Performance Implications of Post-quantum Transport Layer Security in the 5G Core

Sai Anirudh Madhavapeddi
Nokia Technology Standards
Munich, Germany
sai.madhavapeddi@nokia.com

Aritra Banerjee
Nokia Technology Standards
Cambridge, United Kingdom
aritra.banerjee@nokia.com

Wolfgang Kellerer
Technical University of Munich
Munich, Germany
wolfgang.kellerer@tum.de

***Abstract*—Post-quantum cryptography (PQC) will become an essential component of securing 5G and future 6G systems, yet the performance implications of deploying PQC within the 5G Core (5GC) remain poorly understood. In this work, we present a complete 5G standalone testbed featuring post-quantum TLS for all inter–network-function (NF) interfaces, enabling detailed measurement of latency, CPU utilization, and handshake behavior under realistic core network procedures. We characterize the TLS connection patterns inherent to the 5GC for a range of deployment configurations, including communication via the Service Communication Proxy (SCP) and direct Network Repository Function (NRF)-based discovery. We analyze how startup strategies influence handshake patterns and core availability, for different levels of TLS handshake overhead.**

Our results show that PQC integration in the 5GC is feasible with minimal impact on system performance. The primary NIST algorithms, ML-KEM and ML-DSA, closely match elliptic curve cryptography (ECC) baselines in both latency and computational cost, while alternatives such as HQC and SLH-DSA introduce manageable overheads. We further observe that cryptographic computation time dominates TLS-induced latency, rather than transmission overheads and MTU-induced fragmentation. Finally, we derive deployment-oriented guidelines to support PQC-enabled 5GC operations with minimal disruption. These findings indicate that quantum-safe migration of 5G core security functions can be achieved efficiently on existing infrastructure.

***Index Terms*—5G; Core Network; Transport Layer Security; Post-Quantum Cryptography; 3GPP; NIST**

I. Introduction

Public-key infrastructure (PKI) underpins the security behind most modern communication networks, including the 5G system. Shor's algorithm efficiently solves the mathematical problems behind Elliptic-curve cryptography (ECC) and RSA, the most widely used asymmetric cryptosystems in PKI [1]. Experts believe that a quantum computer powerful enough to break these cryptosystems - a cryptographically relevant quantum computer (CRQC) - could arrive in the upcoming decades [2]. Moreover, the threat of the store-now-decrypt-later attack already exists today.

To be prepared for this quantum-threat, the National Institute of Standards and Technology (NIST) has selected 5 new post-quantum cryptography (PQC) algorithms to replace existing asymmetric cryptography in PKI [3]. Projects such as the Open Quantum Safe (OQS) project [4] have already published open-source implementations for these algorithms as well. Thus the time is ripe to start migrating to quantum-secure alternatives in cybersecurity at large.

However, there are still practical concerns with respect to the migration to PQC in operator deployments. These include the larger object sizes, i.e. key and ciphertext sizes, for PQC algorithms which are often in the kilobyte range, higher computational requirements, and interoperability with existing hardware and software stacks.

Within the 5G system, several functions and interfaces rely on asymmetric cryptography, such as primary authentication of user equipment (UE), network domain links secured by IPSec, and— the focus of this paper — the 5G Core (5GC), whose interfaces use the Transport Layer Security (TLS) protocol [5]. With the advent of cloud-native and distributed 5G cores [6], the security and performance of TLS interfaces become increasingly critical. Integration of PQC into the primary authentication procedure has been studied, such as in [7] and [8]. There are also studies on PQC integration into security protocols including TLS, where metrics such as handshake runtime, latency [9], [10] and memory requirements [11] are evaluated.

This paper presents a system-level analysis and measurement study of PQC integration in the 5GC, focusing on TLS handshake overheads and operational configurations. Using a dockerized Open5GS [12] and UERANSIM [13] testbed integrated with an OpenSSL build supporting PQC, we measure the impact of various cryptographic algorithms, startup methods, and network function topologies (involving the Service Communication Proxy - SCP) on network procedures. Our contributions are:

- A characterization of TLS handshake patterns in the 5GC, for different core network configuration and startup scenarios
- A comparison of with-SCP vs. without-SCP inter-NF communication and the resulting number and timing of TLS handshakes.
- A study of startup strategies and their effects on handshake consistency, core network startup, and UE registrations
- A performance evaluation of NIST-standardized PQC algorithms based on end-to-end latency and CPU utilization measurements in the aforementioned procedures

979-8-3195-0167-7/26/$31.00 ©2026 IEEE

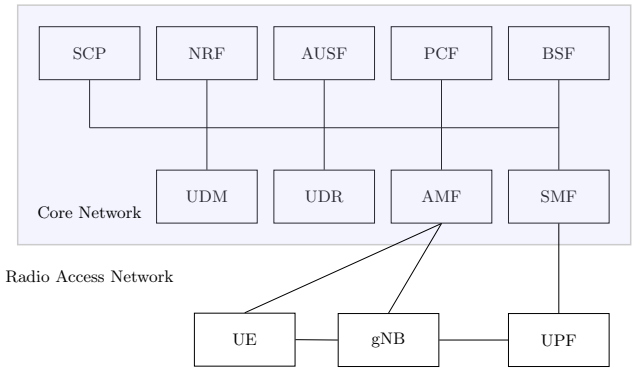


Fig. 1: 5G standalone system with minimal core network

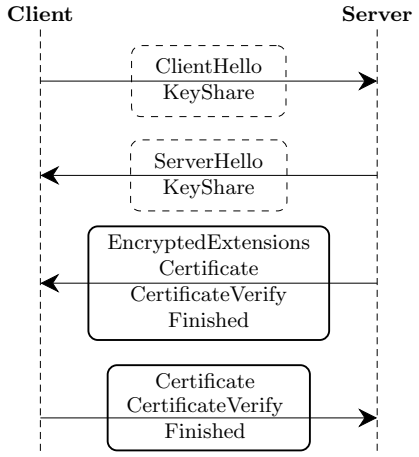


Fig. 2: mTLS 1.3 protocol flow diagram

II. Background

A. 5G Service-based Architecture

The 5GC follows a service-based architecture (SBA) where NFs communicate through HTTP/2-based APIs as defined in 3GPP TS 23.501 [14]. In this work, we focus on the NFs required for UE registration and PDU session setup in a representative standalone Open5GS deployment. Fig. 1 shows the evaluated standalone topology.

The evaluated deployment includes NRF, SCP, AMF, SMF, AUSF, UDM, UDR, PCF, and BSF. The associated signaling paths for registration/session procedures determine the TLS connections analyzed in the following sections.

