PQBench: Post-Quantum Encrypted Network Traffic Classification Benchmarking

Eylon Y. Katan*, Noam Leshem*, Kostas Pashiourtides†, Amit Dvir, IEEE Senior Member‡, Angelos K. Marnerides, IEEE Senior Member†, Ran Dubin‡, Revital Marbel§, Chen Hajaj¶
*School of Computer Science, Ariel University, Israel
[eylonyaa.katan,noam.leshem]@msmail.ariel.ac.il
†Dept. of Electrical & Computer Engineering & KIOS CoE, University of Cyprus, Nicosia, Cyprus
[marnerides.angelos,pashiourtides.costas]@ucy.ac.cy
‡Dept. of Computer & Software Engineering, Ariel Cyber Innovation Center, Ariel University, Israel
[rand,amitdv]@ariel.ac.il
§School of Computer Science, Holon Institute of Technology, Israel
marbelr@hit.ac.il
¶Dept. of Industrial Engineering and Management & Data Science and AI Research Center, Ariel University, Israel
chenha@ariel.ac.il

***Abstract*—Post-quantum cryptography (PQC) is expected to revolutionize secure communications in next-generation digital ecosystems. While prior studies show that different PQC algorithms can significantly impact traffic latency, there is still no standardized approach for detecting the presence of PQC or identifying the specific algorithm in encrypted traffic for traffic engineering purposes.**

We present PQBench, the first comprehensive framework for benchmarking the classification of encrypted Internet traffic using PQC algorithms. Our approach involves data collection from multiple network environments across browsers, operating systems, and cryptographic algorithms, resulting in two novel international datasets. Using time–direction–length flow features, we train machine learning models to perform traffic analysis, achieving 93% accuracy in PQC detection, 95% in browser identification, 99% in operating system classification, and 78% in PQC algorithm differentiation.

This research offers practical insights into the performance and security implications of quantum-resistant cryptography in real-world scenarios, enabling network optimization and helping cybersecurity practitioners develop PQC-based architectures. By establishing a standardized and reproducible pipeline, PQBench paves the way for replicating existing work and integrating novel approaches. In the spirit of open science, both our framework and datasets are publicly available in our GitHub repository.

***Index Terms*—Post-Quantum Cryptography, Classification, Encrypted Traffic, Benchmarking**

I. Introduction

The field of cryptography stands at a critical point as the advent of quantum computing threatens to undermine the security of current cryptographic systems. This impending challenge has given rise to the rapidly evolving field of Post-Quantum Cryptography (PQC), which aims to develop cryptographic systems resistant to attacks from quantum computers. At the forefront of this effort stands the National Institute of Standards and Technology (NIST) [1], which selected a set of PQC algorithms for digital signature and key establishment protocols such as ML-KEM and CRYSTALS-Kyber [2]. These algorithms were chosen for their strong resistance to quantum attacks and their efficient implementation. For instance, CRYSTALS-Dilithium and FALCON are based on the difficulty of lattice-based problems, striking a balance between performance and security. Additionally, other PQC approaches have been tested and implemented, such as SPHINCS+, a stateless hash-based signature scheme recognized for its robust security guarantees, although it does result in larger signature sizes.

As these algorithms are gradually integrated into real-world systems and protocols such as TLS, their influence extends beyond cryptographic security into other domains of network analysis. One such domain is encrypted traffic classification, where the introduction of PQC brings new types of metadata, handshake patterns, and algorithm identifiers that may impact how traffic is analyzed and understood. The field of encrypted traffic classification addresses the challenge of understanding and categorizing network traffic increasingly protected by sophisticated encryption methods. This task has become increasingly complex over recent years, prompting researchers to explore innovative approaches. Specifically, integrating deep learning and artificial intelligence models has introduced additional and improved research perspectives to these approaches [3]–[8], enabling more nuanced and accurate classification of encrypted traffic.

Although current PQC research has primarily focused on characterizing the performance overhead of quantum-resistant algorithms [9]–[15], the classification of meta-data within PQC traffic represents a critical gap in the research, highlighting the lack of studies in this area. Addressing this gap is crucial, particularly in light of the upcoming changes to network protocols. For instance, the list of PQC algorithms a client uses, currently visible in the ”supported groups” and ”signature algorithms” fields of TLS Client Hello packets, will soon be encrypted using the Encrypted Client Hello (ECH) TLS extension.

In this paper, we tackle several classification tasks, aiming to utilize parameters unrelated to the encrypted content but focused solely on metadata. Specifically, we employ a set of classifiers to investigate how simplistic traffic properties such as packet size, time, and direction (TDL) can enhance our understanding of the properties of PQC-enabled traffic, thereby establishing a benchmark for general PQC classification performance. In particular, we aim to further analyze the traffic to accurately predict the underlying browser and operating system (OS) of the traffic. This finding has multiple implications, both in the privacy and cybersecurity aspects.

By emphasizing real-world applicability beyond mere performance analysis, our methodology offers valuable insights into the behavior and characteristics of PQC in practical network environments. Such insights can be instrumental for network administrators, security researchers, and protocol designers seeking to effectively implement and optimize PQC solutions in the face of evolving quantum computing capabilities.

The following research questions highlight the areas of interest:

- From a network perspective, is TDL sufficient to differentiate between PQC-encrypted traffic and No-PQC-encrypted traffic?
- From a privacy perspective, given PQC-encrypted traffic, how accurately can we determine the OS and browser used by the client, which may implement different PQC algorithms?
- From a cybersecurity perspective, can we accurately classify the cryptographic protocols (e.g., CRYSTALS-Kyber, ML-KEM) based on their usage of different PQC algorithms?

The abbreviations used throughout this paper are defined in Table I.

TABLE I: List of abbreviations used in this paper

ADB	AdaBoost
DT	Decision Tree
GB	Gradient Boosting
IAT	Inter-Arrival Time
KEM	Key Encapsulation Mechanism
KNN	K-Nearest Neighbors
LR	Logistic Regression
MLP	Multi-Layer Perceptron
NB	Naive Bayes
NIST	National Institute of Standards and Technology
OS	Operating System
POB	PQC, OS, Browser
PQC	Post-Quantum Cryptography
RF	Random Forest
TDL	Time-Direction-Length
XGB	XGBoost
ACC	Accuracy
PRE	Precision
REC	Recall
F1	F1-Score
STD DEV	Standard Deviation

II. Contributions

We introduce a novel framework, referred to as PQBench, for classifying encrypted network traffic that utilizes PQC algorithms.[1] This approach addresses a significant gap in our understanding of next-generation secure communications.

