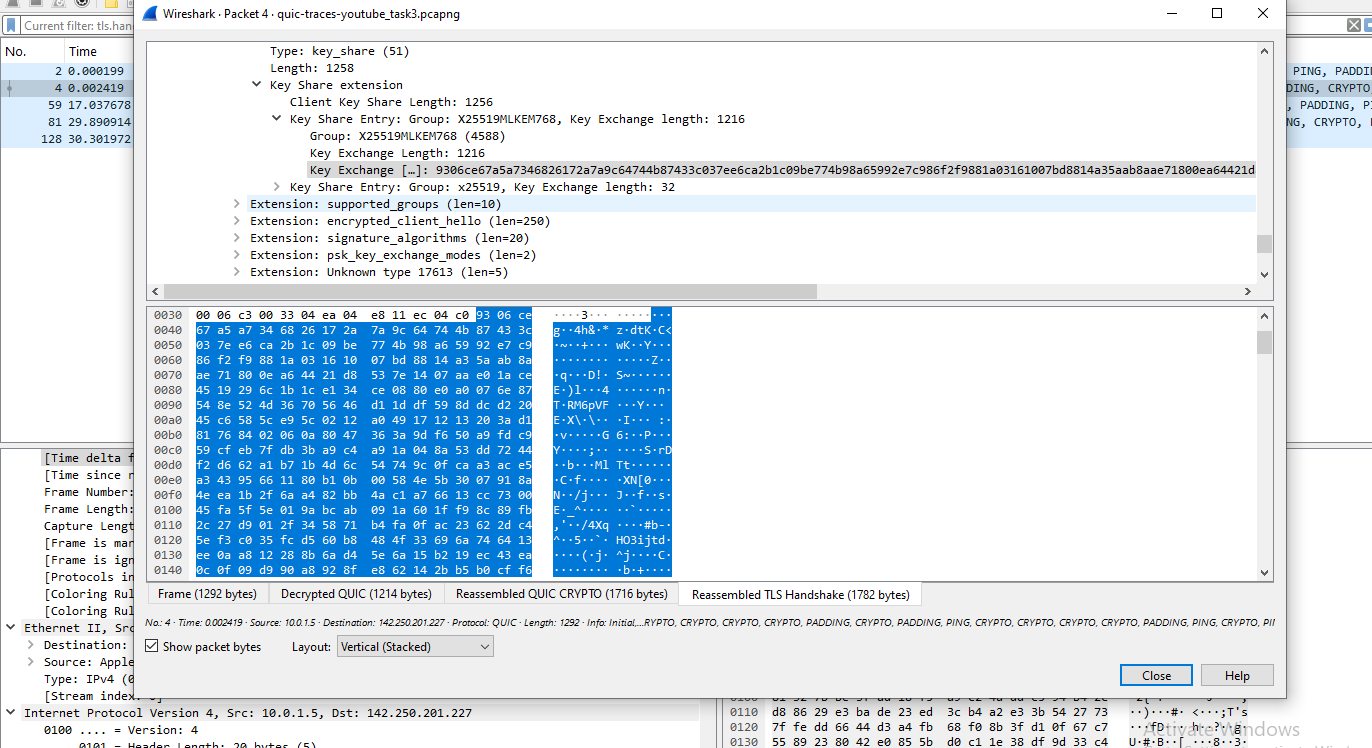


Info exchanged:

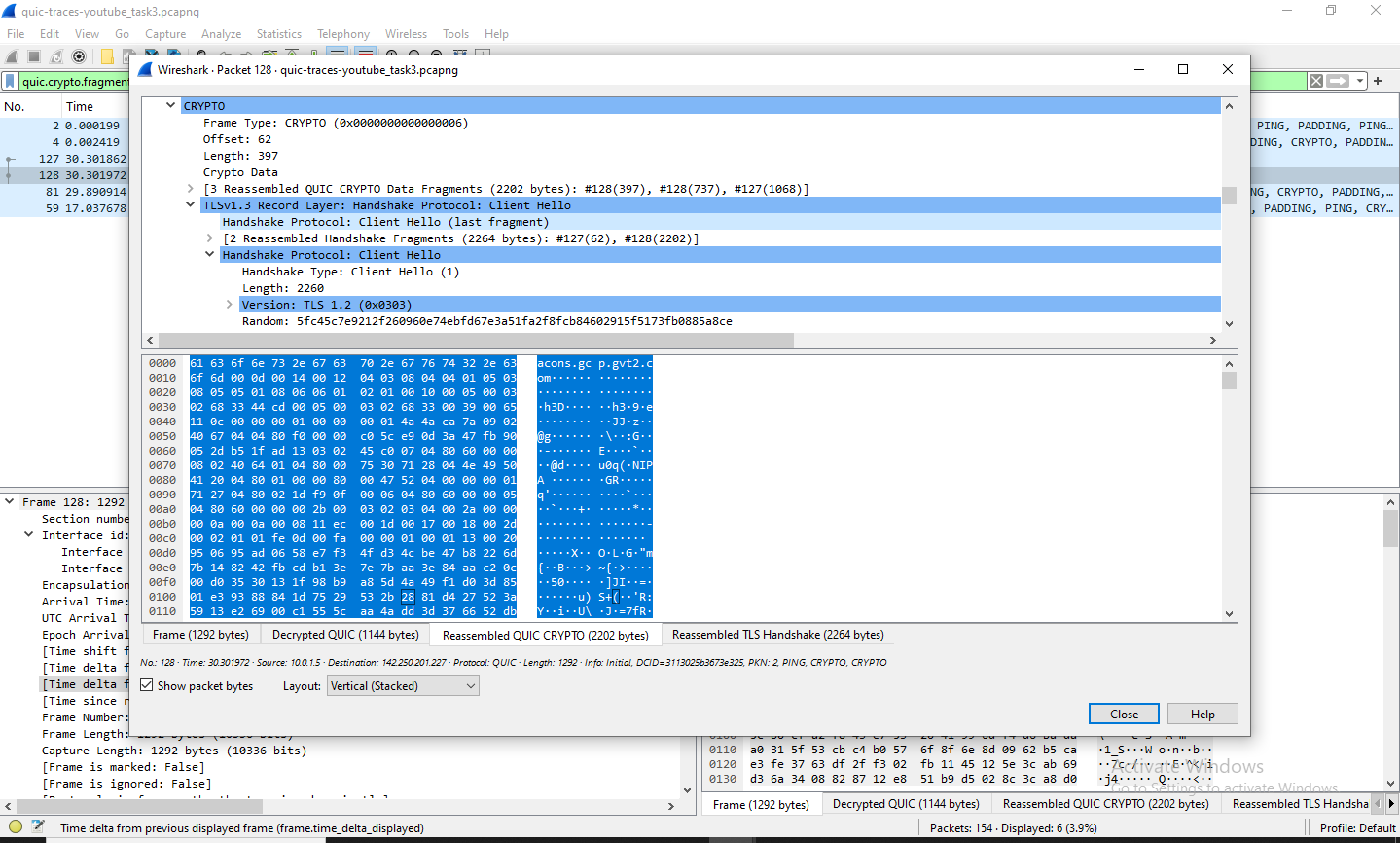
Inside **Client Hello** we can read the actual fields sent by the client:

