# Integrating Post-Quantum TLS into the Control Plane of 5G Networks

Yacoub Hanna*, Diana Pineda*, Maryna Veksler*, Manish Paudel*, Kemal Akkaya*,
Mila Anastasova†, and Reza Azarderakhsh†

*Advanced Wireless and Security Lab, Florida International University, Miami, FL USA 33174
Email: {yhann002, dpine033, mveks001, mpaud002, kakkaya}@fiu.edu

†Dept. of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL USA 33431
Email: {manastasova2017, razarderakhsh}@fau.edu

***Abstract*—Significant performance improvements in bandwidth and latency make 5G a suitable candidate for a wide range of applications, particularly those requiring real-time communication, such as Industrial Control Systems (ICS) and autonomous vehicles. However, today's security, including modern cryptographic systems, is prone to different attacks caused by the high computational power of quantum computing, emphasizing the requirements for including quantum-resistant security measures. To accommodate attacks targeted at 5G networks, there are efforts to move towards TLS-based security, which is the widely accepted standard on other networks. However, integrating post-quantum algorithms must also be considered in such a transition. Thus, this paper is the first to perform the integration of Post-quantum TLS (PQ-TLS) protocols into 5G networks and offer a realistic performance evaluation. Our approach focuses on bringing PQ-TLS within the 5G control plane (CP) without needing a major overhaul, thus ensuring communications' interoperability even with legacy components of 5G, which may not support TLS. Specifically, we have transitioned the registration and authentication protocols for the core network functions and the user equipment (UE) by following a TLS tunneling approach using virtualization. We then evaluate the performance and feasibility of PQ-TLS in enhancing the security of 5G communications on an actual testbed. Our results demonstrate that while PQ algorithms introduce some overhead, they remain viable for 5G applications, particularly for protocols that can run on the core network.**

***Index Terms*—5G, AKA, TLS, Tunneling, Post Quantum Cryptography, Edge computing, Interoperability, Performance.**

## I. Introduction

5G mobile technology provides anywhere, anytime connection across billions of devices. The faster speeds, reduced latency, and enhanced coverage of this technology introduced new classes of applications, such as Enhanced Mobile Broadband (eMBB), Ultra-Reliable and Low-Latency Communications (URLLC), and massive machine-type communications (mMTC). With the vision of 6G, there is now a push toward the integration of more intelligence and automation capabilities. Nevertheless, while 5G technology and network applications are rapidly expanding, new security threats and attacks emerge, requiring a thorough analysis of the existing security specifications.

The Authentication and Key Agreement (AKA) Protocol [1] is one of the crucial security mechanisms in 5G that ensures the initialization of a secure communication channel between subscribers and carriers. Nonetheless, the current AKA standard created and approved by 3GPP has shown to underspecify the security requirements of the protocol, resulting in the vulnerable 5G connections [2]. Therefore, more advanced security features are required to strengthen the security of 5G protocols. Moreover, since 5G often serves as an enabler for real-world solutions that support time-sensitive applications [3] across a variety of domains such as the Industrial Internet of Things (IIoT), Cyber-Physical Systems (CPS), and Metaverse [4], these protocols should be both secure and efficient.

Consequently, the Transport Layer Security (TLS) protocol has been touted as a more secure alternative to be used in 5G systems since it provides a state-of-the-art security mechanism across a wide range of networks. In particular, Since TLS allows easy termination security directly in the network functions instead of gateways, it becomes an ideal candidate for 5G deployment with multi-tenancy. However, despite TLS being secured against the majority of classic attacks, its asymmetric cryptography is vulnerable to attacks from quantum computers [5]. For example, Shor's algorithm solves prime factorization and discrete logarithms in polynomial time, posing a significant threat to most of the widely used public key cryptosystems [6], with its implications demonstrated on a smaller scale. To address these challenges, various Post-Quantum Cryptography (PQC) algorithms are being actively designed and are currently in the testing stages. One such open-source PQC library is provided by the Open Quantum Safe (OQS) project, which claims to provide the security levels specified by the National Institute of Standards and Technology (NIST) [7].

While there have been multiple attempts to incorporate TLS protocol for 5G communications [8], [9], they are limited to the implementations within the core network of 5G and lack any PQ support. Thus, in this paper, we propose integrating PQ-TLS into 5G using existing open-source implementations without requiring a major overhaul of their implementations. This is particularly critical since any changes to the protocols would raise interoperability challenges among different components of 5G as well as among different telecom operators in case of roaming needs. We target control plane (CP) communications as

there are many control protocols within 5G ecosystem that utilize secure connection establishment among different components. Note that in the data plane, once a symmetric key is created, data can be encrypted using existing standards such as AES-256 to be PQ-compliant. Another goal of this work is to examine PQ-TLS performance and feasibility for use in specific 5G applications. This is because 5G systems allow rapid and dynamic deployment/disposal of a large number of network slices that can serve different users' needs, and thus, scalability becomes a concern with more stringent security protocols.

Integrating PQ-TLS across 5G communications is not an easy task due to the network complexity. Therefore, we opt to utilize a tunneling-based TLS approach to secure end-to-end Control Plane (CP) communications. Specifically, tunneling should be implemented as a client-server architecture across four primary components of 5G systems: User Equipment (UE), the Next Generation Node B (gNB), Network Registration Function (NRF), and specific core network functions such as Service Communication Proxy (SCP) and Access and Mobility Management Function (AMF). We identify three distinct pairs of server-client pairs to create TLS tunnels in the CP as follows: NRF-SCP, AMF-gNB, and gNB-UE. As a result, we create a chain of the PQ-TLS tunnels across 5G connections to achieve fully secure end-to-end communications in 5G. Our approach does not require any direct modification of existing 5G components or infrastructure. We rely on virtualized interfaces that can host the TLS tunnel on the client and server.

