Post-Quantum Key Exchange and ID Encryption Analyses for 5G Mobile Networking

Qaiser Khan, Sourav Purification, Sang-Yoon Chang
University of Colorado Colorado Springs, Colorado Springs, CO, 80918, USA
Email: {qkhan, spurific, schang2}@uccs.edu

***Abstract*—5G technology addresses user privacy concerns in cellular networking by encrypting subscriber identifier with elliptic curve-based encryption and then transmitting it as ciphertext known as Subscriber Concealed Identifier (SUCI). However, an adversary equipped with a quantum computer can break a discrete logarithm-based elliptic curve algorithm and the user privacy in 5G is at stake against quantum attacks. In this paper, we study the incorporation of the post-quantum ciphers in the SUCI calculation both at the user equipment and at the core network, which involves the shared key exchange and then using the resulting key for the ID encryption. We experiment on different hardware platforms to analyze the PQC key exchange and encryption using NIST-standardized CRYSTALS-Kyber. Our analyses focus on the performances and compare the Kyber-based key exchange and encryption with the current (pre-quantum) Elliptic-curve Diffie–Hellman (ECDH). The performance analyses are critical because mobile networking involves resource-limited and battery-operating mobile devices. We measure and analyze not only the time and CPU-processing performances but also the energy and power performances. Our results show that Kyber-512 is the most efficient and even has better performance (i.e., faster computations, and lower energy consumption) than ECDH.**

***Index Terms*—Post-Quantum Cryptography, 5G, SUCI, Elliptic Curve Integrated Encrypted Scheme, CRYSTALS-Kyber**

I. Introduction

In telecommunications networking, user privacy is very important against tracking the behavior and the location of the mobile user. Before 5G, there has been no ID confidentiality protection. The ID of the user equipment (UE) was transmitted in plaintext before 5G, including 4G and any of the previous cellular generations. Because cellular connections in wireless rely on open-air medium, the adversary can access, capture, and track the UE ID, e.g., IMSI catching [1], [2]. UE tracking can result in the breach of the location (where the mobile UE is located) and behavior (how and when it uses the connectivity) of the mobile UE and the human owner using the UE.

Telecommunications networking has evolved in security. The most recent generation in cellular technology, 5G introduced authentication and key agreement (AKA) that conceal the actual permanent identity of the user to enable the confidentiality protection of the user ID to protect the user privacy. In contrast, previous generations do not conceal user identities. While enabling other security objectives (confidentiality, integrity, authentication, and non-repudiation), 5G AKA includes key exchange and encryption. More specifically, 5G AKA uses the Elliptic Curve Integrated Encryption Scheme (ECIES) [3]. ECIES involves the public-key key exchange to establish the shared key, which in turn can be used for encryption for the confidentiality protection of the ID. The UE ID is called International Mobile Subscriber Identity (IMSI) in 4G and earlier generations. In 5G, this identifier has evolved to become the Subscriber Permanent Identity (SUPI). 5G optionally encrypts the SUPI plaintext to generate and transmit the Subscriber Concealed Identifier (SUCI), including the ciphertext/encryption output of SUPI/IMSI.

5G SUCI calculation (encrypting the SUPI plaintext) however is vulnerable against quantum-equipped adversary. Assuming quantum computing, Shor's algorithm [4] can break the mathematical problem (discrete-logarithm problem in this case) anchoring the cipher security and finding the key. Recent prototypes in quantum computing, e.g., IBM, and Google, continue to evolve to support greater qubits to realize such threats to break the current ciphers.

Because of the security concerns due to the emerging quantum computing, the National Institute of Standards and Technology (NIST) standardized Post-quantum cryptographic (PQC) algorithms. By late 2017, 69 out of 82 candidate ciphers were selected in the first round, meeting suitability criteria. In subsequent rounds, NIST revised the selection process based on analysis, feedback, and performance metrics, ultimately finalizing one public-key encryption/key encapsulation mechanism (PKE/KEM) and three digital signatures in July 2022. Thanks to NIST standardization of the post-quantum cipher, the commerce software and systems in the US have been mandated to apply and incorporate post-quantum ciphers by 2035 [5]. Among the standardized PQC, our work focuses on the PKE/KEM to use the public-key encryption to establish the symmetric key while protecting confidentiality; the only PQC standardized algorithm for PKE/KEM is CRYSTALS-Kyber. [6]

Because the UE in cellular communications is mobile, lightweight, and battery-operating, we analyze the performance impacts of the post-quantum transition for SUCI in 5G. Our performance analyses are more relevant to such mobile/5G networking than other applications with greater resources and looser performance requirements.

In this research work, we study the performance analyses of post-quantum cipher in the 5G AKA identification and registration. We build our scheme on the 3rd Generation Partnership Project (3GPP) standard 5G-AKA protocol [7]. While the current 3GPP-defined 5G-AKA protocol uses the pre-quantum elliptic curve algorithms that are vulnerable against

the quantum-equipped adversary (specifically, secp256r1 and X22519), we replace the pre-quantum ciphers and incorporate the post-quantum Kyber-512 [8] into 5G AKA and analyze the pre-quantum vs. post-quantum 5G AKA.

We compare the performances of the existing pre-quantum ciphers (the elliptic curve KEM ciphers of secp256r1 and X22519) with the post-quantum cipher (CRYSTALS-Kyber-512) when implemented in 5G AKA. For fairness, we compare the ciphers with the same security levels and fix the computing platform on which we simulate the 5G AKA. Because the mobile UE is resource-constrained, we compare the performances of the ciphers with the shortest key/parameter lengths. We implement and measure performance in processing time, power, energy, and CPU cycles. While the memory and byte overhead increases from pre-quantum to post-quantum, our results show that the post-quantum Kyber is more efficient in terms of time and energy than pre-quantum PKE/KEM.

