National College of Ireland

The Performance Impact of Post-Quantum Cryptography on User Experience – Configuration Manual

MSc Research Project
MSc Cybersecurity

Damien Nalty
Student ID: x23272414

School of Computing
National College of Ireland

Supervisor: Raza Ul Mustafa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: .....Damien Nalty.....

Student ID: .....x23272414.....

Programme: .....MSCCYBE_JAN24_OL..... **Year:** 2025-2026

Module: .....Research Project Practicum II.....

Supervisor: .....Dr. Raza Ul Mustafa.....

Submission Due Date: .....28th January 2026.....

Project Title: The Performance Impact of Post-Quantum Cryptography on User Experience...

Word Count: .....5286..... **Page Count**.....30.....

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Damien Nalty

.....

Date: 24/01/2026

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Damien Nalty
X23272414

1 Introduction

This document serves as the configuration manual for Cassini, a Post-Quantum TLS test framework developed to support the thesis: "*The Performance Impact of Post-Quantum Cryptography on User Experience.*"

The manual describes the specific hardware and software environments, and the technical architecture of the tool itself. It also outlines the experimental setup and execution workflows used to generate the performance metrics associated with the research and analysis presented in the thesis report.

2 System Configuration

This section provides a detailed description of the hardware and software components; overall architecture and topology used to execute the testbed and retrieve the performance results. The results are subsequently used to facilitate the analysis and derive insights on the adoption of post-quantum cryptography and migration from classical cryptography for standard web-based users and applications across the security levels defined.

2.1 Hardware and Operating System Environment

The following specification outlines the hardware[1] and software components; modules and drivers used to build and deploy the testbed.

[1] *The hardware specification used describes a system that was sufficient to facilitate the deployment of the developed test harness, along with supporting the orchestration of a commercial test tool to enrich the test data with real-world traffic emulation results, to outline how the migration may impact standard web-based users and applications.*

Table 1 – Machine Hardware, Kernel/Operating System and Software dependency Environment

Machine Hardware Specification	**Description:** Mid-Range 2U Rack Server **Product:** PowerEdge R740 (OEM RXL) **Vendor:** Dell EMC Inc. **Memory:**192GiB **CPU:** 2 x Intel® Xeon Gold 6140 2.30GHz **Configuration:** Cores=18 per socket - 36 Cores in total **Network Interfaces:** eno1 10 Gigabit Ethernet X550 eno2 10 Gigabit Ethernet X550 eno3 1 Gigabit Network - I350 eno4 1 Gigabit Network - I350 enp134s0f0np0 40GbE XL710 QSFP+ enp134s0f1np1 40GbE XL710 QSFP+ enp175s0f0np0 100GbE MT27800[ConnectX-5] enp175s0f1np1 100GbE MT27800 [ConnectX-5]
Operating System	Ubuntu 24.04.3 LTS
Backend System Components	=== PACKAGES & VERSIONS === --- Virtualisation Components --- libvirt: libvirtd (libvirt) 10.0.0 QEMU: emulator version 8.2.2 (Debian 1:8.2.2+ds-0ubuntu1.10) virsh: 10.0.0 system bridge-utils sshpass hwloc ethtool vim-tiny --- Docker Core Components --- Docker version 29.0.0, build 3d4129b Docker Compose version v2.40.3 containerd: containerd containerd.io v2.1.5 fcd43222d6b07379a4be9786bda52438f0dd16a1 runc: runc version 1.3.3 --- Docker Engine Details (docker info) --- Server Version: 29.0.0 Storage Driver: overlay2 Logging Driver: json-file Cgroup Driver: systemd Cgroup Version: 2 Runtimes: io.containerd.runc.v2 runc containerd version: fcd43222d6b07379a4be9786bda52438f0dd16a1

	runc version: v1.3.3-0-gd842d771
Software Components	
	See Appendix 5.5 – Software Bill of Materials for full list of software components

2.1.1 Test Utility Architecture - Cassini

The Cassini Test Utility is a lightweight Docker-based, web-driven test utility designed to facilitate comparison data measurements of Classical, Hybrid & Post-Quantum key exchange and signatures. These measurements are used to evaluate user experience and performance across the following NIST Security Levels (Technology, 2025)

- which corresponds to security strength levels of classical algorithms - AES and SHA to quantify and evaluate Post-Quantum Ciphers and Signatures

- Level 1: AES-128 Equivalent
- Level 3: AES-192 Equivalent
- Level 5: AES-256 Equivalent

The Testbed consists of two docker containers designed to execute specific tests using nominated classical, hybrid and post-quantum cryptographic profiles – one container acting as a server entity and the other container acting as the client entity along with UI functions to expose the configuration, results dashboard and results retrieval to the operator for post-analysis.

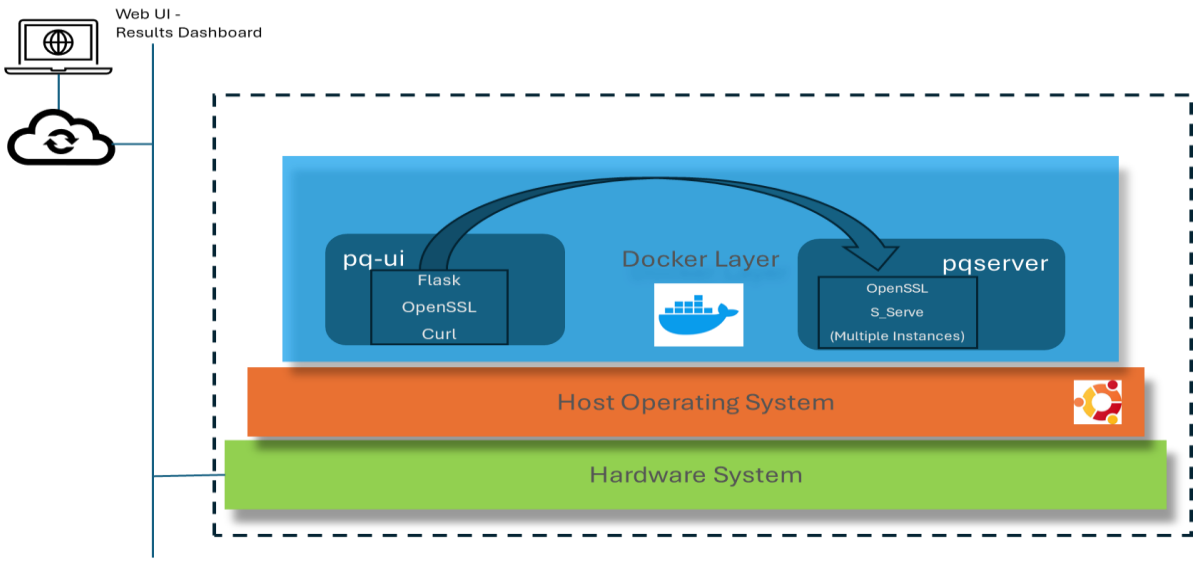

Figure 1 – Cassini PQ-TLS Test Utility - Component Architecture

2.1.2 Build Directory Structure - Cassini

```
Cassini/ ├── docker-compose.yml
         ├── README.md
         ├── LICENCE
         ├── install.sh
         ├── prereqs.sh
         ├── server/
         │   ├── Dockerfile
         │   └── entrypoint.sh
         ├── ui/
         │   ├── Dockerfile
         │   ├── app.py
         │   └── templates/
         │       └── index.html
         └── helper_scripts/
             └── analyse_pcap_final.py
```

The file structure is based around a docker compose script – *docker-compose.yml*, which builds the two containers, defined in their respective *Dockerfile*s in each 'server' and 'ui' directory.

The root directory also consists of a *README* markdown file, an abridged version of this document which focuses on installation and basic operations. A Licence declaration - *LICENCE* based on Apache framework. An installer script – *'install.sh'*, which automates the build process and installation of dependencies and configurations. A '*prereqs.sh'* script, called by the installer to verify that the target system contains or downloads the necessary dependencies to complete the build process.

Each sub-directory has operational files used by the containers to implement the test utility. In the 'server' directory there is a file invoked by the 'pqserver' container, called '*entrypoint.sh'*, which builds and executes the cryptographic key exchange profiles and signatures establishing a server instance with a unique socket connection for each. The 'ui' directory contains a python script – *'app.py'* which runs the web application supporting the helper functions presented in the web front-end, test execution, results collection and download.

There is also a 'Templates' sub-directory which contains the *'index.html'* file. This file defines the structure and styles of the web page and JavaScript helper functions to present the data to the user.

