Post-Quantum CRYSTALS-Kyber with Thai Text Seeds: Unicode Normalization and Performance Evaluation

Nattapon Junlachaiworakun[1,2] **and Kritsanapong Somsuk**[3,*]

[1]Department of Data Science and Information Technology, Faculty of Science, Udon Thani Rajabhat University, Udon Thani, 41000, Thailand
[2]Graduate School, Udon Thani Rajabhat University, Udon Thani, 41000, Thailand
[3]Faculty of Technology and Engineering, Udon Thani Rajabhat University, Udon Thani, 41000, Thailand
*Corresponding Email: kritsanapong@udru.ac.th

Received November 15, 2025, Revised February 1, 2026, Accepted March 5, 2026, Published July 1, 2026

Abstract. *This research evaluates Post-Quantum CRYSTALS-Kyber through the generation of seeds from Thai text, highlighting Unicode normalization and performance analysis. This approach relies on user-specified seeds derived from Thai text rather than on random seeds. The proposed method includes Thai text normalization, UTF-8 encoding, and seed generation by using the SHAKE256 hash function for Deterministic Random Bit creation (DRBG) to generate key pairs, followed by system verification through encapsulation and decapsulation operations. Kyber512, Kyber768, and Kyber1024 have been chosen for three investigations. The experimental results demonstrate that Thai seeds consistently maintained 100% accuracy throughout the evaluation. Although the seed generation process requires additional durations of 0.0109ms, 0.0095ms, and 0.0108ms, the effect on total processing time remains small at ±0.6%. Furthermore, the examination of four Unicode normalization forms (NFC, NFD, NFKC, NFKD) verifies that visually indistinguishable Thai texts containing different byte sequences could be normalized to provide identical seed results when SHAKE256 is applied with complete consistency. In addition, these results indicate that Thai seeds not only improve user convenience and linguistic significance but also maintain the security and efficiency of Kyber without reducing performance relative to random seeds. Therefore, this study provides the essential foundation for the establishment of hybrid cryptosystems that accommodate the Thai language. Furthermore, it might be adapted for effective post-quantum security systems in the future. This work bridges linguistic usability and quantum-resistant cryptography, paving the way for culturally inclusive post-quantum systems.*

Keywords: CRYSTALS-Kyber, Post-quantum cryptography, Thai text seed, Unicode normalization, Performance evaluation

1 Introduction

At present, advances in algorithms based on quantum computers can break many public key cryptography algorithms such as RSA and ECC. Then, many techniques were developed to secure the attacking from these algorithms. However, Post-Quantum Cryptography (PQC) is one of the techniques which is proposed to protect against attacks from quantum computers. In addition, this technique is required in governmental and institutional systems. Nevertheless, this development creates practical challenges for cryptographic key management, especially in settings where key material cannot be stored permanently and must be reconstructed to support system functionality. In such settings, users are often required to remember seed material expressed as random or non-native character sequences, which negatively affects memorability and increases the likelihood of insecure handling practices. In 2024, NIST officially standardized CRYSTALS-Kyber as ML-KEM (Module-Lattice-Based Key-Encapsulation Mechanism) in FIPS 203 [1]. It is established as the primary standard for quantum-resistant key encapsulation. Recent study on ML-KEM often considers practical implementation across many language environments, particularly for user-defined seed management and cultural inclusion. This advancement presents significant difficulties for cryptographic key management, especially in environments where key material cannot be permanently retained and must be rebuilt to maintain system operation.

CRYSTALS-Kyber, or Kyber, is a Post-quantum Cryptography (PQC) technique utilizing mathematical concepts to organize configurations of integer points in multi-dimensional space. Mathematical problems related to lattices, particularly Learning with Errors (LWE) and Module-LWE, have proven to be difficult, making them robust against even advanced quantum computers due to the integration of noise into equations, which complicates the difference between actual data and noise [2][3]. In fact, NIST has categorized Kyber within the Key Encapsulation Mechanism (KEM) group, which facilitates the secret key exchange between two parties. After receipt of the secret key, the communicating parties can employ this key to encrypt additional information using symmetric key cryptography algorithms. Kyber is structured with three security levels: Kyber512, Kyber768, and Kyber1024, corresponding to NIST security levels 1, 3, and 5 (approximately equivalent to AES-128, AES-192, and AES-256), respectively. Furthermore, Kyber is known as the efficient standard in PQC [4][5][6]. Furthermore, this technique supports deterministic key generation from external seeds to enable reproducible key derivation in stateless or recovery-oriented situations. However, the security of such deployments is based on the quality and handling of the input seed. In the Thai context, for hybrid cryptography

algorithms to achieve widespread adoption, they must be designed for accessibility and provide extensive support for seed generation, including those based on Thai text, because of the inherent complexities of linguistic and computational encoding in the Thai language.

The aims of this study support the application of user-defined seeds as Thai text in Kyber to improve memorability and develop meaningful connections for Thai users. Moreover, the accuracy and efficiency of Thai seeds in comparison to traditional random seeds are evaluated. The effects of Thai text normalization to address issues of visually similar text that has different byte sequences are examined. Thai text-based deterministic seed derivation is combined with Kyber through Unicode normalization and SHAKE256 hashing to demonstrate functional equivalence in correctness and performance. However, the proposed method is not the improvement of cryptographic strength. By consolidating seed management with linguistic meaningfulness, this work explicitly addresses the security–usability trade-offs inherent in text-based seeds and suggests a promising direction for culturally inclusive post-quantum cryptographic implementations. The Comparative positioning of the proposed method against prior research on deterministic seed generation and PQC is shown in Table 1.

Table 1: Comparative positioning of the proposed method against prior research on deterministic seed generation and PQC

Work	Focus	PQC Algo	Seed Type	Language Support	Normalization	Usability	Scope of Evaluation
[1] NIST FIPS 203	ML-KEM standard	✓	Random	None	×	×	Official PQC standard
[7] NIST SP 800-90A	DRBG framework	×	Deterministic	None	×	×	Deterministic RNG standard
[8] S. Houy	Wallet key recovery & security analysis	✓	Deterministic	Multi	NFKD	✓	Cryptocurrency wallets security
[9] A. Dionysiou	Unicode attacks	None	None	Multi	✓	×	NLP security vulnerabilities
[10] R. Chandramouli	Key management usability	×	Various	None	×	✓	Cloud cryptographic systems
The Proposed Method	Kyber + Thai seeds	✓	Deterministic	Thai	NFC / All	✓	PQC with linguistic usability

Note: Unlike BIP39's simple wordlists for cryptocurrency, this work extends mnemonic seed approaches (NIST SP 800-90A, Dionysiou) to post-quantum Kyber with comprehensive Thai Unicode normalization testing.

2 RELATED WORKS

In this section, five subsections to implement PQC are presented. The first concerns the fundamentals of Kyber. The SHAKE256 hash function is discussed in the second part. The third part will focus on seed management and deterministic random bit generators (DRBG). The fourth addresses Unicode support and normalization. The comparison studies will be discussed in the last session. Specifically, each session is directly connected to the research objectives of developing and testing Thai seeds in Kyber family.