Throughout network operation, the NRF records the profiles and statuses of other NFs, while the optional SCP serves as a proxy that reduces the number of direct inter-NF connections. We focus on two NF service-communication models defined in Annex E of [14]: Model B and Model D. In Model B (direct communication), an NF service consumer queries the NRF, selects a producer, and invokes the service directly. In Model D (delegated discovery), the consumer hands discovery and selection to the SCP, which performs NRF-based lookup and forwards the request. These two models capture the architectural alternatives in our testbed: one without SCP mediation and one with the SCP interposed between NFs.

B. Transport Layer Security in 5G Core

According to the 3GPP standard for 5G Security architecture, TS 33.501 [15], the use of TLS (specifically mutual-TLS or mTLS, where both the server and client authenticate each other) for securing service-based interfaces (SBIs) in the 5GC must be supported, but is not mandatory to use. The 5G Core adopts a cloud-native, service-oriented model via RESTful APIs over IP networks, increasingly deployed in virtualized and containerized environments [6].

It has been shown in [16] that enabling TLS in the 5GC adds negligible performance overhead in long-term operation—less than 1% increase in both UE registration time and CPU utilization—although registration time is roughly 30% higher in a freshly started network, which we explore further. Both TLS 1.3 [17] and 1.2 [18] are prescribed in the standard; we evaluate TLS 1.3 only, as it completes the handshake in a single round trip and is the preferred version for new deployments.

A diagram of the TLS 1.3 mutual-authentication handshake is shown in Fig. 2. The handshake contains two key steps involving asymmetric cryptography. In the KeyShare step, the client and server exchange public keys using traditional Diffie–Hellman (DH), which are used to derive a shared secret. In post-quantum variants, Key Encapsulation Mechanisms (KEMs) replace DH: the client provides a public key, the server encapsulates a secret, and returns the resulting ciphertext.

In the certificate exchange step, each entity presents a certificate signed by a certificate authority (CA), and proves possession of the corresponding private key through a `CertificateVerify` message signed using a Digital Signature Algorithm (DSA). The derived shared secret is then used to establish symmetric session keys. Asymmetric operations are confined to this initial handshake, which completes in a single round-trip in TLS 1.3; subsequent communication proceeds using symmetric encryption. TLS further supports session-resumption tickets to avoid repeating full handshakes after disconnection.

C. NIST-standardized PQC Algorithms

We evaluate standardized PQC families relevant to TLS migration in 5GC: ML-KEM [19] and HQC [20] as key encapsulation mechanisms (KEMs); ML-DSA [21], FN-DSA [22], and SLH-DSA [23] as digital signature algorithms (DSAs); and CROSS [24] as an additional fallback signature candidate. For the considered configurations, cryptographic object sizes span from hundreds of bytes to tens of kilobytes, substantially larger than ECC counterparts.

- **ML-KEM** (Module Lattice-based KEM) [19]: primary NIST-recommended KEM, based on the Module Learning With Errors (MLWE) problem.

- **HQC** (Hamming Quasi-Cyclic) [20]: code-based KEM fallback with larger keys and ciphertexts than ML-KEM.
- **ML-DSA** (Module Lattice-based DSA) [21]: primary NIST-recommended signature scheme, also based on MLWE.
- **FN-DSA** (Falcon) [22]: lattice-based DSA with the smallest standardized signatures, but requiring floating-point arithmetic.
- **SLH-DSA** (Stateless Hash-based DSA) [23]: conservative hash-based fallback; fast and slow variants trade signature size against signing cost.

Among the standardized schemes, ML-KEM and ML-DSA are the primary NIST recommendations and form our baseline PQC path. We additionally include CROSS-RSDP-128 (balanced) as a non-standard but practically relevant code-based DSA candidate, analogous to HQC among KEMs. These choices span the latency and object-size range reflected in our measurements.

Each family is standardized at multiple NIST security levels, with computational cost and on-the-wire object sizes increasing accordingly. We evaluate level-1 parameter sets for all schemes except ML-DSA-44 (level 2, the lowest standardized ML-DSA level), matching the ECC baseline strengths used in Section VI.

III. Related work

The work [16] evaluates the performance impact of enabling TLS in the 5G core, using metrics similar to ours—UE registration time and CPU utilization percent—across different TLS versions, AEAD ciphers, and certificate types (RSA and ECDSA). Our study extends this line of investigation in three key ways. First, we incorporate post-quantum security, evaluating both the cryptographic latency and system-level effects of PQC-enabled TLS within the 5GC. Second, we characterize the structure and timing of TLS handshakes in the core network under different topologies and startup behaviors, an aspect not addressed in prior TLS–5GC studies. Third, we introduce core startup time and cumulative CPU utilization as metrics that more completely capture the cost of TLS handshakes, whereas UE registration time alone hides much of the handshake overhead behind RAN procedures.

There are several works introducing post-quantum security in the 5G core to various degrees. The works [25] and [26] also integrate PQC into open-source 5G core simulations (Open5GS and free5GC, respectively). The evaluation is restricted to point-to-point metrics between NFs, such as handshake latency, bandwidth usage, and retransmission rate. In [27], the authors evaluate UE registration time with post-quantum KEM-based authentication, but this step is added outside the existing TLS protocol in Open5GS (which uses traditional quantum-vulnerable cryptography). Another work [28] implements VPN-over-TLS tunnels between the NFs, focusing on tunnel latency and packet size metrics. They evaluate UE registration with the AMF and NF registration with the NRF across a few post-quantum DSAs only, without post-quantum key exchange.

Our setup introduces the full range of NIST-standardized PQC into the existing TLS functionality in Open5GS, with realistic function calls between NFs. We also consider broad, network-level metrics that not only capture the full TLS handshake overhead but are also directly relevant to both the UE and the network operator.

IV. Methodology - Capturing TLS overhead

Asymmetric cryptographic overhead in TLS is concentrated in the handshake phase. In our system measurements, handshake activity is concentrated in three scenarios:

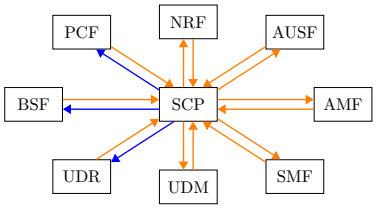
- **Core network startup:** When the core network is first initialized, each NF establishes a TLS connection with a central NF—either the SCP if deployed, or the NRF otherwise. This phase accounts for the majority of handshake activity during system initialization.
- **First UE registration:** When a UE tries to register with the network, NFs exchange additional control messages for authentication and session establishment. When this happens for the first time after the startup of the core network, it triggers further TLS handshakes—either routed through the SCP or established directly between the relevant NFs—depending on the deployment configuration.
- **NF restart or scaling events:** When an NF instance is restarted or a new instance is introduced through autoscaling, a subset of the handshakes from the previous two phases occurs (not evaluated here). The number of handshakes depends on the NF's role and its connectivity with other NFs.