The rest of the paper is organized as follows: Section II describes the motivation and background that highlight the rationale and context of the research. We discuss the related work in Section III. Section IV describes our implementation and experimentation procedure. Section V focuses on the experimentation analysis, presenting the results and observations. Section VI concludes the paper, summarizing the key findings, contributions, implications of the study, and future directions.

II. Motivation and Background

A. Motivation: PQC Transition

Public-key ciphers are vulnerable to quantum computers due to the challenges posed by prime factorization and discrete logarithms. In 1994, Peter Shor introduced quantum algorithms for prime factorization [4], while Grover's algorithm [9] significantly accelerates key searches. The combined impact of the Shor's and Grover,s algorithms compromises the security of current cryptographic systems, as demonstrated in Table I.

In October 2023, IBM announced the development of the latest Quantum computer with a processor containing over 1000 qubits [10]. Martin et al. [11] have shown that calculating the elliptic curve discrete logarithm on a quantum computer requires a considerably smaller number of Quantum Bits (qubits) compared to addressing the integer factorization problem. While breaking RSA-2048 would necessitate millions of qubits, which is still years away, the urgency for considering quantum attacks arises from several factors: compatibility of PQC with classical computers, the uncertainty surrounding the development of quantum computers globally, the security of PQC ciphers against both quantum and classical attacks, and the government agencies taking early steps to use PQC for better security.

The National Institute of Standards and Technology (NIST) introduced post-quantum cryptographic algorithms in 2016 to address the quantum resistance issues. In late 2017, 82 candidate ciphers were submitted for review, and in December of the same year, 69 algorithms were selected in the first round based on meeting minimum suitability criteria. By January 2019, NIST advanced 26 ciphers to the second round, based on internal analysis, theoretical security assessments, and public feedback. By July 2020, 15 ciphers were chosen for round 3 and categorized into finalized and alternative groups based on factors such as performance, computational cost, data transfer cost, and implementation feasibility.

TABLE I: Security level for pre- quantum vs post-quantum ciphers. Shor's Algorithm breaks the pre-quantum RSA and EC ciphers by finding the key in polynomial time.

Security level (bits)	Symmetric	RSA	EC	Post Quantum (NIST Round 4)
128	AES-128	3072 bits	256 bits	Kyber-512
192	AES-192	7680 bits	384 bits	Kyber-768
256	AES-256	15360 bits	512 bits	Kyber-1024

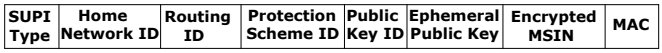


Fig. 1: Subscriber Concealed Identifier (SUCI) data fields

After a comprehensive evaluation across multiple rounds, NIST identified four candidate ciphers for potential standardization in July 2022. These include one PKE/KEM, namely CRYSTALS-Kyber, and three Digital Signatures (DS), namely CRYSTALS-Dilithium, FALCON, and SPHINCS+, all originating from the third round. PKE/KEM is employed to ensure message confidentiality, while DS is used to ensure message integrity.

NIST conducted a thorough analysis of the security strengths of post-quantum ciphers and compared the brute-force difficulty with pre-quantum ciphers. Table I outlines the pre-quantum security levels of symmetric (AES) and public-key ciphers, including RSA and elliptic curve (EC) algorithms, in 128-bit, 192-bit, and 256-bit security contexts. It contrasts these with the Kyber family of post-quantum cryptographic schemes, specifically Kyber-512, Kyber-768, and Kyber-1024. The table demonstrates that Kyber-512 offers equivalent 128-bit security as EC-256 (including algorithms like X25519 and secp256r1). Furthermore, it highlights the vulnerability of traditional public-key algorithms to quantum attacks, notably Shor's algorithm, whereas Kyber-512 remains secure against quantum threats due to its reliance on different mathematical hard problems.

We utilize NIST security-level analyses to ensure fairness in our comparison study, selecting ciphers of the same security levels for comparison purposes.

B. Background on 5G AKA and Cryptographic Primitives

In this paper, we use acronyms standardized by 3GPP for 5G telecommunication networks. Table II provides a list of the acronyms used throughout this paper.

1) 5G-AKA and SUCI: 5G Authentication and Key Agreement (5G-AKA) is a mutual authentication protocol in the 5G cellular network between the user equipment (UE) and the core network (CN). The UE is the beneficiary of the cellular service provision receiving the connectivity access while the CN is the cellular service provider network providing the connectivity access to the UE. The 5G-AKA protocol consists

TABLE II: Acronyms used in this research

AKA	Authentication and Key Agreement
CN	Core Network
ECIES	Elliptic Curve Integrated Encryption Scheme
IMSI	International Mobile Subscriber Identity
KEM	Key Encapsulation Mechanism
KDF	Key Derivation Function
MSIN	Mobile Subscriber Identity Number
PQC	Post Quantum Cryptography
SUCI	Subscriber Concealed Identifier
SUPI	Subscriber Permanent Identifier
UE	User Equipment
USIM	Universal Subscriber Identity Module

of two phases: the identification phase and the challenge-response phase. The identification phase starts with a registration request by the UE. Upon powering on, the UE initiates the registration request to the CN by sending its subscription identifier. This identifier is stored in an integrated chip called universal subscriber identity module (USIM). The USIM in the UE also stores a public key of the CN. In 4G, the IMSI identifier is sent in plaintext during authentication initialization.