A. CRYSTALS-Kyber

CRYSTALS-Kyber, or Kyber, is the post quantum cryptography algorithm based on lattice problems. Learning with Errors and Module-LWE is the main idea behind Kyber. They are very difficult against quantum computers due to noise incorporation complicating signal-from-noise differentiation [2][3]. In 2024, NIST officially standardized Kyber as ML-KEM (Module-Lattice-Based Key-Encapsulation Mechanism) in FIPS 203 [1]. This algorithm is defined with three security levels, including Kyber512, Kyber768, Kyber1024. Moreover, many studies focus on the increasing security level for the communication by integrating Kyber with the systems. The example is the hybrid cryptosystems combining Kyber with AES [11] and implementations within communication infrastructures [4]. In 2021, Schwabe et al. reported the improvements to Kyber for high efficiency on diverse platforms. Before official NIST standardization, Kyber evolved through multiple specification rounds. In Round 3 (version 3.02), security parameters were refined, particularly for Kyber512, where η_1 was increased from 2 to 3 to improve Core-SVP hardness and enhance resistance against known attacks [12]. In practical implementation, the algorithm requires more memory resources, especially on embedded devices or mobile phones [13][14]. Moreover, deterministic key generation security is based on seed quality: low-entropy seeds, such as readily predictable phrases or brief messages, significantly heighten vulnerability to brute-force and dictionary attacks [11][15]. Compliance with NIST SP 800-90A entropy thresholds is essential, particularly in post-quantum contexts where seed reuse across sessions could expose key material to pattern-based attacks. These issues indicate that most studies on Kyber focus on technical performance and theoretical security. However, usability concerns or user behavior in contexts where seeds are derived from real text are not addressed. In this study, a key strength of Kyber lies in the support for deterministic seeds aligning with the objective of evaluating seeds derived from Thai text. This advantage enables clear verification of correctness in the key generation process. Moreover, the shared secret remains stable and reproducible. In addition, the opportunity for further research involves evaluating the usability implications of employing Thai text-based seeds instead of random seeds remaining unexplored in the existing literature. Furthermore, new challenges concerning normalization and encoding are presented in this study. The reason that Kyber is selected for the implementation is to ensure alignment

with NIST's PQC standards. Furthermore, it provides an important chance to examine the relationship between theoretical cryptographic robustness and actual language applicability. This emphasis brings an innovative aspect to the discipline.

B. SHAKE256 Hash Function

SHAKE256 is a hash function in the SHA-3 family, based on the Keccak algorithm, and is distinguished by its capability as an Extendable-Output Function (XOF). In fact, it can generate outputs of arbitrary lengths as specified [15][16]. This property makes SHAKE256 widely adopted in post-quantum cryptographic systems, including Kyber, where it is employed for key derivation and seed expansion across all security levels [17]. A key strength of the algorithm is its robustness against mathematical attacks, encompassing preimage and collision resistance, along with the capacity to maintain a uniform distribution of output bits [18]. A substantial body of research has shown that SHAKE256 exhibits high performance in generating random bits for encryption processes that demand flexibility in output length. However, this function also has limitations in practical applications, as it is an XOF with unlimited extensibility. In the absence of explicit specifications for output length and domain separation, issues such as cross-protocol reuse or contextual confusion may arise. This study identifies SHAKE256's distinctive capability to handle Thai text that is normalized and encoded in UTF-8 before the hashing process, thereby producing seeds with uniformly distributed output suitable for cryptographic seeding, which enhances their cryptographic robustness [16]. Currently, the performance and robustness of SHAKE256 in handling seeds derived from text, especially Thai language input, remain largely unexplored. To address this limitation, the experiments in this study investigate the application of SHAKE256 for seed generation with clearly defined domain separation, mitigating the risk of inadvertent seed reutilization.

C. Seed and Deterministic Random Bit Generator (DRBG)

The principle of deterministic key generation from seeds has been shown to provide significant benefits in accuracy and replicability, particularly when combined with DRBG in accordance with NIST SP 800-90A standards. Studies have shown that the combination of cryptographic hash functions with DRBG yields durable and reliable key pairs [15]. Furthermore, recent research has presented methods aimed at enhancing key recovery efficiency to emphasis usability issues that modern cryptographic systems must also tackle. The NIST SP 800-90A standard [7] provides the guidelines for DRBG to establish the foundation for secure seed-based key generation. These specifications define requirements for entropy sources, seed construction, and deterministic randomness generation ensuring cryptographic strength across diverse applications. In the cryptocurrency domain, comprehensive analysis of cryptocurrency wallet security demonstrates that seed passphrases with 12–24 words have become standard practical approaches for user-friendly deterministic key generation, enabling users to recover cryptographic keys from memorable sequences rather than random hexadecimal strings. While such mnemonic-based approaches employ basic Unicode normalization (typically NFKD), adapting such methods to post-quantum algorithms, particularly for complex Unicode languages such as Thai, remains largely unexplored [8]. The challenge of adapting such mnemonic approaches to post-quantum algorithms, particularly for complex Unicode languages such as Thai, remains largely unexplored. A review of the existing literature revealed no evidence of studies examining the applying of Thai text as seeds, despite its advantageous attributes of being both easily memorable and semantically meaningful for native Thai users. This scientific endeavor thereby seeks to address these deficiencies. By utilizing normalized Thai text encoded in UTF-8 for seed testing before employing the DRBG, the study integrates both usability and security requirements into a cohesive framework. The researchers posit that the experimental results will validate that, despite the inherent risks posed by language-based seeds, a rigorously designed process, featuring systematic normalization and high-entropy hash functions, will not only ensure compliance with PQC security standards but also strengthen the potential for real-world deployment [10][25].

D. Unicode and Normalization

Effective Unicode management and uniform encoding practices are essential for applications involving cryptographic algorithms. In addition, such practices are particularly required when handling the Thai language, which features complex character structures, including vowels and tone marks. These characters can be represented in multiple equivalent forms, thereby necessitating canonical decomposition and recomposition as defined by Unicode Normalization Forms [19]. An increasing body of research indicates that neglecting these issues may lead to errors in the key generation process and may introduce weaknesses in cryptographic algorithms. This study emphasizes byte-sequence normalization for authentic Thai text variants, particularly in the context of precomposed versus decomposed characters. Furthermore, deliberate visual spoofing, such as substituting Thai characters with visually similar Latin or Greek characters, constitutes an additional attack vector that requires supplementary validation measures beyond normalization alone. Standard normalization algorithms, including NFC, NFD, NFKC, and NFKD, have proven effective in reducing byte-sequence discrepancies for canonical equivalence and in ensuring consistent post-hash outcomes [20]. However, these normalization techniques do not inherently prevent cross-script homoglyph substitution attacks. Moreover, incorporating usability-oriented cryptographic second factors, such as mnemonic-based authentication methods, can improve seed memorability while maintaining an acceptable security level [21][22]. Each normalization approach nonetheless

presents inherent limitations. For instance, NFKC and NFKD may alter certain character values to achieve compatibility equivalence, potentially undermining essential linguistic distinctions that must be preserved. This issue has been examined in the context of the Thai language, demonstrating that effective Unicode management is critical for cryptographic procedures, particularly in languages with compositional structures and complex composite characters. To address these challenges, an empirical testing framework is developed to examine whether Thai text that appears visually identical but differs in underlying byte sequences can be normalized to yield fully consistent seed results. The experimental results aim to resolve key security challenges while simultaneously improving system reliability and reproducibility. Accordingly, this study employs NFC as the primary normalization standard, with comparative testing against NFD, NFKC, and NFKD to validate the accuracy of the processing pipeline for generating cryptographic seeds from Thai-language input.