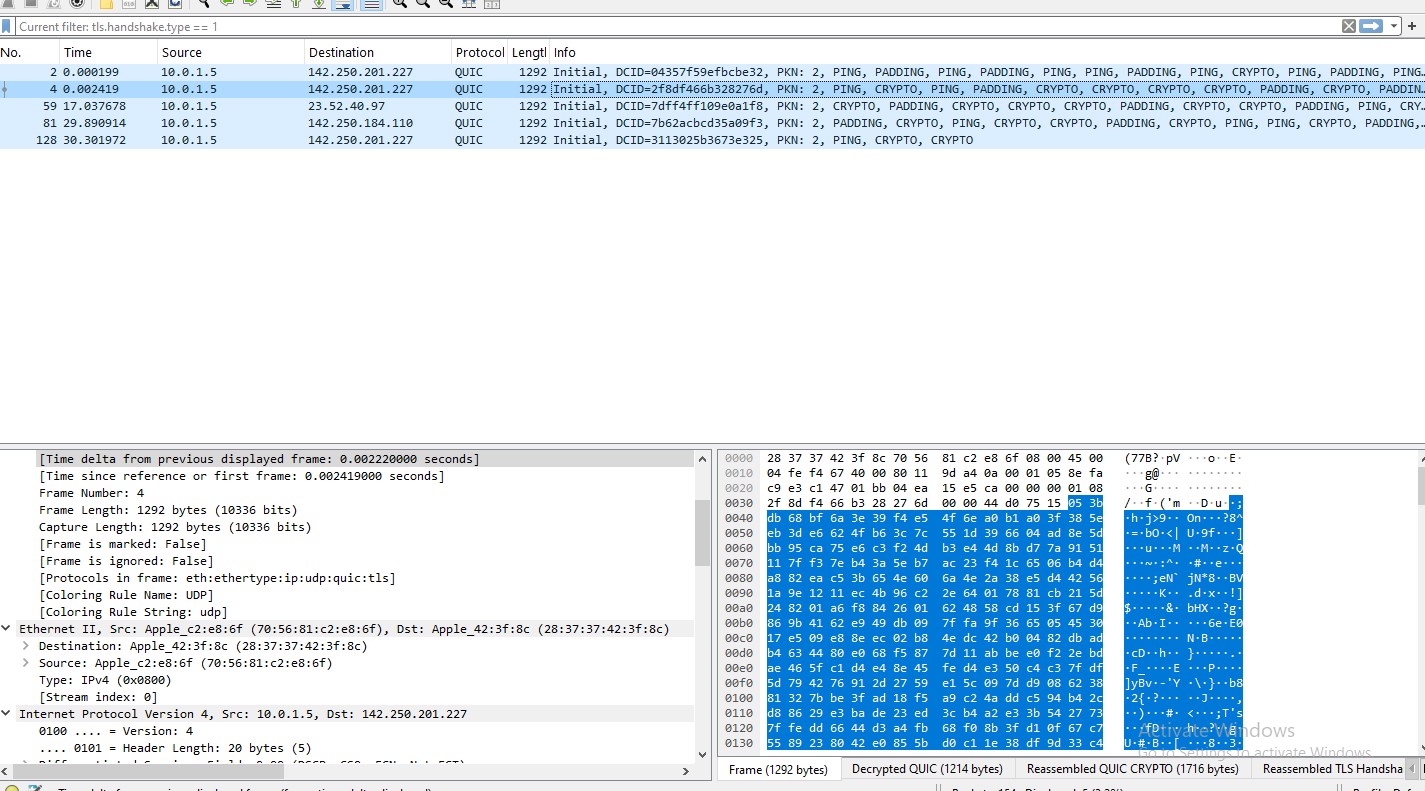
|  |  |
| --- | --- |
|  |  |
| **Server Name Indication (SNI)** | The hostname the client wants (e.g. beacons.gcp.gvt2.com). |
| **ALPN (Application Layer Protocol Negotiation)** | Protocols supported (e.g. h3 for HTTP/3). |
| **Cipher Suites** | Encryption algorithms the client supports. |
| **Supported Groups / Key Share** | Parameters for key exchange (Elliptic Curve, etc.). |
| **Session Ticket / PSK** | Data used if the client attempts 0-RTT resumption. |
| **Random** | Client random value for key generation. |