After these handshakes, inter-NF traffic proceeds over persistent TLS channels; therefore, long-term overhead is small and lifecycle events dominate the impact. We first determine minimal/consistent handshake counts by topology, then analyze startup policies and PQC choices using startup time, registration time, and cumulative CPU utilization.

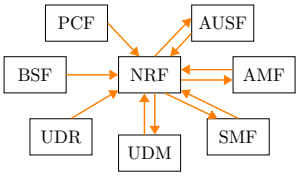
A. Configuration Models

We have already discussed that the SCP acts as a proxy for inter-NF communication, simplifying the core network topology and reducing the total number of TLS connections required. The effect becomes more pronounced at larger scales, with multiple instances of each NF within the same core network. Nevertheless, the benefit is still evident in our simplified setup containing a single instance of each NF (nine including the SCP). We enumerate below the TLS handshakes required for the complete operation of the core network, including both the startup phase and the first UE registration. They are displayed in Fig. 3.

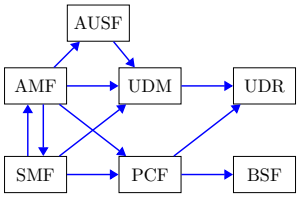
- **With SCP (Model D):** During startup, every other NF (as client) connects to the SCP (as server). The SCP itself, as client, connects to NRF, AUSF, AMF, SMF, and UDM as servers. Upon receiving the first registration request, the SCP additionally connects as client to UDR, BSF, and PCF as servers. In total, there are 16 handshakes—13 during startup and 3 after the first UE registration.



(a) With SCP



(b) Without SCP - Startup



(c) Without SCP - First registration

Fig. 3: TLS handshakes (direction: client → server) during startup (orange) and first registration (blue)

- **Without SCP (Model B):** During startup, every other NF (as client) connects to the NRF (as server). The NRF, as client, also connects to AUSF, AMF, SMF, and UDM as servers. When the first UE registration is triggered, additional handshakes occur directly between the NFs responsible for authentication and session establishment for the UE. In total, there are 22 handshakes—11 during startup and 11 after the first UE registration.

Even in this simple core network topology, the presence of the SCP reduces the overall number of handshakes required within the core. The number of handshakes triggered after the first UE registration decreases significantly, with the trade-off of slightly more handshake activity during startup. This benefit is expected to scale more sharply in larger and more distributed deployments.

B. Startup Methods

Enabling TLS introduces ordering constraints during core initialization, since each NF must be ready to accept TLS connections when other NFs attempt to establish them. If connection attempts occur before an endpoint is fully initialized, retries are triggered and startup time increases. As 3GPP does not prescribe a startup sequence, we evaluate different initialization strategies while keeping a consistent NF launch order, starting with the central functions (NRF and SCP), followed by the remaining NFs according to their logical dependencies [29].

We consider four startup methods, summarized in Table I, which differ only in the waiting time between NF launches. Sequential startup waits for each NF to fully initialize before launching the next. Rolling startup introduces a shorter, fixed delay T_{roll} between launches. Staggered startup starts the central NF(s) first, followed by all remaining NFs in parallel after a delay T_{stag}. Concurrent startup launches all NFs simultaneously; in practice, this frequently leads to failed handshakes and is therefore not evaluated further.

TABLE I: Description of startup methods based on waiting time between NF initializations.

Startup method	**Waiting time between NF initializations**
Sequential	NF initialization time
Rolling	T_{roll} < NF initialization time
Staggered	T_{stag} after NRF, SCP 0 for all other NFs
Concurrent	0

V. Experimental Setup

A. Testbed Overview

The testbed runs on a workstation equipped with a 12-core 3.50 GHz processor and 64 GB of RAM, running Ubuntu 22.04. It uses Docker's native OS-level virtualization, with each container based on the same operating system. The 5G core network is emulated using Open5GS, where each NF runs in its own container, while the gNB and UE are simulated using UERANSIM in separate containers. The NFs included in this setup are NRF, SCP, AUSF, UDM, UDR, SMF, AMF, PCF, and BSF. All core NF containers are launched from a common Docker image using a Docker Compose file. The core containers are connected to one Docker network, and the AMF, gNB, and UE containers to another. Both networks use bridged networking, allowing packet capture and inter-container routing while keeping the core and RAN segments logically isolated.

Each NF is configured to use TLS for inter-NF communication, with customizable certificates injected at container launch. Open5GS performs TLS operations using APIs from the `libcurl` library, which by default relies on the system's OpenSSL installation. However, the OpenSSL version distributed through Ubuntu 22.04's `apt` repositories does not support post-quantum cryptography (PQC). To enable PQC, the common "Open5GS" image first compiles a custom OpenSSL (version 3.5.4) from source in a dedicated directory. Next, `libcurl` is rebuilt against this custom OpenSSL, followed by Open5GS itself, which is then linked to both of these custom builds.

In addition to OpenSSL, the LibOQS library and OQSProvider [4] are built into the same image to provide a broader set of PQC algorithms accessible through OpenSSL's provider interface. For testing, the Open5GS source code is lightly modified in `/lib/sbi/nghttp2-server.c` and `/path/sbi/client.c` to pin a specific key encapsulation mechanism (KEM) during the TLS handshake. In standard TLS negotiation, multiple KEMs are advertised and one is selected; this modification ensures a fixed algorithm for repeatable measurements. The entire build and image-generation process is automated using Dockerfiles to ensure reproducibility. The gNB and UE containers are launched from a separate Docker image built with UERANSIM. To support

reproducibility, the experiment artifacts are publicly available at https://github.com/saianirudhm/pq_5gc.

B. Measurement Collection

- **PCAP-based timing:** We define the core network startup time as the interval from the launch of the first NF until the final NF becomes ready, i.e., until the last TLS handshake required for core initialization completes (assuming supporting services such as databases are already running). In practice, we measure this using packet capture (PCAP) as the time between the start of the first NF and the final `ServerHello` message observed after startup. Subsequent TLS 1.3 messages are encrypted and therefore not visible in PCAP; readiness may lag slightly, but this offset is largely algorithm-independent. PCAP traces are also used to identify and count TLS handshakes in the evaluated scenarios.
- **UE registration timing:** This metric represents the time elapsed from the first registration request sent by the UE until PDU session establishment is completed. A timer is initiated when the registration request is sent (`/src/ue/nas/mm/mm.hpp`) and stopped when the PDU session is established (`/src/ue/nas/sm/establishment.cpp`). To minimize log clutter, the logging level in `/src/utils/logger.cpp` is restricted to error messages, which now include the timing output. For simplicity, we call this registration time, though it is actually registration and PDU session establishment time.
- **CPU utilization:** CPU usage is sampled every 0.05 s using `/proc/$pid/cgroup` and `cpu.stat` across all Docker containers. The increment in CPU time between two consecutive samples is divided by the real elapsed time to obtain a CPU utilization percentage per container. Because multiple cores are available, this percentage can exceed 100%.