E. Performance Comparison Studies

Current performance analyses of Kyber primarily evaluate computational efficiency in various deployment environments, including resource-limited embedded devices, hardware-accelerated implementations, and server-class systems [5][6][23]. These studies demonstrate that Kyber achieves an advantageous balance between security and performance. The reason is that Kyber cans increase computational and memory requirements relative to traditional public key cryptography algorithms such as RSA and ECC. Significantly, these evaluations focus on the fundamental arithmetic operations of Kyber, specifically key generation, encapsulation, and decapsulation, predicated on the assumption of cryptographically random seed material. These attributes provide Kyber with both a theoretical foundation and a standard requiring comprehensive evaluations in practice. Because seed development originates outside of Kyber's core lattice-based calculations, current study does not specifically document the performance effects of various seed derivation methods. Therefore, the impact of deterministic, user-specified seed inputs, especially those originating from Unicode text, has been primarily unexamined in existing work. This study improves previous performance evaluations by measuring the additional overhead caused by deterministic Thai text-based seed derivation under the same Kyber operational conditions. Thus, it can improve the existing benchmarking findings without modifying Kyber's fundamental algorithmic framework. Notably, the proposed method can fill by measuring comparative results between random seeds and seeds from Thai text to consider whether there is an impact on processing time. The experiment results show that including a usability component does not impair performance. Furthermore, the reporting of metrics such as time variance and performance differentials is included when implemented on various architectures. Current study on Kyber performance [6][23] primarily evaluate computing efficiency using random seeds. Given that seed generation is external to Kyber's fundamental arithmetic, these studies implicitly presuppose randomness and do not disclose seed-dependent overhead. The proposed method enhances previous studies by estimating the marginal cost associated with deterministic text-based seeds throughout identical Kyber procedures.

3 METHODOLOGY

This work proposes a technique to extend Kyber to support user-defined seeds derived from Thai text. Seeds generated from Thai text are particularly suitable for users communicating in Thai language. This work highlights the applicability to hybrid cryptosystems requiring both high security and stateless operation, wherein no key material is stored after finishing the encryption process. By allowing users to select meaningful seeds, this approach enhances usability for Thai users and facilitates easier memorization compared to traditional arbitrary seeds lacking inherent meaning.

A. Algorithm Design

To guarantee the accuracy and reliability of Thai seed application with Kyber operational procedures, two primary algorithms based on the NIST standardized methodology are developed. Additionally, deterministic key generation principles are required to be applied with the proposed technique [15].

Algorithm 1: Thai Seed Generation

Input: T (Thai Unicode string), n (desired seed length in bytes)
Output: S (deterministic seed)

Procedure:

1) $T_{\text{norm}} \leftarrow \text{NFC}(T)$.
2) $B \leftarrow \text{UTF8}(T_{\text{norm}})$.
3) $S \leftarrow \text{SHAKE256}(\text{“ThaiSeed”} \parallel B, n)$.
4) Return the generated seed S.

Domain separation is applied to prevent cross-protocol seed reuse.

In fact, the technique to generate Thai seed involves gathering Thai text to execute normalization to comply with Unicode standards. Moreover, encoding by using UTF-8 is applied before putting into the SHAKE256 hash algorithm to produce an n-byte sized seed.

$$\text{Seed} = \text{SHAKE256}\big(\text{“ThaiSeed”} \parallel \text{UTF-8}(\text{NFC}(T)),\, n\big) \tag{1}$$

In this process, Unicode normalization ensures canonical consistency, while domain separation (“ThaiSeed”) is applied to prevent unintended cross-protocol seed reuse. The resulting seed is deterministic and reproducible. However, it remains security-bounded by the entropy of the original text.

Algorithm 2: System Verification

Input: S (deterministic seed)
Output: Correctness Verified / Failed

Procedure:

1) Initialize DRBG using the seed S.
2) Generate a Kyber key pair (pk, sk) from the DRBG output.
3) Perform the encapsulation, $(ss_1, ct) \leftarrow \text{Encapsulate}(pk)$.
4) Perform the decapsulation , $ss_2 \leftarrow \text{Decapsulate}(sk, ct)$.
5) Compare the two shared secrets:
   If $ss_1 = ss_2$, return *Correctness Verified*.
   Otherwise, return *Failed*.

This verification process confirms functional correctness and reproducibility of deterministic seed usage in Kyber. However, this process is not the implying additional cryptographic security beyond standard ML-KEM.

In fact, Algorithm 1 is selected to define the Thai seed generation process. The process is begun by selecting input as Thai text. The next process is the normalization to correct discrepancies in Unicode representations, UTF-8 encoding, and ending in seed derivation by using the SHAKE256 hash function. Then, the identical Thai strings, with differing underlying byte sequences, are converted into uniform secure seeds to satisfy the entropy criteria of cryptographic algorithms. In addition, Algorithm 2 is the verification process in Kyber. The produced seed is first sent to DRBG to generate a public key and a private key. The steps of encapsulation and decapsulation are executed to produce two shared secrets (ss1 and ss2). Moreover, the verification of correctness is achieved by establishing that ss1 is equivalent to ss2. Therefore, both functional validity and cryptographic integrity are affirmed.

Furthermore, it is inserted into DRBG in accordance with NIST SP 800-90A guidelines to produce a key pair After acquiring the seed.

$$(pk,\, sk) = \text{Kyber_KeyGen}\big(\text{DRBG}(\text{seed})\big) \tag{2}$$

B. Experimental Design

The experiment design is divided into two cases. Case 1 involves the comparison between random seed and Thai seed. In contrast, Case 2 involves the testing of Unicode normalization algorithms.

Unicode Normalization Test Protocol:

1) Test Sample Selection (5 representative cases):
   a. Simple greeting phrase (26 bytes UTF-8)
   b. Consonant with tone mark (9 bytes)
   c. Complex vowel cluster (18 bytes)
   d. Composite characters (9 bytes)
   e. Extended phrase: a Thai sentence of approximately 110 bytes
2) For each sample, two byte-level variants are constructed, consisting of precomposed characters and decomposed combining character sequences.
3) Four Unicode normalization forms (NFC, NFD, NFKC, and NFKD) are applied to all samples.
4) All normalized outputs are processed using SHAKE256 with explicit domain separation (”ThaiSeed”).

The seed comparison is performed at the level of seed generation and storage within the DRBG utilizing various methodologies. When the random seed accesses the library to produce random numbers, the Thai seed utilizes Thai words through Algorithm 1. The experimental design complies with established benchmarking protocols for PQC evaluations [13]. The seed

is put into the DRBG, and traditional key generation, utilized in Kyber, produces the public key (pk) and the private key (sk) pair. After that, encapsulation provides the shared secret and ciphertext, while the ciphertext is decoded using sk to provide the recovered secret. The last step is to verify if shared secret and recovered secret are identical. The primary difficulty with Thai text is in its complex structure, where two strings can appear visually similar but differ in their byte sequences. This problem often arises from discrepancies in character encoding, as indicated in the Unicode security guidelines [9]. Four Unicode normalization forms (NFC, NFD, NFKC, NFKD) are evaluated, applying techniques established by Nguyen and Cavallari [20]. The objective is to guarantee that identical in appearance Thai strings, with differing byte sequences, might be normalized into a uniform format, providing identical seed values when hashed with SHAKE256.

