**TASK # 06:**

**For the QUIC based website access**

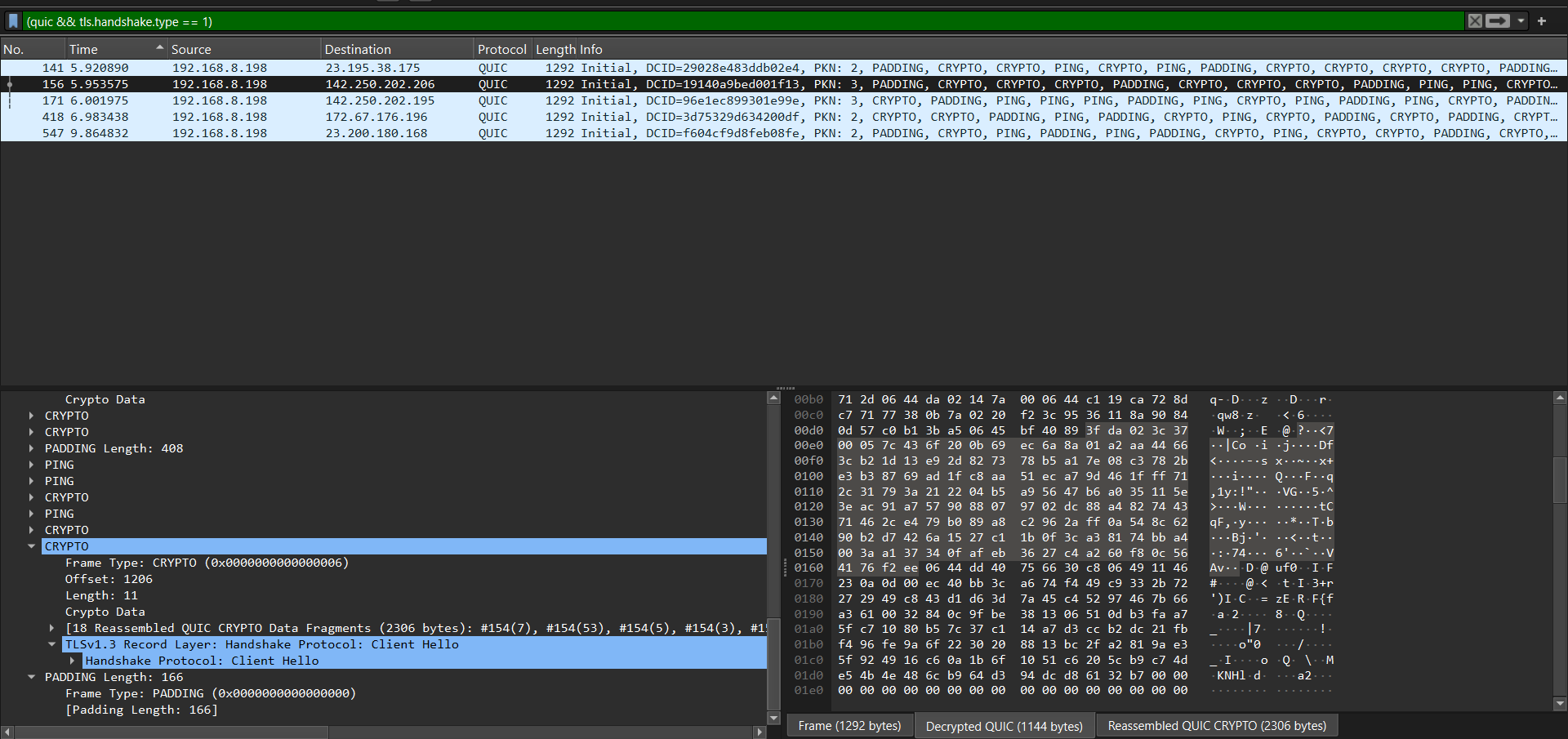
1. **What is the name of website?**

The website which we find that is working on **quic** is <https://www.pearson.com/>

**A screenshot of a computer

AI-generated content may be incorrect.**

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

****

**TLS Handshake** includes

* **Supported TLS version** (always TLS 1.3 in QUIC).
* **Supported cipher suites** (e.g., AES-128-GCM, AES-256-GCM, ChaCha20-Poly1305).
* **Supported key exchange groups** (e.g., X25519, P-256).
* **Supported signature algorithms**.
* **ALPN extension** → tells the server it wants HTTP/3 (h3).
* **SNI (Server Name Indication)** → the website’s domain name.
* Random + session resumption data (if 0-RTT is attempted).

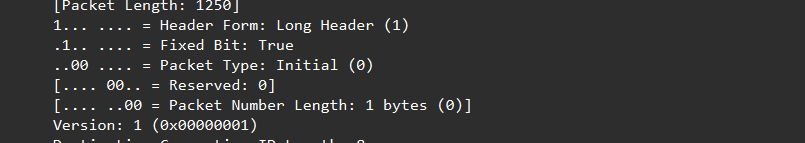
1. **Identify the QUIC packet that contains the TLS ClientHello (QUIC embeds TLS handshake inside QUIC).**