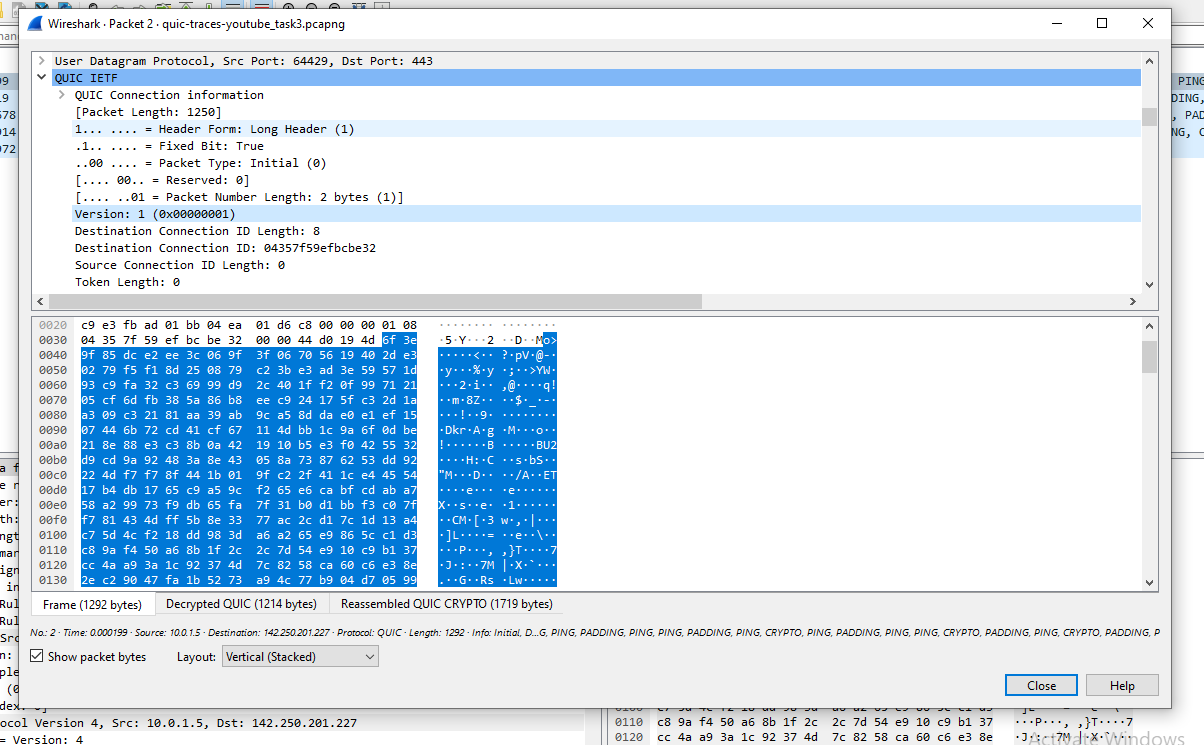




3. Identify the QUIC packet that contains the TLS ClientHello (QUIC embeds TLS handshake inside QUIC).