VI. Performance Measurements

The primary focus of this study is a comparative performance evaluation of NIST-standardized PQC algorithms used in TLS within the 5G core network. We use P-256 ECDH for key exchange and ED25519 for authentication as baselines, according to widely used recommendations [30]. We evaluate ML-KEM-512 and HQC-128 as KEMs, and ML-DSA-44, FN-DSA-512, SLH-DSA-128 (both fast and slow variants) and CROSS-RSDP-128 (balanced) as DSAs—all corresponding to NIST security level 1, except ML-DSA-44, which represents level 2, the lowest standardized level.

A. Handshake Latency

Before evaluating network-level metrics, we first analyze the point-to-point latency of a single TLS handshake (say, T_{HS}) for the full range of PQC algorithms considered. Although similar benchmarks have been reported in prior work [25] [26], we present this evaluation specific to our setup. The values are close to those reported in [25] that also uses Open5GS, and the trends across PQC algorithms are similar to both works. Later, this helps us guide the selection of parameters T_{roll} and T_{stag} in the rolling and staggered startup scenarios. Measurements were obtained over 1000 point-to-point `OpenSSL` connections executed on the host system.

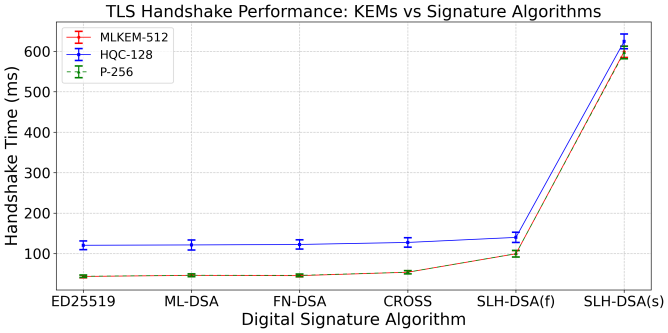


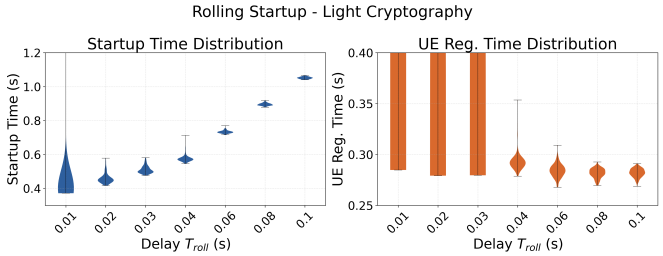
Fig. 4: Single mTLS handshake latency microbenchmarks (mean with standard deviation error bars)

The results of the benchmarking study are shown in Fig. 4. T_{HS} ranges from 40ms for the lightest combination, to 600ms for the heaviest combination. Notably, ML-KEM performs nearly identically to the baseline P-256, and ML-DSA and FN-DSA show similar latency to ED25519, highlighting the computational efficiency of lattice-based PQC algorithms despite their larger object sizes. HQC and CROSS, the code-based algorithms, exhibit slightly higher latency than their lattice-based counterparts. Of particular interest is the contrast between the SLH-DSA fast and slow variants: the slow variant incurs substantially higher handshake latency, even though its signature size is smaller. This implies that cryptographic computation time dominates the overall handshake delay, while packet transmission and fragmentation overheads remain negligible.

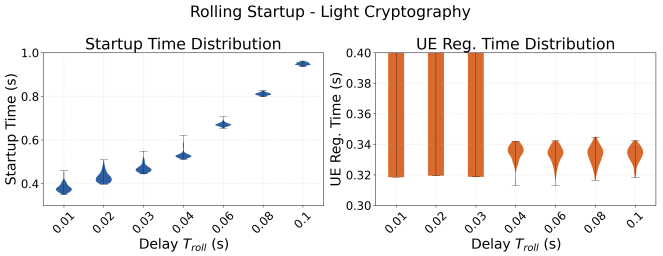
B. Startup and Configuration

In this subsection, we evaluate core startup and first UE registration time under different architectural and cryptographic configurations. We consider deployments with and without the SCP, and compare "light" (P-256 ECDH with ED25519) and "heavy" (HQC-128 with SLH-DSA-128 slow) cryptography, representing the lower and upper bounds of TLS handshake cost. Although the light configuration reflects typical deployment choices, the heavy case captures worst-case overhead and fallback scenarios. All results report the mean and standard deviation over 100 independent runs; the remaining variance is attributed to system-level nondeterminism (e.g., process scheduling and container overhead) rather than insufficient sampling.

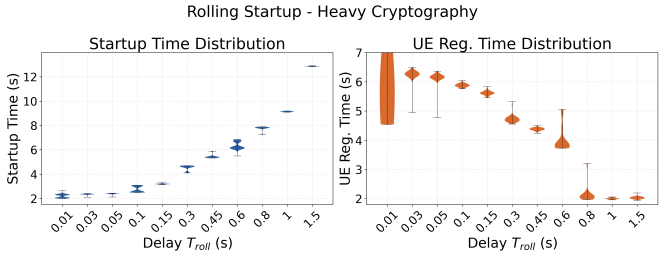
1) Sequential startup: We begin with the most deterministic scenario of sequential start. As we have already noted, 13+3 handshakes are involved in the case with SCP, and 11+11 handshakes in the case without SCP. There is no variance observed in the number of handshakes across all runs, as this startup method is fully deterministic. Table II shows the



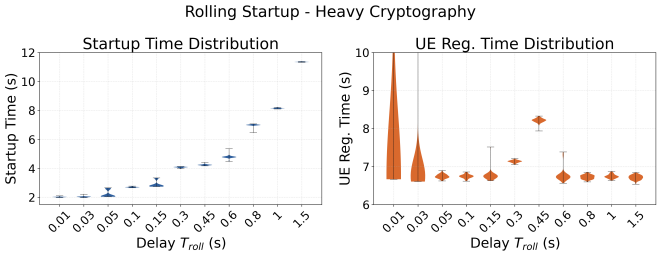
(a) With SCP, "light" cryptography



(b) Without SCP, "light" cryptography



(c) With SCP, "heavy" cryptography



(d) Without SCP, "heavy" cryptography

Fig. 5: Comparison of startup and first registration time distributions for different rolling startup delays (T_{roll})

TABLE II: Sequential startup results for "light" (top row) and "heavy" (bottom row) cryptography.