We implemented the proposed approach into two testbed setups using 5G network functions deployed at Florida International University (FIU) and the Google Cloud environment. We consider multiple PQ algorithms to assess their implications on the performance of 5G connections between different network components. We compared the performance of our proposed PQ-TLS approach against traditional and TLS-based services and discussed the potential trade-off between security and communications overhead. The results indicate that bringing TLS to 5G CP comes with some overhead, but the right choice of PQ algorithms can maintain the performance of existing TLS solutions based on classical algorithms such as RSA or ECDSA.

The remainder of the paper is organized as follows. First, we present state-of-the-art works on secure communications in 5G networks in Section II. In Section III, we present the background on TLS, 5G, and PQ algorithms We present our approach in Section IV. In Section V, we evaluate the performance of our method in various settings and discuss the results. Finally, we present the conclusions in Section VI.

## II. Related Work

Open-source 5G deployment lacks an efficient implementation and analysis of TLS protocol. Authors in [8] implemented a TLS communication within the core network and evaluated the impact of different cipher suites on the network performance. Their results indicate that the communication delay increases proportionally to the cipher size, while TLS 1.3 is indicated to outperform TLS 1.2 in terms of overhead and the number of messages required to establish a secure connection between NFs. However, their implementation details and tools are missing. Later, Linh et al. [9] analyzed the vulnerabilities and 3GPP compliance of TLS implementations within three major open-source 5G networks, namely free5GC, open5GS, and OAU 5G CN. They determined that while both free5GC and open5GS provide TLS support for core network communications, neither of them is fully compliant with 3GPP requirements except for the key exchange algorithm and recommended TLS version.

3GPP defines EAP-TLS as a standard security protocol for 5G communications. Specifically, it is used for UE authentication in limited use cases such as IoT environments or private networks. Subsequently, the authors in [2] attempted to analyze and verify the security of 5G EAP-TLS. They designed a ProVerif model checker that revealed multiple weaknesses and design flaws, which can break indented security. However, this work does not provide a practical implementation and evaluation of EAP-TLS for 5G communications.

Unlike the existing works for TLS in 5G, which focus either on the core network or UE authentication, we integrate TLS across all components of the 5G network within the CP. Furthermore, we implement a tunneling TLS approach that does not require direct modification of existing 5G components or infrastructure. Thus, our framework can be applied across various existing 5G implementations. To the best of our knowledge, this is the first work to successfully combine PQ and TLS technologies to secure end-to-end 5G communications within a real testbed environment.

## III. Background

In this section, we describe briefly the fundamental concepts and technologies we use in our work.

### A. Transport Layer Security (TLS) Handshake

The TLS handshake is a crucial component of the TLS protocol. It establishes trust between two parties, a client and a server, via a series of messages that define a shared key to encrypt their communications. The complete process of the TLS handshake is shown in Figure 1. Eventually, a TLS session offers a symmetric key to be used for the confidentiality and integrity of the data. Moreover, TLS relies on certificates for authentication of the server and the client. As can be seen in Fig. 1, the server responds with a *Server Hello* message, which is responsible for transmitting a certificate to the client for the purpose of validating its identity. These certificates are based on the existing standard signature algorithms (e.g., RSA or ECDSA).

### B. Extensible Authentication Protocol-Transport Layer Security (EAP-TLS)

EAP-TLS is a mutual authentication method that an EAP peer and server can use to authenticate each other. The Authenticator(entity initiating EAP authentication) and the peer first negotiate EAP by exchanging EAP-Request/Identity and EAP-Response/Identity packets in EAP-TLS. Later on, the authenticator will repeat all EAP packets to a backend EAP server,

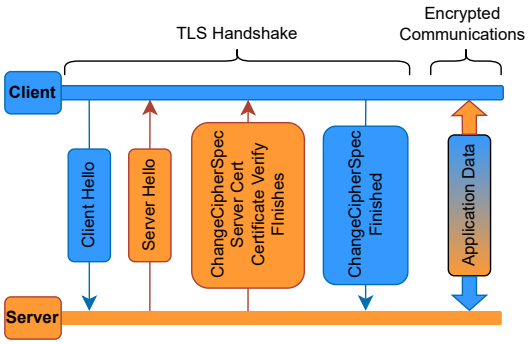


Fig. 1: TLS Handshake Messages

whose response to the peer's identity is to send an EAP-TLS/Start packet to start up the EAP-TLS conversation [10]. For implementation within 3GPP networks, the specific requirements regarding supported TLS versions and cipher suites are detailed in the 3GPP TS 33.310 [11] standard, which should be followed to ensure compliance and security.

In 5G architecture, the EAP framework involves the UE as the Client, the Session Management Function (SMF) as the Authenticator, and a RADIUS Server or AAA Server as the Authentication Server. The SMF initiates the authentication procedure, sending start messages to the AAA server and establishing an authentication channel with the UE. Then they authenticate each other by one of the protocols like EAP-TLS or EAP-PEAP [12].

### C. Post-Quantum Algorithms

The advancement of quantum computing uncovered the weaknesses of standard algorithms used for encryption. For example, Shor's algorithm is capable of breaking the commonly used discrete logarithm and integer factorization problems. Thus, post-quantum cryptography provides methods to design public-key cryptosystems resistant to quantum computers [6]. In this paper, we use the three signature scheme families, namely *Dilithium*, *Falcon*, and *SPHINCS (S+SHA)*, that US NIST has selected as the first three post-quantum signature algorithms to be standardized [13].

### D. 5G Background

5G provides multiple advantages over traditional cellular communications, including enhanced connectivity, reduced latency, and improved coverage. This is achieved by completely transitioning to a *service-based architecture* using virtualization and software-defined networking (SDN). 5G network slicing technology can create sub-networks tailored to specific requirements that enable prioritization of the emergency connection and prevention of network overloads.