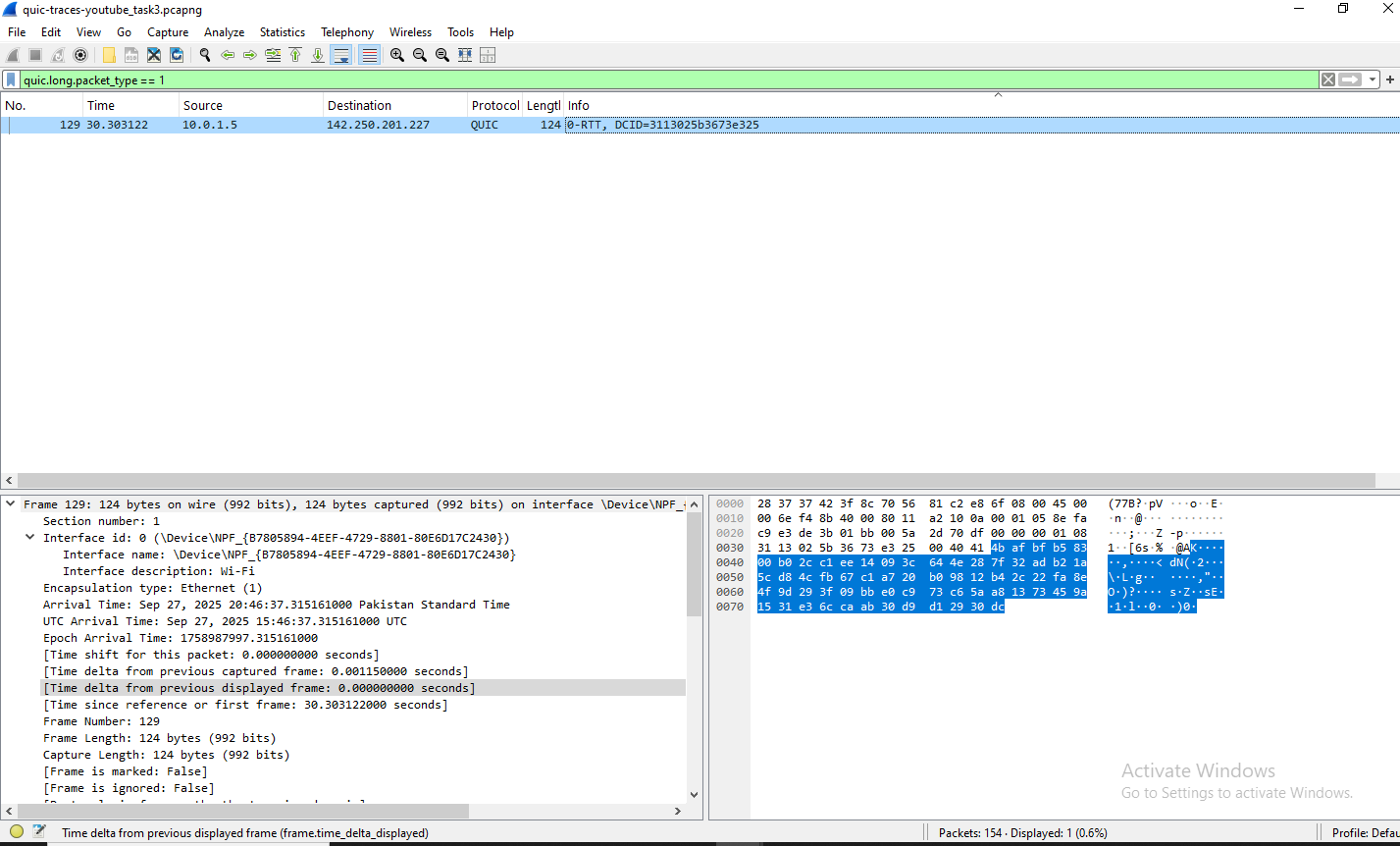


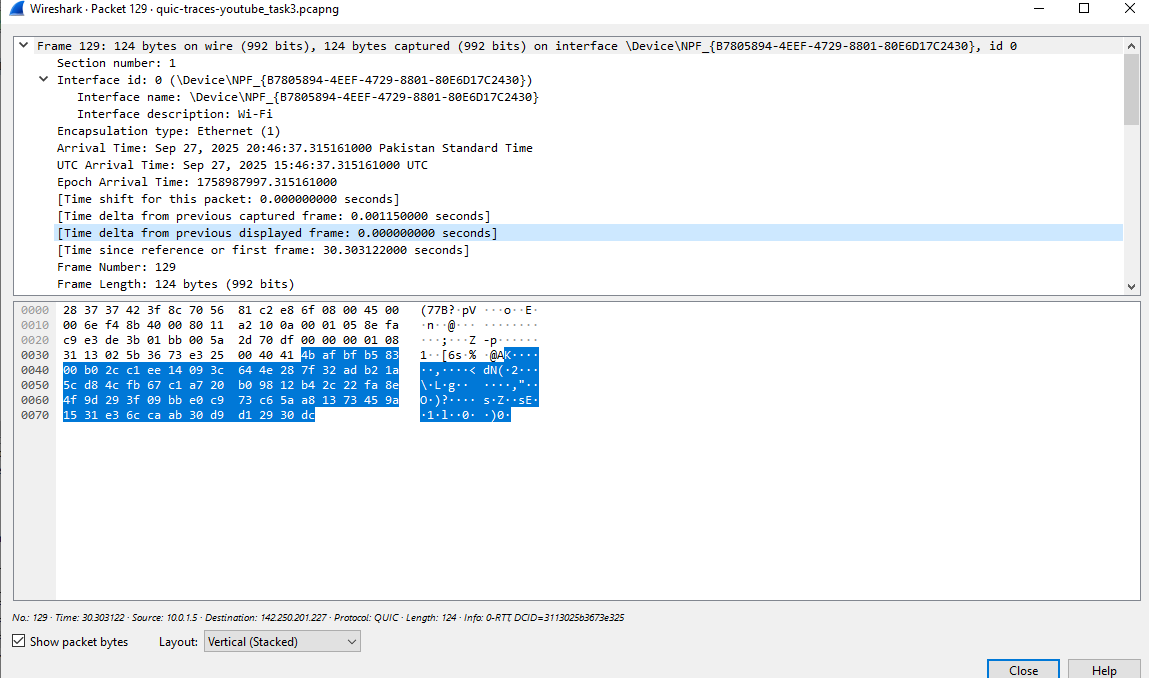


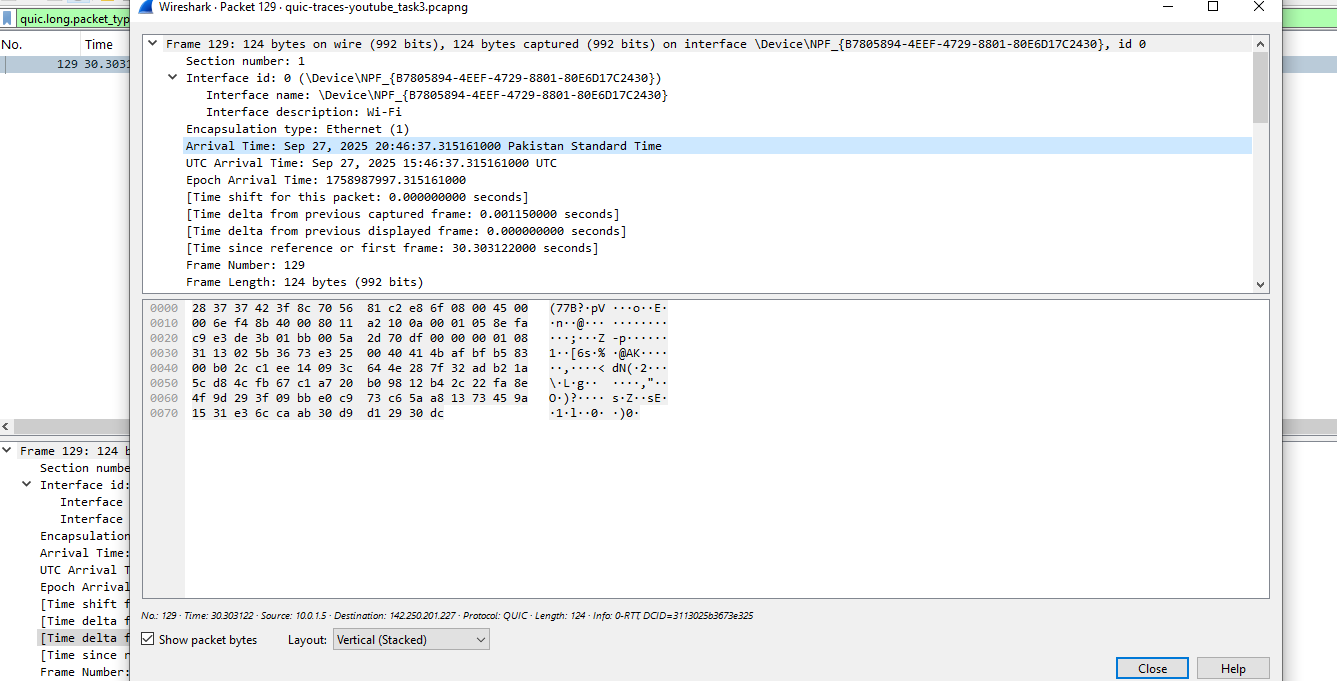
4. Which QUIC version is used in your trace? 