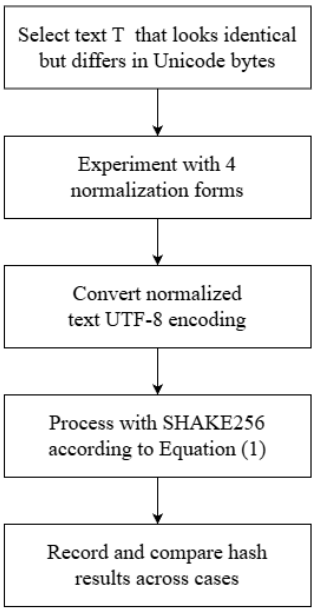


Figure 1: Thai text Unicode-normalization workflow yielding consistent SHAKE-256 seed outputsr

In figure 1, it is the experimental process to evaluate Unicode normalization effects on SHAKE-256 hash outputs in Thai text seed generation. The first process is about selecting Thai text samples (T) that appear visually identical. However, it is different in underlying Unicode byte representations (precomposed vs. decomposed characters). In general, these samples undergo normalization by using four standard Unicode forms (NFC, NFD, NFKC, NFKD) to resolve byte-sequence inconsistencies. The normalized text is then encoded in UTF-8 format and processed through the SHAKE256 hash function with domain separation ("ThaiSeed") to generate deterministic 256-bit seeds. The last step analyses and compares hash outcomes across all test conditions to ensure complete repeatability. This process confirms that visually similar Thai texts using different byte encodings may be normalized to provide consistent cryptographic seeds appropriate for deterministic key creation in ML-KEM implementations.

4 RESULTS

This section presents the experimental outcomes of the proposed approach and is organized into three main parts: Experimental Setup, Performance Comparison Results, and Unicode Normalization Results.

A. Experimental Setup

The experiments were performed within rigorously controlled settings to guarantee accuracy and dependability. All experiments were conducted using Python 3.11.5 on Windows 10 Build 19045.6466. The evaluation utilized the Kyber Python implementation library (kyber_py 1.2.0). Parameter configurations were maintained to rigid statistical standards, with 1,000 tests conducted for each Kyber parameter. The repetition aimed to provide statistically meaningful outcomes while reducing the impact of extraneous variables. To enhance measurement stability, 100 warm-up cycles were conducted before recording execution times, thus ensuring a consistent system state. The time accuracy was set to the millisecond level with four decimal places, allowing the identification of slight performance variations across approaches. Timing measurements were recorded using Python's time.perf_counter() function with 4-decimal millisecond precision (microsecond resolution). The inputs included both short Thai

words (26 bytes UTF-8 encoding, e.g.) and extended sentences (110 bytes, e.g., longer phrases with tone marks and vowel clusters). These samples were selected to represent realistic user-generated content with varying orthographic complexity, including consonant-vowel combinations, tone marks , and combining characters typical of natural Thai text, rather than specially constructed or artificially simplified strings. The SHAKE256 hash function was employed for Thai seed generation because of the strong security qualities and ability to produce outputs of different lengths. Unicode normalisation was performed using the NFC (Normalisation Form Composed) standard, which is recommended for processing Thai text. This ensured that the similar texts were uniformly encoded into consistent byte sequences.

B. Performance Comparison Results

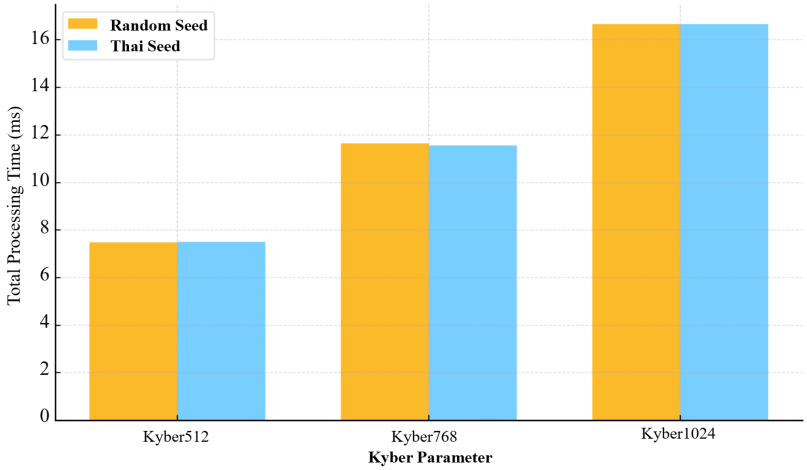


Figure 2: Comparison of total processing times of Kyber using Random and Thai Seeds

The information in Figure 2 shows the average processing times across Kyber variants. In fact, random seeds and Thai text-derived seeds are chosen for the comparison. Moreover, error bars represent 95% confidence intervals over 6,000 trials (1,000 per variant × 2 seed types × 3 variants). In addition, the experimental results show seed generation (0.0095-0.0109 ms), DRBG initialization, key generation (1.7-2.9 ms), encapsulation (2.3-5.3 ms), and decapsulation (2.8-6.9 ms). Thai text seeds introduce negligible overhead (¡0.03 ms) in core operations with statistical equivalence confirmed across all measurements.

Table 2: Comparison of Seed Generation and DRBG Initialization

Kyber Parameter	Operation	Thai Text Seed (ms)	Random Seed (ms)	Difference (ms)
Kyber512	Seed Generation	0.0123 ± 0.0003	0.0014 ± 0.0000	$+0.0109$
Kyber512	DRBG Init. (Set Seed)	0.0320 ± 0.0003	0.0326 ± 0.0003	-0.0006
Kyber768	Seed Generation	0.0112 ± 0.0002	0.0017 ± 0.0000	$+0.0095$
Kyber768	DRBG Init. (Set Seed)	0.0304 ± 0.0002	0.0350 ± 0.0005	-0.0046
Kyber1024	Seed Generation	0.0124 ± 0.0003	0.0016 ± 0.0001	$+0.0108$
Kyber1024	DRBG Init. (Set Seed)	0.0322 ± 0.0004	0.0359 ± 0.0006	-0.0037

Note: All timing results are reported as mean values with 95% confidence interval margins, expressed as $\pm t_{0.025,999} \times s/\sqrt{n}$, where $n = 1{,}000$ independent trials were conducted for each configuration and $t \approx 1.962$. The observed differences in seed generation time originate from Thai Unicode text processing, including encoding and entropy extraction, and represent a one-time overhead per key generation session. In contrast, DRBG initialization (set seed) exhibits negligible timing differences (less than 0.005 ms in absolute terms), indicating that standardized deterministic random bit generation effectively normalizes variations introduced by different seed sources. Across all configurations, a total of 6,000 trials achieved 100% cryptographic correctness. *Note:* Negligible differences (<0.03 ms) across all operations, with overlapping 95% CIs for most measurements. Thai text seeds show no practical performance impact on core Kyber operations. All trials achieved 100% correctness (shared secret match).