As shown in Fig. 2, different from 4G/LTE, the architecture of 5G Core (5GC) is divided into two separate planes: the Control Plane (CP) and the User Plane (UP). This division is commonly referred to as Control User Plane Separation (CUPS). The primary purpose behind separating the CP and the UP is to enable the centralization of CP Functions while concurrently facilitating the placement of UP devices closer to end-users. Consequently, this arrangement leads to reduced latency and enhanced data transmission speed, thus resulting in an overall improvement in user experience [14]. Moreover, within the 5GC, there are various Network Functions (NFs), each designed with specific roles and responsibilities. Some of these functions that relate to our work are as follows:

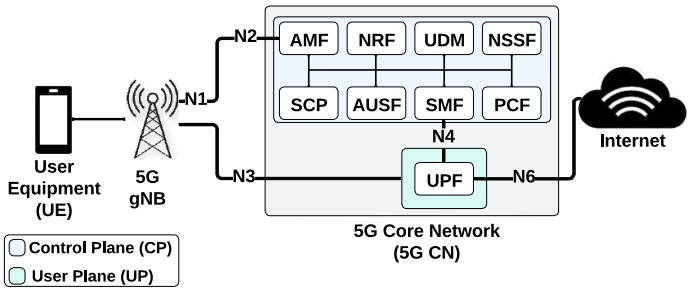


Fig. 2: 5G Core Architecture

- **Access and Mobility Management Function (AMF):** The primary operations of the AMF are registration, connection, as well as mobility management of UE devices [15].
- **User Plane Function (UPF):** UPF is used to connect the user, data coming from the gNB to the Data Network (DN), which is referred to as an internet service provider.
- **NF Repository Function (NRF):** NRF has a significant impact on the network by overseeing various network functions and enabling the storage and transmission of their data within the 5G core network [16].
- **Authentication Server Function (AUSF):** During the registration, the AMF communicates with the AUSF to verify the identity of the UE.
- **Service Communication Proxy (SCP):** This function is responsible for managing and optimizing the communication between network functions within the 5G Core [17].

## IV. Proposed Approach

### A. Problem and Motivation

The 5G core network oversees comprehensive network operations, including data routing and mobility management. However, due to their advanced capabilities and increased connectivity, 5G networks introduce security challenges. Thus, securing communication within this highly interconnected network is critical.

To this end, 3GPP standards recommended the deployment of TLS within the 5G core and the other control operations related to UEs. By incorporating TLS into the infrastructure, the 5G core network can efficiently reduce the threats linked to unauthorized entry and data leaks, thereby enhancing the network's overall security. The other benefit is that TLS can be configured to support PQ algorithms, guaranteeing resilience against any attack from future quantum computers.

Nevertheless, many open-source 5G implementations still lack universal TLS support across all CP components, impacting comprehensive testing of this infrastructure with respect to the evaluation of TLS performance. This limitation also restricts the adoption and evaluation of PQ algorithms, which is crucial for future security. Indeed, even if these implementations start offering TLS capabilities, there will still be a transition period where there is a need to support legacy security protocols of 5G

and TLS simultaneously to eliminate any interoperability issues. Therefore, there is an urgent need to bring comprehensive TLS support to 5G CP and enable convenient performance testing under various conditions, including PQ capabilities.

Note that 3GPP recommends using EAP-TLS as a security protocol for 5G communications. However, EAP-TLS has some security concerns that have been exposed recently: 1) For instance, EAP-TLS integrity protection does not include EAP header fields, Code, Identifier, Length, Type, and Flags, which an attacker can modify [18]. Such a vulnerability could result in denial-of-service (DoS) attacks by modifications in the Type fields, enabling attackers to change the TLS-based EAP method, potentially enabling them to complete sessions undetected. 2) The RFC [10] does not specify whether clients using EAP should verify a Message-Authenticator in an Access-Accept without an EAP-Message. Clients that skip this check are vulnerable to attacks from colliding packets without an EAP message or Message-Authenticator. This will allow unauthorized access [19]. 3) During the initiation of protocol interaction, an attacker can intercept initial information from the UE to the Home Network and Server Network without a Subscription Permanent Identifier (SUPI) and establish a TLS handshake. This might enable the intruder to act as a man-in-the-middle attack, modifying the interactions between UE and Network [20].

Due to such issues in EAP-TLS, we opt for a solution that also prevents such security concerns while enabling PQ integration, as detailed next.

### B. Integration of VPN TLS Tunnel

Our approach is based on an open-source TLS Tunnel integration into 5G without re-hauling the existing implementation codes. For tunneling, we utilize a Virtual Private Network (VPN), which is a networking feature that establishes a secure and protected virtual communication pathway that overlays existing physical data transmissions. Therefore, data packets flow through encrypted tunnels that wrap payloads over unsecured public networks [21].

In our approach, we establish TLS tunnels by modifying the client and server through network virtualization to serve as pathways linking two defined devices to encrypt data and verify transfers to guarantee the integrity of the data. Note that when using EAP-TLS, the data transmission may continue without further encryption once authenticated. On the other hand, VPN over TLS establishes a secure private tunnel that encrypts every traffic between the client and server for the entire session and not just for the authentication phase.

The tunnel creation consists of two main steps: the setup phase (performed only once) and the TLS handshake. During the setup phase, the server and client load their respective configurations and set up necessary parameters such as network addresses, port numbers, and certificates. Both the server and the client create SSL contexts to encrypt their communication. The client sends an authentication packet containing its credentials to the server. The server verifies these credentials against its protected and hashed database as part of the TLS initiation process. After successful authentication the server sends a configuration packet to the client, including the IP address, Netmask, and Maximum Transmission Unit (MTU) for the client's Tunnel (TUN) interface. This IP address is assigned from a pool of available addresses maintained by the server, ensuring that each client receives a unique address for the duration of the session.

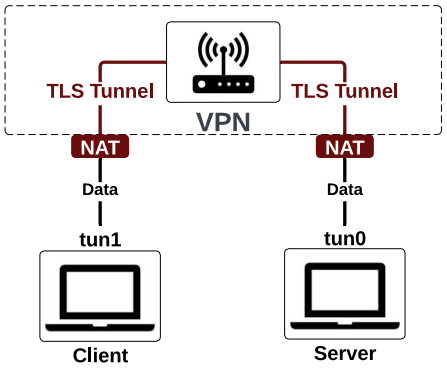


Fig. 3: VPN Over TLS (TLS tunneling).