Finally, the '*helper_scripts*' directory contains a python script *'analyse_pcaps_final.py'*. This script can be used for the post-analysis of pcapng files produced during the test activities to produce relevant data output in CSV format from the raw capture files to compare to results produced from the tool. The script can be downloaded and run manually within a folder

containing pcapng files and produces a CSV file of results called *"pcap_analysis_results.csv"* with the following column entries:

pcap_name, num_conns, cps, ct_min_ms, ct_avg_ms, ct_max_ms, hs_min_ms, hs_avg_ms, hs_max_ms, ttfb_min_ms, ttfb_avg_ms, ttfb_max_ms, hs_bytes_avg, hs_pkts_avg, avg_pkts_per_conn

2.1.3 Cassini - Docker Container Functions

The following is a description of each individual container function:

pqserver: This container's only function is to serve as the high-performance OpenSSL server. The *'entrypoint.sh'* script detects available Classical, Hybrid and PQC algorithms (like ML-DSA) and KEM groups (like ML-KEM) and launches the server instances to be used in testing each cipher/signature pairing.

pq-ui: This container runs the Python Flask application to execute the tests and provide results. The container also serves the Web-based UI and dashboard and provides two key API endpoints:

- ***/run_test***: Executes *'openssl s_time'* to measure connection throughput (Connections Per Second).
- ***/curl_timing***: Executes a curl command to the pqserver entity to measure specific application latency TCP Connect, TLS Handshake, and Time to First Byte(TTFB)

3 Installation and Operation

The following sections outline the pre-requisites and instructions necessary to access and install the Cassini test utility to a Linux based host or virtual machine.

3.1 Pre-requisites

Linux Host (or virtual machine) running Ubuntu Server 24.04.3 LTS

- openssh should be installed – for remote access and management
- git - version control system utility – accessed by running 'sudo apt-get install git'

3.2 Cassini – Accessing Software Repository

To access and download Cassini to a new target system perform the following command in a terminal session from the target host

'git clone https://github.com/da0ncirl/Cassini.git'

Username for 'https://github.com': <Your Username>

Password for 'https://YourUsername@github.com': <TYPE/PASTE YOUR TOKEN HERE>

3.3 Cassini – Installation Procedure

The Cassini Utility contains an installation script, which is used to

- Call the pre-requisite packages check script
- Install all packages and dependencies necessary for operating the docker-based system
- Deploy the docker containers necessary to operate the Cassini test utility

To execute the installation procedure, follow these steps:

1. Navigate into 'Cassini' directory (created during the ‘git clone’ step above)
2. run the 'install.sh' script bash to install docker and dependencies to the target host machine. The installer wrapper script calls the prereqs.sh script to install the build dependencies necessary for step 2 - building the *pqserver* and *pq-ui* containers to the target host.

   ~$***sudo /bin/bash ./install.sh***
3. Follow onscreen output and instructions to complete the install.

NB for an initial install, the user will be prompted to log out and log back in to the host to ensure that all required permissions are sourced for the docker commands (to run without ‘sudo’ permissions.)

4. When complete the user will observe two containers 'pqserver' and 'pq-ui' on the target host, by running:

   ~$docker ps -a

3.4 Accessing the Cassini Test Utility

Once the containers are running, open a web browser and navigate to:

http://<host IP>:8080

OPTIONAL: If running via Virtual Machine based target host, local port forwarding from your client to the target VM via the physical host can be setup.

For Example - using a Virtual Machine in a NAT configuration with the Host:

- Host Mgmt IP - 10.1.1.10/24 (Virtual Machine IP 192.168.122.1)
- Virtual Machine - 192.168.122.101

Cassini running on a container in the virtual machine is accessible via

http://192.168.122.101:8080

Use the following Local port forwarding to access the container UI directly from your client PC on local port 9999:

1. Execute the following command from your local client PC to the target host:
   ~$***ssh -L 9999:192.168.122.101:8080 user@10.1.1.10***
2. Once the ssh session is established to target host 10.1.1.10 it is configured to tunnel all traffic destined for 192.168.122.101:8080 via the active ssh session using local port 9999
3. it is then possible to open a web browser on the local client PC to the following url - ***http://localhost:9999***

this will access the UI running on the container inside the virtual machine equivalent to running ***http://192.168.122.101:8080*** on the target host.

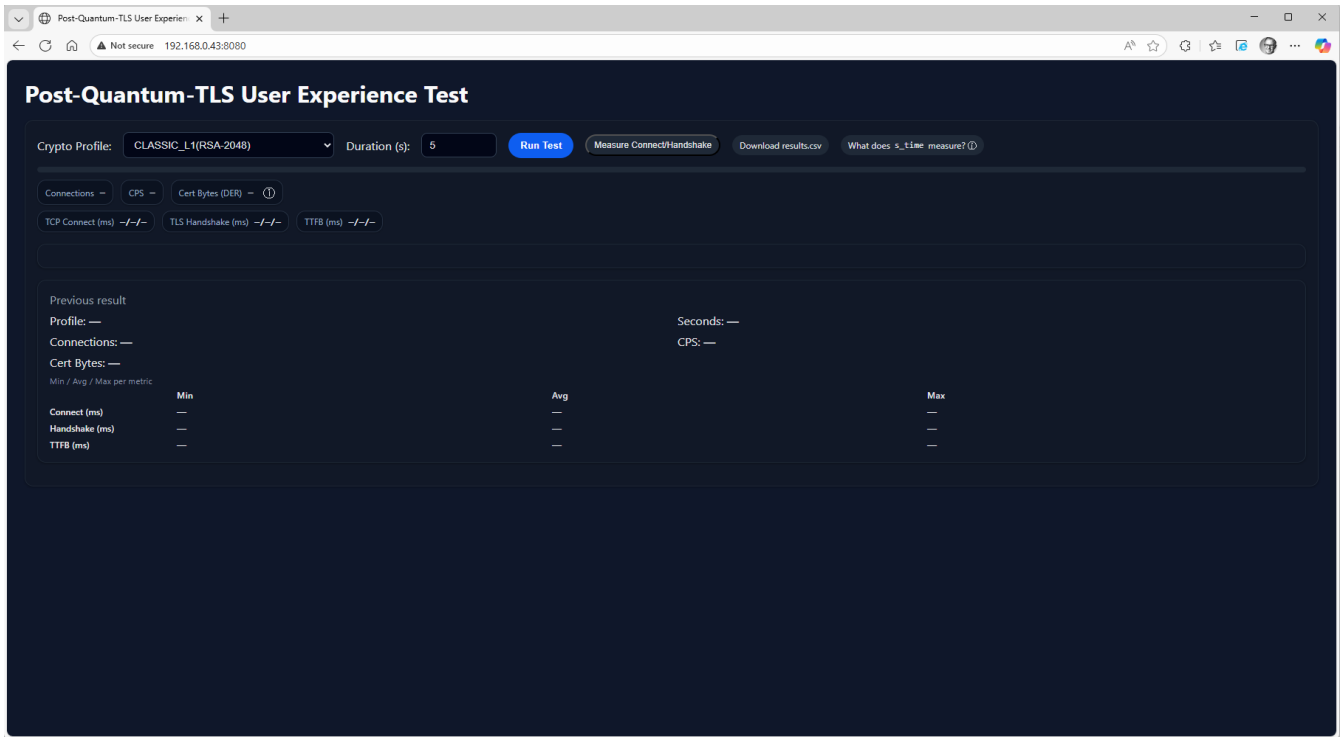

Figure 2 – Cassini PQ-TLS Test Utility Home Page

3.5 How To Run a Cryptographic/Signature Performance Test :

i. Select a "Crypto Profile" from the dropdown.

ii. Select a "Duration (s)" for the test.

iii. Click 'Run Test'

This will execute ‘openssl s_time’ for the chosen profile and record all results to the triplet fields displayed for Connection details and also handshake time details.

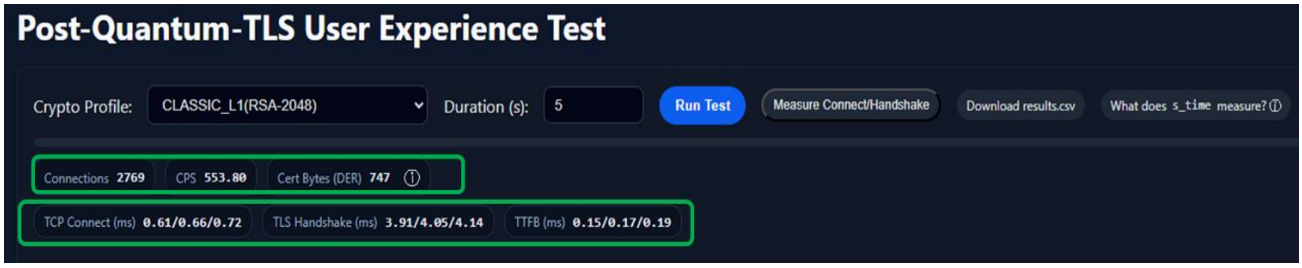

Figure 3 – Cassini – Test Utility – Test Results Triplet Fields

After the initial test and as subsequent tests are executed, a dashboard is initialised in the web UI that provides the user with visibility of how the key performance indicators for each Cryptographic profile compare with results values for each test run.