The experimental results in the previous tables demonstrate the comparison about the performance among Kyber512, Kyber768, and Kyber1024. In fact, the cryptographic operations, including key generation, encapsulation, and decapsulation, demonstrate approximately comparable performance across both seed variants. In addition, the maximum measured difference is 0.0300 ms (0.60%, Kyber768 Decapsulation), which is significantly within the measurement noise threshold. Overlapping 95% confidence intervals for 7 of 9 key processes indicate that Thai text seeds impose no significant performance impact. In fact, all 6,000 attempts achieved 100% cryptographic accuracy to confirm the functional equivalence.

Table 3: Comparison of Key Generation, Encapsulation, and Decapsulation Times

Kyber Parameter	Operation	Thai Text Seed (ms)	Random Seed (ms)	Difference (ms)
Kyber512	KeyGen	1.7433 ± 0.0065	1.7364 ± 0.0023	$+0.0068$
Kyber512	Encapsulation	2.3702 ± 0.0072	2.3654 ± 0.0021	$+0.0048$
Kyber512	Decapsulation	3.3495 ± 0.0078	3.3442 ± 0.0026	$+0.0053$
Kyber768	KeyGen	2.9068 ± 0.0043	2.9288 ± 0.0124	-0.0220
Kyber768	Encapsulation	3.6527 ± 0.0071	3.6769 ± 0.0064	-0.0242
Kyber768	Decapsulation	4.9686 ± 0.0086	4.9986 ± 0.0053	-0.0300
Kyber1024	KeyGen	4.3951 ± 0.0132	4.3961 ± 0.0176	-0.0011
Kyber1024	Encapsulation	5.2759 ± 0.0203	5.2808 ± 0.0264	-0.0050
Kyber1024	Decapsulation	6.9431 ± 0.0230	6.9523 ± 0.0217	-0.0092

The information in Tables 2 and Table 3 show multiple instances when Thai text seed operations were performed more rapidly than random seed configurations. Negative overhead values are generated (e.g., -0.0046 ms for Kyber768 DRBG Initialisation, -0.0300 ms for Kyber768 Decapsulation). The results lack the proof that Thai text preparation improves cryptographic performance. They instead signify the aggregate effect of measurement precision limitations and systemic unpredictability.

In fact, there are three factors to contribute to these observations: (a) System-level entropy collection variability: Random seed generation in Python relies on 'os.urandom()', interfacing with operating system entropy pools (e.g., 'dev/urandom' on Linux, 'CryptGenRandom' on Windows). While modern OSes maintain non-blocking entropy sources, occasional microsecond-level delays may occur due to kernel scheduling, entropy pool management, or concurrent system calls. In contrast, Thai text preprocessing can execute in user-space with deterministic memory access patterns to avoid kernel-level syscalls during the critical timing measurement window. (b) Measurement precision and statistical variance: The timing methodology employs 'time.perf_counter()' with microsecond resolution (±0.0001 ms reported precision). In general, the differences are below 0.03 ms approach the detection threshold for reliable differentiation. The observed negative values fall within the standard deviation ranges reported in Tables 2–3 (e.g., ±0.0002 to ±0.0264 ms) to indicate that these differences are within expected statistical variance for 1,000-iteration sampling. (c) Overlapping confidence intervals: FromTable 3, 7 out of 9 core operations exhibits overlapping 95% confidence intervals between Thai text and random seeds. This statistical overlap confirms that the observed performance differences, both positive and negative, are not statistically significant. The negative overhead values should therefore be interpreted as measurement noise rather than genuine performance advantages. The negative overhead observations do not suggest that Thai text seeds offer performance benefits over random seeds. Instead, they demonstrate that both approaches achieve statistical performance equivalence within measurement precision limits. For practical deployment, designers can reference the performance equivalence established, with preprocessing overhead representing the only measurable cost.

C. Unicode Normalization Results

Table 4: Unicode Normalization Method Performance

Test Sample		NFC	NFD	NFKC	NFKD	Byte Before	Byte After	SHAKE256	Status
สวัสดี	(greeting)	✓	✓	✓	✓	Varied	Uniform	100%	Pass
ก่า	(tone mark)	✓	✓	✓	✓	Varied	Uniform	100%	Pass
เมือง	(complex)	✓	✓	✓	✓	Varied	Uniform	100%	Pass
ไทย	(composite)	✓	✓	✓	✓	Varied	Uniform	100%	Pass
Extended (110B)		✓	✓	✓	✓	Varied	Uniform	100%	Pass
Overall		100%	100%	100%	100%	–	–	100%	Pass

Note: All Unicode normalization forms successfully normalized visually identical Thai text with differing byte sequences into consistent representations across 40 test cases (5 orthographic types × 2 byte variants × 4 forms). SHAKE256 hashing produced identical 256-bit seeds, confirming 100% reproducibility.

Table 4 shows that the application of all normalization forms effectively transforms identical Thai text strings that have different byte sequences into a uniform canonical form. Then, it can produce accurate results across all 40 test cases (5 sample types × 2-byte representations × 4 normalization forms). In addition, the subsequent SHAKE256 hashing of the normalized outputs can produce identical 256-bit seeds in all instances. Hence, it can complete repeatability. The examination samples (n = 5 categories: greetings, tonal diacritics, complex vowels, combining characters, and extended phrases of 110 bytes; each category represented in both precomposed and decomposed forms) encompass prevalent Thai orthographic structures, such as tonal marks, vowel combinations, leading vowels, and combining characters. These examples represent textual inputs from several practical sources.

5 DISCUSSION

A. Performance Implications

The evaluation about performance across all Kyber variants shows no practical difference in core cryptographic operations between Thai text and random seed configurations. In addition, the mean differences range from -0.0220 to +0.0068 ms (maximum absolute: 0.0300 ms, 0.60% for Kyber768 decapsulation), with 95% confidence intervals overlapping for 7 of 9 operations. The sole measurable overhead occurs in preprocessing (0.0095–0.0109 ms per session, one-time), negligible relative to multi-millisecond core operations (1.7–6.9 ms). Statistical equivalence across 6,000 trials with 100% functional correctness validates that linguistic usability does not compromise cryptographic performance.