As shown in Fig. 3, the TUN interface is a virtual network device used by both the server and client to handle network traffic, acting as a bridge between the client's virtual network and the server's network. On the server side, Network Address Translation (NAT) is used to forward traffic from the TUN interface (tun0) to the outside (tun1) and vice versa. Meanwhile, the client reads and writes data from its TUN interface. Specifically, the TUN interface is set as the default gateway.

After establishing a connection between the client and server, the second phase in the procedure is the execution of a TLS handshake as described in Section III-A.

We would like to note that VPN Over TLS can be implemented in UE devices, whether they are IoT or Mobile Devices, such as iOS and Android, through dedicated applications. For iOS, the *Network Extension* framework provides APIs to manage VPN connections and configure the virtual network [22]. On Android, the *VpnService API* allows the creation of VPN applications that manage virtual network interfaces [23]. IoT devices may have constrained resources, and implementing VPN over TLS involves integrating or developing lightweight VPN clients that support TUN interfaces. Many IoT devices run Linux-based systems, making using tools like OpenVPN [24] or strongSwan [25] to establish secure connections feasible.

### C. TLS Integration into NF Registration

5G has the advantage over prior generations in supporting a service-based architecture where all network functions can register themselves and their supported services with the NRF without introducing new protocols or interfaces. The registration procedure involves any network function (NF) in the core sending the necessary profile registration information to the NRF, which then replies whether the NF has been registered successfully [26].

Since NFs send the required information during this process, as shown in Fig. 4, there is a concern about unauthorized parties intercepting these messages and impersonating NFs or

altering the transmitted data. Therefore, integrating TLS into the communication between NFs within the 5G Core Network is crucial. Note that when TLS communication is integrated, the NF must establish a TLS connection before authenticating with another CP function.

As shown in Fig. 4, to integrate TLS between NFs, we utilize VPN over TLS. In this setup, the NRF behaves as a server waiting for the SCP to establish a TLS tunnel. The SCP acts as an intermediary, facilitating secure communication between NFs by encrypting the data transmitted over the TLS tunnel. Any NF that uses the SCP to communicate with the NRF or other NFs will have its data encrypted, ensuring that sensitive information remains protected from potential security threats such as interception, impersonation, and data manipulation. VPN tunnels will help maintain a continuous connection during the communication session.

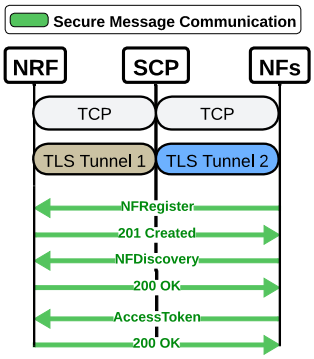


Fig. 4: NF Registration protocol and exchanged messages.

### D. TLS Integration to UE Registration

During the UE registration phase in 5G networks, all the NFs have already registered themselves with the NRF and are prepared to handle UE CP messages. When a UE initiates a connection to the 5G network, it begins the registration process to gain access. The UE registration protocol in 5G involves initial communication with the gNB, followed by interaction with AMF, which then triggers the authentication procedure with the AUSF and other NFs in the control plane [27].

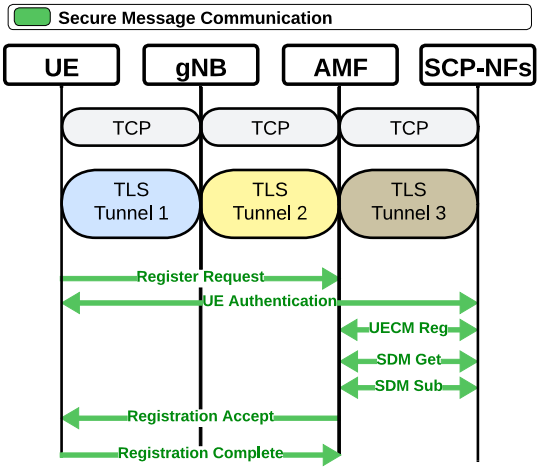


Fig. 5: UE registration and authentication protocol elements.

Integrating TLS into this registration process follows a similar concept as discussed in the earlier subsection (IV-C), where VPN over TLS is utilized. However, due to the involvement of multiple components, maintaining a continuous TLS connection for communication between all these entities becomes challenging. Therefore, our approach divides the communication into three distinct tunnels, as illustrated in Fig. 5. In the VPN over TLS setup, the communication pairs are defined as follows: the gNB acts as the server, waiting for the UE's TLS connection initiation; the AMF serves as the server for interactions with the gNB; and the SCP functions as the server for communications with other NFs. Using this segmented approach, we ensure that each communication segment is secured through TLS encryption across the UE registration process in the 5G network. The handoff process between these tunnels is facilitated by the 5G component (e.g., UE, gNB, AMF) deploying them. Specifically, each tunnel is identified by the IP address assigned during the initial setup phase, which helps the hosting components accurately direct traffic between the tunnels.

## V. Performance Evaluation

This section summarizes the experiment setup, metrics, and performance evaluation results.

### A. Setting up the 5G Testbed Environment

The proposed approach has been deployed and tested in the 5G open-source testbed, based on the work in [28]. This experimental setup includes various open-source projects that emulate the CP, User Plane (UP), gNB, and UE components of the 5G infrastructure. We utilize the open-source Open5GS project[1] to implement the 5G Network Core, while UERANSIM is used to realize UE and gNB. To test the efficiency of our approach, we set up two testbeds hosted by physical devices and cloud as indicated in Figure 6. The motivation behind these two setups is to discern the impact of computation and communication overhead. *Cloud Testbed* is more realistic as, typically, the core network functions will stay in remote locations, potentially on the private clouds of the telecom operators. At the same time, UE and gNBs will be much closer geographically.