Core startup time (s)		First registration time (s)	
With SCP	Without SCP	With SCP	Without SCP
10.54 ± 0.04	9.24 ± 0.05	0.28 ± 0.01	0.33 ± 0.01
14.84 ± 0.08	13.37 ± 0.05	2.02 ± 0.03	6.72 ± 0.05

average and standard deviations for core startup and first registration time, across the scenarios a) with and without SCP, and b) with "light" and "heavy" cryptography as we have defined. The startup time with SCP is longer than without SCP, and vice versa for the first registration time. This is in accordance with the number of handshakes occurring in each case.

2) Rolling startup: We now evaluate the rolling startup scenario for a range of delay values T_{roll} between successive NF launches. The distributions are shown as violin plots in Fig. 5 to illustrate variability.

We first consider subfigure 5a (with SCP, "light" cryptography). For very small T_{roll}, startup appears fast on average, but many handshakes fail because endpoints are not yet ready. Consequently, the UE often fails its first registration attempt and succeeds only after the default retry timer of approximately 10 s, leading to the widely spread registration-time distribution for $T_{roll} < 0.04$s. For intermediate values of T_{roll} (0.04s and 0.06s), comparable to the handshake latency T_{HS}, some handshakes still fail during startup but are completed during registration, resulting in slightly prolonged yet successful registrations. When T_{roll} exceeds T_{HS} (0.08s and 0.1s), startup and registration become fully consistent—matching sequential startup in reliability but with significantly reduced startup time. Consistency improves with T_{roll} and stabilizes shortly beyond T_{HS}. A similar trend appears without SCP (subfigure 5b), and all considered rolling cases remain faster than sequential startup (10.54 s with SCP, 9.24 s without).

Under "heavy" cryptography, even smaller values of T_{roll} (e.g., 0.03s) allow successful first registration, albeit with longer registration times. The increased handshake duration appears to introduce an inherent buffering effect during startup. As before, startup time grows linearly with T_{roll}, and once T_{roll} exceeds roughly T_{HS}, both startup and registration stabilize, with no further reduction in registration time. In the absence of the SCP (subfigure 5d), a minor increase in registration time is observed around $T_{roll} = 0.45$s, slightly below $T_{HS} \approx 0.6$s. Although PCAP traces did not reveal a definitive cause, a plausible explanation is that delayed startup handshakes overlap with registration-triggered handshakes, creating transient contention and a small increase in registration latency.

3) Staggered start: We also evaluated the staggered startup strategy for the same configurations, where the NRF is started first, followed by the SCP (if present) and then all remaining NFs concurrently after a short delay. For "light" cryptography, this approach yielded reliable operation and reduced overall core startup time compared to rolling startup, at the cost of slightly increased first UE registration time.

Under "heavy" cryptography, however, staggered startup proved highly inconsistent. In the absence of the SCP, 20–30% of first registration attempts failed and were only completed after retry timeouts, even for larger delay values. This behavior indicates handshake contention at the central NF during concurrent initialization, where multiple expensive TLS handshakes overload the NRF. The SCP topology mitigates this effect, but overall staggered startup becomes sensitive to handshake cost and central function load under heavier

TABLE III: PQC relative performance, averaged per combination across all metrics

Algorithm	Startup Time (%)	UE Registration (%)
KEM Comparison (as % of ECDH)		
ML-KEM-512	99.9%	99.9%
HQC-128	116.0%	139.7%
DSA Comparison (as % of ECDSA)		
ML-DSA	100.7%	103.5%
FN-DSA	100.3%	102.3%
CROSS	102.9%	113.3%
SLH-DSA(f)	113.7%	167.9%
SLH-DSA(s)	320.7%	1179.1%

cryptographic configurations.

C. Startup Time and UE Registration

Following the preliminary analysis of startup methods and network configurations with and without SCP, we now extend the evaluation to the full set of PQC algorithms considered. We use the rolling start method with $T_{roll} = 0.05s$, as this ensures reliable startup while also making the variation across PQC algorithms more pronounced. Results are presented for both deployment scenarios—with SCP and without SCP. We include a baseline corresponding to "no-TLS" for both startup and UE registration time. This baseline, being independent of the cryptography used, is shown as a horizontal reference line. All reported values are averaged over 100 independent runs, and the mean startup and first registration times are summarized in Fig. 6.

The observed trends for both startup and registration times closely mirror the point-to-point TLS handshake benchmarks discussed earlier. The primary NIST-recommended algorithms, ML-KEM and ML-DSA, exhibit performance closely comparable to their elliptic-curve counterparts. As a quantitative example in the SCP-present case, the elliptic curve combination of ECDSA + P-256 ECDH has a 55.6% higher startup time and 15.5% higher first registration than the no-TLS baseline. On the other hand, the primary recommended PQC combination of ML-DSA + ML-KEM has a 55.8% higher startup time and 21.4% higher first registration time. Relative to the ECC combination, startup and first registration times increase by 0.1% and 5.1%, respectively. The overhead for each algorithm with respect to its elliptic-curve counterpart, calculated per combination and averaged across both metrics and configurations, is shown in Table III.

Overall, these results reinforce the earlier observation that cryptographic computation, not transmission or protocol-level overhead, dominates the total latency. The equivalence of ML-KEM and ML-DSA to their ECC baselines underscores their practicality for 5G core deployments, while the heavier code- and hash-based algorithms remain viable but less efficient alternatives.

1) Concurrent UE registration: In practice, a freshly started or restarted core may receive multiple UE registration requests concurrently. Although the total number of TLS handshakes remains unchanged, parallel requests introduce concurrent handshake threads, increasing instantaneous computational load and potentially extending per-UE registration time compared to the single-UE case. We therefore evaluate the average per-UE registration time as a function of the number of simultaneous UEs. In addition to the "light" and "heavy" configurations, we include an intermediate "medium" combination (HQC-128 with CROSS) with a handshake duration of approximately 120 ms. Results for the with-SCP topology are shown in Fig. 7; without SCP, concurrent registrations frequently failed (likely a testbed artifact in our Open5GS setup).

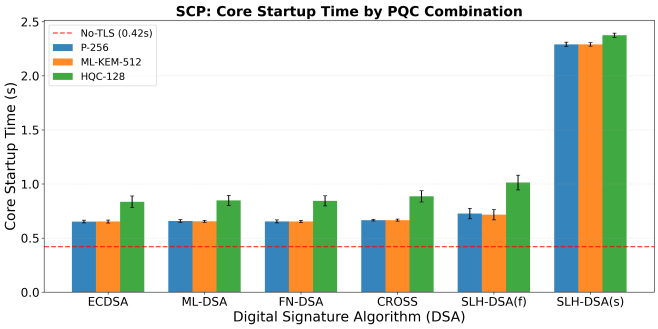
For the "light" and "medium" configurations, per-UE registration time increases modestly with the number of concurrent UEs. It even slightly drops in going from 1 to 2 UEs, possibly due to the second UE triggering idle threads. In contrast, the "heavy" configuration exhibits a pronounced growth in both mean and variance of per-UE registration time. This indicates that while lighter PQC schemes remain robust under moderate concurrency, heavier combinations amplify computational contention during simultaneous handshake processing.