3.5.1 Cassini Results Dashboard

Within the dashboard, a TLS handshake Timeline Comparison view is presented. This results view uses ‘Red/Amber/Green’ heatmap-based colour coding and length-based visibility to convey TLS handshake timing for each Cryptographic profile. From this view, we can immediately see which profile has the shortest TLS handshake and which has the longest.

Figure 4 – Cassini Test Utility Results Dashboard– TLS Handshake Timeline Comparison View

The Dashboard also populates a view based on the number of connections and connections per second for each profile. The test operator can see how efficient each profile is at creating connections within the specified duration of the tests executed.

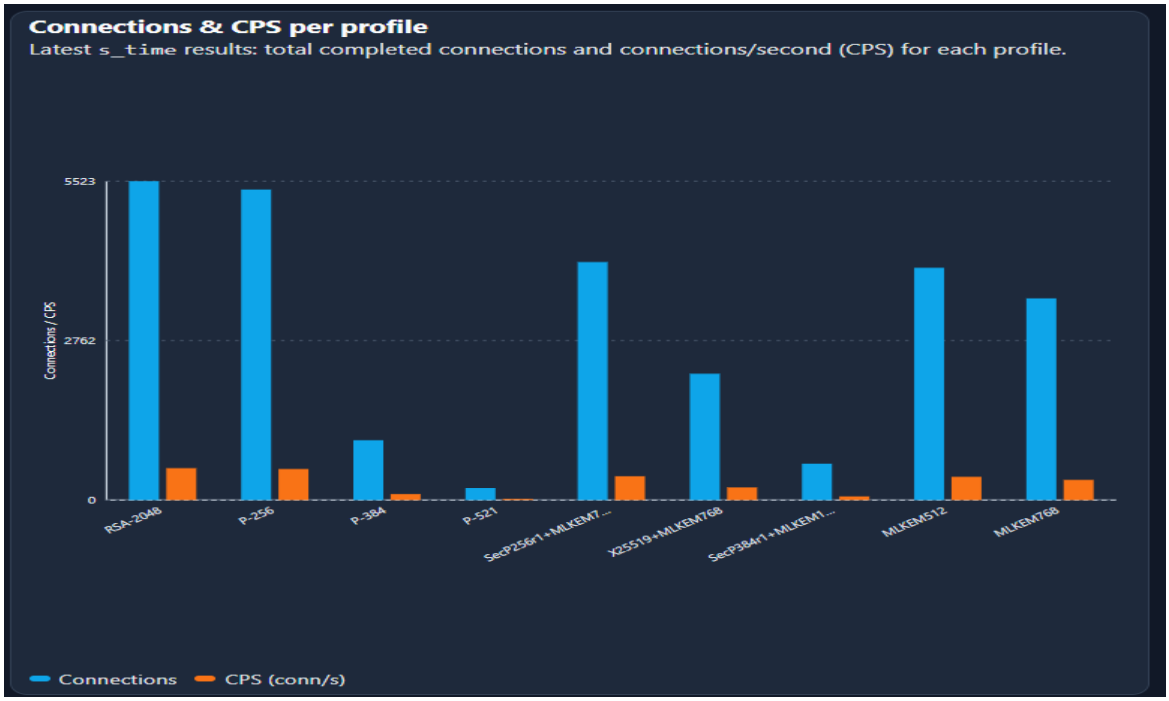

Figure 5 – Cassini Test Utility – Results Dashboard – Connections/CPS Comparison View

A scatter plot-based graph is also presented, comparing TLS handshake time – the centre is based on the average handshake time and whisker plots convey the minimum and maximum values for comparison purposes. The x-axis is based on the Cert Bytes – the amount of data transferred between client and server to authentication and verify the sessions. This presents a unique view of TLS performance, with a context for the overhead associated with the secure establishment of each connection.

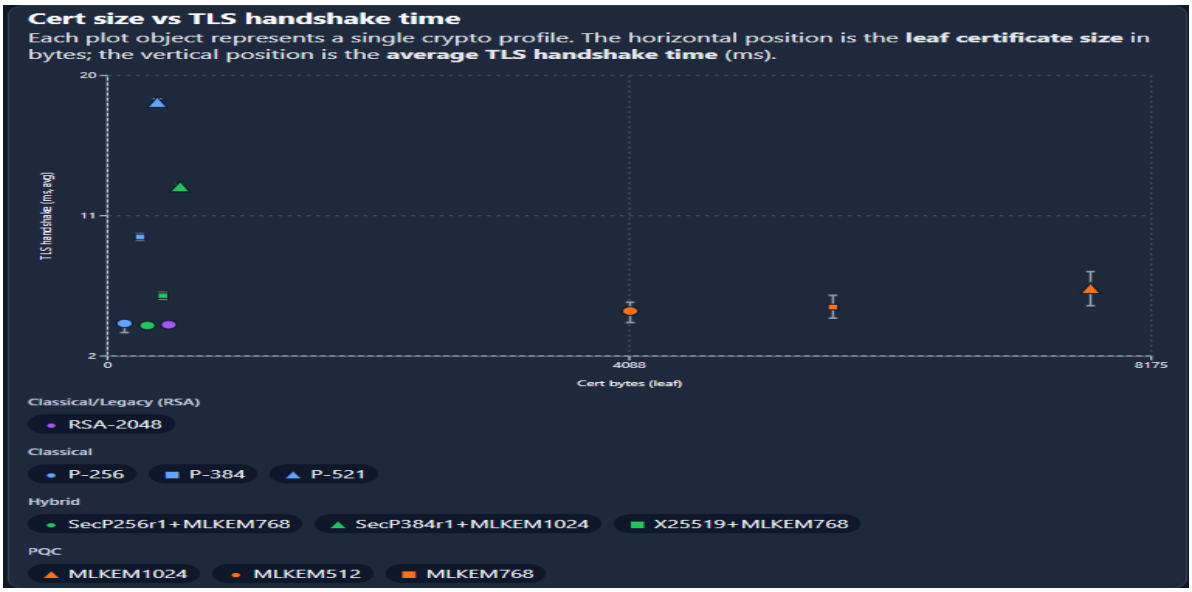

Figure 6 – Cassini Test Utility – Results Dashboard – TLS Handshake vs. Cert Bytes

Each profile is represented by

- A unique icon representing the Security Level
  - Circle – Security Level 1
  - Square – Security Level 3
  - Triangle – Security Level 5
- Colour-coded Grouping:
  - Legacy – Purple
  - Classical – Blue
  - Hybrid – Green
  - Post-Quantum – Orange

3.5.2 Measure Connect/Handshake (Latency):

To run this test click the ‘Measure Connect/Handshake’ button.

This executes the ‘curl_timing’ function towards the pqserver container to gather min, avg, and max statistics for TCP Connect, TLS Handshake, and TTFB.

This populates the "Handshake timeline" and the "Cert size vs TLS handshake time" scatter plot, if different from the current data plotted.

Note: The Connect/Handshake test action is triggered automatically during a "Run Test" to provide a comprehensive view of performance. The independent test provides granularity and

control for any potential artefacts that may be observed in test results that require further validation to test variance.

3.5.3 Previous result view

The final dashboard view consists of the triplet-based views as presented from the top of results dashboard in figure 3 above but populated for the previous test run. This allows the test operator to compare the numerical values for the previous test to the current test run – this could be based on a difference security profile or comparing test runs of the same profile for variance.

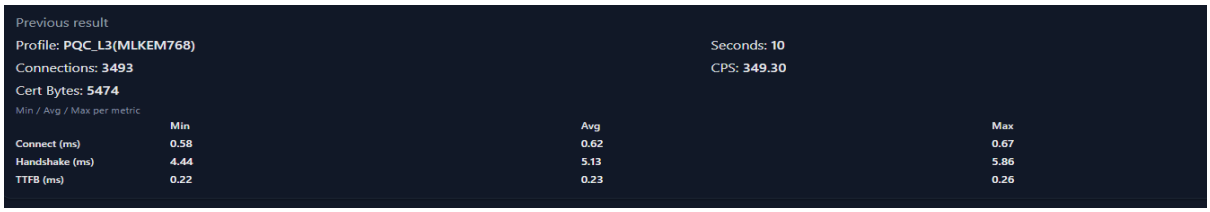

Figure 7 – Cassini Test Utility – Results Dashboard – Previous Results view

3.5.4 Test Data Results Retrieval

All results data produced by the test executions are written to a results file, in CSV format. This file can be downloaded on demand by the test operator for post-analysis. The UI has a dedicated button which can be used as required by the test operator. Each test run is written in sequence to results.csv file.