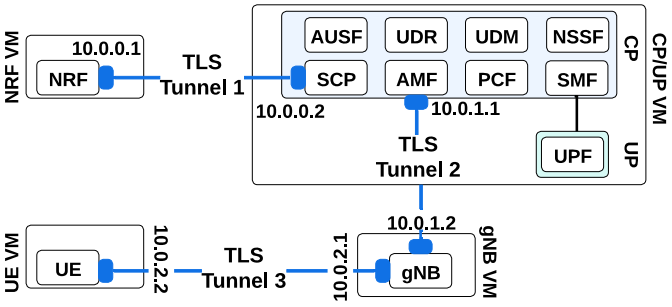


Fig. 6: 5G Testbed with TLS capabilities.

In our *Local Testbed*, the NRF runs as a standalone function on the Dell Precision Workstation Intel i7 10$^{th}$ generation processor. We use three laptops running Ubuntu 20.04, 8GB RAM, and a 3.4GHZ Intel i5 8$^{th}$ generation processor to deploy CP functions, gNB, and UE. The connection was established among the

[1]https://github.com/open5gs/open5gs

machines by setting up a Hotspot network in which one device acts as an access point, creating a wireless network that other devices may connect to and communicate locally over Wi-Fi. We use the proposed approach to establish TLS tunneling between four devices (which are shown as VMs in Fig. 6). For our *Cloud Testbed*, we duplicated this setup on the Google Cloud Platform (GCP) using four distinct virtual machines (VMs) in the East and Central US regions.

We utilized the OQS-OpenSSL library [29] to implement PQ-TLS on each machine within our testbed. We make the complete code for our setup available under GitHub[2]. We used Wireshark to capture and analyze the traffic generated by both testbeds. Moreover, we examine various PQ algorithms in order to evaluate their impact on the efficiency of 5G connections across diverse network elements. The PQ signature algorithms have been applied in our experiments to create a certificate that the root CA verifies. These PQ certificates verify the identity of the server. We assume that the PQ public key of the CA will be available at the client of each machine.

### B. Metrics and Benchmarks

We assess the performance of our approach using four overhead metrics: *latency*, *packet size*, *number of messages*, and *energy consumption*. The latency measures the time required to establish a secure connection between the 5G network components. The number of messages and packet size correspond to the total number of messages exchanged during connection establishment and the size of each packet, respectively. Energy consumption is the total amount of electrical power the clients use while performing tasks under the different cryptographic algorithms.

We use these metrics to evaluate the efficiency of PQ-TLS for 1) NRF initialization with the 5G core network and 2) UE registration to the 5G core network.

In the experiments, we tested five different certificates: ECDSA, RSA, Falcon512 (Fal512), Dilithium2 (Dil2), and SPHINCS SHA (S+SHA). We selected two primary benchmark approaches. First, we consider cases where TLS is not employed or existing 4/5G security solutions are used. For instance, for UE registration, we compare a traditional 4/5G AKA implementation without TLS against our case when TLS tunneling is applied. In the case of NRF function registration, we use an approach where there is no TLS and security. These approaches are depicted as *noTLS* in the figures. As another benchmarking, we compare the PQ algorithms to the RSA/ECDSA implementation of TLS.

### C. Performance Results

This section presents the performance evaluation of 5G networks utilizing TLS by measuring latency in various experiment setups in local and cloud testbeds.

*1) NRF Registration with the 5G Core Network:*

[2]https://github.com/adwise-fiu/5G-PQ-TLS-Tunnel

*a) NRF Registration at Local Testbed:* We first conducted experiments in our Local Testbed to measure the overhead of TLS integration. We looked at two cases: 1) NRF-SCP, which measures the TLS overhead when NRF authenticates SCP only; 2) NRF-CP, which measures the TLS overhead when all the remaining CP functions are registered.

As shown in Table I, the results indicated that TLS brings in certain overhead in packet sizes and latency. For instance, registering the SCP function to the NRF without TLS takes 0.072 seconds, while using TLS with ECDSA increases the value to 0.143 seconds and RSA to 0.155 seconds. Also, message size increases from 32 to 53 bytes when TLS with ECDSA and RSA are used. These increases are due to the introduction of TLS handshakes in the process. Note that NRF-CP takes longer as it waits for all the NFs to register using TLS sequentially.

TABLE I: Average TLS Overhead for NRF Registration under Different Traditional Signatures in the Local Testbed

| | | noTLS | ECDSA | RSA |
|---|---|---|---|---|
| **Delay (sec)** | **NRF-SCP** | 0.072 | 0.143 | 0.155 |
| | **NRF-CP** | 0.124 | 0.378 | 0.39 |
| **Packet Size (Byte)** | **NRF-SCP** | 6351 | 15581 | 15778 |
| | **NRF-CP** | 16kB | 50kB | 52kB |
| **No of msgs** | **NRF-SCP** | 32 | 53 | 53 |
| | **NRF-CP** | 81 | 93.7 | 95.1 |

When we check the impact of PQ overhead, we see that the additional overhead depends on the type of PQ certificate used, as shown in Table II. For instance, when NRF registers the CP functions, the increase in latency compared to existing RSA-based TLS to PQ-based Falcon512 is 7.94%. For others, such as Dilithium2 and S+SHA, there is a significant increase in delay. Therefore, Falcon512 is a viable alternative to RSA, while others should not be used if performance concerns exist for specific applications. For the packet size and number of messages, the increase compared to RSA is not significant except for S+SHA.

TABLE II: Average TLS Overhead for NRF Registration under Different PQ Signatures in the Local Testbed

| | | RSA | Fal512 | Dil2 | S+SHA |
|---|---|---|---|---|---|
| **Delay (sec)** | **NRF-SCP** | 0.155 | 0.186 | 0.341 | 0.838 |
| | **NRF-CP** | 0.390 | 0.421 | 0.576 | 1.073 |
| **Packet Size (Byte)** | **NRF-SCP** | 15778 | 16851 | 21801 | 59392 |
| | **NRF-CP** | 52kB | 53kB | 56kB | 82kB |
| **No of msgs** | **NRF-SCP** | 53 | 56 | 60 | 84 |
| | **NRF-CP** | 96.2 | 99 | 106.8 | 130.8 |

*b) NRF Registration on the Cloud Testbed:* In our cloud testbed experiment, as shown in Fig.7, the NRF function and CP are located in Ohio and South Carolina, respectively. Looking at the Tables III and IV, the results reflect slightly higher values compared to the *Local Testbed* due to additional propagation delay in the cloud environment.