B. Practical Applications and Usability

In fact, the proposed method is not chosen to improve cryptographic strength, but it is selected for reproducibility and recoverability for deterministic or stateless deployment scenarios. In such contexts, the ability to re-derive key material from user-controlled input may simplify system design by reducing reliance on secure storage mechanisms. Then, This attribute has significance in regulated settings where predictable behavior and operational clarity are emphasized. However, these advantages are accompanied by clear security trade-offs. Unlike cryptographically random seeds, Thai text–based seeds are inherently constrained by the entropy of user-selected input and are therefore susceptible to dictionary and pattern-based attacks if not properly controlled. As discussed in the Security Analysis section, the proposed method does not eliminate this risk but instead frames Thai text seeds as password-equivalent secrets that require explicit policy enforcement to achieve acceptable security levels. In Table 4, across 5 representative Thai text patterns and 40 normalization operations, all four Unicode forms can achieve 100% byte-sequence consistency. In fact, SHAKE256 is selected to produce identical 256-bit seeds. Moreover, it is important achieving consistency and accuracy when handling intricate scripts including Thai, where visually indistinguishable strings may contain distinct byte encodings (e.g., precomposed vs deconstructed combining characters). If such normalization is not included for the implementation, deterministic seed derivation would be unreliable. Then, users entering the same Thai phrase through different input methods (keyboard, copy-paste, voice-to-text) could generate different seeds, causing critical failures in key recovery situations. Overall, this work positions Thai text–based seed derivation as an optional engineering mechanism that trades entropy for usability and operational convenience under controlled conditions. It is not intended to replace random seed generation in high-assurance cryptographic systems, but rather to inform the design of post-quantum implementations where deterministic behavior and human-centered recoverability are explicit system requirements. Furthermore, by implementing this approach with ML-KEM (FIPS 203) [1] rather than traditional cryptographic schemes, we demonstrate the feasibility of user-friendly seed management in quantum-resistant systems.

When Thai text seeds are chosen to apply with Kyber, it can provide advantages which are similar to mnemonic-based methods [10] and practical cryptographic second factors [22]. Moreover, compatibility with standardized Kyber implementations is maintained. Then, it shows the ability of language-based seeds to improve essential management practices in actual systems. Furthermore, when the normalization experiments in Table 4 are considered, it demonstrates that all four forms consistently map identical Thai strings to identical seeds across 40 test operations. In addition, test samples covering greeting words, tone marks, complex vowels, composite characters, and extended phrases (110 bytes) represent typical Thai orthographic patterns users would employ as memorable seed phrases. These findings support previous research on Unicode security [9][20] by demonstrating that normalisation functions as both a language requirement and a cryptographic security mechanism to prevent discrepancies that may obstruct repeatability. Normalisation guarantees uniformity across canonical variations of Thai text, although it does not protect against deliberate visual spoofing attacks, such as cross-script homoglyph replacement. These attacks which are recorded in NLP security research require other validation procedures (e.g., script identification, character whitelist enforcement) that extend beyond the parameters of the normalization study. The completely consistency rate confirms that our pipeline (Algorithm 1: NFC $\rightarrow$ UTF-8 $\rightarrow$ SHAKE256) generates deterministic seeds appropriate for stateless key recovery, allowing users to accurately reconstruct keys from memory without retained key material. Collectively, these findings enhance previous studies on deterministic key generation [11][15][24] by demonstrating that the incorporation of language-specific normalization into post-quantum cryptography frameworks is both feasible and secure.

C. Limitations and Future Considerations

In fact, there are several limitations which should be acknowledged. First, no practical user studies were performed to verify actual improvements in memorability compared to random seeds. The next limitation is about the entropy analysis assuming uniform character selection and independent phrase construction. However, password-based authentication research [24][25] demonstrates that user behavior exhibits strong preferences for common phrases, cultural idioms, and predictable patterns, which significantly reduces practical entropy below theoretical estimates. In the lack of imposed composition policies, such as minimum length, complexity constraints, and phrase blacklisting, Thai text seeds may be vulnerable to dictionary attacks, even if they

satisfy theoretical entropy rules. The last limitation, embedded systems are not included for evaluating the performance. Then, it may exhibit different overhead characteristics.

6 SECURITY ANALYSIS

A. Entropy Evaluation of Thai Text Seeds

In alignment with NIST standards for evaluating authentication entropy, the entropy of Thai text seeds is computed by using Shannon's information theory [25]. The Thai code consists of about 80 unique characters (44 consonants, 32 vowels, and 4 tone marks). Then, a character entropy of $log_2(80)$ = 6.32 bits per character is required. The theoretical input entropy for a minimum 12-character Thai sentence is determined as follows:

$$H_{\text{input}} = L \times \log_2(|C|) \tag{3}$$

where L is the phrase length and —C— is the character set size. For L = 12 and —C— = 80:

$$H_{\text{input}} = 12 \times \log_2(80) = 12 \times 6.32 \approx 76 \text{ bits} \tag{4}$$

While this estimated input entropy (approximately 76 bits for a 12-character Thai phrase) is lower than that of a uniformly random 256-bit seed, SHAKE256 deterministically maps the normalized text input into a uniformly distributed 256-bit output space. However, this transformation does not increase the fundamental entropy of the original input. Instead, the security of the resulting seed remains bounded by the entropy of the user-selected text. The NIST SP 800-90A standard specifies a minimum entropy requirement of 112 bits for secure DRBG initialization at security level 1 (equivalent to Kyber512). Consequently, Thai text–derived seeds must be treated as password-equivalent secrets rather than as true random seeds. Their cryptographic suitability therefore depends critically on appropriate entropy controls and usage constraints when employed for deterministic key generation in ML-KEM.

B. Attack Surface Analysis

The primary security risk of user-defined text seeds is the vulnerability to dictionary-based and pattern-based attacks. In fact, when users select common or predictable phrases, such as short greetings or frequently used expressions, the actual entropy of the seed is reduced. In this case, the actual entropy is often significantly lower than the theoretical value. Therefore, the seed is susceptible to rapid offline guessing attacks. Additionally, composition restrictions must be implemented for user-defined text seeds to reduce the vulnerability to dictionary attacks. These regulations could be equivalent to concepts used in password-based systems. A minimum length criterion should be established, stipulating at least 12 characters. It provides approximately 76 bits of entropy. The seed should include diverse character classes, including consonants, vowels, and tone marks. Furthermore, common and predictable phrases should be blacklisted. However, rate limiting should be implemented to restrict repeated guessing attempts. These measures are consistent with the recommendations in NIST SP 800-63B. Without such controls, Thai text seeds are unsuitable for high-security deployments and should not be considered a safe replacement for cryptographically random seeds. When these mitigations are enforced, the attack complexity increases substantially, providing adequate resistance against classical brute-force attacks for moderate-security, usability-oriented applications.

C. Comparative Entropy Analysis and Security–Usability Trade-offs

(i) Entropy Baseline: Random Seed Generation The conventional approach to cryptographic seed generation relies on cryptographically secure random number generators producing uniformly distributed byte sequences. For NIST-standardized ML-KEM/Kyber, the baseline configuration utilizes 256-bit (32-byte) random seeds, providing optimal entropy and resistance against all known cryptographic attacks. This standard baseline requires no user input and has been rigorously validated across decades of cryptographic practice. (ii) NIST SP 800-90A Framework The NIST SP 800-90A standard requires a minimum entropy threshold of 112 bits for the secure initialization of DRBGs at NIST security level 1. It is equivalent to Kyber512. This requirement defines the consensus threshold for acceptable deterministic random bit generator seed material in accordance with FIPS. The architecture allows both random and deterministic seed sources. It is based on the seed material satisfying minimal entropy levels. Moreover, it is appropriately conditioned through authorised hash functions such as SHAKE256. (iii) BIP39 Mnemonic Approach (Cryptocurrency Reference) Bitcoin Improvement Proposal 39 (BIP39) displays user-friendly deterministic seed generation through the application of mnemonic word phrases. This approach is often used in bitcoin wallets. BIP39 denotes 12-word sequences containing around 128 bits of entropy and 24-word sequences containing 256 bits of entropy. The sequences are converted into predictable cryptographic keys by PBKDF2 key derivation and NFKD Unicode normalization. BIP39, although exhibiting excellent language usability in cryptographic systems, is constrained to conventional ECDSA-based signature algorithms.