In 5G, the identifier is known as SUPI which is sent in an encrypted format known as SUCI. The CN public key stored in the USIM encrypts the SUPI into SUCI. Fig. 1 shows the SUCI format. The SUCI contains a SUPI type indicator, a core network identifier, a protection scheme ID, a core network public key ID, and a protection scheme's output. The protection scheme's output contains a public key, message authentication code (MAC), and encrypted MSIN (Mobile Subscriber Identity Number), either SUPI or IMSI. When the UE sends the SUCI to the core network it decrypts the SUCI using its private key and identifies the subscriber successfully. The SUCI format protection scheme identifies the algorithm used for the encrypt and decrypt function.

Table III describes the protection schemes where the protection scheme ID is a numerical value ranging from 0 to 15. It signifies whether a protection scheme is null (0) or non-null (1-15) for the given SUCI value. When a SUPI value is provided, the Protection Scheme ID determines the corresponding scheme applied, resulting in an output scheme. In the null scheme, no encryption is applied to the provided SUPI. Instead, the output scheme consists of the MSIN component of the IMSI.

Our work focuses on the 5G protection schemes beyond the null scheme which uses encryption to protect the confidentiality of the UE ID. More specifically, we focus on the two schemes called Profile A and Profile B using distinct ciphers. In both Profiles A and B, ECIES generates the SUCI from a given SUPI value at the UE-side. Similarly, at the CN-side, ECIES is utilized to decipher the SUPI value from the received SUCI value. The key difference between Profile A and Profile B arises in the Elliptic Curve domain parameters, Elliptic Curve Diffie-Hellman (ECDH) primitive, and point compression. However, other elements remain consistent across both profiles. Profile A uses Curve25519 as its domain parameter and X25519 as an ECDH primitive while Profile B uses secp256r1 as its domain parameter and Elliptic Curve Cofactor Diffie-Hellman as ECDH Primitive. Additionally, Profile B implements point compression to reduce overhead, while Profile A does not apply this compression technique.

2) Pre-Quantum (X25519 and secp256r1): X25519 and secp256r1 (NIST P-256) are both elliptic curve cryptography (ECC) algorithms used for key exchange but they differ in terms of the elliptic curve they use, the underlying mathematics, and certain characteristics.

Although X25519 is not NIST-standardized, it is widely used and considered secure and is defined in RFC 7748 [12]. Curve25519 is designed to be fast and secure. It is a Montgomery curve, which provides certain performance advantages in implementations. X25519 uses the elliptic curve Curve25519, defined over a prime field defined by the prime number $2^{255} - 19$. X25519 is often faster in computation, which makes it particularly efficient for key exchange in various protocols. Secp256r1, also known as NIST P-256, is a NIST-recommended elliptic curve [13]. Secp256r1 is a Weierstrass curve and is widely implemented in cryptographic libraries, and security protocols, including TLS.

3) Post-Quantum, CRYSTALS-kyber: CRYSTALS-Kyber [8] is the only PKE/KEM cryptographic algorithm selected by NIST in the 4th round of the Post-Quantum Cryptography Standardization process. Kyber is specifically a post-quantum cryptographic algorithm, which means that it provides security against attacks from both classical and quantum computers. The mathematical hardness of the Kyber relies on problems believed to be hard even for quantum computers. One such problem is the hardness of solving certain instances of the Learning With Errors (LWE) problem over module lattices. These problems involve finding a secret vector given some noisy linear equations.

When the protection scheme ID is 1 then X25519 is used and when the ID is 2 secp256r1 is used. 3GPP also standardizes protection scheme ID 3$\sim$11 for future use, while 12$\sim$15 for the use of the proprietary home operator.

4) Key Encapsulation Mechanism (KEM): Because symmetric-key cryptography is more efficient than public-key cryptography, the general cryptographic practices use the public-key encryption (PKE) for key encapsulation mechanism (KEM) to share and establish the symmetric key (which key can then be used to encrypt the communications and messages). KEM/PKE solves the problem by securely transmitting the symmetric key while encrypting the symmetric key with public-key cryptography. Hence, KEM enables two communicating parties to derive a shared key by leveraging public-key cryptographic encryption. This section fills the gap between the SUCI calculation and our implementation design.

KEM consists of the following three algorithms. The details of KEM inclusion in SUCI calculation are described in Section IV-B.

KEM.KeyGen: A probabilistic key generation algorithm

TABLE III: Protection Schemes

Protection Scheme	Protection Scheme Identity	Algorithm	Output Scheme
Null Scheme	0	N/A	Size of MSIN (If use IMSI)
Profile A	1	ECIES (X25519)	256 bits-Public keys + 64 bits MAC + Cipher-text value
Profile B	2	ECIES (secp256r1)	256 bits-Public keys + 64 bits MAC + Cipher-text value
Reserved for future standardization	3-11	N/A	N/A
Reserved for proprietary (home operator)	12-15	N/A	N/A

KEM.KeyGen which generates a public-private key pair.
KEM.Encapsulation: A probabilistic encapsulation algorithm *KEM.Encapsulate* that takes the public key of the receiver as input. Then it selects a shared key from randomness, encrypts the shared key using the receiver public key, and outputs the ciphertext, which is also known as an encapsulated shared key.
KEM.Decapsulation: A deterministic decapsulation algorithm that takes as input a ciphertext (encapsulated shared key) and private key of the receiver to generate the shared key or report a failure.

III. Related Work

A. SUCI Protection in 5G

The introduction of encrypted subscriber identity in 5G comes from the fact that in previous generations (2G, 3G, 4G) technology, plaintext transmission of subscriber identity (e.g. IMSI) during user registration suffers from user privacy leakages. An attacker, with open-source hardware and software tools, can act as a man-in-the-middle between the UE and the base station to eavesdrop on the plaintext IMSI without the knowledge of the network operator or the user. The captured IMSI is then utilized to track the user location, and eavesdrop on user private conversations [1]. For example, Mjølsnes et al. [14] realized this attack (also known as *IMSI catcher*) in 4G/LTE network using software-defined radios (Ettus USRP B210) and open-source software tools (OpenBTS and Open Air Interface). While there are research works to reveal the presence of IMSI catchers [15]–[17], the 3GPP standard addresses the issue by introducing Subscription Concealed Identifier (SUCI) in 5G authentication and key agreement protocol [7].