For example, when comparing Dilithium2 on local and cloud testbeds for NRF registration to CP, we find minimal variation; it is 0.421 seconds in the local and 0.48 seconds in the cloud testbed, resulting in an acceptable overhead increase of only 14.27%. S+SHA benefits the most from the cloud testbed as the latency goes down.

These results suggest that Falcon512 is still the best option in realistic deployments. Other alternatives, such as Dilithium and S+SHA, can also be considered as they do not significantly increase the latency. Indeed, given its implementation simplicity due to being stateless, S+SHA can be attractive for large-scale 5G systems. One interesting observation is that both Dilithium and S+SHA perform better in latency compared to the *Local Testbed*. This indicates that these algorithms are advantageous regarding message size and count. We will discuss this in Section VI.

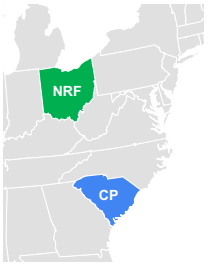


Fig. 7: NRF and CP locations on the Cloud Testbed.

TABLE III: Average TLS Overhead for NRF Registration with Different Traditional Signatures in the Cloud Testbed

| | | noTLS | ECDSA | RSA |
|---|---|---|---|---|
| **Delay (sec)** | **NRF-SCP** | 0.107 | 0.26 | 0.281 |
| | **NRF-CP** | 0.110 | 0.431 | 0.452 |
| **Packet Size (Byte)** | **NRF-SCP** | 6kB | 14kB | 15kB |
| | **NRF-CP** | 14kB | 62kB | 64kB |
| **No of msgs** | **NRF-SCP** | 34 | 48 | 50 |
| | **NRF-CP** | 64 | 78 | 80 |

TABLE IV: Average TLS Overhead for NRF Registration with Different PQ Signatures in the Cloud Testbed

| | | RSA | Fal512 | Dil2 | S+SHA |
|---|---|---|---|---|---|
| **Delay (sec)** | **NRF-SCP** | 0.281 | 0.291 | 0.312 | 0.385 |
| | **NRF-CP** | 0.452 | 0.481 | 0.529 | 0.653 |
| **Packet Size (Byte)** | **NRF-SCP** | 15kB | 16.5kB | 20kB | 50kB |
| | **NRF-CP** | 64kB | 64.4kB | 68.48kB | 97kB |
| **No of msgs** | **NRF-SCP** | 50 | 54 | 58 | 82 |
| | **NRF-CP** | 80 | 84 | 88 | 112 |

*2) UE Registration with the 5G Core Network:* Using the same setups, we conducted experiments to measure the overhead of bringing TLS to the UE registration protocol.

*a) UE Registration in the Local Testbed:* Table V and VI shows the average overhead during registration of UE to AMF. For the traditional signature algorithm, there is a delay overhead of 8.24% with ECDSA and 11.65% for RSA when compared with no TLS. On the other hand, Falcon512 and Dilithium2 have an overhead of 7.61% and 45.45% compared to RSA, respectively.

Also, the UE-AMF message count shows a similar trend: Falcon512 and Dilithium2 have 9.048% and 18.09% more messages compared to RSA, respectively. Consequently, based on these results, Falcon512 is again a viable alternative for UE registration.

We note that since UE registration suffers from a higher baseline delay and happens more frequently than NRF registration, this results in a more significant impact on the overall system performance.

*b) UE Registration in the Cloud Testbed:* We have deployed the cloud setup shown in Fig. 8 for UE registration where the NRF-CP (i.e., core network) is located in Iowa, while both gNB and UE are located in South Carolina.

TABLE V: Average TLS Overhead for UE Registration with Different Traditional Signatures in the Local Testbed

| | | noTLS | ECDSA | RSA |
|---|---|---|---|---|
| **UE-AMF** | **Delay (sec)** | 0.365 | 0.3951 | 0.4075 |
| | **Packet Size (Byte)** | 13kB | 21kB | 25kB |
| | **No of msgs** | 30 | 43 | 43.1 |

TABLE VI: Average TLS Overhead for UE Registration with Different PQ Signatures in the Local Testbed

| | | RSA | Fal512 | Dil2 | S+SHA |
|---|---|---|---|---|---|
| **UE-AMF** | **Delay (sec)** | 0.407 | 0.438 | 0.592 | 1.09 |
| | **Packet Size (Byte)** | 25kB | 29kB | 32kB | 67kB |
| | **No of msgs** | 43.1 | 47 | 50.9 | 75 |

Table VII and VIII show the UE registration overhead. There is a noticeable increase when using RSA, with 0.624 seconds compared to 0.521 seconds when using noTLS. Moreover, ECDSA and RSA require 37 messages, whereas noTLS requires only 21. On the other hand, Falcon512 and Dilithium2 show an increase in the latency overhead of 2.88% and 4.65%, compared to RSA. Additionally, the packet size of Falcon512 increases by 8.11%.

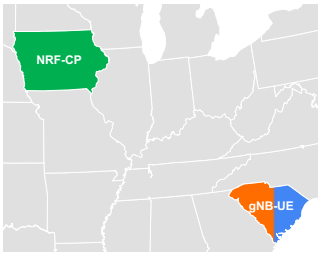


Fig. 8: UE, gNB, and CP locations on the Cloud Testbed.

Although S+SHA contains 69 messages compared to Falcon512's 41 in cloud environments, the delays remain comparable. This is because the cloud has optimized CPU usage and faster transmission speeds, which better accommodate the large signature size and computationally demanding certificates, like Dilithium2 and S+SHA, over smaller and less demanding certificates like RSA and Falcon512.