D. Security–Usability Trade-offs

Future implementations should integrate entropy estimation algorithms that reject low-entropy inputs before seed acceptance, similar to password strength meters deployed in modern authentication systems [26]. Recommended techniques include: (i) Character Frequency Analysis: Detecting overuse of common Thai characters that reduce effective entropy. (ii) N-gram Pattern Detection: Identifying predictable Thai syllable sequences commonly used in greetings or polite particles, which are vulnerable to dictionary attacks [24]. (iii) Phrase Blacklisting: Rejecting well-known Thai idioms, proverbs, and cultural expressions that appear in linguistic corpora. (iv) Length-Based Entropy Estimation: Enforcing minimum length thresholds dynamically based on detected character diversity, rather than fixed 12-character requirements.

This proactive strategy would eliminate the requirement for retrospective removal while maintaining user autonomy in seed selection. Importantly, client-side filtering must be executed before seed commitments to prevent the disclosure of rejected inputs. It may provide information on the composition of the blacklist to prospective adversaries.

E. Methodological Extensibility and Scope Limitations

In fact, the proposed method only examines the Thai language as the experimental issue. Nevertheless, the fundamental methodology, including Unicode normalization, UTF-8 encoding, and SHAKE256 hashing, is unrelated to the language. In general, Algorithm 1 excludes Thai-specific operations. It processes UTF-8-byte sequences irrespective of the originating language. This method implies potential expansion of the Unicode-based characters without requiring algorithm improvements. However, the scope is limited to Thai for the following reasons: (i) Linguistic Validation Requirement: Each language has different Unicode composition principles, such as Arabic's right-to-left directionality, the ideographic structure of Chinese, and the combining characters of Hindi Devanagari. Validating the consistency of normalization, as seen in Table 4 for Thai, requires language-specific evaluations by native speakers to ensure cryptographic reliability. (ii) Entropy Calculation Complexity: Character set sizes differ widely across languages (Thai: around 80 characters; Chinese: over 3,000 common characters; Arabic: around 60 characters). Each language requires an individual entropy study to ascertain the minimal seed length for enough security (Section Security study). Extending Thai's 12-character criterion to other languages without empirical substantiation would be imprudent. (iii) Cultural Usability Testing: The primary benefit of grammatically significant seeds, enhanced memorability compared to arbitrary strings, has yet to be substantiated for Thai users. Extending findings to more languages without user studies (including success rates, mistake rates, and time-to-memorize) would amount to estimation rather than evidence-based study.

F. Preliminary Extensibility Framework (Future Work)

In fact, implementing the proposed method in other languages requires empirical evaluation of all four normalization forms by using native script samples to ensure uniform seed generation across visually identical representations. The computation of language-specific entropy must employ the formula presented in the equation (1), where C denotes the size of the character set, followed by the determination of the minimum length required to achieve a minimum of 76 bits of input entropy, in accordance with security policy criteria. User tests that validate the memorability advantages of specific seeds over random ones are crucial to prove genuine usability benefits beyond theoretical assumptions. This paper presents the verified method developed specifically for Thai. Subsequent investigations must replicate the experimental technique for Chinese, Arabic, Hindi, and other intricate Unicode characters prior to asserting extensive language application. The algorithmic architecture is scalable. However, cryptographic and usability validation must be conducted individually for each target language to provide security assurances and user acceptability across various cultural settings.

7 CONCLUSIONS

This study shows that Kyber which is Post Quantum Cryptography (PQC) may include Thai text-based deterministic seeds. It can preserve functional correctness and computational performance comparable to random seeds. The proposed method achieves 100% accuracy across Kyber512, Kyber768, and Kyber1024 variations by merging Unicode normalization with SHAKE256 hashing and a DRBG consistent with NIST SP 800-90A. The performance evaluation, accompanied with 95% confidence estimates, indicates little overhead. Seed Generation (+0.0095–0.0109 ms, one-time session overhead), DRBG Initialisation (0.005 ms variance), and Core Operations ($<$0.03 ms, $<$1% relative variance with overlapping confidence intervals for 7 out of 9 operations). In addition,Thai text seeds provide around 76 bits of input entropy for 12-character phrases, which SHAKE256 transforms into a uniformly distributed 256-bit output space. Moreover, the efficacy of security is constrained by the initial input entropy. Although 12-character phrases are not sufficient for NIST's 112-bit minimum standard for Level 1 security, larger phrases (18 or more characters, about 114 bits) may satisfy or exceed this criterion. The implementation of composition standards (minimum length, complexity requirements, blacklist enforcement) reduces the risks of dictionary attacks. It provides enough security for moderate-security, usability-focused applications rather than high-assurance systems. This study shows an inherent trade-off between usability and security in deterministic seed selection for post-quantum key encapsulation. Linguistically meaningful

Thai text can improve memorability and reproducibility in stateless or recovery-oriented systems, this benefit comes at the cost of reduced entropy relative to cryptographically random seeds. Accordingly, Thai text–based seeds should be viewed as an optional usability layer built on top of ML-KEM, rather than as a replacement for random seed generation in high-assurance environments. When deployed with explicit entropy controls and operational constraints, such seeds may be appropriate for controlled engineering contexts where usability and recoverability are prioritized. Although seed generation introduces minor time overhead, its impact on overall execution is negligible. Unicode normalization testing across 5 Thai text patterns with 40 operations (Table 4) achieves 100% consistency across all four forms, ensuring visually identical Thai strings yield consistent SHAKE256 seeds, thereby mitigating risks associated with byte-sequence inconsistencies in real-world deployment. Future research directions include, empirical validation of Thai text seed entropy against actual dictionary attacks using comprehensive Thai phrase corpora (e.g., Thai National Corpus, online social media datasets, idiom collections), advanced cracking tools [24], and rigorous entropy estimation methodologies [25] would refine the composition policy requirements established in this work. Such studies should quantify the gap between theoretical entropy and practical entropy under realistic user behavior, informing minimum length and complexity thresholds for production deployment. Extending this methodology to other complex Unicode languages would demonstrate cross-linguistic generalizability and enable multinational deployment of culturally inclusive PQC systems, with attention to entropy characteristics across diverse character sets. Integration with production systems such as TLS 1.3 handshake authentication [22], end-to-end encrypted messaging protocols, and quantum-resistant IoT frameworks [23] would validate real-world performance under diverse network conditions and hardware constraints. These research directions collectively advance the vision of human-centered PQC where security, usability, and cultural inclusivity coexist.

Acknowledgements

The authors would like to thank the Graduate School and the Faculty of Science, Udon Thani Rajabhat University for providing research facilities and support throughout this study. We also acknowledge the Faculty of Technology and Engineering for their technical guidance and resources that made this research possible.