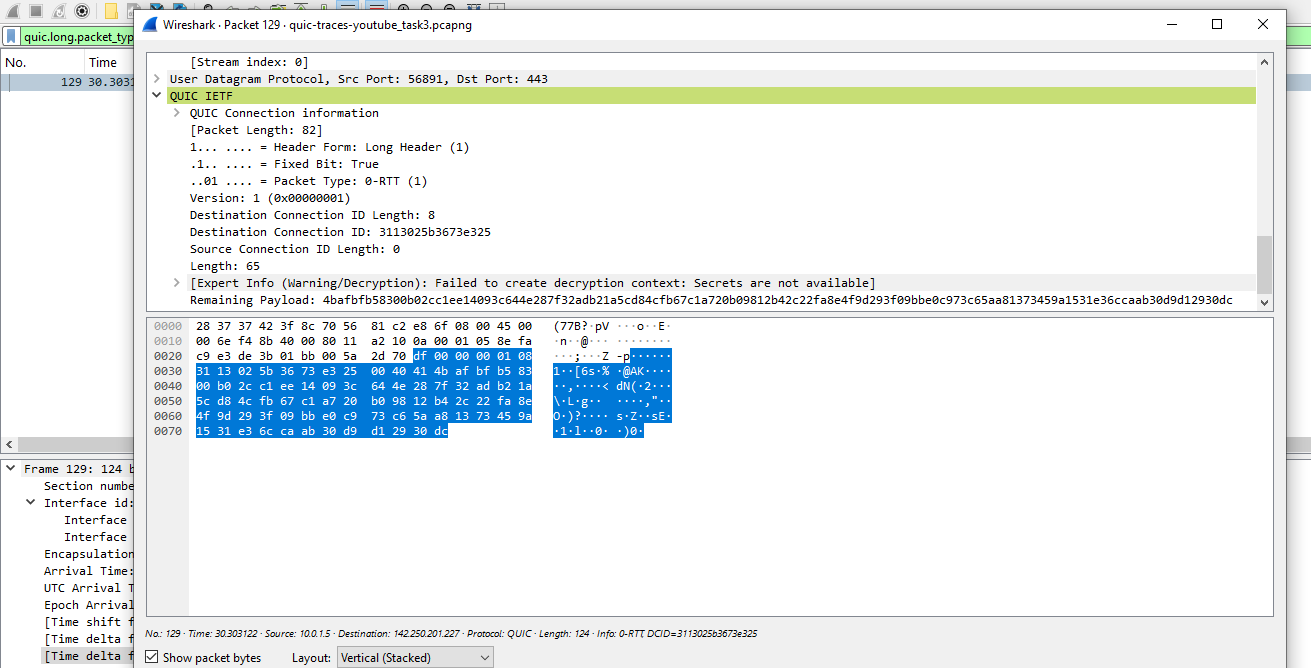
5. Locate the packet where 0-RTT or 1-RTT keys are first used?

* 1. 0-RTT packets will say *Long Header* with *Packet Type: 0-RTT (0x01)*.
  2. 1-RTT packets will say *Short Header* and show *Protected Payload* or *STREAM* frames (if you have keys for decryption).