D. CPU Utilization Analysis

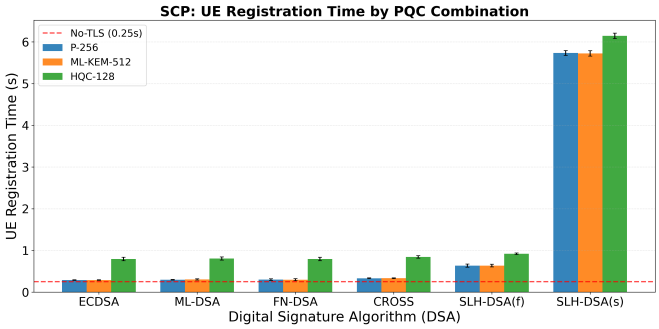
We now evaluate the variation in CPU utilization across the different PQC configurations for the same network procedures. CPU usage for each NF container is measured using Linux `cgroup v2` statistics. Specifically, we sample the `cpu.stat` file at fixed intervals of 50 ms to obtain the cumulative CPU time consumed in microseconds (`usage_usec`). The instantaneous CPU utilization percentage is computed as the difference in `usage_usec` between consecutive samples, divided by the elapsed wall-clock time, and multiplied by 100. This metric naturally accounts for multi-core usage, such that values exceeding 100% indicate parallel utilization of multiple CPU cores.

For each experimental run, we collect a continuous time-series of CPU usage covering the entire procedure, from core startup through UE registration and PDU session establishment. To enable direct comparison across PQC algorithms and configurations, we consider the total integrated CPU-seconds consumed per run. This value is obtained by summing the final cumulative `usage_usec` across all NF containers and normalizing to seconds. The resulting metric represents the total computational load incurred by the system, independent of variations in wall-clock execution time, and therefore provides a more stable measure of cryptographic and protocol processing cost.

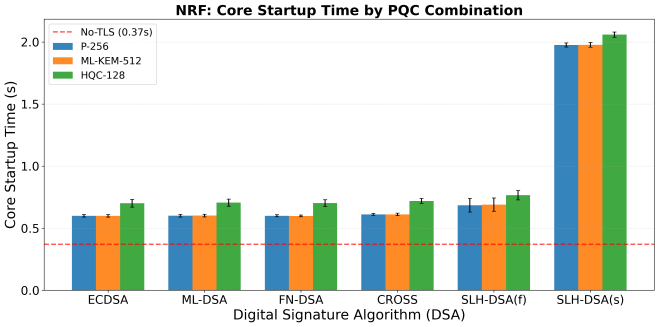
1) Instantaneous CPU utilization percent: To illustrate the origin of CPU utilization differences across PQC configurations, we examine instantaneous CPU traces for a representative NF (AMF) under "light" and "heavy" cryptography. The SCP is disabled to maximize handshake activity, and a rolling startup with $T_{roll} = 2\,\text{s}$ is used to separate startup events clearly. The resulting traces are shown in Fig. 8. The AMF process starts at approximately 12 s, visible in both cases as an initial peak of about 120%.



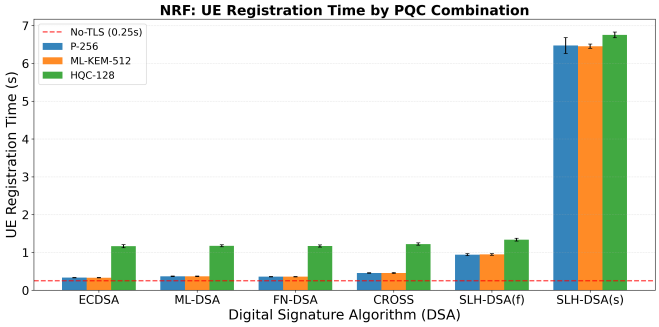
(a) Startup time (with SCP)



(b) Registration time (with SCP)



(c) Startup time (without SCP)



(d) Registration time (without SCP)

Fig. 6: Comparison of startup and registration performance across all PQC combinations with and without SCP

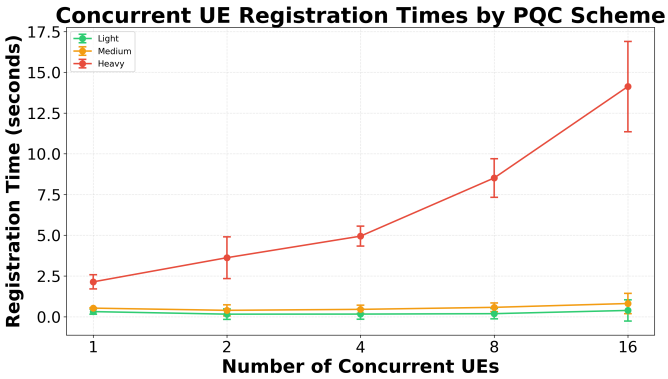


Fig. 7: Concurrent UE registration time vs number of UEs

The subsequent peaks correspond to TLS handshake operations. Under "light" cryptography, these peaks remain moderate, whereas in the "heavy" configuration they are substantially higher, reaching around 120% during each handshake. The mean utilization level (horizontal dashed line) is also elevated in the heavy case, reflecting increased cumulative CPU usage. Overall, the figure demonstrates that the primary difference between light and heavy cryptography lies in the magnitude and frequency of handshake-induced CPU spikes, which directly translate into the cumulative CPU utilization reported later.

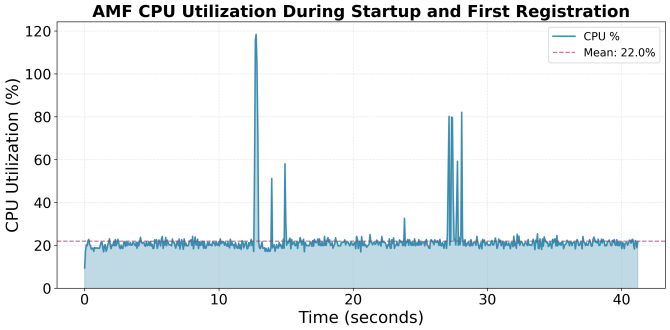
2) Cumulative CPU utilization time: We now analyze cumulative CPU utilization time, aggregated across all NF containers, for the complete set of PQC algorithms. To isolate cryptographic effects, we use the sequential startup method, which ensures a deterministic number of handshakes and avoids startup-induced retries. For a fixed cryptographic configuration, peak instantaneous CPU utilization across other startup methods remained similar; differences in cumulative utilization were therefore primarily due to variations in handshake count. Focusing on sequential startup thus enables direct comparison of cryptographic cost, particularly between light and heavy PQC schemes.