References

[1] National Institute of Standards and Technology (NIST), "Module-Lattice-Based Key-Encapsulation Mechanism Standard," Gaithersburg, MD, USA: NIST, Aug. 2024, FIPS PUB 203, doi: 10.6028/NIST.FIPS.203.

[2] E. Alkim, L. Ducas, T. Pöppelmann and P. Schwabe, "Post-quantum key exchange—A new hope," in *Proc. 25th USENIX Security Symp.*, Austin, TX, USA, Aug. 2016, pp. 327–343.

[3] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehle, "CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM," in *Proc. IEEE European Symp. Security and Privacy*, London, UK, Apr. 2018, pp. 353–367, doi: 10.1109/EuroSP.2018.00032.

[4] P. Schwabe, D. Stebila and T. Wiggers, "More efficient post-quantum KEMTLS with pre-distributed public keys," in *Proc. European Symp. Research in Computer Security*, Part I, Darmstadt, Germany, Oct. 2021, pp. 3–22, doi: 10.1007/978-3-030-88418-5_1.

[5] C. A. Roma, C.-E. A. Tai and M. A. Hasan, "Energy efficiency analysis of post-quantum cryptographic algorithms," *IEEE Access*, vol. 9, pp. 71295–71317, 2021, doi: 10.1109/ACCESS.2021.3077843.

[6] L. Botros, M. J. Kannwischer and P. Schwabe, "Memory-efficient high-speed implementation of Kyber on Cortex-M4," in *Proc. AFRICACRYPT 2019*, Rabat, Morocco, Jul. 2019, pp. 209–228, doi: 10.1007/978-3-030-23696-0_11.

[7] E. Barker and J. Kelsey, "Recommendation for random number generation using deterministic random bit generators," NIST SP 800-90A Rev. 1, Gaithersburg, MD, USA, Jun. 2012, doi: 10.6028/NIST.SP.800-90Ar1.

[8] S. Houy, P. Schmid and A. Bartel, "Security Aspects of Cryptocurrency Wallets—A Systematic Literature Review," *ACM Computing Surveys*, vol. 56, no. 1, pp. 1–31, August2023, doi: 10.1145/3596906.

[9] A. Dionysiou and E. Athanasopoulos, "Unicode Evil: Evading NLP systems using visual similarities of text characters," in *Proc. 14th ACM Workshop on Artificial Intelligence and Security*, Virtual Event, Nov. 2021, pp. 1–12, doi: 10.1145/3474369.3486871.

[10] R. Chandramouli, M. Iorga, and S. Chokhani, "Cryptographic key management issues and challenges in cloud services," NISTIR 7956, 2013, pp. 1–36, doi: 10.6028/NIST.IR.7956.

[11] N. Bindel, J. Brendel, M. Fischlin, B. Goncalves and D. Stebila, "Hybrid key encapsulation mechanisms and authenticated key exchange," in *Proc. Int. Conf. Post-Quantum Cryptography*, Chongqing, China, May 2019, pp. 206–226, doi: 10.1007/978-3-030-25510-7_12.

[12] S. Li, Y. Chen, L. Chen, J. Liao, C. Kuang, K. Li, W. Liang and N. Xiong, "Post-Quantum Security: Opportunities and Challenges",*Sensors*, vol. 23, pp. 8744, 2023, doi: 10.3390/s23218744

[13] M. A. G. de la Torre, I. A. M. Sandoval, G. T. F. de Abreu and L. H. Encinas, "Post-quantum wireless-based key encapsulation mechanism via CRYSTALS-Kyber for resource-constrained devices," *IEEE Access*, vol. 13, pp. 66714–66725, 2025, doi: 10.1109/ACCESS.2024.3479215.

[14] [14] J. Señor, J. Portilla and M. Portela-García, "Performance analysis of postquantum cryptographic schemes for securing large-scale wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 20, no. 10, pp. 12339–12349, Oct. 2024, doi: 10.1109/TII.2024.3423641.

[15] G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, "Keccak," in *Proc. EUROCRYPT*, Athens, Greece, May 2013, pp. 313–314, doi: 10.1007/978-3-642-38348-9_19.

[16] J. Kelsey, S. Chang, and R. Perlner, "SHA-3 derived functions: cSHAKE, KMAC, TupleHash, and ParallelHash," Gaithersburg, MD, USA, Dec. 2016, NIST SP 800-185, doi: 10.6028/NIST.SP.800-185.

[17] M. Bisheh-Niasar, R. Azarderakhsh, and M. Mozaffari-Kermani, "Instruction-Set Accelerated Implementation of CRYSTALS-Kyber", *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 11, pp. 4648-4659, November 2021, doi: 10.1109/TCSI.2021.3106639.

[18] National Institute of Standards and Technology (NIST), "SHA-3 standard: Permutation-based hash and extendable-output functions," Gaithersburg, MD, USA, Aug. 2015, FIPS PUB 202, doi: 10.6028/NIST.FIPS.202.

[19] N. Ansary, Q. A. Rahman Adib, T. Reasat, A.S. Sushmit, A.I. Humayun, S. Mehnaz, K. Fatema, M. M. Or Rashid and F.Sadeque, "Unicode Normalization and Grapheme Parsing of Indic Languages," in *Proc. 2024 Lang. Resources and Evaluation Conf. (LREC-COLING)*, pp. 17019–17030, May 2024.

[20] H. Nguyen and S. Cavallari, "Neural multi-task text normalization and sanitization with pointer-generator," in *Proc. First Workshop on Natural Language Interfaces*, Jul. 2020, pp. 37–47, doi: 10.18653/v1/2020.nli-1.5.

[21] J. Lang, A. Czeskis, D. Balfanz, M. Schilder, and S. Srinivas, "Security keys: Practical cryptographic second factors for the modern web," in *Proc. Financial Cryptography and Data Security*, Christ Church, Barbados, Feb. 2016, pp. 422–440, doi: 10.1007/978-3-662-54970-4_25.

[22] D. Sikeridis, P. Kampanakis and M. Devetsikiotis, "Post-quantum authentication in TLS 1.3: A performance study," in *Proc. The Network and Distributed System Security Symp.*, San Diego, CA, USA, Feb. 2020, pp. 1–16, doi: 10.14722/ndss.2020.24041.

[23] T. M. Fernández-Caramés, "From pre-quantum to post-quantum IoT security: A survey on quantum-resistant cryptosystems for the Internet of Things," *IEEE Internet of Things J.*, vol. 7, no. 7, pp. 6457–6480, 2020, doi: 10.1109/JIOT.2019.2956244.

[24] A. Narayanan and V. Shmatikov, "Fast dictionary attacks on passwords using time-space tradeoff," in *Proc. 12th ACM Conf. Comput. Commun. Security*, Alexandria, VA, USA, Nov. 2005, pp. 364–372, doi: 10.1145/1102120.1102168.

[25] E. Simion, "Entropy and Randomness: From Analogic to Quantum World," *IEEE Access*, vol. 8, pp. 74553–74561, 2020, doi: 10.1109/ACCESS.2020.2988658.

[26] P. A. Grassi, M. E. Garcia, and J. L. Fenton, "Digital identity guidelines: Authentication and lifecycle management," NIST SP 800-63B, Gaithersburg, MD, USA, Jun. 2017, doi: 10.6028/NIST.SP.800-63b.