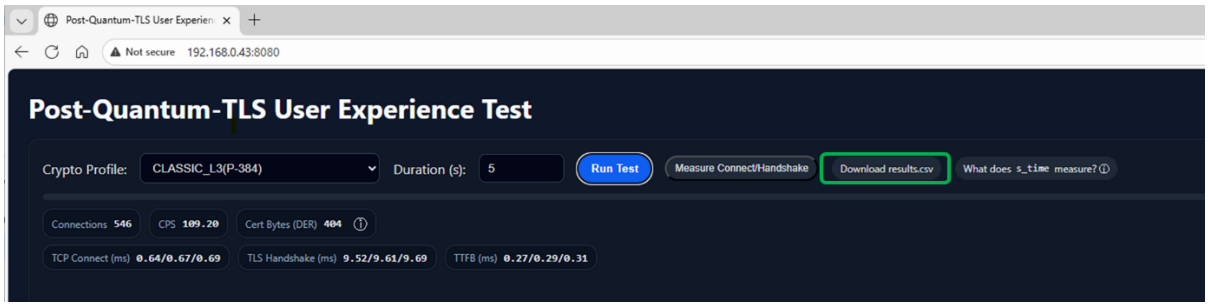

Figure 8 – Cassini Test Utility – Results Retrieval Button

The resulting file can be downloaded to a client host PC for investigation and post-analysis. The structure of the CSV file columns is outlined below:

"timestamp","profile","seconds","conns","cps","bytes_read","cert_bytes","ct_min_ms","ct_avg_ms","ct_max_ms","hs_min_ms","hs_avg_ms","hs_max_ms","ttfb_min_ms","ttfb_avg_ms","ttfb_max_ms"

timestamp	profile	seconds	conns	cps	bytes_read	cert_bytes	ct_min_ms	ct_avg_ms	ct_max_ms	hs_min_ms	hs_avg_ms	hs_max_ms	ttfb_min_ms	ttfb_avg_ms	ttfb_max_ms
2025-12-07T18:13:22Z	CLASSIC_L1(RSA-2048)	5	2808	561.6	6003504	747	0.58	0.65	0.78	3.9	4.09	4.29	0.16	0.17	0.19
2025-12-07T18:13:40Z	CLASSIC_L1(P-256)	5	2977	595.4	13464971	342	0.58	0.62	0.67	3.46	3.49	3.52	0.28	0.29	0.3
2025-12-07T18:13:55Z	CLASSIC_L3(P-384)	5	648	129.6	2930256	404	0.58	0.64	0.78	9.49	9.64	9.93	0.26	0.29	0.36
2025-12-07T18:14:06Z	CLASSIC_L5(P-521)	5	253	50.6	1144319	479	0.6	0.64	0.67	17.48	17.66	17.89	0.26	0.29	0.33
2025-12-07T18:14:22Z	HYBRID_L1(SecP256r1+MLKEM768)	5	2300	460	10455800	342	0.64	0.7	0.8	3.8	3.85	3.94	0.29	0.31	0.33
2025-12-07T18:14:37Z	HYBRID_L3(X25519+MLKEM768)	5	1097	219.4	4977089	404	0.59	0.65	0.73	5.71	5.78	5.88	0.21	0.27	0.29
2025-12-07T18:14:54Z	HYBRID_L5(SecP384r1+MLKEM1024)	5	385	77	1751750	479	0.59	0.63	0.67	12.34	12.47	12.67	0.26	0.26	0.27
2025-12-07T18:15:06Z	PQC_L1(MLKEM512)	5	2184	436.8	9891336	3945	0.58	0.62	0.66	3.96	4.23	4.51	0.17	0.21	0.24
2025-12-07T18:15:21Z	PQC_L3(MLKEM768)	5	1813	362.6	8211077	5474	0.6	0.65	0.68	4.48	4.7	5.06	0.23	0.24	0.27
2025-12-07T18:15:32Z	PQC_L5(MLKEM1024)	5	1470	294	6662040	7432	0.6	0.65	0.7	5.43	6.53	10.29	0.24	0.26	0.27

Figure 9 – Output of results.csv file for test run (shown in Microsoft Excel)

3.5.5 Cassini – Test Utility - Other Features

The UI also contain a number of helper tooltips, that elaborate on how the testing is performed, the functions and calculations used to derive results, with an explanation of the key performance indicators.

Figure 10 – Cassini Test Utility - UI Tooltips Describing Test Functions

These tooltips bridge execution data and KPIs to surface the performance and user experience context from the testing activity.

4 References

OpenSSL. (2025, 12 01). Retrieved from OpenSSL Library Home Page: https://www.openssl.org/

Solutions, V. (2025, November). *VIAVI TeraVM - Application Emulation and Security Validation*. Retrieved from VIAVI Solutions Prodcuts and Services: https://www.viavisolutions.com/en-uk/products/teravm

Technology, N. I. (2025, Sept 30). Post-Quantum Cryptography Standardization - Security (Evaluation Criteria). Gaithersburg, MD, USA.

5 Appendices

5.1 Commercial Test Tool Validation

In order to validate the test utility and its results a commercial tool, TeraVM an application emulation and security traffic performance test suite was used to:

- Compare baseline primitive testing in Cassini for similar performance and behaviour
- Execute SaaS traffic profile to observe if the baseline findings are consistent with real-world traffic functions.

The architecture of the commercial test tool is described below. TeraVM, developed by VIAVI Solutions (Solutions, 2025) is virtual machine-based and consists of a controller instance, comprising of a Web-based UI, incorporating an embedded Java client user interface. Provisioning and execution of the tests and presentation of results are managed from this instance. The provisioned traffic profiles are copied down to the dedicated application traffic generation modules and generated across the nominated interfaces into the network. Each application traffic module also records metrics for each PDU sent and received, reporting to the controller. This communication is managed via a dedicated internal Linux bridge operating as a virtual switch. This ensures that the control aspects and the test data are isolated from each other minimising contamination and ensuring the integrity of results.

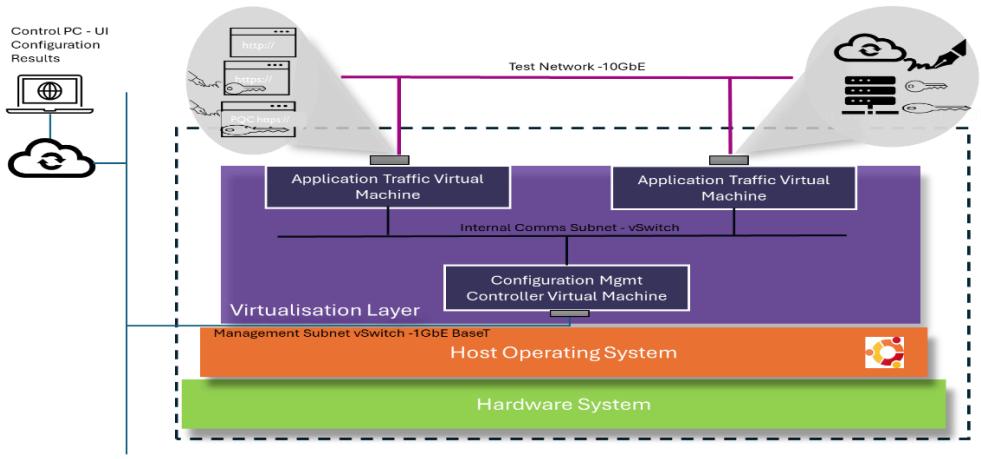

Figure 11 – TeraVM Test Tool Virtual Machine Architecture

One key feature of TeraVM is the ability to instigate a PDU capture before and during an active test run. This facilitates an adjacent validation of the test activities, by saving pcapng traces for each test profile for post analysis.