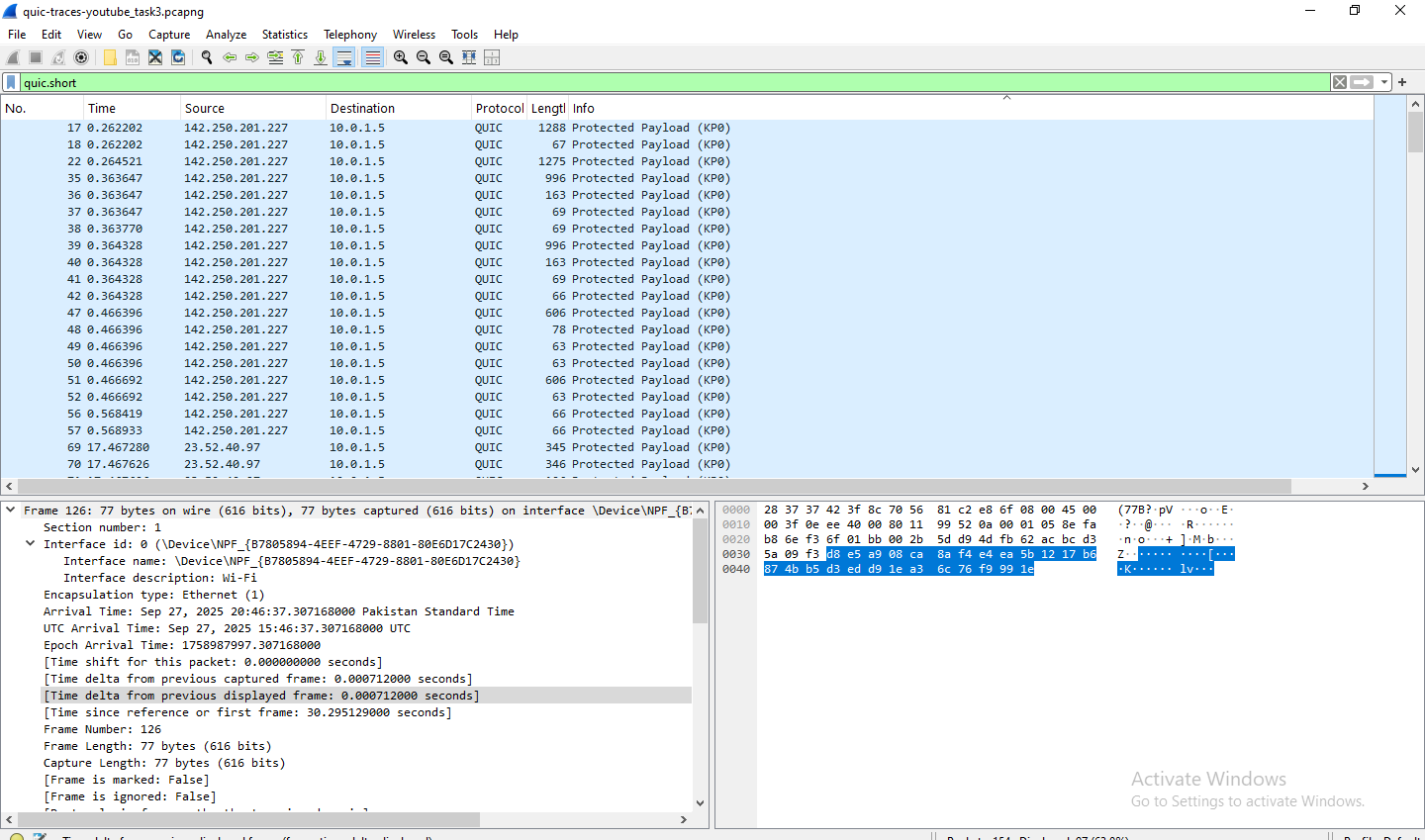




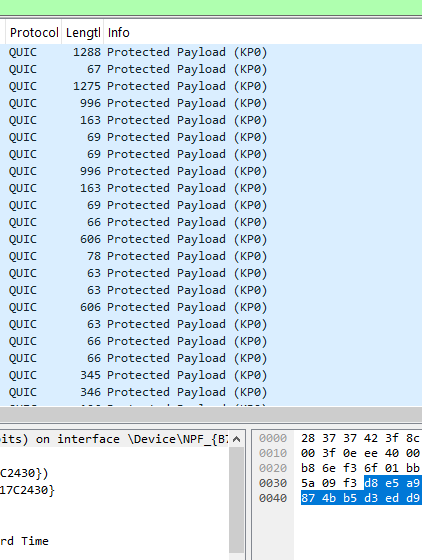


1-RRT

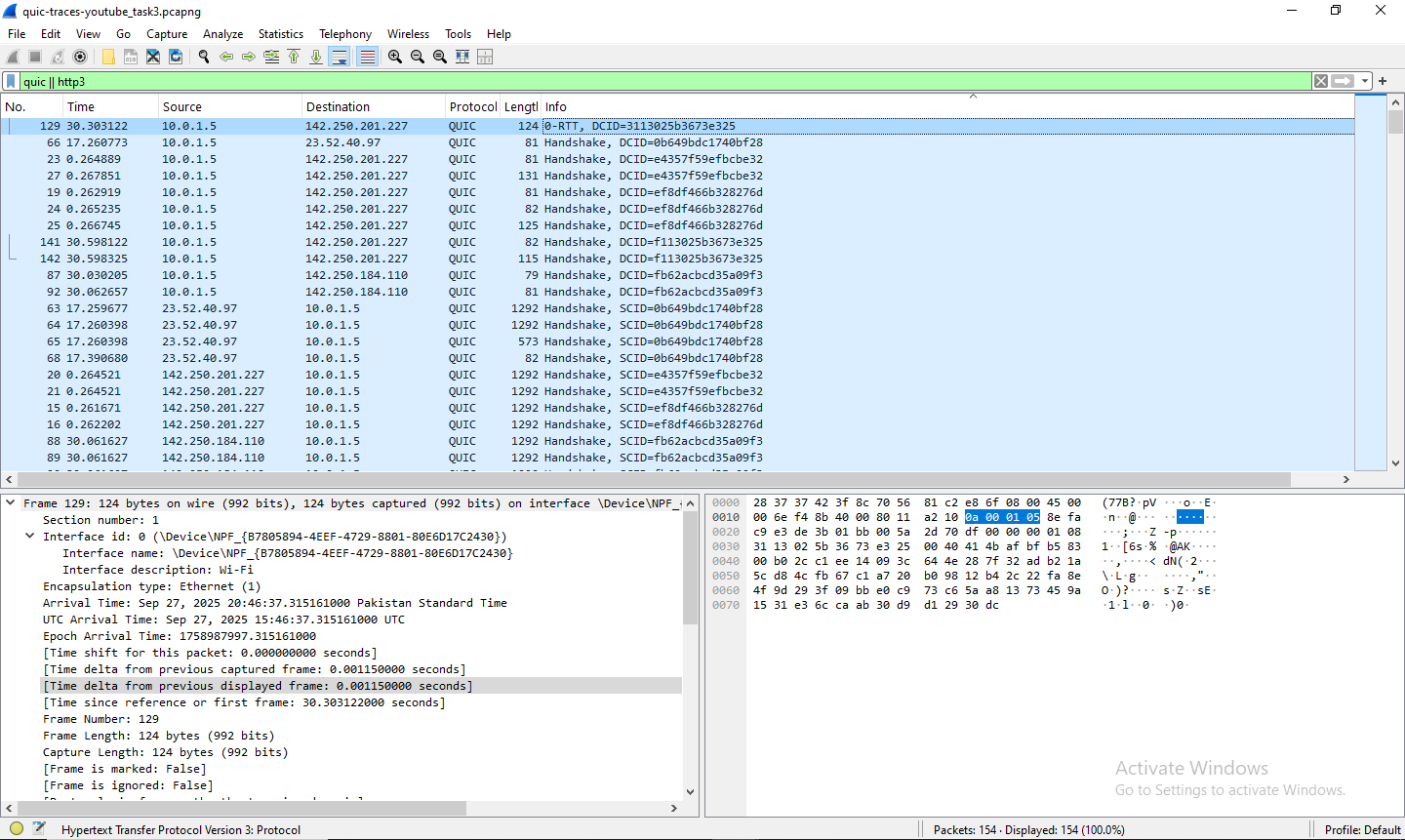
QUIC **short-header** packets carry **1-RTT protected** data.