TABLE VII: Average TLS Overhead with Different Traditional Signatures on the Cloud Testbed.

| | | noTLS | ECDSA | RSA |
|---|---|---|---|---|
| **UE-AMF** | **Delay (sec)** | 0.521 | 0.611 | 0.624 |
| | **Packet Size (Byte)** | 2kB | 4.4kB | 4.44kB |
| | **No of msgs** | 21 | 37 | 37 |

TABLE VIII: Average TLS Overhead with Different PQ Signatures on the Cloud Testbed.

| | | RSA | Fal512 | Dil2 | S+SHA |
|---|---|---|---|---|---|
| **UE-AMF** | **Delay (sec)** | 0.624 | 0.642 | 0.653 | 0.686 |
| | **Packet Size (Byte)** | 4.44kB | 4.8kB | 12.4kB | 39kB |
| | **No of msgs** | 37 | 41 | 45 | 69 |

### *D. Comparing EAP-TLS and TLS Tunneling*

We also wanted to evaluate how much, if any, overhead TLS tunneling brings compared to regular EAP-TLS as we advocate for it. To this end, we performed a different experiment. We deployed a Raspberry PI 4 as the client UE and the server in the cloud. We measured the energy consumption from the UE perspective and latency for the whole TLS handshake.

Table IX shows the additional energy overhead compared to a noTLS approach for both approaches under different signatures. Note that the local setup also offered the same results, so we did not include them separately. As can be seen, the energy burden it brings to the client side, even on an RPi, is minimal. Most importantly, the energy consumption for EAP-TLS and TLS tunneling is the same.

TABLE IX: Energy Consumption Comparison of EAP-TLS and TLS Tunnel (Watts) in the Cloud Testbed.

| | **TLS** | **TLS Tunneling** |
|---|---|---|
| **ECDSA** | 0.4084 | 0.4090 |
| **RSA** | 0.4378 | 0.4394 |
| **Fal512** | 0.4706 | 0.4738 |
| **Dil2** | 0.4789 | 0.4794 |
| **S+SHA** | 0.4864 | 0.4880 |

We then looked at the latency of the TLS handshake for both cases, which are reported in Table X. In both setups, the latency difference is negligible. These results suggest that utilizing TLS tunneling does not bring any additional overhead with respect to regular EAP-TLS.

TABLE X: Latency comparison of EAP-TLS with TLS Tunneling in Local and Cloud Testbeds

| | **Latency in Local (sec)** | | **Latency in Cloud (sec)** | |
|---|---|---|---|---|
| | **EAP-TLS** | **TLS Tunneling** | **EAP-TLS** | **TLS Tunneling** |
| **ECDSA** | 0.065 | 0.066 | 0.180 | 0.179 |
| **RSA** | 0.079 | 0.0793 | 0.200 | 0.201 |
| **Fal512** | 0.110 | 0.109 | 0.234 | 0.245 |
| **Dil2** | 0.2649 | 0.2649 | 0.266 | 0.281 |
| **S+SHA** | 0.762 | 0.762 | 0.328 | 0.351 |

### *E. Discussion*

Analyzing the *Local* and *Cloud Testbeds*, we find a discrepancy, with a significant increase in latency moving from Falcon512 to S+SHA in the *Local Testbed*, while this increase is not significant in the *Cloud Testbed*. For instance, analyzing Table II and Table IV, there is a notable increase in latency when moving from Falcon512 with 0.186 seconds to S+SHA with 0.838 seconds for registering NRF to CP in the *Local Testbed*. On the other hand, the cloud environment shows a smaller difference in latency between RSA at 0.481 seconds and S+SHA at 0.653 seconds.

This difference is due to the variance in cloud infrastructure as well as in key and signature sizes between Falcon512 and S+SHA. When comparing Falcon512 and S+SHA in terms of their key and signature sizes, we observe the following: Falcon512 has a public key size of 897 bytes, a secret key size of 1281 bytes, and a signature size of 752 bytes, while S+SHA has a significantly smaller public and secret key sizes at 32 and 64 bytes, but much larger signature size of 17088 bytes.

The overall latency consists of propagation delay, transmission of packets, and computational delay of certificates. Locally, transmission and computational delay are dominant factors, as propagation delay is negligible. Although the cloud introduces some propagation delay, it manages the overall latency effectively with better CPU utilization and faster transmission speeds. We find that the smaller certificates with low computational demands perform efficiently in local environments, while the cloud environment performs significantly better when the signature size and computational demands are larger. Therefore, certificates like Dilithium2 and S+SHA perform better in cloud environments than in local setups. Finally, using TLS tunneling instead of EAP-TLS is more suitable as it adds comprehensive security without any additional delay or energy overheads.

To conclude our findings, across all setups, Falcon512 stands out as the most practical choice when compared to traditional RSA certificates due to its modest overhead increase. In the *Local Testbed*, we observed that Dilithium2 and S+SHA exhibited a substantial spike in latency. At the same time, only Falcon512 maintained an acceptable level of latency, indicating it as the only feasible option for implementation. However, when real-life and closely configured *Cloud Testbeds* are used, the impact of latency overhead becomes less relevant for Dilithium2 and even for S+SHA compared to the *Local Testbed*, making Dilithium2 a viable alternative. Additionally, S+SHA also remains an option for cloud implementation since its algorithm is stateless and can be easier to implement in complex systems.

## VI. Conclusion

In this paper, we designed an end-to-end TLS integrated into 5G communications. We applied a tunneling approach, ensuring all network components can establish a reliable TLS connection using a VPN. In addition, we integrated PQ algorithms, Falcon512, Dilithium2, and S+SHA, with TLS to further secure 5G communications against emerging threats of quantum computing. We implemented the proposed approach on two test beds consisting of real physical devices and cloud-based virtual machines and conducted a series of experiments to assess its feasibility. Our results indicate that while PQ-TLS approaches introduce slight overhead for communications, some of them match the performance of classical TLS approaches. For instance, we identified PQ-TLS using Falcon512 as the most efficient implementation, combining both security and minimized delays. Finally, TLS tunneling also matched the performance of a regular EAP-TLS implementation.


## Acknowledgment

This research was funded by US NSF under the grant No. 2147196.