The following is a view of the TeraVM web-based user interface displaying the provisioning UI with views of the TLS configuration dialogue for one of the baseline tests (RSA-2048)

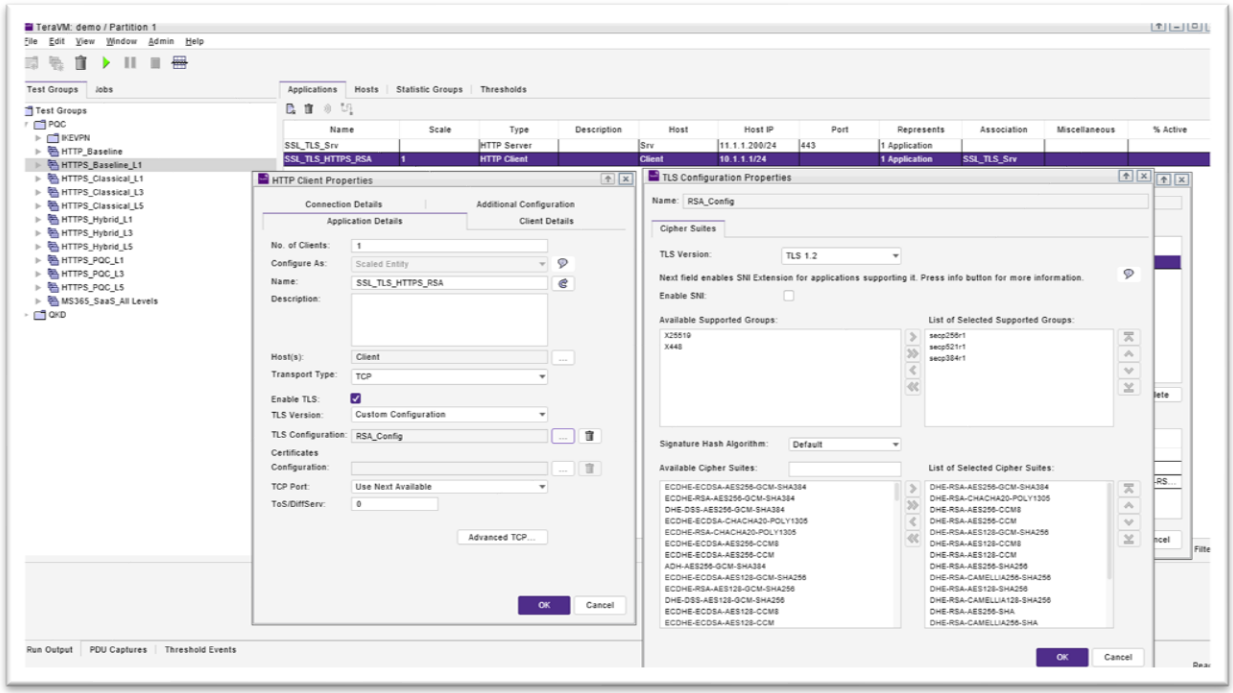

Figure 12 – TeraVM User Interface – TLS provisioning dialogue

The Test cases are executed in strict sequence as demonstrated in the Cassini test utility, 10 Bytes for 10 seconds for comparison purposes.

5.2 Wireshark Packet Analysis Verification

Wireshark is used in the manual analysis of raw capture traces to verify that the right profile and signature are being used for the test runs in Cassini test utility:

i. TLS Client Hello:
   ⇨ Expand Transport Layer Security -> TLSv1.3 Record Layer -> Handshake Protocol: Client Hello.
   Look for the Extension: key_share.
There should be a Key Share entry specifically for the algorithm ID associated.

ii. TLS Server Hello:
   Look for the Extension: key_share.
   The server must select and respond with the same algorithm group ID.

Legacy_L1: RSA2048 (Port 9843)

i. Check that the protocol version is TLS 1.2 (not 1.3).
ii. Check the Client Hello Cipher Suite: It must be *TLS_RSA_WITH_AES_256_GCM_SHA384* in Server Hello
iii. Check Server Hello Certificate algorithmIdentifier Algorithm ID - *sha256WithRSAEncryption*
iv. Verify there is no Server Key Exchange packet.

Classical_L1: P-256 (Port 8843)

i. Check that the protocol version is TLS 1.3
ii. Check the Client Hello Cipher Suite: TLS1.3 compatible (*TLS_AES_256_GCM_SHA384; TLS_CHACHA20_POLY1305_SHA256;TLS_AES_128_GCM_SHA256*) and Extension key_share: *secp256r1*
iii. Check Server Hello Certificate: confirm Extension key_share: *secp256r1*; Cipher Suite *TLS_AES_256_GCM_SHA384*
iv. Verify there is no Server Key Exchange packet.

Classical_L3: P-384 (Port 8943)

i. Check that the protocol version is TLS 1.3
ii. Check the Client Hello Cipher Suite: TLS1.3 compatible (*TLS_AES_256_GCM_SHA384; TLS_CHACHA20_POLY1305_SHA256;TLS_AES_128_GCM_SHA256*) and Extension key_share: *secp384r1*
iii. Check Server Hello Certificate: confirm Extension key_share: *secp384r1*; Cipher Suite *TLS_AES_256_GCM_SHA384*

Classical_L5: P-521 (Port 11443)

i. Check that the protocol version is TLS 1.3
ii. Check the Client Hello Cipher Suite: TLS1.3 compatible (*TLS_AES_256_GCM_SHA384; TLS_CHACHA20_POLY1305_SHA256;TLS_AES_128_GCM_SHA256*) and Extension key_share: *secp521r1*
iii. Check Server Hello Certificate: confirm Extension key_share: *secp521r1*; Cipher Suite *TLS_AES_256_GCM_SHA384*

Hybrid_L1: SecP256r1+MLKEM768 (Port 8643)

i. Check that the protocol version is TLS 1.3
ii. Check the Client Hello Cipher Suite: TLS1.3 compatible (*TLS_AES_256_GCM_SHA384; TLS_CHACHA20_POLY1305_SHA256;TLS_AES_128_GCM_SHA256*) and Extension key_share: *secp256r1MLKEM768*
iii. Check Server Hello Certificate: confirm Extension key_share: *secp256r1MLKEM768*; Cipher Suite *TLS_AES_256_GCM_SHA384*

Hybrid_L3: X25519+MLKEM768 (Port 8743)

i. Check that the protocol version is TLS 1.3
ii. Check the Client Hello Cipher Suite: TLS1.3 compatible (*TLS_AES_256_GCM_SHA384; TLS_CHACHA20_POLY1305_SHA256;TLS_AES_128_GCM_SHA256*) and Extension key_share: *X25519MLKEM768*
iii. Check Server Hello Certificate: confirm Extension key_share: *X25519MLKEM768*; Cipher Suite *TLS_AES_256_GCM_SHA384*

Hybrid_L5: SecP384r1+MLKEM1024 (Port 10443)

i. Check that the protocol version is TLS 1.3

ii. Check the Client Hello Cipher Suite: TLS1.3 compatible (*TLS_AES_256_GCM_SHA384; TLS_CHACHA20_POLY1305_SHA256;TLS_AES_128_GCM_SHA256*) and Extension key_share: *SecP384r1MLKEM1024*
iii. Check Server Hello Certificate: confirm Extension key_share: *SecP384r1MLKEM1024*; Cipher Suite *TLS_AES_256_GCM_SHA384*

PQC_L1 MLKEM512 (Port 8443):

i. Check that the protocol version is TLS 1.3
ii. Check the Client Hello Cipher Suite: TLS1.3 compatible (*TLS_AES_256_GCM_SHA384; TLS_CHACHA20_POLY1305_SHA256;TLS_AES_128_GCM_SHA256*) and Extension key_share: *MLKEM512*
iii. Check Server Hello Certificate: confirm Extension key_share: *MLKEM512*; Cipher Suite *TLS_AES_256_GCM_SHA384*

PQC_L3 MLKEM768 (Port 8543):

i. Check that the protocol version is TLS 1.3
ii. Check the Client Hello Cipher Suite: TLS1.3 compatible (*TLS_AES_256_GCM_SHA384; TLS_CHACHA20_POLY1305_SHA256;TLS_AES_128_GCM_SHA256*) and Extension key_share: *MLKEM768*
iii. Check Server Hello Certificate: confirm Extension key_share: *MLKEM768*; Cipher Suite *TLS_AES_256_GCM_SHA384*

PQC_L5 MLKEM1024 (Port 9443):

i. Check that the protocol version is TLS 1.3
ii. Check the Client Hello Cipher Suite: TLS1.3 compatible (*TLS_AES_256_GCM_SHA384; TLS_CHACHA20_POLY1305_SHA256;TLS_AES_128_GCM_SHA256*) and Extension key_share: *MLKEM1024* or supported groups *MLKEM1024*
iii. Check Server Hello Certificate: confirm Extension key_share: *MLKEM1024*; Cipher Suite *TLS_AES_256_GCM_SHA384*

5.3 tshark Analysis

Use 'tshark' to export specific fields (Frame Number, Time, Packet Length, Protocol, Info) into a CSV format, which can then be post-analysed in Excel or Python.