Since datasets with the above labels are not publicly available, and to effectively assess PQBench, we collected and provided two new comprehensive datasets of PQC and No-PQC-based traffic patterns from various operating systems, web browsers, and PQC algorithms, alongside an open-source analysis toolkit [16], which the research community may reuse for the study of PQC traffic patterns classification. Our provided datasets differ from other PQC benchmarking datasets by being solely network-based while utilizing application-layer protocols, whereas datasets such as the ones provided by Mallick et al. [17] or Rocha et al. [18] suggest Memory-based data, such as object and text files, and do not simulate a network or application-layer protocols at all.

We were able to achieve high accuracy in classifying Post-Quantum Key Encapsulation Mechanism (KEM) algorithms while utilizing various data processing methods to augment classification quality. This sets a benchmark for advancements in the field, fosters collaborative progress in cybersecurity, and lays the groundwork for more resilient and efficient network infrastructures in the post-quantum era.

Specifically, the different contributions of this work can be summarized as follows:

[1]From this point onwards, we refer to traffic deploying PQC algorithms as PQC encrypted network traffic

- Conceptualizing and developing PQBench, a novel classification framework capable of identifying PQC encryption usage and specific Post-Quantum KEMs from encrypted traffic.
- Introduction of an open-source PQC traffic classification toolkit [16], which includes components for capturing algorithm-level traffic across multiple OS environments.
- Creation of two publicly available, diverse, and reusable datasets containing PQC and No-PQC encrypted network traffic, enabling reproducibility and benchmarking for future PQC traffic classification research.
- Demonstration of strong classification performance in identifying PQC-encrypted traffic and additional derived data, such as browsers, operating systems, and KEMs, from real-world encrypted streams.

The remainder of this paper is structured as follows: Section III reviews the background and existing work on PQC and traffic classification. Section IV introduces the mechanism and composition of the PQBench framework. Section V discusses our experimental setup, while Sections VI, VII are dedicated to analyzing and interpreting our results. Finally, Section VIII provides insights from the results, summarizes, and concludes this work.

III. Background and Related Work

In this section, we discuss the current state of modern PQC algorithms (KEM and Digital Signatures) and their deployment to present-day systems. Then, we review recent confluences between PQC and modern internet networking, while examining security and performance measurements. Finally, we inspect PQC classification experiments, the environment and conditions in which they were performed, and their results.

A. *PQC algorithms overview*

Each cryptography algorithm has three stages: key exchange, sign, and encrypt, utilizing different methods and techniques to ensure quantum resiliency. Recently, NIST has announced the standardization of selected PQC key encapsulation mechanisms and digital signature algorithms.

Module-Lattice-Based Key-Encapsulation Mechanism **(ML-KEM)** (*NIST FIPS 203*) [19] ensures secure key establishment in the presence of quantum-capable adversaries. ML-KEM is derived from **CRYSTALS-Kyber**, a member of the Cryptographic Suite for Algebraic Lattices (CRYSTALS). The construction of Kyber follows a two-stage approach: First, we employ Kyber.CPAPKE—an IND-CPA-secure public-key encryption scheme, meaning that no Probabilistic Polynomial-Time (PPT) adversary can distinguish ciphertexts of any two chosen equal-length messages with non-negligible advantage—to encrypt 32-byte plaintext messages. Then, a Fujisaki–Okamoto (FO) transform is applied to construct the IND-CCA2-secure KEM [20]. ML-KEM's security relies on the hardness of the Module Learning with Errors (MLWE) problem. It specifies three parameter sets: ML-KEM-512, ML-KEM-768, and ML-KEM-1024, offering increasing levels of security.

Module-Lattice-Based Digital Signature Algorithm **(ML-DSA)** (*NIST FIPS 204*) [21] inherits design principles from **CRYSTALS-Dilithium**, whose signatures rely on the Fiat–Shamir heuristic to convert an interactive identification protocol into a non-interactive one. Building on Dilithium [22], ML-DSA applies a Fiat–Shamir-with-Aborts transformation and augments it with both hedged (randomized) and deterministic signing modes, achieving strong existential unforgeability under chosen-message attack in the quantum random-oracle model. ML-DSA defines three parameter sets: ML-DSA-44, ML-DSA-65, and ML-DSA-87.

Stateless Hash-Based Digital Signature Algorithm **(SLH-DSA)** (*NIST FIPS 205*) [23] provides post-quantum security grounded solely in the pre-image and collision resistance of SHA-2 or SHAKE rather than on number-theoretic assumptions. SLH-DSA is based on **SPHINCS+**, which is an advanced version of the original Stateless Practical Hash-based Incredibly Nice and Compact Signatures (SPHINCS) scheme, and is recognized for its robust security. Unlike other signature schemes that require maintaining state information (such as a counter), SPHINCS+ is stateless, simplifying its implementation; it relies on the Winternitz One-Time Signature Plus (WOTS+) and a Forest of Random Subsets (FORS) for generating signatures, ensuring strong security through hash-based cryptographic techniques [24]. A signature hashes the message with a per-signature randomizer, signs part of that digest with a pseudorandomly selected FORS key, and authenticates that key through a hypertree of eXtended Merkle Signature Scheme (XMSS) layers built from Winternitz one-time signatures, allowing unlimited signing without maintaining state. Twelve approved parameter sets balance signature size versus signing speed across NIST security categories 1, 3, and 5, making SLH-DSA practical for high-assurance governmental and commercial applications.

NIST will release a FALCON-based standard soon, where **FALCON** algorithm is a post-quantum digital signature scheme engineered to provide robust security against quantum attacks while maintaining computational and storage efficiency. It is grounded in the Number Theory Research Unit (NTRU) lattice problem, which relies on the difficulty of finding the shortest vector in a lattice generated by a polynomial ring and is considered difficult to solve by both classical and quantum computers. NIST has also standardized the Hamming Quasi-Cyclic **(HQC)** algorithm, a code-based key encapsulation mechanism designed to ensure robust security against quantum attacks while maintaining computational practicality and compact key sizes. HQC's security is rooted in the computational complexity of decoding random errors from structured quasi-cyclic codes, a problem recognized as computationally infeasible for both classical and quantum adversaries. This security foundation positions HQC as an efficient and reliable candidate for safeguarding future cryptographic systems against quantum threats.

B. *PQC Characterization in Network Traffic*

Recently, several works have focused on the integration of PQC into core internet networking, examining the performance and feasibility of PQC ciphers in conjunction with key enablers of secure internet networking, including Public-Key Infrastructure (PKI) for trust and protocols such as TLS 1.3 and QUIC for secure session establishment. Such works [9]–[15] try to understand how PQC algorithms integrate and impact the existing network infrastructure.