## References


[1] T. Liu, F. Wu, X. Li, and C. Chen, "A new authentication and key agreement protocol for 5g wireless networks," Telecommunication Systems, vol. 78, pp. 317–329, 2021.

[2] J. Zhang, L. Yang, W. Cao, and Q. Wang, "Formal analysis of 5g eap-tls authentication protocol using proverif," IEEE access, vol. 8, pp. 23 674–23 688, 2020.

[3] S. K. Rao and R. Prasad, "Impact of 5G technologies on industry 4.0," Wireless personal communications, vol. 100, pp. 145–159, 2018.

[4] Z. Huang, C. Xiong, H. Ni, D. Wang, Y. Tao, and T. Sun, "Standard evolution of 5g-advanced and future mobile network for extended reality and metaverse," IEEE Internet of Things Magazine, vol. 6, no. 1, pp. 20–25, 2023.

[5] H. T. Larasati and H. Kim, "Quantum cryptanalysis landscape of shor's algorithm for elliptic curve discrete logarithm problem," in Information Security Applications: 22nd International Conference, WISA 2021, Jeju Island, South Korea, August 11–13, 2021, Revised Selected Papers 22. Springer, 2021, pp. 91–104.

[6] D. Stebila and M. Mosca, "Post-quantum key exchange for the internet and the open quantum safe project," in International Conference on Selected Areas in Cryptography. Springer, 2016, pp. 14–37.

[7] National Institute of Standards & Technology, "Post-quantum cryptography," Jul. 22, 2022. [Online]. Available: https://csrc.nist.gov/Projects/post-quantum-cryptography

[8] A. Hosseini Vasoukolaei, "Tls performance evaluation in the control plane of a 5g core network slice," Ph.D. dissertation, Carleton University, 2021.

[9] A. B. N. Linh, D. Rupprecht, E. Poll, and K. Kohls, "Analysing open-source 5g core networks for tls vulnerabilities and 3gpp compliance," 2023.

[10] D. Simon, R. Hurst, and D. B. D. Aboba, "The EAP-TLS Authentication Protocol," RFC 5216, Mar. 2008. [Online]. Available: https://www.rfc-editor.org/info/rfc5216

[11] Tech Invite. (2023) Ts 33.310 - 3gpp specification. Accessed: 2024-07-22. [Online]. Available: https://www.tech-invite.com/3m33/tinv-3gpp-33-310.html

[12] Q. Hao, L. Sun, S. Guo, R. Dou, H. Liu, and D. Qian, "5g secondary authentication based on eap-tls protocol," in 2021 International Conference on Computer Technology and Media Convergence Design (CTMCD), 2021, pp. 296–300.

[13] National Institute of Standards & Technology, "Nist announces first four quantum-resistant cryptographic algorithms," Jul. 5, 2022. [Online]. Available: https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms

[14] "Control and user plane separation of epc nodes (cups)," 2017. [Online]. Available: https://www.3gpp.org/news-events/3gpp-news/cups

[15] "5g access and mobility management function (amf)," 2023. [Online]. Available: https://techcommunity.microsoft.com/t5/azure-for-operators-blog/what-is-the-5g-access-and-mobility-management-function-amf/ba-p/3707685

[16] "5g system;network function repository services;stage 3(3gpp ts 29.510 version 15.3.0 release 15)," 2019. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/129500_129599/129510/

[17] M. Toy and A. Toy, "Overall network and service architecture," Future Networks, Services and Management: Underlay and Overlay, Edge, Applications, Slicing, Cloud, Space, AI/ML, and Quantum Computing, pp. 93–155, 2021.

[18] D. Simon, R. Hurst, and D. B. D. Aboba, "Rfc 5216: The eap-tls authentication protocol," Mar 2008. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc5216#page-28

[19] Radius/udp considered harmful. [Online]. Available: https://www.blastradius.fail/pdf/radius.pdf

[20] J. Zhang, Q. Wang, L. Yang, and T. Feng, "Formal verification of 5g-eap-tls authentication protocol," in 2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC), 2019, pp. 503–509.

[21] H. Akter, S. Jahan, S. Saha, R. H. Faisal, and S. Islam, "Evaluating performances of vpn tunneling protocols based on application service requirements," in Proceedings of the Third International Conference on Trends in Computational and Cognitive Engineering: TCCE 2021. Springer, 2022, pp. 433–444.

[22] S. Demetriou, N. Zhang, Y. Lee, X. Wang, C. A. Gunter, X. Zhou, and M. Grace, "Hanguard: Sdn-driven protection of smart home wifi devices from malicious mobile apps," in Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2017, pp. 122–133.

[23] Y. Song and U. Hengartner, "Privacyguard: A vpn-based platform to detect information leakage on android devices," in Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices, 2015, pp. 15–26.

[24] A. Alshalan, S. Pisharody, and D. Huang, "A survey of mobile vpn technologies," IEEE Communications Surveys Tutorials, vol. 18, no. 2, pp. 1177–1196, 2016.

[25] A. Mense, S. Steger, D. Jukic-Sunaric, A. Mészáros, and M. Sulek, "Open source based privacy-proxy to restrain connectivity of mobile apps," in Proceedings of the 14th International Conference on Advances in Mobile Computing and Multi Media, 2016, pp. 284–287.

[26] D. Chandramouli, R. Liebhart, and J. Pirskanen, 5G for the Connected World. John Wiley & Sons, 2019.

[27] 3rd Generation Partnership Project (3GPP), "Technical Specification Group Services and System Aspects and Procedures for the 5G System (5GS); Stage 2; (R16)," 3GPP, Tech. Rep. TS 23.502, September 2020.

[28] D. Pineda, R. Harrilal-Parchment, K. Akkaya, A. Ibrahim, and A. Perez-Pons, "Design and analysis of an open-source sdn-based 5g standalone testbed," in IEEE INFOCOM 2023-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, 2023, pp. 1–6.

[29] "Oqs-openssl," Feb 2022. [Online]. Available: https://github.com/open-quantum-safe/openssl/blob/OQS-OpenSSL_1_1_1-stable/README.md