Command to extract Packet Sizes: Run against a .pcap file:

 # Syntax: tshark -r <input_file> -T fields -e <field1> -e <field2> ... -E separator=, > output.csv

Example (for RSA) below:

```
~$tshark -r rsa_test.pcap -T fields \
   -e frame.number \
   -e frame.time_relative \
   -e frame.len \
```

*-e ip.src *

*-e tcp.srcport *

*-e tcp.dstport *

*-e tls.handshake.type *

*-e tls.record.content_type *

-E header=y -E separator=, -E quote=d > packet_analysis_RSA.csv

frame.len: This provides an exact wire size of the packet.

tls.handshake.type: identification of Client Hello (1), Server Hello (2),

tls.record.content_type: identification of Handshake (22), Application Data (23), etc.

5.3.1 Useful tshark and pcap commands and example outputs

Count the server packets (assuming server is port 8443) sent during the handshake
~$tshark -r pqc_test.pcap -Y "tcp.srcport == 8443 and tls" | wc -l

Provide summary details for the pcapng file
~$capinfos RSA_finalFilter10sec.pcap

File name: RSA_finalFilter10sec.pcap
File type: Wireshark/tcpdump/... - pcap
File encapsulation: Ethernet
File timestamp precision: microseconds (6)
Packet size limit: file hdr: 262144 bytes
Number of packets: 45 k
File size: 16 MB
Data size: 15 MB
Capture duration: 10.505136 seconds
First packet time: 2025-11-24 21:42:12.972070
Last packet time: 2025-11-24 21:42:23.477206
Data byte rate: 1476 kBps
Data bit rate: 11 Mbps
Average packet size: 340.87 bytes
Average packet rate: 4332 packets/s
SHA256:
a133ff05e64c70412ee6dfa0e5311ee007372eb2571bdddc0ed93fe0e3d7e8aa
RIPEMD160: 1ed78b9fae1eaa76a0f56624991a9056951cdf3e
SHA1: 5f59caaf6f0ebeccfa890ffa7a2684a935f53b5f
Strict time order: True
Number of interfaces in file: 1
Interface #0 info:
Encapsulation = Ethernet (1 - ether)
Capture length = 262144
Time precision = microseconds (6)
Time ticks per second = 1000000

Number of stat entries = 0
Number of packets = 45517

Count the TCP Connections in a pcapng file
~$tshark -r RSA_finalFilter10sec.pcap -Y "tcp.flags.syn==1 && tcp.flags.ack==0" | wc -l
~$3250

Output I/O statistics of the discrete TCP Connections in a pcapng file
~$tshark -r RSA_finalFilter10sec.pcap -q -z io,stat,1,"tcp.flags.syn==1 && tcp.flags.ack==0"

```
===================================================
| IO Statistics                                   |
|                                                 |
| Duration: 10.505136 secs                        |
| Interval:  1 secs                               |
|                                                 |
| Col 1: tcp.flags.syn==1 && tcp.flags.ack==0     |
|-------------------------------------------------|
|          |1               |                     |
| Interval | Frames | Bytes |                     |
|--------------------------|                      |
|  0 <>  1 |    217 | 16058 |                     |
|  1 <>  2 |    352 | 26048 |                     |
|  2 <>  3 |    318 | 23532 |                     |
|  3 <>  4 |    403 | 29822 |                     |
|  4 <>  5 |    379 | 28046 |                     |
|  5 <>  6 |    462 | 34188 |                     |
|  6 <>  7 |    331 | 24494 |                     |
|  7 <>  8 |    242 | 17908 |                     |
|  8 <>  9 |    287 | 21238 |                     |
|  9 <> 10 |    247 | 18278 |                     |
| 10 <> Dur|     12 |   888 |                     |
===================================================
```

5.4 Useful Docker Administration Commands

#To bring system down and remove cache:
~$*docker compose down*
~$*docker compose down --remove-orphans*
~$*docker compose down --rmi all*
~$*docker compose down -v*

#To clear disk space used by containers on a host (useful if using virtual machines):
~$*docker volume prune*
~$*docker builder prune*
~$*docker builder prune -a*

~$*docker system prune -a*

#To build docker containers from scratch without using locally cached components
~$*docker compose build --no-cache*
~$*docker compose up --force-recreate*

#To manually build system and check docker logs for each container:
~$*docker compose up -d --build*
~$*docker ps -a*
~$*docker compose logs pqserver*
~$*docker compose logs pq-ui*

#To use Docker to login to containers:
~$*docker compose exec pqserver /bin/bash*
~$*docker compose exec pq-ui /bin/bash*

#To use Docker to login to containers and run remote openssl admin commands in a single output:
~$*docker exec -it pqserver openssl list -tls-groups*
~$*docker exec -it pqserver openssl list -kem-algorithms*
~$*docker exec -it pqserver openssl list -key-exchange-algorithms*
~$*docker exec -it pqserver openssl list -signature-algorithms*
~$*docker exec -it pqserver openssl list -tls-signature-algorithms*
~$*docker exec -it pqserver openssl list -tls1_3 -signature-algorithms*
~$*docker exec -it pqserver openssl list -tls1_2 -signature-algorithms*

#Use Docker to login to containers and run remote openssl execution commands in a single output (back-end troubleshooting if any test fails in Web UI):
~$*docker exec -i pq-ui sh -lc "openssl s_time -connect pqserver:8643 -www / -tls1_3 -new -time 5"*
~$*docker exec -i pq-ui sh -lc "openssl s_time -connect pqserver:8743 -www / -tls1_3 -new -time 5"*
~$*docker exec -i pq-ui sh -lc "openssl s_time -connect pqserver:11443 -www / -tls1_3 -new -time 5"*
~$*docker exec -i pq-ui sh -lc "openssl s_time -connect pqserver:10443 -www / -tls1_3 -new -time 5"*
~$*docker exec -i pq-ui sh -lc "openssl s_time -connect pqserver:8643 -www / -tls1_3 -new -time 5"*
~$*docker exec -i pq-ui sh -lc "openssl s_time -connect pqserver:8443 -www / -tls1_3 -new -time 5"*
~$*docker exec -i pq-ui sh -lc "openssl s_time -connect pqserver:9843 -www / -tls1_2 -new -time 5"*

5.5 Software Bill of Materials - Containers

Exrtracted using ‘syft’ utility which generates software bill of materials (SBOM) for a host system. The embedded Microsoft Excel Sheets contain a detailed list of software packages, version and type used.

pqserver

✓ Loaded image
pqtls_test-ui:latest
✓ Parsed image
sha256:500f6adbed4cb8b5bfea5145407f1db67c4045e0021caafd1d7669e9e82e927c
✓ Cataloged contents
97160207fa7708498e0492a62216003cf06c648e84b82ddde8d2b4040ec69c72
├── ✓ Packages [137 packages]
├── ✓ Executables [794 executables]
├── ✓ File metadata [2,763 locations]
└── ✓ File digests [2,763 files]

ServerContainerPackages.xlsx

pq-ui

✓ Loaded image
pqtls_test-pqserver:latest
✓ Parsed image
sha256:496014c4c8a16bf6f04cabb434ac80e930d7d9769c2fee9e5315dbc0d9d3c4a7
✓ Cataloged contents
7a69beff671d85d951fd54de44150a657ec8460f1e5674ee61667d026b770ba5
├── ✓ Packages [244 packages]
├── ✓ File metadata [8,110 locations]
├── ✓ Executables [1,338 executables]
└── ✓ File digests [8,110 files]

UIContainerPackages.xlsx

5.6 Analysis Results Sheets

The attached sheets provide a view of the test data derived from both test systems and the analysis of the data produced to derive insights

results_10s_All.xlsx

SaaS_analysis_results.xlsx

5.7 OpenSSL Commands and Outputs

Provides prescriptive guidance on the scope and boundaries of the ciphers, signatures and key exchange algorithms used for this project. (OpenSSL, 2025)

~$openssl -version
OpenSSL 3.5.1 1 Jul 2025 (Library: OpenSSL 3.5.1 1 Jul 2025)

~$openssl list -all-tls-groups
secp256r1:P-256:secp384r1:P-384:secp521r1:P-521:x25519:x448:brainpoolP256r1tls13:brainpoolP384r1tls13:brainpoolP512r1tls13:ffdhe2048:ffdhe3072:ffdhe4096:ffdhe6144:ffdhe8192:MLKEM512:MLKEM768:MLKEM1024:SecP256r1MLKEM768:X25519MLKEM768:SecP384r1MLKEM1024