In 5G, the user encrypts its identifier (IMSI/SUPI) using public-key cryptography turning into a concealed identifier (SUCI), and serves as a component of 5G-AKA during authentication and key agreement protocol. The 3GPP proposes two discrete logarithm-based Elliptic Curve Cryptography (ECC) algorithms to calculate the SUCI. The protection scheme identifier field in the SUCI format specifies the algorithms as X25519 and secp256r1 as discussed in Section II-B1.

B. Post-Quantum for 5G KEM

Among the research for studying the PQC incorporation to 5G networking, e.g., [18]–[22], the most closely related to our work is by Ulitzsch et al. [22] and Damir et al. [18]. Damir et al. [18] proposes a novel authentication and key agreement protocol based on a KEM for 5G where the user identity and forward secrecy are preserved. Their proposed protocol is compatible with post-quantum cryptography and analyzes the performance of the scheme using NIST round 4 finalists. Ulitzsch et al. [22] focuses more on the SUCI calculation of 5G-AKA. They propose a post-quantum secure SUCI calculation scheme called *KEMSUCI*. They also evaluate their framework with NIST round 3 finalists using a standard SIM card. However, both of the research identifies CRYSTAL-Kyber as the best suitable KEM amongst post-quantum algorithms. Our work builds on these previous works providing proof-of-concepts [18], [22] but further analyzes the performances to include the comparison between post-quantum (Kyber-512) vs. pre-quantum (X25519 and secp256r1) and to measure energy and power for mobile-resource-constrained applications. We measure the performance of three different algorithms (Kyber-512, X25519, and secp256r1) on three different platforms (Computer, MiniPC, and Raspberry Pi) while providing the time duration, energy consumption, and CPU cycle count analysis focusing on the UE.

IV. Implementation Design

We implement PQC in the 4G/5G context for empirical performance analyses and validations. This section describes our implementation scope, i.e., using the 5G information fields and simulating the participating entities and their SUCI protocol. Our implementation follows the standardized 5G protocol because we focus on the 5G application, and our implementation and experimentation focus on the KEM involving public-key cryptography (requiring the PQC transition for quantum resistance) and not on the symmetric cryptographic operations.

A. SUCI in Existing 5G

This section describes the SUCI calculation in existing 5G that builds on Section II-B1. In the current standard of 5G AKA two protection schemes X25519 and secp256r1 exist. Below are the steps for generating the protection scheme output fields of the SUCI for both of the ECDH algorithms. We refer to each relevant field in the SUCI format depicted in Fig. 1.

The CN generates elliptic curve (EC) public-private key pairs and shares the public key with the user by integrating it into the USIM. This is the core network public key discussed in Section II-B1. Further discussion on each of these steps follows below.
1. Public-Private Key Generation at UE: During the registration process, the UE generates the EC public-private key pair. The UE's public key is used by CN to de-concealed SUPI from the received SUCI while the UE's private key serves as input for the key agreement function in the subsequent step.

2. Key Agreement: The input to this function includes the CN's public key stored in the USIM and the UE's private key obtained from step 1, resulting in the generation of a shared key. In contrast to the use of ECDH in other contexts which exchange the public keys via digital networking, our key agreement makes use of the public keys shared during the UE registration, as described in Section II-B.
3. Encryption Key: The UE then uses a key derivation function on the shared key to generate an encryption key k_{enc} for the symmetric cipher, an initial counter block (ICB), and a k_{mac} for the message authentication code (MAC).
4. Symmetric Encryption: The UE uses the encryption key k_{enc} and the ICB to encrypt the plaintext (SUPI or IMSI) depending on the SUPI Type field in the SUCI format. Also, the UE generates the MAC field using k_{mac}. The output ciphertext value becomes the part of scheme output of the SUCI.

The UE then transmits the finalized output of the UE-side encryption process to the CN for registration and identification. This output consists of the concatenation of the UE's public key, ciphertext, and MAC.
5. Decryption: When the CN receives the registration request along with the SUCI, it uses UE's public key (from step 1) and CN's private key to generate the shared key at CN side. The shared key is further used to generate a decryption key k_{dec} same as k_{enc} to decrypt the SUCI, using a key derivation function.

In our experimentation, we implement steps 1, 2, and shared key part of step 5 for our performance analysis of existing SUCI calculation. Steps 3 and 4 use symmetric cryptography which is not in our research contribution.

B. Post-Quantum Cryptography for Subscriber Concealed Identifier (SUCI)

In this section, we describe the SUCI generation using post-quantum cryptography. Unlike Section IV-A the post-quantum cryptography algorithms use a KEM to generate a shared key. We describe the related background information for KEM in Section II-B4 and build on it. Although we use KEM, the information elements of the SUCI format are intact hence we omit redundant information about the fields in SUCI because it is already described in Section IV-A. The steps for generating protection scheme output for SUCI are as below:

The CN generates the public-private key pair using *KEM.KeyGen* function. The resulting CN's public key is stored in the USIM.
1. Public-Private Key Generation at UE: The UE generates the public-private key pair using *KEM.KeyGen* function.
2. Encapsulation (UE): To encrypt SUPI/IMSI the UE uses the function *KEM.Encapsulate* to generate initial randomness, derive the shared key from this randomness, and compute a ciphertext using CN's public key. The generated ciphertext is the encapsulated shared key.
3. Symmetric Encryption: After generating the shared key, the UE encrypts the plaintext (SUPI or IMSI) with a shared key that is computed using function *KEM.Encapsulate*.