Measurements are averaged over 100 independent runs covering both startup and registration phases. CPU usage is aggregated across all NFs and reported as stacked bars separating startup and registration contributions (Fig. 9). The trends closely mirror those observed in timing results, confirming that TLS-induced overhead in the 5G Core is dominated by cryptographic computation rather than transmission effects. For all NIST security level 1 algorithms tested, maximum instantaneous CPU utilization remained below 150%, well within the capacity of the 12-core test platform.

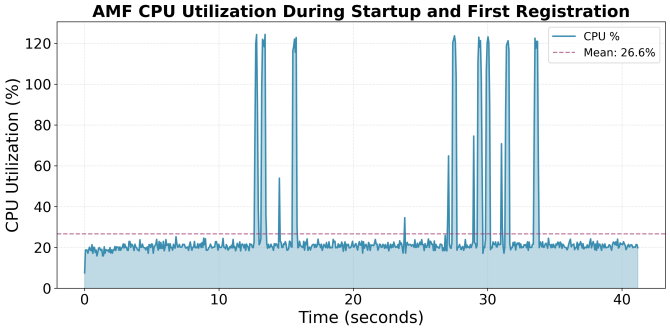
VII. Discussion

A. Operational Impact

The observed TLS overhead in the 5G Core is concentrated in lifecycle events such as core startup, first UE registration, and NF restart or scaling. In these phases, SCP-assisted deployments reduce direct inter-NF TLS connections and improve robustness under concurrent registrations, with the trade-off of slightly longer startup and an additional central

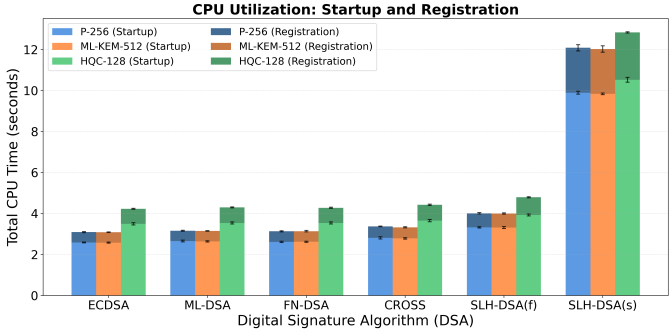


(a) "Light" cryptography

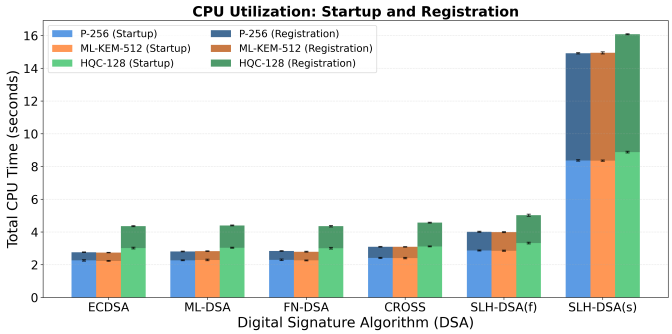


(b) "Heavy" cryptography

Fig. 8: Instantaneous CPU utilization (%) traces for AMF container during startup and first registration.



(a) With SCP



(b) Without SCP

Fig. 9: Cumulative CPU utilization time during startup and registration for different PQC combinations

dependency. The results therefore point to a configuration choice: prioritize simpler handshake topology and registration stability with SCP, or reduce centralization by relying on direct NRF-mediated discovery.

Startup policy also has a direct effect on system behavior. Rolling and staggered methods can reduce total initialization time compared to strictly sequential startup, but they may defer or duplicate handshakes and increase first-registration delay when endpoints are not yet ready. This effect becomes more visible under high handshake cost and concurrent activity, where contention at central functions can cause instability. In practical terms, handshake duration and startup sequencing should be treated as orchestration parameters, especially for rapid recovery and scaling scenarios.

From a cryptographic standpoint, the main NIST path (ML-KEM with ML-DSA) remains close to ECC baselines in end-to-end impact, supporting deployment on existing infrastructure. Heavier alternatives such as HQC and SLH-DSA increase transient CPU and latency costs, but these costs remain bounded to handshake-heavy phases and do not materially change steady-state control-plane throughput once TLS channels are established.

B. Limitations and Future Work

This study uses a controlled single-host Docker testbed that isolates cryptographic cost well, but does not fully capture distributed-cloud behavior such as cross-host RTT, shared-network contention, and heterogeneous resource scheduling. While the qualitative trends are expected to hold, absolute latency and contention profiles may differ in multi-host deployments.

Future work should validate these results on constrained edge hardware and in geographically distributed setups, where compute limits and transport effects can shift trade-offs across PQC choices. It should also extend the same measurement framework beyond 5GC TLS to other asymmetric-crypto control-plane mechanisms, including IKE/IPSec and roaming-related interfaces, to complete the system-level view of post-quantum migration.

VIII. Conclusion

We presented a system-level evaluation of post-quantum TLS integration in the 5G Core, implementing standardized PQC algorithms within an Open5GS testbed and measuring their impact on startup behavior, UE registration, and CPU utilization. Our results show that, despite substantially larger cryptographic object sizes, transmission effects are negligible in this deployment context and overall latency is dominated by cryptographic computation. Performance impact is largely confined to transient lifecycle phases such as startup and first registration, and can be controlled through architectural and orchestration choices, including SCP-mediated signaling and calibrated NF initialization. These findings indicate that the 5G Core is a practical entry point for quantum-safe migration with ML-KEM, ML-DSA, and appropriate startup orchestration.

ACKNOWLEDGMENTS

This work is partially supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy as part of the project “6G Future Lab Bavaria”.