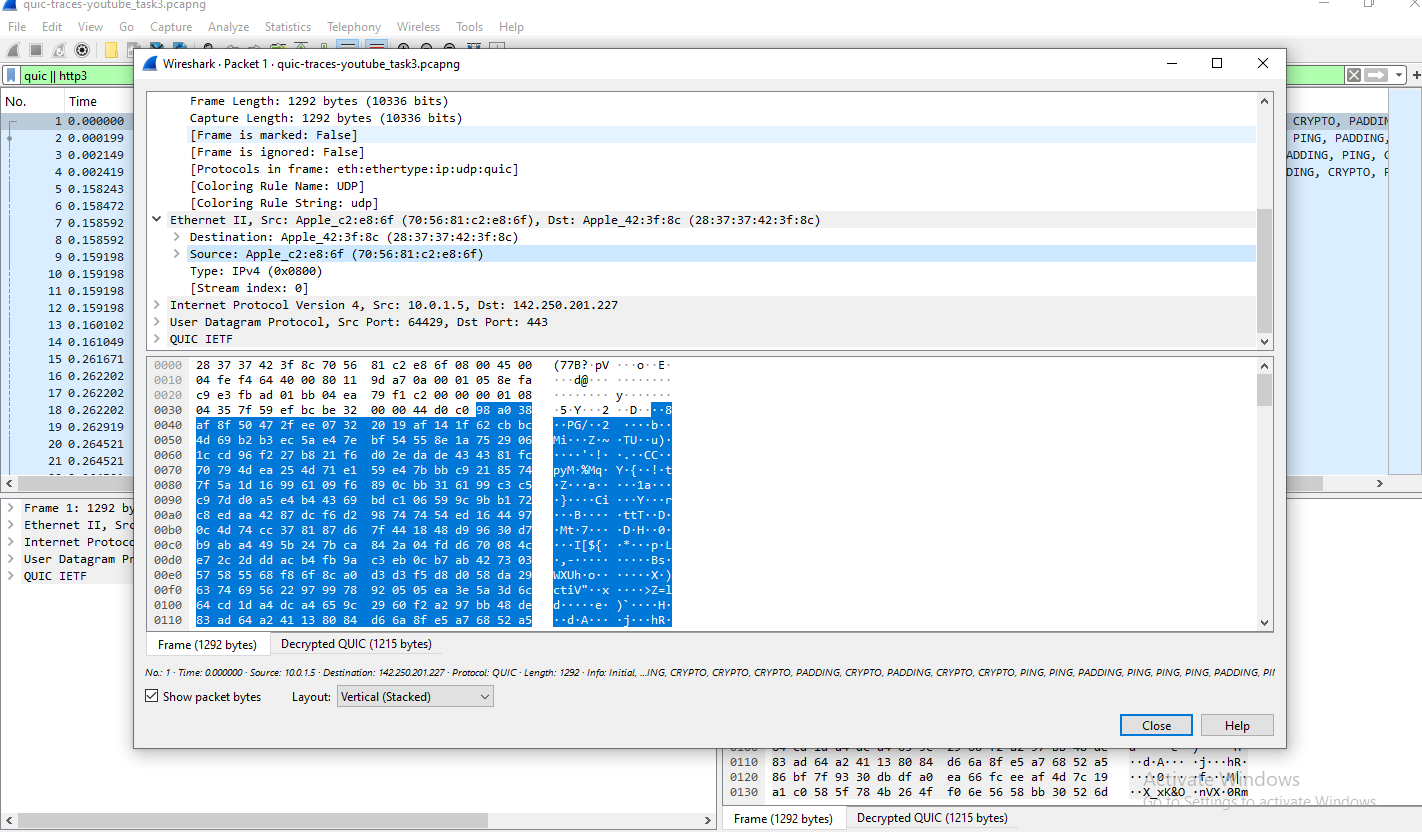


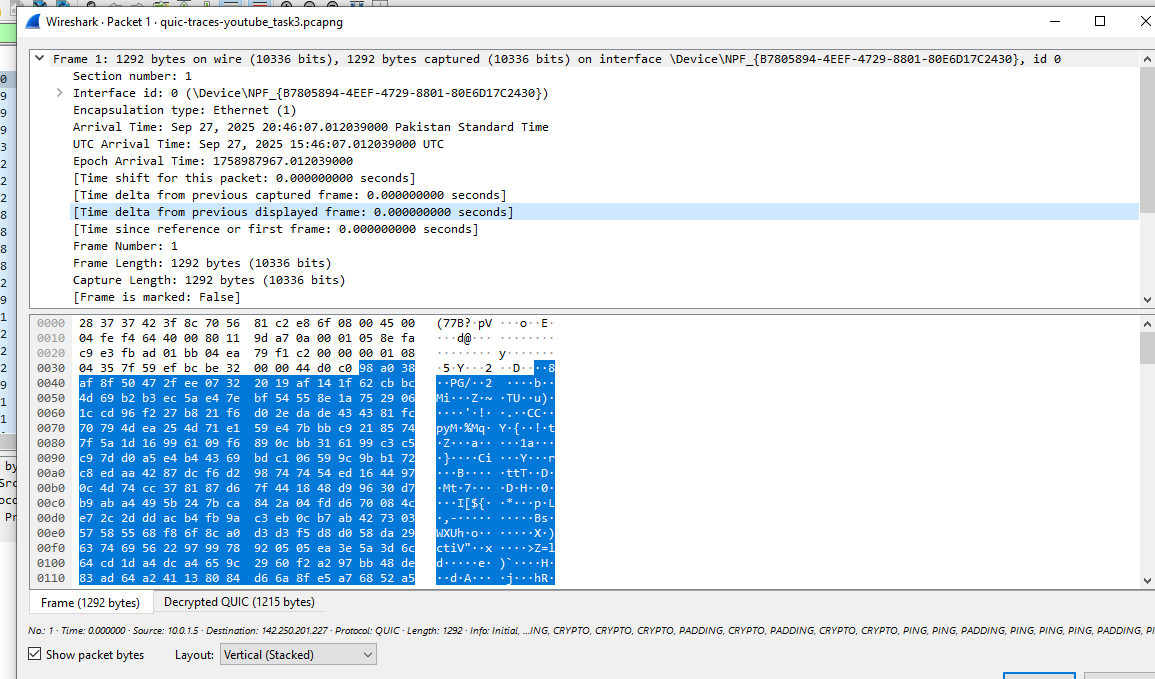
All data is protected

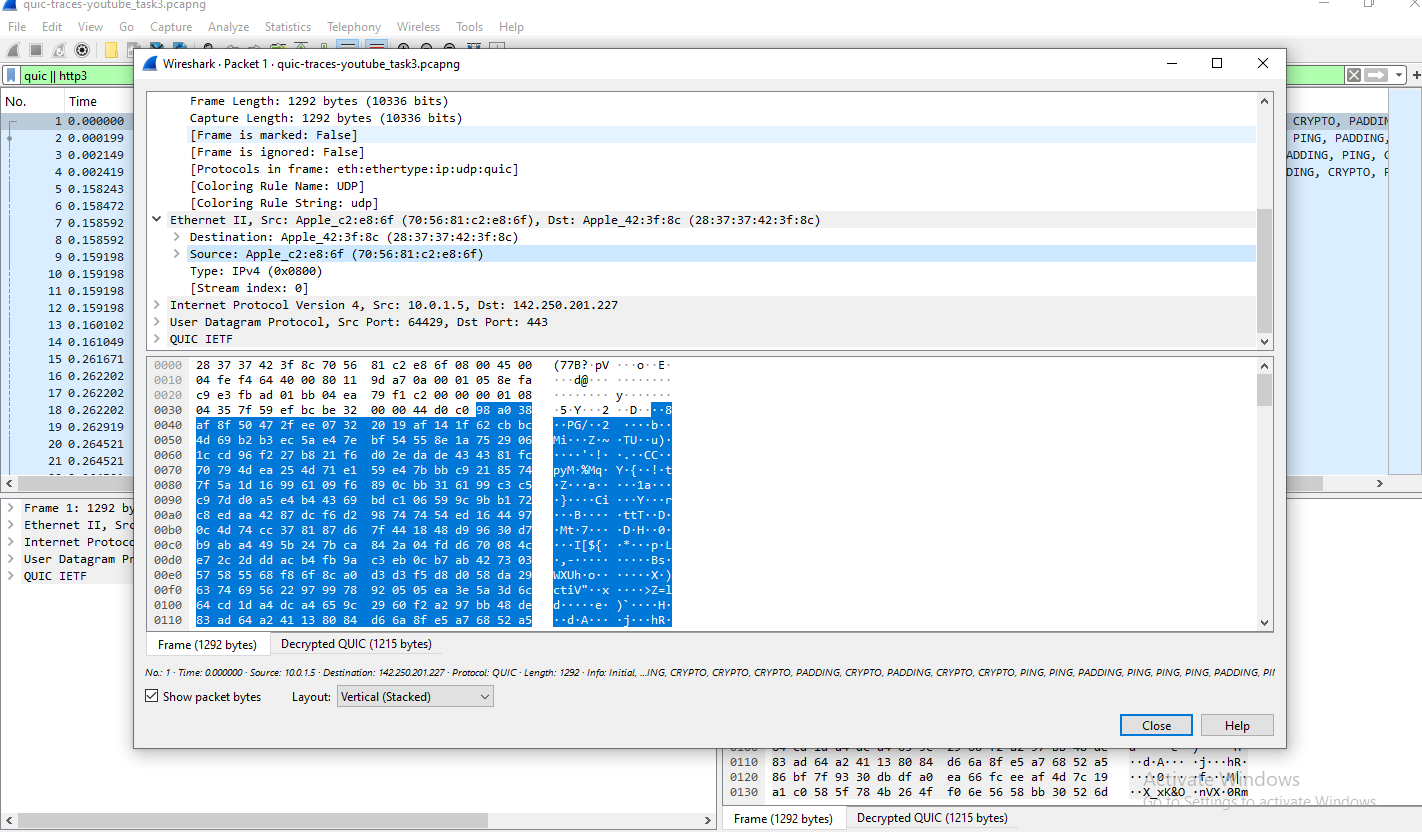


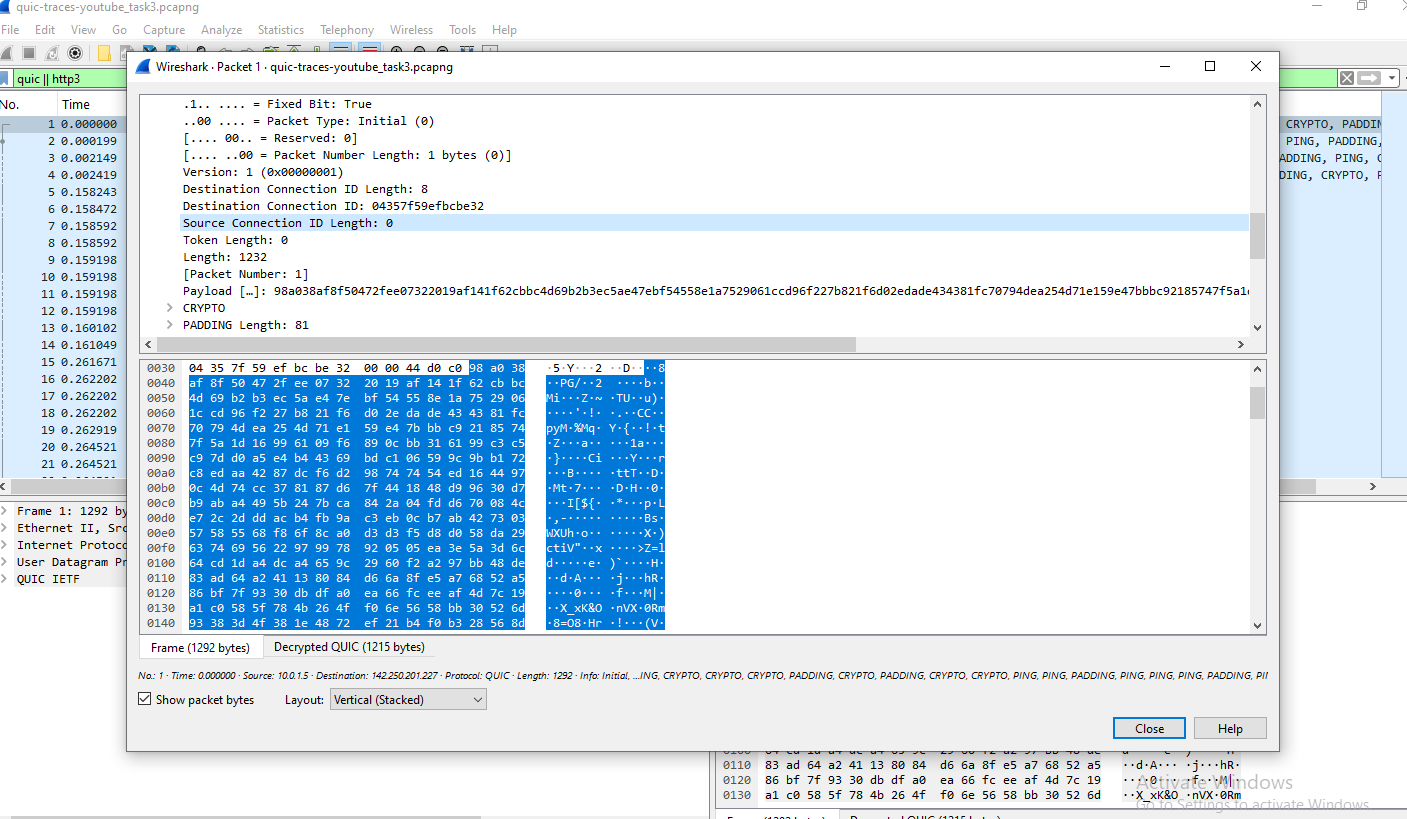
6. Find the first packet that carries application data (HTTP/3). How does this differ from HTTP over TCP











**How this differs from HTTP over TCP (HTTPS):**

* **Transport:**

HTTP/3 runs over **QUIC (UDP)**

HTTP over TLS (HTTPS) runs over **TLS over TCP**. (RFC 9114 / RFC 9000).

* **TLS integration:**

QUIC integrates TLS 1.3 directly into the transport (handshake + encryption are part of QUIC) instead of TLS over an established TCP connection.

* **Multiplexing & head-of-line blocking:**

QUIC provides independent streams with per-stream flow control and avoids TCP’s head-of-line blocking — a lost QUIC packet only stalls streams that actually used that packet, not all streams on the connection.

* **Latency:**

QUIC can achieve 0-RTT early data (if allowed) and generally needs fewer round trips for new connections than TLS+TCP.

* **Visibility in Wireshark:**

HTTP/3 frames are inside QUIC STREAM frames

HTTPS (HTTP/2 or 1.1) appears as TLS Application Data over TCP and is handled by the TLS dissector + HTTP dissector.