Rios et al. [25] measured the number of successful TLS handshakes over a specific period and discovered that complex PQC algorithms tend to create more transmission overhead compared to their No-PQC counterparts at equivalent security levels. Additionally, in their evaluation of the impact on handshake performance, they observed that using post-quantum primitives generally results in higher amount of completed TLS handshakes compared to classical algorithms. Sikeridis et al. [9] conducted a comprehensive performance study on the impact of post-quantum signature algorithm candidates on TLS 1.3 under realistic network conditions, providing valuable data on how these new algorithms might affect real-world network performance. Building on this, Raavi et al. [10], [11] presented an in-depth analysis of post-quantum digital signature performance overheads regarding computational and memory requirements. The authors' work also included the development of a security comparison model for understanding and comparing the security of algorithms with different hardness problems, offering a pipeline for evaluating the trade-offs between security and performance in PQC algorithms.

A study by Henrich et al. [12] delved into the specifics of the TLS handshake and KEMs under varying network characteristics, aiming to identify suitable quantum-safe algorithms within TLS 1.3 and considering various PQC KEMs and KEM parameters. Given the prevailing network quality, the authors provided valuable recommendations regarding the use of various algorithms and configurations. Notably, their work demonstrated how unwanted scheduling of operating system processes may affect performance and how TCP control mechanisms heavily influence handshake performance. Corsi et al. [26] measured the influence of Post-Quantum KEM algorithms on TLS CPU utilization compared to standard algorithms in 5G networks. By repeatedly registering a User Equipment (UE) using KEM as measurement, they discovered that PQC causes significant CPU utilization, with Kyber being the lowest and HQC being the highest. Additionally, their results indicate that while Kyber-768 and Kyber-1024 exhibit median performance comparable to TLS, the overall impact of KEMs on CPU usage varies significantly, highlighting the need for careful algorithm selection in high-demand scenarios.

In the mobile domain, Mankowski et al. [14] analyzed TLS usage by the highest-ranked apps from the Google Play Store. This work assessed the potential overhead from adopting post-quantum algorithms in mobile applications, providing insights into the practical implications of transitioning to PQC in the mobile ecosystem.

C. *PQC Classification*

In the realm of PQC-encrypted network traffic classification, a study conducted by Mallick et al. [17] has attempted to fingerprint both the KEM and the Digital-Signature algorithm used across multiple operating systems from different libraries and protocol implementations by relying solely on CPU and memory-level features. Leveraging coarse memory metrics together with per-core cycle counters, their XGBoost classifier separates classical from post-quantum key-exchange traffic with 98–100% accuracy, identifies the exact PQC KEM in a seven-class task with 97% accuracy, and reaches 86% accuracy when distinguishing among digital-signature families. The same feature set exposes implementation-level differences: Kyber and FrodoKEM binaries compiled with `liboqs` or `CIRCL` are discriminated with roughly 96–100% accuracy, and CIRCL's hybrid X25519+Kyber handshakes are distinguished from their pure-PQC counterparts at about 97%.

Another experiment in PQC classification was conducted by Rocha et al. [18] who attempted to classify various Post-Quantum KEMs (Kyber, etc.) from encrypted text files, based on various file statistics (Frequency, Overlapping, etc.) while changing the encrypted file size. They have received up to 94% accuracy in the multi-classification task. Their method leverages the NIST SP 800-22 statistical test suite to extract distinguishing features from ciphertexts, which are then processed through multiple machine learning models.

Those classification experiments highlight the necessity of our paper, being the first network-based classification benchmarking framework utilizing application-layer protocols, with results based on real-world traffic scenarios.

IV. PQBench

This section introduces the core elements of our PQC classification (PQBench) framework. Our approach combines sophisticated data collection methodologies with advanced feature extraction techniques to develop a robust classification system to identify subtle patterns in encrypted traffic. We begin by detailing the dataset collection process, which provides the empirical foundation for our research. Following this, we explain the feature extraction process, a crucial step that distills traffic representations into fundamental characteristics such as packet size, timing, and directional flow. These features form the basis for training a comprehensive suite of classification models. Each model is evaluated on various classification tasks, ranging from the binary PQC/No-PQC distinction to more complex multi-class tasks, including algorithm classification.[2] The full PQBench pipeline can be inferred from Fig. 1. Each different component will be explained in the following sections.

A. Lab Environment

The data collection process for our research was organized into several clearly defined phases to ensure a comprehensive and accurate representation of PQC-based network traffic across different environments.

Data was recorded in two countries: Israel and Cyprus. In Israel, we utilized three different sites for data collection, while in Cyprus, all the data was collected from a single site. A breakdown of the machine configurations used for data capture, categorized by collection sites, is presented in Table II.

Environment Setup: We configure various operating systems (i.e., Windows, macOS, Linux) and web browsers (i.e., Chrome, Firefox), with the PQC flag set to true for enabling PQC traffic, and false for capturing No-PQC traffic. This setup ensures we capture various environments to simulate real-world conditions accurately. Additionally, two PQC algorithms for encryption are integrated into the system. Since TLS authentication must be executed when a connection or conversation is established, rather than throughout the entire lifespan of the data transmission, as of today, browsers are not signing their information using PQC algorithms. Moreover, the consensus is that there are currently no strong enough quantum computers that can forge or break the signature, so signing isn't needed, yet mitigation will need to occur eventually.

[2]From this point onwards, we refer to the KEM algorithms as PQC algorithms

Automation and Interaction Simulation: We employ web automation tools such as Selenium [27] to replicate human browsing interactions. These tools allow us to create crawlers that mimic user behavior, interacting with websites just as a human user would. The system automatically navigates websites, streams content, and performs interactions to simulate typical user activity.

Traffic Capture: During this phase, we capture encrypted network traffic generated by the simulated interactions. The traffic includes the combinations of operating systems, web browsers, and PQC algorithms. Once the traffic is captured, we extract and analyze it.

Data Anonymization and Storage: In the second traffic recording phase, we ensure all data is anonymized according to EU GDPR privacy regulations [28] before storage. This step ensures compliance with data privacy standards while preserving the integrity of the dataset for future analysis.

Feature Extraction: We focus on the unique characteristics of modern secure communication protocols, highlighting the differences between PQC and traditional cryptographic methods, as well as the variations among different PQC algorithms. Specifically, our approach relies on three fundamental features: Packet timestamp, Packet direction, and Packet size. In particular, from each packet within a flow, we extract its relative time (to the first packet in the flow), its direction (client-to-server/server-to-client), and its length (size in bytes). The rationale behind choosing these features is twofold: (1) traditional features such as TCP window size and Server Name Indication (SNI) will soon be inaccessible due to the increasing adoption of QUIC, which uses UDP; and (2) adding authenticity and encryption introduces overhead [10], [11], which affects packet transmission times.

B. Datasets

We generated two different and diverse datasets for various classification tasks of the PQBench framework, where each dataset was specially collected for a different classification task.