**A screen shot of a computer

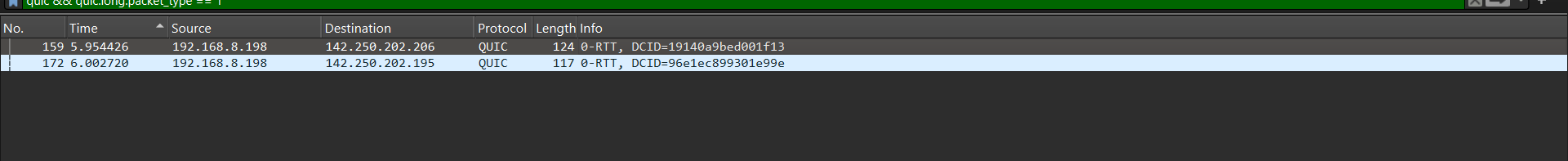
AI-generated content may be incorrect.**

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

**Version 1**

****

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

****

**Packet number 159**

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

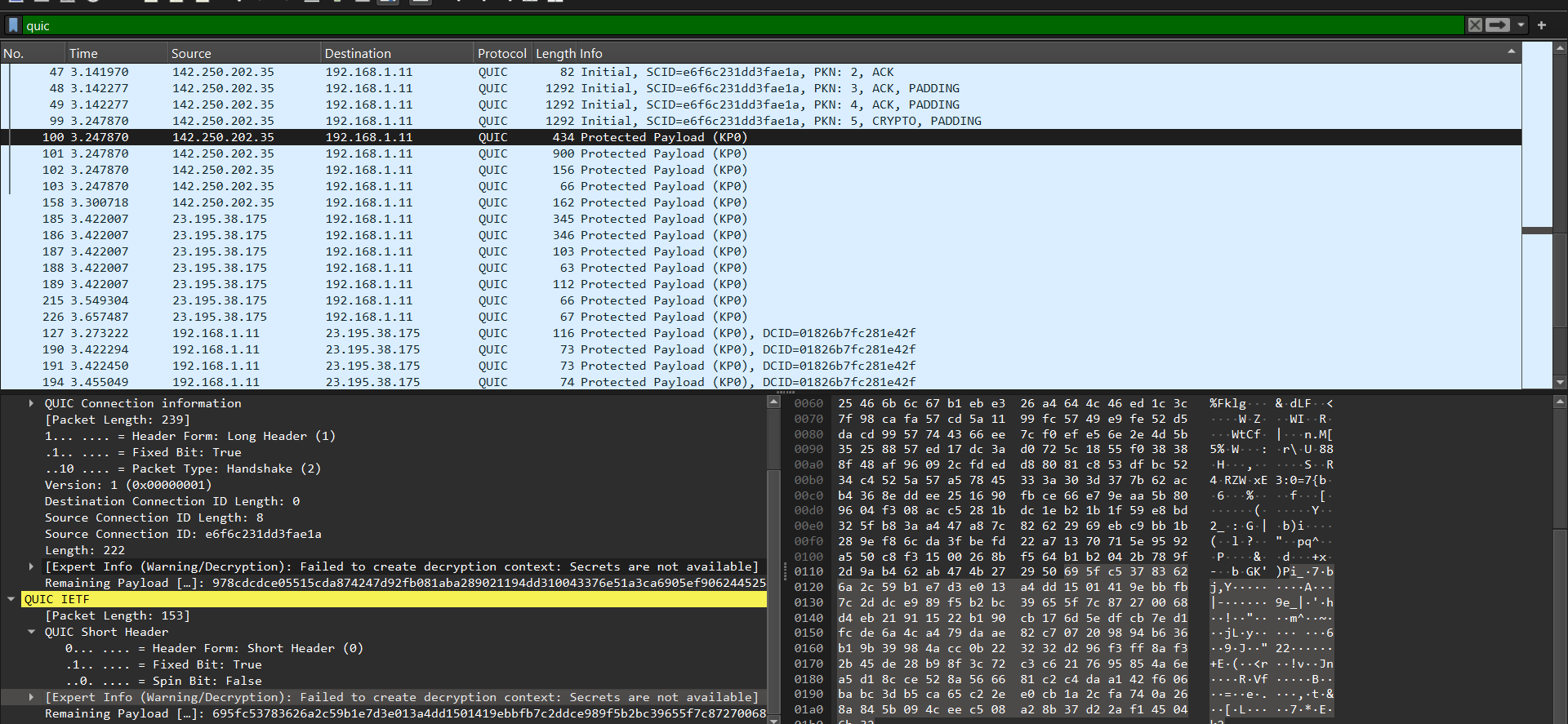
**LOCATING APPLICATION DATA PACKETS**

**Handshake ends at CRYPTO (0x06) frames**

* + These carry ClientHello, ServerHello, EncryptedExtensions, Finished, etc.
  + Once you no longer see CRYPTO frames, the handshake is over.

**Next packets = protected application data**

* + Look for QUIC packets with **Long Header → 1-RTT** or **Short Header**.
  + These packets carry HTTP/3 data (application data), but since they’re encrypted, Wireshark can’t label them as STREAM without TLS secrets.



**HTTP/3 over QUIC:**

* Runs directly on UDP, not TCP.
* QUIC provides its own encryption (TLS 1.3 inside QUIC handshake).
* Multiplexing: Multiple streams run independently → one stream’s loss doesn’t block others.
* Faster start: With 0-RTT/1-RTT, application data can be sent immediately after handshake

**HTTP/2 or HTTP/1.1 over TCP:**

* Runs on TCP (a connection-oriented protocol).
* Requires a full 3-way TCP handshake + TLS handshake before data flows.
* Head-of-line blocking: If one packet is lost, all streams on that TCP connection are stalled until it’s retransmitted.