~$openssl list -tls-groups
secp256r1:secp384r1:secp521r1:x25519:x448:brainpoolP256r1tls13:brainpoolP384r1tls13:brainpoolP512r1tls13:ffdhe2048:ffdhe3072:ffdhe4096:ffdhe6144:ffdhe8192:MLKEM512:MLKEM768:MLKEM1024:SecP256r1MLKEM768:X25519MLKEM768:SecP384r1MLKEM1024

~$openssl list -kem-algorithms
{ 1.2.840.113549.1.1.1, 2.5.8.1.1, RSA, rsaEncryption } @ default
{ 1.2.840.10045.2.1, EC, id-ecPublicKey } @ default
{ 1.3.101.110, X25519 } @ default
{ 1.3.101.111, X448 } @ default
{ 2.16.840.1.101.3.4.4.1, id-alg-ml-kem-512, ML-KEM-512, MLKEM512 } @ default
{ 2.16.840.1.101.3.4.4.2, id-alg-ml-kem-768, ML-KEM-768, MLKEM768 } @ default
{ 2.16.840.1.101.3.4.4.3, id-alg-ml-kem-1024, ML-KEM-1024, MLKEM1024 } @ default
X25519MLKEM768 @ default
X448MLKEM1024 @ default
SecP256r1MLKEM768 @ default
SecP384r1MLKEM1024 @ default

~$openssl list -tls-signature-algorithms
ecdsa_secp256r1_sha256:ecdsa_secp384r1_sha384:ecdsa_secp521r1_sha512:ed25519:ed448:ecdsa_sha224:ecdsa_sha1:ecdsa_brainpoolP256r1tls13_sha256:ecdsa_brainpoolP384r1tls13_sha384:ecdsa_brainpoolP512r1tls13_sha512:rsa_pss_rsae_sha256:rsa_pss_rsae_sha384:rsa_pss_rsae_sha512:rsa_pss_pss_sha256:rsa_pss_pss_sha384:rsa_pss_pss_sha512:rsa_pkcs1_sha256:rsa_pkcs1_sha384:rsa_pkcs1_sha512:rsa_pkcs1_sha224:rsa_pkcs1_sha1:dsa_sha256:dsa_sha384:dsa_sha512:dsa_sha224:dsa_sha1:mldsa44:mldsa65:mldsa87

~$openssl list -cipher-algorithms
Legacy:
AES-128-CBC
AES-128-CBC-HMAC-SHA1

AES-128-CBC-HMAC-SHA256
id-aes128-CCM
AES-128-CFB
AES-128-CFB1
AES-128-CFB8
AES-128-CTR
AES-128-ECB
id-aes128-GCM
AES-128-OCB
AES-128-OFB
AES-128-XTS
AES-192-CBC
id-aes192-CCM
AES-192-CFB
AES-192-CFB1
AES-192-CFB8
AES-192-CTR
AES-192-ECB
id-aes192-GCM
AES-192-OCB
AES-192-OFB
AES-256-CBC
AES-256-CBC-HMAC-SHA1
AES-256-CBC-HMAC-SHA256
id-aes256-CCM
AES-256-CFB
AES-256-CFB1
AES-256-CFB8
AES-256-CTR
AES-256-ECB
id-aes256-GCM
AES-256-OCB
AES-256-OFB
AES-256-XTS
aes128 => AES-128-CBC
aes128-wrap => id-aes128-wrap
aes128-wrap-pad => id-aes128-wrap-pad
aes192 => AES-192-CBC
aes192-wrap => id-aes192-wrap
aes192-wrap-pad => id-aes192-wrap-pad
aes256 => AES-256-CBC
aes256-wrap => id-aes256-wrap
aes256-wrap-pad => id-aes256-wrap-pad
ARIA-128-CBC
ARIA-128-CCM
ARIA-128-CFB
ARIA-128-CFB1
ARIA-128-CFB8
ARIA-128-CTR
ARIA-128-ECB

ARIA-128-GCM
ARIA-128-OFB
ARIA-192-CBC
ARIA-192-CCM
ARIA-192-CFB
ARIA-192-CFB1
ARIA-192-CFB8
ARIA-192-CTR
ARIA-192-ECB
ARIA-192-GCM
ARIA-192-OFB
ARIA-256-CBC
ARIA-256-CCM
ARIA-256-CFB
ARIA-256-CFB1
ARIA-256-CFB8
ARIA-256-CTR
ARIA-256-ECB
ARIA-256-GCM
ARIA-256-OFB
aria128 => ARIA-128-CBC
aria192 => ARIA-192-CBC
aria256 => ARIA-256-CBC
bf => BF-CBC
BF-CBC
BF-CFB
BF-ECB
BF-OFB
blowfish => BF-CBC
CAMELLIA-128-CBC
CAMELLIA-128-CFB
CAMELLIA-128-CFB1
CAMELLIA-128-CFB8
CAMELLIA-128-CTR
CAMELLIA-128-ECB
CAMELLIA-128-OFB
CAMELLIA-192-CBC
CAMELLIA-192-CFB
CAMELLIA-192-CFB1
CAMELLIA-192-CFB8
CAMELLIA-192-CTR
CAMELLIA-192-ECB
CAMELLIA-192-OFB
CAMELLIA-256-CBC
CAMELLIA-256-CFB
CAMELLIA-256-CFB1
CAMELLIA-256-CFB8
CAMELLIA-256-CTR
CAMELLIA-256-ECB
CAMELLIA-256-OFB

camellia128 => CAMELLIA-128-CBC
camellia192 => CAMELLIA-192-CBC
camellia256 => CAMELLIA-256-CBC
cast => CAST5-CBC
cast-cbc => CAST5-CBC
CAST5-CBC
CAST5-CFB
CAST5-ECB
CAST5-OFB
ChaCha20
ChaCha20-Poly1305
des => DES-CBC
DES-CBC
DES-CFB
DES-CFB1
DES-CFB8
DES-ECB
DES-EDE
DES-EDE-CBC
DES-EDE-CFB
des-ede-ecb => DES-EDE
DES-EDE-OFB
DES-EDE3
DES-EDE3-CBC
DES-EDE3-CFB
DES-EDE3-CFB1
DES-EDE3-CFB8
des-ede3-ecb => DES-EDE3
DES-EDE3-OFB
DES-OFB
des3 => DES-EDE3-CBC
des3-wrap => id-smime-alg-CMS3DESwrap
desx => DESX-CBC
DESX-CBC
id-aes128-CCM
id-aes128-GCM
id-aes128-wrap
id-aes128-wrap-pad
id-aes192-CCM
id-aes192-GCM
id-aes192-wrap
id-aes192-wrap-pad
id-aes256-CCM
id-aes256-GCM
id-aes256-wrap
id-aes256-wrap-pad
id-smime-alg-CMS3DESwrap
idea => IDEA-CBC
IDEA-CBC
IDEA-CFB

IDEA-ECB
IDEA-OFB
rc2 => RC2-CBC
rc2-128 => RC2-CBC
rc2-40 => RC2-40-CBC
RC2-40-CBC
rc2-64 => RC2-64-CBC
RC2-64-CBC
RC2-CBC
RC2-CFB
RC2-ECB
RC2-OFB
RC4
RC4-40
RC4-HMAC-MD5
seed => SEED-CBC
SEED-CBC
SEED-CFB
SEED-ECB
SEED-OFB
sm4 => SM4-CBC
SM4-CBC
SM4-CFB
SM4-CTR
SM4-ECB
SM4-OFB

Provided:

{ 1.2.410.200046.1.1.12, ARIA-256-CBC, ARIA256 } @ default
{ 2.16.840.1.101.3.4.1.22, AES-192-CBC, AES192 } @ default
{ 2.16.840.1.101.3.4.1.4, AES-128-CFB } @ default
{ 1.2.410.200046.1.1.38, ARIA-192-CCM } @ default
{ 1.2.410.200046.1.1.1, ARIA-128-ECB } @ default
{ 2.16.840.1.101.3.4.1.2, AES-128-CBC, AES128 } @ default
{ 2.16.840.1.101.3.4.1.24, AES-192-CFB } @ default
{ 1.2.392.200011.61.1.1.1.2, CAMELLIA-128-CBC, CAMELLIA128 } @ default
{ 1.2.392.200011.61.1.1.1.4, CAMELLIA-256-CBC, CAMELLIA256 } @ default
{ 1.2.410.200046.1.1.35, ARIA-192-GCM } @ default
{ 2.16.840.1.101.3.4.1.42, AES-256-CBC, AES256 } @ default
{ 2.16.840.1.101.3.4.1.28, AES-192-WRAP-PAD, AES192-WRAP-PAD, id-aes192-wrap-pad } @ default
{ 1.2.410.200046.1.1.36, ARIA-256-GCM } @ default
{ 1.3.111.2.1619.0.1.2, AES-256-XTS } @ default
{ 2.16.840.1.101.3.4.1.8, AES-128-WRAP-PAD, AES128-WRAP-PAD, id-aes128-wrap-pad } @ default
{ 1.2.840.113549.1.9.16.3.6, DES3-WRAP, id-smime-alg-CMS3DESwrap } @ default
{ 2.16.840.1.101.3.4.1.48, AES-256-WRAP-PAD, AES256-WRAP-PAD, id-aes256-wrap-pad } @ default
{ 1.2.156.10197.1.104.3, SM4-OFB, SM4-OFB128 } @ default
{ 2.16.840.1.101.3.4.1.25, AES-192-WRAP, AES192-WRAP, id-aes192-wrap } @ default
{ 2.16.840.1.101.3.4.1.41, AES-256-ECB } @ default

{ 0.3.4401.5.3.1.9.49, CAMELLIA-256-CTR } @ default
{ 1.2.410.200046.1.1.2, ARIA-128-CBC, ARIA128 } @ default
{ 2.16.840.1.101.3.4.1.6, aes-128-gcm, id-aes128-GCM } @ default
{ 0.3.4401.5.3.1.9.41, CAMELLIA-256-ECB } @ default
{ 2.16.840.1.101.3.4.1.44, AES-256-CFB } @ default
{ 1.2.156.10197.1.104.4, SM4-CFB, SM4-CFB128 } @ default
{ 0.3.4401.5.3.1.9.4, CAMELLIA-128-CFB } @ default
{ 1.2.410.200046.1.1.39, ARIA-256-CCM } @ default
{ 1.2.410.200046.1.1.14, ARIA-256-OFB } @ default
{ 2.16.840.1.101.3.4.1.46, aes-256-gcm, id-aes256-GCM } @ default
{ 0.3.4401.5.3.1.9.9, CAMELLIA-128-CTR } @ default
{ 2.16.840.1.101.3.4.1.23, AES-192-OFB } @ default
{ 1.2.156.10197.1.104.1, SM4-ECB } @ default
{ 2.16.840.1.101.3.4.1.7, aes-128-ccm, id-aes128-CCM } @ default
{ 2.16.840.1.101.3.4.1.47, aes-256-ccm, id-aes256-CCM } @ default
{ 1.2.410.200046.1.1.7, ARIA-192-CBC, ARIA192 } @ default
{ 2.16.840.1.101.3.4.1.45, AES-256-WRAP, AES256-WRAP, id-aes256-wrap } @ default
{ 1.2.410.200046.1.1.15, ARIA-256-CTR } @ default
{ 1.2.410.200046.1.1.3, ARIA-128-CFB } @ default
{ 1.2.410.200046.1.1.34, ARIA-128-GCM } @ default
{ 1.2.410.200046.1.1.6, ARIA-192-ECB } @ default
{ 2.16.840.1.101.3.4.1.26, aes-192-gcm, id-aes192-GCM } @ default
{ 0.3.4401.5.3.1.9.29, CAMELLIA-192-CTR } @ default
{ 0.3.4401.5.3.1.9.43, CAMELLIA-256-OFB } @ default
{ 1.2.156.10197.1.104.2, SM4, SM4-CBC } @ default
{ 1.2.410.200046.1.1.37, ARIA-128-CCM } @ default
{ 2.16.840.1.101.3.4.1.27, aes-192-ccm, id-aes192-CCM } @ default
{ 1.3.14.3.2.17, DES-EDE, DES-EDE-ECB } @ default
{ 1.2.410.200046.1.1.11, ARIA-256-ECB } @ default
{ 1.3.111.2.1619.0.1.1, AES-128-XTS } @ default
{ 2.16.840.1.101.3.4.1.5, AES-128-WRAP, AES128-WRAP, id-aes128-wrap } @ default
{ 2.16.840.1.101.3.4.1.3, AES-128-OFB } @ default
{ 0.3.4401.5.3.1.9.3, CAMELLIA-128-OFB } @ default
{ 0.3.4401.5.3.1.9.1, CAMELLIA-128-ECB } @ default
{ 1.2.840.113549.3.7, DES-EDE3-CBC, DES3 } @ default
{ 0.3.4401.5.3.1.9.44, CAMELLIA-256-CFB } @ default
{ 1.2.410.200046.1.1.10, ARIA-192-CTR } @ default
{ 0.3.4401.5.3.1.9.23, CAMELLIA-192-OFB } @ default
{ 0.3.4401.5.3.1.9.24, CAMELLIA-192-CFB } @ default
{ 1.2.410.200046.1.1.9, ARIA-192-OFB } @ default
{ 1.2.410.200046.1.1.13, ARIA-256-CFB } @ default
{ 2.16.840.1.101.3.4.1.1, AES-128-ECB } @ default
{ 1.2.410.200046.1.1.8, ARIA-192-CFB } @ default
{ 1.2.156.10197.1.104.7, SM4-CTR } @ default
{ 2.16.840.1.101.3.4.1.43, AES-256-OFB } @ default
{ 1.2.410.200046.1.1.4, ARIA-128-OFB } @ default
{ 1.2.392.200011.61.1.1.1.3, CAMELLIA-192-CBC, CAMELLIA192 } @ default
{ 0.3.4401.5.3.1.9.21, CAMELLIA-192-ECB } @ default
{ 1.2.410.200046.1.1.5, ARIA-128-CTR } @ default
{ 2.16.840.1.101.3.4.1.21, AES-192-ECB } @ default

NULL @ default
AES-128-CBC-CTS @ default
AES-192-CBC-CTS @ default
AES-256-CBC-CTS @ default
AES-256-CFB1 @ default
AES-192-CFB1 @ default
AES-128-CFB1 @ default
AES-256-CFB8 @ default
AES-192-CFB8 @ default
AES-128-CFB8 @ default
AES-256-CTR @ default
AES-192-CTR @ default
AES-128-CTR @ default
AES-256-OCB @ default
AES-192-OCB @ default
AES-128-OCB @ default
AES-128-SIV @ default
AES-192-SIV @ default
AES-256-SIV @ default
AES-128-GCM-SIV @ default
AES-192-GCM-SIV @ default
AES-256-GCM-SIV @ default
{ AES-256-WRAP-INV, AES256-WRAP-INV } @ default
{ AES-192-WRAP-INV, AES192-WRAP-INV } @ default
{ AES-128-WRAP-INV, AES128-WRAP-INV } @ default
{ AES-256-WRAP-PAD-INV, AES256-WRAP-PAD-INV } @ default
{ AES-192-WRAP-PAD-INV, AES192-WRAP-PAD-INV } @ default
{ AES-128-WRAP-PAD-INV, AES128-WRAP-PAD-INV } @ default
AES-128-CBC-HMAC-SHA1 @ default
AES-256-CBC-HMAC-SHA1 @ default
AES-128-CBC-HMAC-SHA256 @ default
AES-256-CBC-HMAC-SHA256 @ default
ARIA-256-CFB1 @ default
ARIA-192-CFB1 @ default
ARIA-128-CFB1 @ default
ARIA-256-CFB8 @ default
ARIA-192-CFB8 @ default
ARIA-128-CFB8 @ default
CAMELLIA-128-CBC-CTS @ default
CAMELLIA-192-CBC-CTS @ default
CAMELLIA-256-CBC-CTS @ default
CAMELLIA-256-CFB1 @ default
CAMELLIA-192-CFB1 @ default
CAMELLIA-128-CFB1 @ default
CAMELLIA-256-CFB8 @ default
CAMELLIA-192-CFB8 @ default
CAMELLIA-128-CFB8 @ default
{ DES-EDE3, DES-EDE3-ECB } @ default
DES-EDE3-OFB @ default
DES-EDE3-CFB @ default

DES-EDE3-CFB8 @ default
DES-EDE3-CFB1 @ default
DES-EDE-CBC @ default
DES-EDE-OFB @ default
DES-EDE-CFB @ default
{ 1.2.156.10197.1.104.8, SM4-GCM } @ default
{ 1.2.156.10197.1.104.9, SM4-CCM } @ default
{ 1.2.156.10197.1.104.10, SM4-XTS } @ default
ChaCha20 @ default
ChaCha20-Poly1305 @ default