PQC, OS, Browser (POB) classification - PQC-POB: This dataset is designed to capture three key attributes in network traffic: PQC/NO-PQC, operating system, and browser (hereafter, this dataset is named POB). It contains 2400 recordings, and has 12 classes, based on the underlying OS (Windows, Linux, macOS), the browser used (Firefox, Chrome), and whether the data is PQC encrypted or not, as can be inferred from Table III. To ensure consistency, we used the Kyber algorithm for PQC-encrypted data and X25519 for No-PQC data. To ensure diversity in network behavior and conditions, each set of samples was recorded from

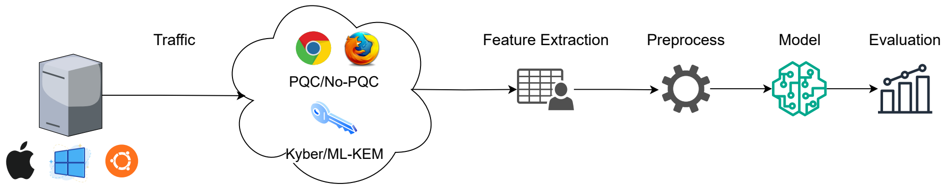

Fig. 1: PQBench framework pipeline

TABLE II: Hardware specifications of recording computers across all sites.

Site(s)	OS	CPU	Cores	RAM (GB)
Cyprus - Site 1	Ubuntu 22.04	Intel i5-10500	12	15
	macOS 15.5	Apple M4 Pro	12	64
	Win 11 Pro	Intel Ultra 7 155H	16	15.4
Israel - Site 1 & 2	Win 10	Intel i7-8565U	8	16
Israel - Site 2 & 3	Win 10 Pro	Intel i7-12700H	20	32
	Arch Linux 6.15.9	Intel i7-12700H	20	32
	macOS 15.3	Intel i5-4260U	4	4

TABLE III: PQC-POB Dataset

OS	Browser	PQC	Samples
Windows	Firefox	X	200
	Firefox	✓	200
	Chrome	X	200
	Chrome	✓	200
MacOS	Firefox	X	200
	Firefox	✓	200
	Chrome	X	200
	Chrome	✓	200
Linux	Firefox	X	200
	Firefox	✓	200
	Chrome	X	200
	Chrome	✓	200

a different physical location (different labs in various countries). This approach introduces variation in data capture, thereby enhancing the dataset's generalizability. For every unique combination of operating system and browser, the dataset includes 200 samples encrypted with PQC and 200 samples without PQC, resulting in 400 samples per OS-browser pair.

Algorithm classification - PQC-Algo: For the PQC algorithm classification task, we created a second dataset, named PQC-Algo. In this dataset, we recorded over a single operating system (Windows) to ensure uniformity, using either Firefox or Chrome, while changing the Kyber and ML-KEM flags to differentiate between the algorithms. That gives us four classes: ChromeML-KEM, Chrome+Kyber, Firefox+MLKEM, and Firefox+Kyber, with each class containing 600 flow samples, totaling 2400 samples as depicted in Table IV. PQC-Algo is designed for the classification of the PQC algorithm used to encrypt the data and a grouping of the algorithm and browser (called a tuple), and it is different than PQC-POB. To ensure sample variety, 300 samples were recorded in the Israeli lab, while the other 300 samples were collected in the Cypriot lab to guarantee diversity in network behavior.

TABLE IV: PQC-Algo Dataset

Browser	Algorithm	Samples
Firefox	Kyber	600
	ML-KEM	600
Chrome	Kyber	600
	ML-KEM	600

Our empirical analysis found that recording 100 connections takes 2000 seconds, with user CPU utilization time being 600 seconds while utilizing 80 MB of memory. Our datasets are based on disjoint entities and environments in the test and training sets. Ensuring that the training and testing datasets do not contain data from overlapping network entities or entire environments is a viable approach in the experimental protocol design. In the spirit of open science, our datasets, as well as our data collection mechanism and feature extraction pipeline, are publicly available on our GitHub repository [16].

C. Datasets Analysis

We performed a statistical analysis for each dataset. Each statistical measure is calculated for the entire flow of each capture file. Tables V and VI present the different datasets, and the different statistical measures calculated for each. Although the range of the packets in each dataset spans from 54 to 16K bytes, 96% of the packets are in the range of 0 to 1500 bytes.

The difference in size in ClientHello Length between Tables V and VI derives from the fact that the PQC-Algo dataset contains only PQC samples, in which the KEM manifests in the ClientHello packet, increasing the size of the packet.

Each flow in PQC-POB contains, on average, 22 ACK packets, while PQC-Algo contains 15 packets. Jerabek et al. [29] have raised a serious problem in many modern encrypted traffic classification ML-based systems, regarding the amount of duplicated samples in the dataset on which they were trained. Therefore, we have measured and calculated `TDL_similarity_pct` - percentage of unique TDL samples in the dataset and `DL_similarity_pct` - percentage of unique DL (no relative time) samples in the dataset, to ensure that such a problem does not affect our models. We achieved a `TDL_similarity_pct` of 1.2% and a `DL_similarity_pct` of 19%, ensuring data complexity and variety.

V. Experiments Design

A. Learning Models

We tested the classification performance using several classification models. Specifically, the models are: Decision Tree, Random Forest, XGBoost, AdaBoost, K-Nearest Neighbors (KNN), Logistic Regression, and Gradient Boosting, Gaussian Naive Bayes and Multi-Layer Perceptron (MLP).

This diverse selection of models allows for a broad exploration of the underlying data structure, enabling us to compare model performance across different methodologies and identify the most suitable approach for our specific problem domain. Tree-based models such as Decision Tree leverage decision nodes to partition the feature space, we also use Random Forest, employing an ensemble of trees grown on randomly selected subsets of features to improve generalizability. Ensemble methods, including XGBoost, AdaBoost, and Gradient Boosting, iteratively build multiple models, with each successive model focusing on correcting the errors of the previous iterations. XGBoost, in particular, is notable for its computational efficiency and effectiveness in handling large datasets. On the other hand, distance-based models, such as KNN, classify data points based on the proximity to their nearest neighbors in the feature space, offering an intuitive approach for classification tasks. Logistic Regression, a discriminative model, estimates the posterior probability of class membership by modeling the decision boundary directly. We also employed the MLP classifier, a type of neural network that captures complex, non-linear relationships by propagating information through multiple hidden layers. It has a single hidden layer with 100 neurons, with Rectified Linear Unit (ReLU) activation function.

B. Metrics

The performance of each model is assessed based on common classification evaluation metrics: accuracy, precision, recall, F1-score, and Area Under the Curve (AUC). These metrics are crucial for determining the most effective and practical models for real-world encrypted internet traffic classification.