The UE then sends the encapsulated shared key and encrypted data to CN.
4. Decapsulation: When the CN receives the registration request, CN uses its private key and encapsulated shared key to generate a shared key by using the decapsulation function *KEM.Decapsulate* and further use that generated shared key for decryption to verify the subscriber.

In our implementation, we use steps 1, 2, and 4 to measure the computational performance of post-quantum KEM. We also calculate the byte size of the public keys because the key lengths in post-quantum cryptography are much greater than the pre-quantum ciphers. Since the SUCI generation occurs at the USIM which is an integrated chip and has memory size limitations (e.g. 256KB), we show the byte size of the public keys in our comparison in Table IV.

V. Experimentation and Performance Analysis

We focus on and compare the post-quantum (Kyber-512) vs. pre-quantum (X25519 and secp256r1) ciphers which are relevant to mobile computing in 5G. We use the key-exchange ECDH utilized in the 3GPP standardized protocol; more specifically, X25519 and secp256r1 which are used in the 5G protection scheme for key exchange as discussed in Section II-B1. To provide a fair comparison, we compare the ciphers which correspond to 128-bit security.

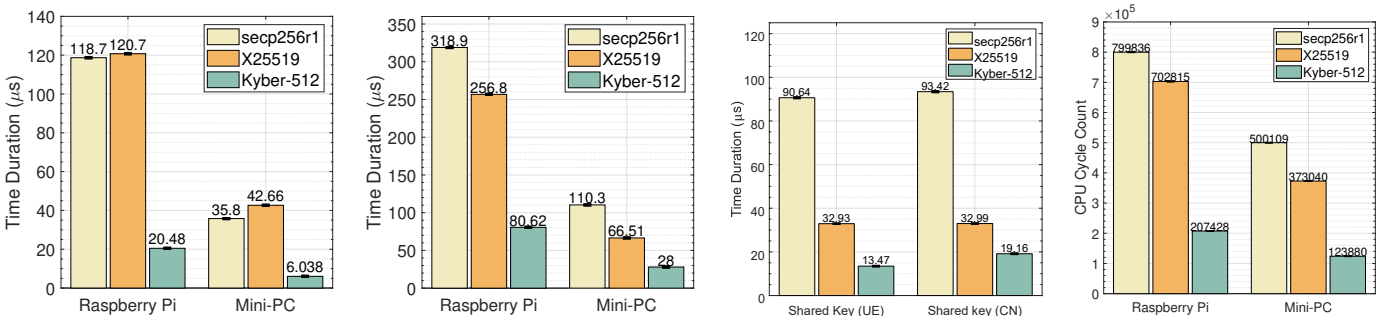
NIST's recommendations for cryptographic algorithms often include different security levels, typically ranging from 80-bits to 256-bits. NIST analyzed such security levels based on the brute-force difficulty, as described in Section II-A. Each level corresponds to a different expected level of security, with higher levels offering stronger protection but potentially requiring more computational resources. We focus on 128-bit security because the UE implementing and applying the ciphers are often mobile and resource-constrained.

This section provides our experimental setup, experimentation, and performance analyses. We measure the computational time in the experimentation and analyze the performance for 100,000 samples. We present average values with 95% confidence interval of the sample data in the analysis except power/energy analysis.

A. Hardware and Software Setup

We implement our system using a computer to simulate both user equipment and the core network. Additionally, we also perform our experimentation on resource-constrained, such as Mini-PC equipped with an Intel 12th N95 processor running at 3.4 GHz with 8 GB RAM, as well as a Raspberry Pi featuring a BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC clocked at 1.8 GHz with 4 GB RAM, to represent UE and implement all three ciphers on both devices for measuring the computational time of public key generation and shared key generation.

We perform our experimentation as implementation design described in Section IV. We follow steps 1, 2, and 5 of Section IV-A for pre-quantum ECIES and use open-source python library *cryptography* for the implementation. For the



(a) Public key generation time (UE)

(b) Shared key generation time (UE)

(c) Shared key generation time (Computer)

(d) CPU Cycles Count (Raspberry Pi and Mini-PC)

Fig. 2: Computational time performances (the left three figures) and CPU performances (Fig. 2d) on Computer, Mini-PC and Raspberry Pi. The average values and the 95% confidence intervals are shown, although they may not be too visible.

post-quantum algorithm, we follow steps 1, 2, and 4 of Section IV-B. We use the open-source Open Quantum Safe (oqs) library *liboqs* [23] to realize the steps. We write a Python script to run the experiments and collect the time measurement data. We measure only the computation time required by each algorithm in key pair generation, encapsulation/encryption, and decapsulation/decryption. The computational time gives us an indication of the energy required for each computation in user equipment.

For the power measurements, we use a power analyzer hardware (ZHURUI PR10-E US15A power meter plug) to capture power consumption during the execution of our cryptographic processes. Specifically, for pre-quantum ECIES, we measure power consumption during steps 1, 2 and 5 of Section IV-A. Meanwhile, for Kyber-512, we follow the procedures outlined in steps 1, 2, and 4 of Section IV-B. After collecting this data, we compute the average power consumption and multiply it by the time duration of the process to obtain the total energy consumed.