REFERENCES

[1] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *SIAM Journal on Computing*, vol. 26, no. 5, p. 1484–1509, Oct. 1997. [Online]. Available: http://dx.doi.org/10.1137/S0097539795293172

[2] M. Mosca and M. Piani, “Quantum threat timeline report 2024: Assessing the impact of quantum computing on cryptography,” Global Risk Institute, Tech. Rep., 2024, accessed: 2025-11-26. [Online]. Available: https://globalriskinstitute.org/publications/quantum-threat-timeline-report-2024/

[3] G. Alagic, M. Bros, P. Ciadoux, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, H. Silberg, D. Smith-Tone, and N. Waller, “Status report on the fourth round of the nist post-quantum cryptography standardization process,” National Institute of Standards and Technology, Tech. Rep. NIST IR 8545, March 2025. [Online]. Available: https://csrc.nist.gov/pubs/ir/8545/final

[4] D. Stebila and M. Mosca, “Post-quantum key exchange for the internet and the open quantum safe project,” in *Selected Areas in Cryptography – SAC 2016*, ser. Lecture Notes in Computer Science, R. Avanzi and H. Heys, Eds., vol. 10532. Springer, 2017, pp. 14–37. [Online]. Available: https://eprint.iacr.org/2016/1017.pdf

[5] T. C. Clancy, R. W. McGwier, and L. Chen, “Post-quantum cryptography and 5g security: tutorial,” in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 285. [Online]. Available: https://doi.org/10.1145/3317549.3324882

[6] G. Yigit and J. Attwood, “Deutsche telekom’s horizontal telco cloud (htc) establishes a new industry blueprint for telco clouds,” Deutsche Telekom, Tech. Rep., February 2026, white paper.

[7] M. T. Damir and V. Niemi, “On post-quantum identification in 5g,” in *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec ’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 292–294. [Online]. Available: https://doi.org/10.1145/3507657.3529657

[8] R. C. Vuppala, D. Kumar, D. Je, N. Sharma, A. Nigam, and D. Kim, “Post-quantum secure hybrid methods for ue primary authentication in 6g with forward secrecy,” in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, 2023, pp. 2590–2595.

[9] K. Bürstinghaus-Steinbach, C. Krauß, R. Niederhagen, and M. Schneider, “Post-quantum tls on embedded systems: Integrating and evaluating kyber and sphincs+ with mbed tls,” in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, ser. ASIA CCS ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 841–852. [Online]. Available: https://doi.org/10.1145/3320269.3384725

[10] M. Sosnowski, F. Wiedner, E. Hauser, L. Steger, D. Schoinianakis, S. Gallenmüller, and G. Carle, “The performance of post-quantum tls 1.3,” in *Companion of the 19th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 19–27. [Online]. Available: https://doi.org/10.1145/3624354.3630585

[11] G. Tasopoulos, J. Li, A. P. Fournaris, R. K. Zhao, A. Sakzad, and R. Steinfeld, “Performance evaluation of post-quantum tls 1.3 on resource-constrained embedded systems,” in *Information Security Practice and Experience: 17th International Conference, ISPEC 2022, Taipei, Taiwan, November 23–25, 2022, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 432–451. [Online]. Available: https://doi.org/10.1007/978-3-031-21280-2_24

[12] S. Jeong, “Open5gs: Open source 5g core network,” GitHub repository, 2025, accessed: 2025-11-26. [Online]. Available: https://github.com/open5gs/open5gs

[13] E. Alperen, “Ueransim: 5g ue and ran simulator,” GitHub repository, 2025, accessed: 2025-11-26. [Online]. Available: https://github.com/aligungr/UERANSIM

[14] 3GPP, “3rd generation partnership project; technical specification group services and systems aspects; system architecture for the 5g system (5gs); stage 2,” 3GPP, Tech. Rep. TS 23.501 V19.5.0 (2025-09), 2025.

[15] ——, “3rd generation partnership project; technical specification group services and systems aspects; security architecture and procedures for 5g system (release 19),” 3GPP, Tech. Rep. TS 33.501 V19.1.0 (2025-01), 2025.

[16] O. Zeidler, J. Sturm, D. Fraunholz, and W. Kellerer, “Performance evaluation of transport layer security in the 5g core control plane,” in *Proceedings of the 17th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec ’24. New York, NY, USA: Association for Computing Machinery, 2024, p. 78–88. [Online]. Available: https://doi.org/10.1145/3643833.3656140

[17] E. Rescorla, “The transport layer security (tls) protocol version 1.3,” Internet Requests for Comments, 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8446

[18] T. Dierks and E. Rescorla, “The transport layer security (tls) protocol version 1.2,” Internet Requests for Comments, August 2008. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc5246

[19] National Institute of Standards and Technology, “Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM) Standard,” U.S. Department of Commerce, Tech. Rep. FIPS 203, August 2024. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf

[20] C. Aguilar-Melchor, O. Blazy, J.-C. Deneuville, P. Gaborit, and G. Zémor, “Efficient encryption from random quasi-cyclic codes,” *IEEE Transactions on Information Theory*, vol. 64, no. 5, pp. 3927–3943, 2018.

[21] National Institute of Standards and Technology, “Module-Lattice-Based Digital Signature Standard,” U.S. Department of Commerce, Tech. Rep. FIPS 204, August 2024. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf

[22] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, “Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU,” NIST Post-Quantum Cryptography Project, Tech. Rep., 2018. [Online]. Available: https://falcon-sign.info/falcon.pdf

[23] National Institute of Standards and Technology, “Stateless Hash-Based Digital Signature Standard,” U.S. Department of Commerce, Tech. Rep. FIPS 205, August 2024. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.pdf

[24] M. Baldi, A. Barenghi, M. Battagliola, S. Bitzer, M. Gianvecchio, P. Karl, F. Manganiello, A. Pavoni, G. Pelosi, F. Pintore, P. Santini, J. Schupp, E. Signorini, F. Slaughter, A. Wachter-Zeh, and V. Weger, “CROSS: Codes and restricted objects signature scheme — specification v2.2,” CROSS Project, Tech. Rep., Jul. 2025, accessed: 2025-11-26. [Online]. Available: https://www.cross-crypto.com/CROSS_Specification_v2.2.pdf

[25] S. Hoque, A. Aydeger, E. Zeydan, and M. Liyanage, “Analysis of post-quantum cryptography in user equipment in 5g and beyond,” 2025. [Online]. Available: https://arxiv.org/abs/2507.17074

[26] P. Scalise, R. Garcia, M. Boeding, M. Hempel, and H. Sharif, “An applied analysis of securing 5g/6g core networks with post-quantum key encapsulation methods,” *Electronics*, vol. 13, no. 21, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/21/4258

[27] 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)*, 2025, pp. 1–6.

[28] Y. Hanna, D. Pineda, M. Veksler, M. Paudel, K. Akkaya, M. Anastasova, and R. Azarderakhsh, “Integrating post-quantum tls into the control plane of 5g networks,” in *2024 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, 2024, pp. 1–8.

[29] H. Supreeth, “docker_open5gs,” GitHub repository, 2025, accessed: 2025-11-26. [Online]. Available: https://github.com/herlesupreeth/docker_open5gs

[30] T. Dierks and E. Rescorla, “Recommendations for secure use of transport layer security (tls) and datagram transport layer security (dtls),” Internet Requests for Comments, November 2022. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc9325