Accuracy measures the proportion of correctly predicted instances out of all instances in the dataset. It is calculated by dividing the sum of true positives and true negatives by the total number of instances (true positives, true negatives, false positives, and false negatives). Accuracy is calculated using the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Precision measures the proportion of correctly predicted positive instances out of all instances predicted as positive. It is calculated by dividing the number of true positives by the sum of true positives and false positives. The Precision formula is as follows:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Recall measures the proportion of correctly predicted positive instances out of all actual positive instances. It is calculated by dividing the number of true positives by the sum of true positives and false negatives. The Recall formula is as follows:

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

F1-score is a harmonic mean of precision and recall that considers both metrics to provide an overall measure of the model's performance. It is calculated by taking the harmonic mean of precision and recall, with higher values indicating better performance. The F1-score formula is:

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4}$$

Finally, AUC measures the overall ability of the model to distinguish between positive and negative classes across various thresholds. It is calculated as the area under the Receiver Operating Characteristic (ROC) curve, which plots the True Positive Rate (Recall) against the False Positive Rate. Higher AUC values reflect a better model performance, showing a greater capacity to accurately identify positive and negative instances

TABLE V: PQC-POB Dataset Statistics

	PQC			**No-PQC**		
	Range	Mean	Std Dev	Range	Mean	Std Dev
Size	[54, 16277]	610.436	887.321	[54, 13098]	589.726	863.429
IAT	[0, 4897]	20.687	190.437	[0, 3884]	20.320	184.547
ClientHello	[1434, 2452]	1646.364	213.980	[571, 914]	646.643	65.096
ServerHello	[1366, 4378]	1971.080	857.101	[278, 3290]	1708.864	403.538

TABLE VI: PQC-Algo Dataset Statistics

	ML-KEM			**Kyber**		
	Range	Mean	Std Dev	Range	Mean	Std Dev
Size	[54, 3912]	398.972	644.101	[66, 15367]	621.863	1222.828
IAT	[0, 1918]	36.448	209.597	[0, 2441]	32.699	196.069
ClientHello	[1454, 2467]	2029.189	239.400	[1801, 2136]	1919.052	83.303
ServerHello	[1434, 3912]	2635.015	1198.745	[1434, 4379]	3329.208	1223.611

at multiple threshold settings. Including AUC in our evaluation offers a broader perspective on the model's performance, as it encapsulates the trade-offs between Recall and False Positive Rate across thresholds.

C. Experimental Setup and Preprocessing

We evaluated our PQBench framework on several classification tasks, as listed in Table VII. The motivation is to further our understanding of the impact PQC has on network traffic across multiple platforms.

To build a real time classification model, we chose to take the first 20 packets from each flow. Then, we extracted the flow's TDL information.

We performed a statistical overview and analysis of the processed data, which is presented in Tables VIII and IX. We omitted the ClientHello and ServerHello statistics from these tables since the results will be the same as in Tables V and VI. We observe that preprocessing resulted in reduced mean packet size and relative times without significantly affecting the dataset's class distribution. The observed differences in Mean Packet Size between Tables V and VIII, as well as between Tables VI and IX, stem from removing Physical and Link layer headers and retaining only IP packet sizes. Additionally, the lower IAT indicates that TCP and TLS handshake packets dominate the initial 20-packet segment, whereas most object transfers occur later, influenced by HTTP. The max-sized packets are singular packets with large sizes, while the rest of the packets don't pass the 1500-byte MTU.

VI. PQC-POB Experimental Results

For each experiment, we used 10-fold stratified cross-validation to verify the robustness of our results.

A. PQC encryption classification

In the first experiment, we classify encrypted traffic to determine whether it employs PQC traffic. Consequently, this experiment involves a binary classification task between PQC and No-PQC data, with results summarized in Table X. Our evaluation indicates that a decision tree can successfully identify the presence of a PQC algorithm with an accuracy of 92%. It demonstrates a high recall, accurately identifying 91% of the samples that utilize the PQC algorithm. When using more complex ensemble-based boosting models such as XGBoost, we can improve our accuracy to 93%, with a recall of 93%.

Increment of packets per flow: In this classification task, we aimed to check how changes in the amount of data in each flow affect our ability to classify whether the data is PQC or not. From Fig. 6, we can see that the increment improved the accuracy result of the PQC binary classification, highlighting the fact that not many packets are required to achieve good results.

B. Browser and Operating System classification

Next, our objective was to determine whether traffic could be accurately classified, assuming that the encrypted traffic employed PQC. We analyzed the subtle distinctions between three operating systems (Windows, Linux, and macOS) and two web browsers (Firefox and Chrome) while using PQC encryption. For these experiments, we used a subset of the dataset consisting solely of PQC samples.

Table XI shows browser classification results, with XGBoost achieving the best performance at 95% accuracy. Table XII presents OS classification results, where Random Forest reaches 99% accuracy. The results reveal that ensemble methods significantly outperform traditional algorithms for browser classification, with a notable performance gap between sophisticated models

TABLE VII: Experiment setup for all classification tasks

Task	Expiriment	Details
POB	PQC binary	Traffic is using a PQC or not
	Browser	Different browsers given PQC
	OS	Different operating systems given PQC
	Tuple (Browser, OS, and PQC)	Specific OS, specific browser, and PQC
	Increment of packet per flow	PQC binary task with increased the number of packets
Algo	ML-KEM	Different KEM algorithms
	Tuple (ML-KEM, Browser)	KEM algorithm alongside the browser used

TABLE VIII: PQC-POB Pre-Processing Statistics

	PQC			No-PQC		
	Range	Mean	Std Dev	Range	Mean	Std Dev
Size	[20, 4364]	406.06	610.82	[20, 3276]	351.85	498.03
IAT	[0, 465]	0.14	4.13	[0, 344]	0.17	4.08

TABLE IX: PQC-Algo Pre-Processing Statistics

	ML-KEM			Kyber		
	Range	Mean	Std Dev	Range	Mean	Std Dev
Size	[52, 3898]	479.35	787.36	[52, 4365]	527.77	890.58
IAT	[0, 123]	0.05	1.35	[0, 76]	0.03	0.70

TABLE X: PQC vs. No-PQC Classification Results

Model	Acc	Pre	Rec	F1	AUC
ADB	0.90	0.90	0.90	0.90	0.95
DT	0.92	0.92	0.91	0.92	0.92
GB	0.92	0.93	0.92	0.92	0.96
KNN	0.87	0.88	0.87	0.87	0.93
LR	0.87	0.88	0.87	0.87	0.92
MLP	0.87	0.87	0.86	0.87	0.91
NB	0.81	0.84	0.81	0.81	0.92
RF	0.90	0.91	0.91	0.91	0.96
XGB	0.93	0.93	0.93	0.93	0.96