B. Computational Time Performance Analysis

As described in Section IV the UE performs public and shared key generation, encapsulation, and encryption processes before transmitting the data to the CN side for decapsulation and decryption. We evaluate and compare the computation times required for generating public keys and shared keys using pre-quantum ciphers (X25519 and secp256r1) and post-quantum cipher (Kyber-512) on two resource-constrained devices (Raspberry Pi and Mini-PC) at the UE side, as shown in Fig. 2a and Fig. 2b respectively. This is essential because the CN requires a more robust device, and the Mini-PC and Raspberry Pi are not suitable for serving as the primary CN device, due to limited computing resources such as processing power, memory, storage, and energy but the UE can be resource-constrained devices.

Fig. 2a shows the computation time for Public key generation on Raspberry Pi and Mini-PC because UE's public key is used by CN to de-concealed SUPI from the received SUCI. By using Raspberry Pi as a UE, the computational time comparison shows that, for Kyber-512, the public key generation requires 20.48μs, which is 82.88% and 82.67% less time consumption than X25519 and secp256r1 have 120.7μs and 118.7μs, respectively. For Mini-PC Kyber-512 consumes 6.038μs, X25519 takes 42.66μs and secp256r1 requires 35.8μs. Kyber-512 exhibits a significant advantage, taking 85.76% and 82.95% less time than X25519 and secp256r1, respectively. Fig. 2a also shows that the Raspberry Pi requires more time as compared to the Mini-PC for all three ciphers this is because of computing resources of Raspberry Pi are less than the Mini-PC.

Similarly, we also measure the computational time for shared key generation on Raspberry Pi and Mini-PC as shown in Fig. 2b because the shared key is used for encapsulation at the UE side and decapsulation at the CN side. The shared key generation time on Raspberry Pi is reduced significantly, standing at 68.57% and 74.69% less than X25519 and secp256r1, respectively. While the time required to generate a shared key on Mini-PC shows that the post-quantum cipher Kyber-512 takes 28μs, demonstrates efficiency, utilizing 57.90% and 74.66% less time than X25519 and secp256r1 having 66.51μs and 110.3μs, respectively.

We also measure the computation time required for shared key generation on the same device (computer) at both the UE and CN sides. Our results demonstrate that Kyber-512 exhibits less computational time at the UE side compared to the CN side, attributable to differing functions performed on the UE and CN, as shown in Fig. 2c. The share key generation for encapsulation/encryption at UE side while using Kyber-512, which utilizes 13.47μs, which utilizes 59.13% and 85.12% less time than X25519 (32.93μs) and secp256r1 (90.64μs) respectively. The shared key generation time at CN for decapsulation/decryption is illustrated in Fig. 2c. The Kyber-512 consume 19.16μs, that is 41.97% and 79.52% less time than X25519 (32.0μs) and secp256r1 (93.42μs) respectively.

Summary: The above results compare the computational time measurements of public key generation and shared key generation for encapsulation/decapsulation operations in the SUCI calculation process as described in Section IV. Post-quantum KEM, Kyber-512 demonstrates shorter processing times for

key generation compared to pre-quantum ciphers X25519 and secp256r1, both at UE and CN sides, across different platforms including computer, Mini-PCs, and Raspberry Pi, which can improve overall performance and responsiveness of mobile devices, crucial for tasks requiring secure communication and data transmission.

C. CPU Processor Analysis

We analyze CPU performance by measuring the CPU usage and the CPU cycle count. Because we run our experiments and the respective cryptographic operations repeatedly to take the statistical measurements while not running other processes, CPU usage is at 100%.

We also measure CPU cycles, which helps to select efficient cryptographic algorithms. Lower CPU cycle counts indicate more efficient algorithms, as they result in reduced processing time and energy consumption, which helps to conserve the battery life of the mobile device. In our experiment, we compute the cycle counts for pre-quantum ciphers (X25519 and secp25r1) described in steps 1, 2, and 5 of Section IV-A. Furthermore, we followed the procedures outlined in Steps 1, 2, and 4 of Section IV-B for the post-quantum cipher (Kyber-512) using both Mini-PC and Raspberry Pi. Our analysis involves computing the cycle count for $100,000$ samples and utilizing a 95% confidence interval to calculate the results.

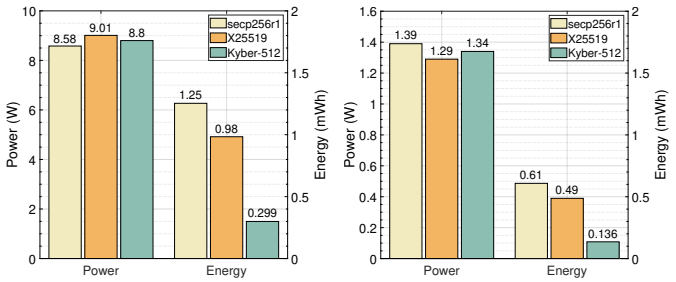
Fig. 2d shows CPU cycle count comparison of pre-quantum and post-quantum ciphers on both Mini-PC and Raspberry Pi. As shown in Fig. 2d, the Raspberry Pi has a low-performance processor and thus requires more CPU cycles to execute tasks than a Mini PC. Mini-PC has a higher performance processor for all ciphers than Raspberry Pi.

For Raspberry Pi, the secp256r1 takes an average of 799836 cycles during steps 1,2 and 5 of existing SUCI calculation as shown in Section. IV-A which is 73.34% higher than Kyber-512, which takes 207428 cycles for SUCI calculation while X25519 takes 702814 cycles. While in Mini-PC, secp256r1 again takes more cycles than X25519 and Kyber-512. secp256r1 takes an average of 500109 cycles which is 75.23% higher than Kyber-512 which takes 123880 cycles.