TABLE XI: Given PQC, Browsers Classification Results

Model	Acc	Pre	Rec	F1	AUC
ADB	0.92	0.92	0.92	0.92	0.96
DT	0.94	0.95	0.95	0.95	0.94
GB	0.95	0.95	0.95	0.95	0.97
KNN	0.84	0.84	0.84	0.84	0.90
LR	0.70	0.72	0.70	0.69	0.75
MLP	0.80	0.80	0.79	0.84	0.84
NB	0.65	0.71	0.65	0.62	0.76
RF	0.94	0.95	0.94	0.94	0.98
XGB	0.95	0.95	0.95	0.95	0.98

(over 94%) and simpler approaches, such as Naive Bayes (65%). Conversely, OS classification demonstrates consistently high performance across all models, indicating

TABLE XII: Given PQC, Operating Systems Classification Results

Model	Acc	Pre	Rec	F1	AUC
ADB	0.98	0.98	0.98	0.98	0.99
DT	0.98	0.98	0.98	0.98	0.99
GB	0.99	0.99	0.99	0.99	1.00
KNN	0.93	0.94	0.93	0.93	0.98
LR	0.92	0.93	0.92	0.92	0.96
MLP	0.95	0.94	0.94	0.92	0.97
NB	0.93	0.94	0.93	0.93	0.98
RF	0.99	0.99	0.99	0.99	1.00
XGB	0.99	0.99	0.99	0.99	1.00

TABLE XIII: Classify Tuple: Browser, OS, and Algorithm (PQC, No-PQC)

Model	Acc	Pre	Rec	F1	AUC
ADB	0.16	0.05	0.16	0.07	0.63
DT	0.88	0.89	0.89	0.88	0.93
GB	0.90	0.90	0.90	0.90	0.99
KNN	0.76	0.77	0.76	0.76	0.94
LR	0.67	0.67	0.67	0.66	0.94
MLP	0.76	0.74	0.74	0.72	0.94
NB	0.51	0.44	0.51	0.43	0.94
RF	0.90	0.90	0.90	0.89	0.99
XGB	0.91	0.91	0.91	0.90	0.99

that operating systems produce more distinct PQC traffic patterns than browsers, likely due to fundamental differences in network stack implementations across Windows, Linux, and macOS platforms.

C. Browser, OS, and PQC (Tuple) Classification

In the next experiment, we addressed the most complex task: classifying encrypted traffic samples based on whether they use PQC and determining the user's parameters, such as operating system and browser. This experiment aimed to identify behavioral differences in real-life scenarios. As shown in Table XIII, XGBoost and Gradient Boosting demonstrate superior performance in both accuracy and AUC, confirming their robustness in handling multi-dimensional classification tasks.

We measured the impact that the size of the dataset has on the F1-score of various models to determine the number of samples needed for successful tuple (browser, OS, PQC) classification. From Fig. 2 we see that an increment in the number of training examples yields better predicting models.

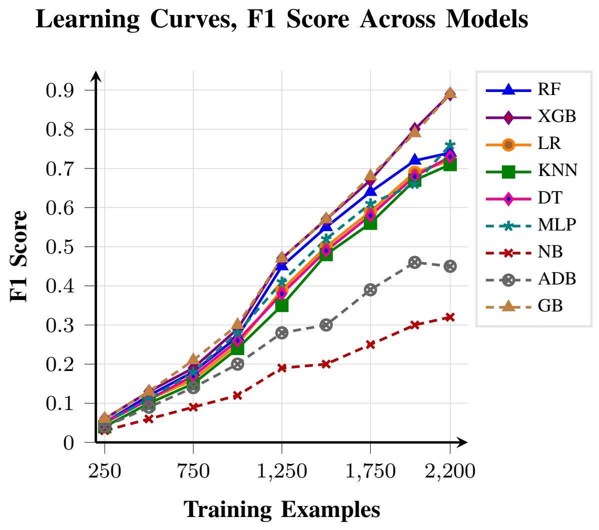

Fig. 2: **Effect of Training Sample Size on F1 Score for Multiple Models in Tuple Classification (POB Task).** XGBoost and Gradient Boosting lead performance, especially as data volume increases.

Then, we used our best-performing model to evaluate its performance in each metric across different sample sizes. We can see from Fig. 3 a linear increase in precision, recall, and F1-score values as the number of samples grows. The accuracy values are the same as the recall values; hence, they're not shown on this graph.

Next, we examine how the number of packets within a flow influences model accuracy for classifying the tuple (browser, OS, PQC). As illustrated in Fig. 4 , the top-performing models already yield strong results after the first 10 packets, a pattern closely linked to the handshake, which is where the KEM effect on the traffic manifests.

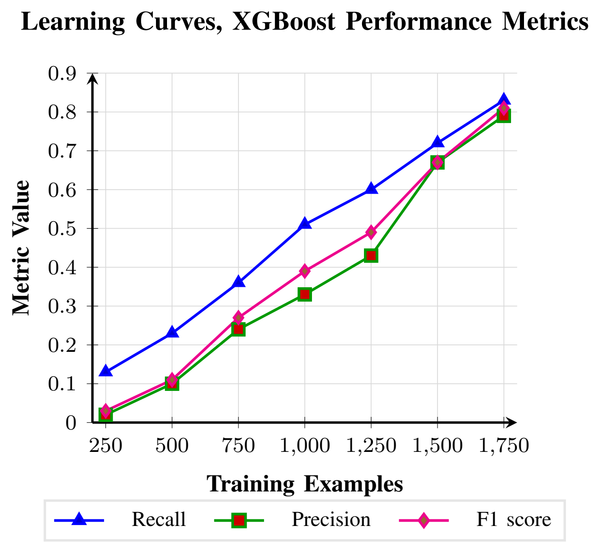

Fig. 3: **Effect of Training Sample Size on XGBoost Performance Metrics.** All metrics rise with sample size, with F1 score near 0.8 by 1750 examples.

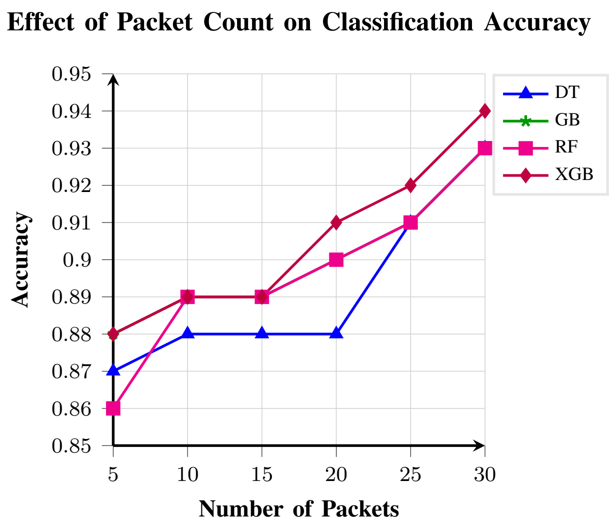

Fig. 4: **Effect of Packet Count on Classification Accuracy for Tuple-Level Prediction in the POB Task.** All models quickly approach high accuracy, with XGBoost achieving the highest score at 30 packets.

Then, we used our best-performing model (XGBoost) to see the impact of packet increment on its performance using several different metrics. From Fig. **??** we can see that, similarly to Fig. 4, 10 packets deliver strong results yet again. This strongly correlates with current studies such as Pesek et al. [30], demonstrating strong classification results using only the first 10 packets of each flow as samples and TDL as their features.

We also examined the feature importance of our model. From Fig. 5, we see that the most important feature is the length of the 4^{th} packet. Upon inspecting our data, we see that the fourth packet is the TLS

ClientHello packet, in which the PQC algorithm chosen manifests as a part of the "supported groups" and "signature algorithms" fields. The second most important feature is the length of the first packet - the SYN.

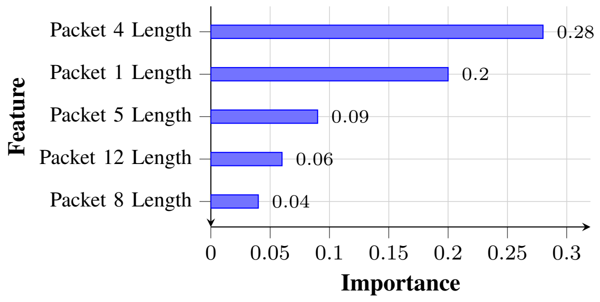

Fig. 5: Feature importance for the Decision Tree in the POB classification task, sorted by descending contribution.

In all experiments for the PQC-POB classification tasks, we have seen that XGBoost achieves the best results in each experiment. We also observed that the Random Forest model consistently delivers strong results. All best results for each classification task can be inferred from Fig. 6.

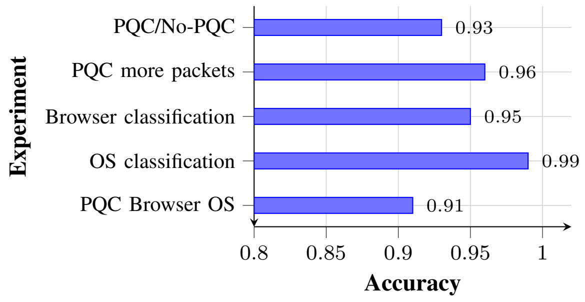

Fig. 6: Summary of accuracy across experiments for the POB classification task.

VII. PQC-Algo Experimental Results

In this section, we use the PQC-Algo dataset, explicitly crafted for this classification task, which contains traffic of the two types of PQC algorithms, ML-KEM and Kyber [19]. We analyze the subtle classification distinctions between the two types of algorithms while using PQC encryption. The objective was to determine whether delicate yet essential traffic data could be accurately classified due to the constant advancements in the PQC realm.

A. PQC Algorithm classification

In this experiment, we evaluate the ability of machine learning models to distinguish between different PQC algorithms (ML-KEM and Kyber) based solely on encrypted traffic patterns. Table XIV presents the results for classifying different PQC algorithms. According to the table, the Ensemble Boosting models (e.g., AdaBoost, Gradient Boosting and XGBoost) outperform other classifiers, achieving the highest accuracy and AUC.

B. Algorithm, Browser (Tuple) classification

The classification results for a combination of each browser (Chrome, Firefox) with each PQC algorithm (ML-KEM, Kyber) are shown in Table XV. This experiment aimed to identify behavioral differences in real-life scenarios. The Random Forest model achieves the highest accuracy and the best AUC.

Similar to the previous POB classification results, XGBoost delivers the best classification results for each experiment yet again.

VIII. Discussion and Conclusions

We introduce two key innovations in classifying whether the user uses PQC and user environments. First, our research leverages actual encrypted traffic that uses PQC for classification, a departure from the common practice of simulating network behaviors. This approach provides more accurate and realistic insights into the performance of PQC protocols in real-world scenarios, thereby enhancing the reliability and applicability of our classification models. Second, we have developed a model that distinguishes between PQC and No-PQC algorithms, adding a critical layer of specificity to the classification process. With the assistance of a PQC algorithm, we demonstrate that both the browser and the operating system used by the user can be accurately identified. The results provided in Sections VI and VII set the first benchmark for accurately classifying traffic encrypted with PQC under various scenarios. As can be inferred from tables X - XIII, the XGBoost and Random Forest classifiers consistently deliver strong performance, particularly when tackling complex classification tasks that involve multiple parameters. This research is significant because it has the potential to enhance the security and efficiency of Internet communications in the era of quantum computing. As quantum computers advance, traditional cryptographic methods face the threat of becoming obsolete. Our research addresses this by classifying encrypted traffic that uses PQC, which is designed to be secure against quantum attacks. Accurately classifying these algorithms is crucial for identifying vulnerabilities and ensuring robust security measures. In addition, by analyzing the effects of browsers used by the user on the identity of the PQC algorithm, critical aspects of user privacy and service optimization further contribute to the security and efficiency of Internet communications. This paper's findings are based on PQC traffic recorded in various network settings. As PQC

TABLE XIV: Algorithm Classification Results

Model	Acc	Pre	Rec	F1	AUC
ADB	0.66	0.66	0.66	0.62	0.85
DT	0.78	0.79	0.79	0.78	0.85
GB	0.78	0.78	0.78	0.78	0.88
KNN	0.74	0.75	0.74	0.74	0.86
LR	0.72	0.73	0.72	0.72	0.85
MLP	0.74	0.74	0.73	0.71	0.84
NB	0.74	0.78	0.74	0.69	0.85
RF	0.78	0.77	0.78	0.78	0.88
XGB	0.78	0.78	0.78	0.78	0.88

TABLE XV: Classify Tuple: Browser and Algorithm (Kyber, ML-KEM)

Model	Acc	Pre	Rec	F1	AUC
AdaBoost	0.55	0.53	0.55	0.46	0.81
DT	0.72	0.73	0.73	0.73	0.82
GB	0.76	0.76	0.76	0.76	0.92
KNN	0.67	0.68	0.67	0.67	0.88
LR	0.66	0.65	0.66	0.65	0.88
MLP	0.69	0.68	0.69	0.65	0.87
NB	0.63	0.62	0.63	0.57	0.83
RF	0.75	0.75	0.74	0.75	0.92
XGB	0.76	0.76	0.76	0.76	0.92

traffic becomes more prevalent, our analysis of real-world ("in the wild") traffic is crucial. In this research, we're leveraging a broader dataset in addition to classifying PQC traffic as a whole and distinguishing specific NIST algorithms within these different environments. Such granularity allows us to deepen our understanding of PQC performance in diverse environments and foster the development of comprehensive security solutions tailored for the post-quantum era. Future research could expand this work by leveraging a broader dataset drawn from widely used applications, such as Zoom [31] and other PQC-utilizing applications. In addition, the ability of the framework to identify the algorithm and OS will set the basis for our ability to investigate the possibility of PQC crypto agility. Based on our results, we aim to determine that the OS of a certain machine has limited computational capabilities and can delegate tasks to another device.