Summary: The analysis provides significant differences in cycle counts between pre-quantum (X25519 and secp256r1) and post-quantum (Kyber-512) ciphers on both the Raspberry Pi and Mini-PC. Kyber-512 stands out favorably in the CPU cycle comparison for mobile devices due to its relatively lower cycle count compared to other cryptographic ciphers like secp256r1 and X25519. This lower cycle count indicates that Kyber-512 requires fewer computational resources, making it more efficient and suitable for resource-constrained devices like mobile devices. Lower cycle counts enable Kyber-512 to execute cryptographic operations more efficiently, resulting in reduced processing time and low energy, which helps to conserve battery life, as discussed in Section V-D.

D. Computational Battery Performance Analysis

Because UE can be mobile and battery-operating, we measure the power and energy performance using two different



(a) Power and Energy using Raspberry Pi

(b) Power and Energy using Mini-PC

Fig. 3: Power (measured) and Energy (calculated) Consumption on Raspberry Pi and Mini-PC

lightweight platforms; the results using Raspberry Pi are shown in Fig. 3a and the results of Mini-PC are shown in Fig. 3b. We measure the power values in watts (W) and then derive and calculate the energy values in milliwatt-hour (mWh); the energy is a product of the power measurements in this section and the time measurements in Section V-B.

In Fig. 3a when simulating UE with a Raspberry Pi, the power consumption of each cipher is closely comparable, ranging from a minimum of 8.58 W for secp256r1 to a maximum of 9.01 W for X25519, with Kyber-512 falling in between at 8.8 W. However, when considering energy consumption the results show incomparable due to the cumulative computational time of public key generation and shared key generation. Kyber-512 proves to be more efficient, requiring only 0.299 mWh due to low computational time, which is notably 69.49% and 76.08% lower than the energy consumption of the X25519 and secp256r1, having 0.98 mWh and 1.25 mWh, respectively.

Similarly, for Mini-PC, the power consumption is comparable and the values are closely related having 1.39 W, 1.29 W, and 1.34 W for secp256r1, X25519, and Kyber-512 respectively. While the energy consumption shows variation, Kyber-512 shows better energy efficiency on Raspberry Pi, consuming only 0.136 mWh. This is significantly lower than the energy consumption of secp256r1 and X25519, which consume 0.61 mWh and 0.49 mWh, respectively, as described in Fig. 3b.

Additionally, Fig. 3a and Fig. 3b illustrate that power and energy consumption are higher on Raspberry Pi compared to Mini-PC, attributed to the lower computing resources of Raspberry Pi.

Summary: We aim to compare the power and energy performances of all three ciphers on resource-constrained devices. The power consumption among all three ciphers is closely comparable on both Mini-PC and Raspberry Pi while post-quantum KEM Kyber-512 demonstrates superior energy efficiency, consuming notably less energy compared to pre-quantum ciphers, secp256r1 and X25519 on both Mini-PC and Raspberry Pi platforms, helping to conserve battery life, thereby allowing users to use their devices for longer periods

TABLE IV: Protection Scheme Output Comparison

Protection Scheme	Ciphertext (Bytes)	Public Key (Bytes)	Scheme Output (Bytes)
Kyber-512	768	800	847
X25519	32	32	111
secp256r1	33	33	112

without needing to recharge.

E. Byte Overhead Analysis

We also compare the protection scheme output byte size analysis in our experimentation. The protection scheme output consists of an encrypted public key or cipher text, MAC code, and the encrypted SUPI. Table IV summarizes the protection scheme output among all three ciphers considering the 64-Bytes MAC size and 15-Bytes SUPI size. The Kyber-512 has a greater scheme output size than the ECIES ciphers because of the ciphertext output size (768 Bytes). Since the protection scheme output is transmitted to the CN from the UE, the communication networking overhead will increase significantly.

However, in our future work, we experiment with the impact of the increased Byte size in networking while prototyping our implementation using open-source software. In terms of storage requirement of the public keys in the USIM (refer to Section IV), the Kyber-512 consumes ($\frac{800}{32} = 25$) of magnitudes of memory size than the ECIES ciphers. The memory size of a standard USIM card is 256 KB [24]; hence, the USIM can store the public keys of the Kyber-512 algorithm and is feasible to use.

Summary: Table IV describe protection scheme output sizes among cryptographic ciphers, with Kyber-512 having a larger output due to its ciphertext size. While this may increase communication overhead it remains feasible for storage of public key within a standard USIM card (has 256 KB memory), facilitating its practical usage in mobile devices.

VI. Conclusion and Future Directions

In this research work, we study the performance analysis of post-quantum (Kyber-512, the only selected KEM cipher by NIST) and pre-quantum (X25519 and secp256r1, currently used for 5G SUCI) in the identification phase of 5G-AKA during registration. The UE calculates the subscription concealed identifier (SUCI) and sends it to the CN during the 5G-AKA protocol for registration. In this study, we show that the SUCI calculation, using the post-quantum Kyber-512 at the UE and CN, performs better than the pre-quantum ECIES ciphers. Focusing on the lightweight ciphers for UE in 5G, PQC KEM-and-encryption based on Kyber-512 performs better in computational time and energy consumption than the pre-quantum equivalent. Due to low energy and time consumption, Kyber-512 has positive effects including extended battery life, data transmission, optimized resource utilization, and enhanced overall performance.

While we focus specifically on the SUCI calculation operations (which utilizes the SIM-hardcoded core network's public key and occurs within the UE before the 5G networking) in this research, taking a systems approach to further analyze the rest of the 5G operations can go beyond the scope of this paper. Hands-on implementations of such 5G networking system, for example, using software-defined radio and 5G software of srsRAN and Open5GS, can validate the 5G integration and facilitate practicality. In addition to the validations, such implementation will enable measuring the networking byte size overhead, the communication overhead in terms of initiating the registration request, and the energy consumption of the user equipment within 5G networking communication session. To measure the scalability of the core network, we plan to measure the maximum number of subscriber identifications that the core network can do with the three ciphers (Kyber-512, X25519, and secp256r1) and compare the latency and throughput of network communication.

While our work simulated and emulated mobile devices using Raspberry Pi and Mini-PC, we can further use a jail-broken phone or an embedded software which are mobile-device friendly. For example, the wolfSSL embedded TLS library is a lightweight and portable SSL/TLS library. It is specifically designed for IoT, embedded, and real-time operating system (RTOS) environments, prioritizing small size, high speed, and comprehensive features. wolfSSL supports ECIES with elliptic curves defined by NIST which can be compromised by quantum computers.

While our research focuses on the current 5G New Radio technology (and shows that it supports the NIST-standardized, post-quantum Kyber), we expect this research to inform the future designs and incorporation of post-quantum ciphers in the next generations of mobile networking (beyond-5G and 6G). The post-quantum Kyber can have comparable or even better time/power performances than ECIES; however, it introduces additional byte overheads in storing the public key and in transmitting the ciphertext.

References

[1] D. Strobel, "Imsi catcher," *Chair for Communication Security, Ruhr-Universität Bochum*, vol. 14, 2007.

[2] F. Van Den Broek, R. Verdult, and J. De Ruiter, "Defeating imsi catchers," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 340–351.

[3] A. Koutsos, "The 5g-aka authentication protocol privacy," in *2019 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2019, pp. 464–479.

[4] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*. Ieee, 1994, pp. 124–134.

[5] The White House, "Promoting United States Leadership in Quantum Computing While Mitigating Risk to Vulnerable Cryptographic Systems," https://www.whitehouse.gov/briefing-room/statements-releases/2022/05/04/national-security-memorandum-on-promoting-united-states-leadership-in-quantum-computing-while-mitigating-risks-to-vulnerable-cryptographic-systems/, 2022, [Online; accessed 31-January-2024].

[6] NIST, "Post quantum cryptography," https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022, 2022, [Online; accessed 10-June-2024].

[7] 3GPP. TS 33.501 version 16.13.0 , "Security architecture and procedures for 5G System," 2023.

[8] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-kyber: a cca-secure module-lattice-based kem," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018, pp. 353–367.

[9] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.

[10] nature, "Work easier Work faster," https://www.nature.com/articles/d41586-023-03854-1, 2023, [Online; accessed 10-January-2023].

[11] V. G. Martínez, L. H. Encinas *et al.*, "A comparison of the standardized versions of ecies," in *2010 Sixth International Conference on Information Assurance and Security*. IEEE, 2010, pp. 1–4.

[12] Internet Research Task Force (IRTF), "Elliptic Curves for Security," https://datatracker.ietf.org/doc/html/rfc7748, January 2016, [Online; accessed 29-January-2025].

[13] M. Adalier and A. Teknik, "Efficient and secure elliptic curve cryptography implementation of curve p-256," in *Workshop on elliptic curve cryptography standards*, vol. 66, no. 446. NIST, 2015, pp. 2014–2017.

[14] S. F. Mjølsnes and R. F. Olimid, "Easy 4g/lte imsi catchers for non-programmers," in *Computer Network Security: 7th International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security, MMM-ACNS 2017, Warsaw, Poland, August 28-30, 2017, Proceedings 7*. Springer, 2017, pp. 235–246.

[15] S. Park, A. Shaik, R. Borgaonkar, and J.-P. Seifert, "Anatomy of commercial imsi catchers and detectors," in *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*, ser. WPES'19. New York, NY, USA: Association for Computing Machinery, 2019, p. 74–86. [Online]. Available: https://doi-org.libproxy.uccs.edu/10.1145/3338498.3358649

[16] A. Dabrowski, N. Pianta, T. Klepp, M. Mulazzani, and E. Weippl, "Imsi-catch me if you can: Imsi-catcher-catchers," in *Proceedings of the 30th annual computer security applications Conference*, 2014, pp. 246–255.

[17] A. Dabrowski, G. Petzl, and E. R. Weippl, "The messenger shoots back: Network operator based imsi catcher detection," in *Research in Attacks, Intrusions, and Defenses: 19th International Symposium, RAID 2016, Paris, France, September 19-21, 2016, Proceedings 19*. Springer, 2016, pp. 279–302.

[18] M. T. Damir, T. Meskanen, S. Ramezanian, and V. Niemi, "A beyond-5g authentication and key agreement protocol," in *International Conference on Network and System Security*. Springer, 2022, pp. 249–264.

[19] 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*, 2019, pp. 285–285.

[20] D. Chawla and P. S. Mehra, "A roadmap from classical cryptography to post-quantum resistant cryptography for 5g-enabled iot: Challenges, opportunities and solutions," *Internet of Things*, p. 100950, 2023.

[21] M. Mehic, L. Michalek, E. Dervisevic, P. Burdiak, M. Plakalovic, J. Rozhon, N. Mahovac, F. Richter, E. Kaljic, F. Lauterbach *et al.*, "Quantum cryptography in 5g networks: A comprehensive overview," *IEEE Communications Surveys & Tutorials*, 2023.

[22] V. Q. Ulitzsch, S. Park, S. Marzougui, and J.-P. Seifert, "A post-quantum secure subscription concealed identifier for 6g," in *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2022, pp. 157–168.

[23] Open-Quantum-Safe, "liboqs version 0.9.2," https://github.com/open-quantum-safe/liboqs.git, 2024, [Online; accessed 26-February-2024].

[24] techredar, "SIM card storage memeory," https://www.techradar.com/sim-only/what-is-stored-on-a-sim-card, 2020, [Online; accessed 10-January-2024].