IX. Acknowledgments

This research was supported by the Ministry of Innovation, Science & Technology, Israel. The EU Horizon COCOON Project supported this work under Grant Agreement No 101120221.

References

[1] NIST, "Nist publications." https://https://csrc.nist.gov/publications/fips, 2024. 30 Oct 2024.

[2] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-kyber," *NIST, Tech. Rep*, 2017.

[3] H. Zhang, L. Yu, X. Xiao, Q. Li, F. Mercaldo, X. Luo, and Q. Liu, "Tfe-gnn: A temporal fusion encoder using graph neural networks for fine-grained encrypted traffic classification," in *ACM Web Conference*, pp. 2066–2075, 2023.

[4] N. Dillbary, R. Yozevitch, A. Dvir, R. Dubin, and C. Hajaj, "Hidden in time, revealed in frequency: Spectral features and multiresolution analysis for encrypted internet traffic classification," in *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*, pp. 266–271, IEEE, 2024.

[5] O. Bader, A. Lichy, A. Dvir, R. Dubin, and C. Hajaj, "Osf-eimtc: An open-source framework for standardized encrypted internet traffic classification," *Computer Communications*, vol. 213, pp. 271–284, 2024.

[6] T.-L. Huoh, Y. Luo, P. Li, and T. Zhang, "Flow-based encrypted network traffic classification with graph neural networks," *IEEE Transactions on Network and Service Management*, 2022.

[7] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Distiller: Encrypted traffic classification via multimodal multitask deep learning," *Journal of Network and Computer Applications*, vol. 183, p. 102985, 2021.

[8] M. Shen, J. Zhang, L. Zhu, K. Xu, and X. Du, "Accurate decentralized application identification via encrypted traffic analysis using graph neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2367–2380, 2021.

[9] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, "Post-quantum authentication in tls 1.3: A performance study," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 71, 2020.

[10] M. Ravvi, *Enabling Post-Quantum Cryptography for Secure Internet Networking: Performance, Feasibility Analyses, and Solutions*. PhD thesis, University of Colorado Colorado Springs, 2023.

[11] M. Raavi, S. Wuthier, X. Zhou, and S.-Y. Chang, "Post-quantum quic protocol in cloud networking," in *European Conference on Networks and Communications and 6G Summit*, pp. 573–578, 2023.

[12] J. Henrich, A. Heinemann, A. Wiesmaier, and N. Schmitt, "Performance impact of pqc kems on tls 1.3 under varying network characteristics," in *Information Security, vol 14411*, pp. 267–287, 2023.

[13] S. P C, K. Jain, and P. Krishnan, "Analysis of post-quantum cryptography for internet of things," in *International Conference on Intelligent Computing and Control Systems*, pp. 387–394, 2022.

[14] D. Mankowski, T. Wiggers, and V. Moonsamy, "Tls $\rightarrow$ post-quantum tls: Inspecting the tls landscape for pqc adoption on android," in *European Symposium on Security and Privacy Workshops*, pp. 526–538, 2023.

[15] T. Liu, G. Ramachandran, and R. Jurdak, "Post-quantum cryptography for internet of things: a survey on performance and optimization," *arXiv:2401.17538*, 2024.

[16] ACIC-ARIEL, "Pqbench dataset and code." GitHub, 2024. https://github.com/ArielCyber/PQBench.

[17] T. Mallick, R. Kompella, A. Kundu, and C. Nita-Rotaru, "Fingerprinting implementations of cryptographic primitives and protocols that use post-quantum algorithms," 2025.

[18] B. Rocha, J. Xexéo, and R. Torres, "Post-quantum cryptographic algorithm identification using machine learning," *Journal of Information Security and Cryptography (Enigma)*, vol. 9, pp. 1–8, 12 2022.

[19] National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard," Federal Information Processing Standards Publication FIPS 203, U.S. Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD, USA, August 2024. Published August 2024.

[20] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, “Crystals-kyber algorithm specifications and supporting documentation,” *NIST PQC Round*, vol. 2, no. 4, pp. 1–43, 2019.

[21] National Institute of Standards and Technology, “Module-Lattice-Based Digital Signature Standard,” Federal Information Processing Standards Publication FIPS 204, U.S. Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD, USA, August 2024. Published August 13, 2024.

[22] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, “Crystals-dilithium algorithm specifications and supporting documentation,” *NIST PQC Round*, vol. 3, no. 4, pp. 1–38, 2021.

[23] National Institute of Standards and Technology, “Stateless Hash-Based Digital Signature Standard,” Federal Information Processing Standards Publication FIPS 205, U.S. Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD, USA, August 2024. Published August 2024.

[24] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, “The sphincs+ signature framework,” in *ACM SIGSAC*, pp. 2129–2146, 2019.

[25] R. Rios, J. A. Montenegro, A. Muñoz, and D. Ferraris, “Towards the quantum-safe web: Benchmarking post-quantum tls,” *IEEE Network*, pp. 1–1, 2025.

[26] A. Corsi, S. Gür, A. Brighente, and M. Conti, “Evaluation of post-quantum key encapsulation methods in 5g core network,” in *2025 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2025.

[27] Selenium, “Selenium automates browsers.” http://www.seleniumhq.org/. Accessed: 2025-08-10.

[28] European Parliament and the Council of the European Union, “Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation),” 4 2016. Entered into force on May 24, 2016; applicable from May 25, 2018.

[29] K. Jerabek, J. Luxemburk, R. Plny, J. Koumar, J. Pesek, and K. Hynek, “When simple model just works: Is network traffic classification in crisis?,” 2025.

[30] J. Pesek, J. Luxemburk, and K. Hynek, “Lightweight traffic classification: A simple baseline matching deep learning performance,” in *2025 9th Network Traffic Measurement and Analysis Conference (TMA)*, pp. 1–4, 2025.

[31] Zoom Video Communications, “Zoom bolsters security offering with the inclusion of post-quantum end-to-end encryption in zoom workplace.” https://news.zoom.com/post-quantum-e2ee/, 2024. Accessed: 2025-08-12.