2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Paper ID #52479

Hands-On Post-Quantum Cryptography Education: Laboratory Modules for Teaching NIST-Standardized Quantum-Resistant Algorithms

Jonah Whitford, East Carolina University

Jonah Whitford is a cybersecurity student at East Carolina University who cares about emerging technologies and making the internet safer. My goal is to help connect what researchers are discovering about new threats, like quantum computing breaking our current encryption, with what we're actually teaching students in the classroom. I want to work on building a curriculum that gives cybersecurity students the practical skills they'll need when they graduate, not just what worked five years ago.

Mr. Colby Lee Sawyer MSSE, East Carolina University

I'm a Teaching Instructor in Technology Systems at East Carolina University, where I teach courses in cybersecurity, IoT, and AI systems. My research focuses on building intelligent, data-driven architectures that connect edge sensing with cloud-based reasoning to improve automation, awareness, and decision-making.

Dr. Ciprian Popoviciu, East Carolina University

Dr. Ciprian Popoviciu has over 26 years of experience working in various technical and leadership roles in the IT industry. He founded and led Nephos6, the first company to enable OpenStack for IPv6 and deploy it in production. Prior to starting Nephos6 he worked for CIsco and he is an industry recognized IPv6 subject matter expert. Currently he is an assistant professor in the college of engineering at East Carolina University and his research is focused on IoT and cybersecurity.

Hands-On Post-Quantum Cryptography Education: Laboratory Modules for Teaching NIST-Standardized Quantum-Resistant Algorithms

Abstract

Most online security today relies on Rivest-Shamir-Adleman (RSA) and elliptic curve cryptographic algorithms. With the latest advancements in Quantum computers, both of these systems will be rendered obsolete. The National Institute of Standards and Technology (NIST) published new quantum-resistant algorithms in 2024, but few higher-education cybersecurity programs teach them. Students graduate without knowing how to work with the latest cryptographic algorithms necessary in a quantum computing enabled cybersecurity world. This paper describes five laboratory modules that teach post-quantum cryptography through hands-on practice. Students implement NIST's standardized algorithms, CRYSTALS-Kyber and CRYSTALS-Dilithium, and compare them with traditional cryptographic methods. The labs cover hybrid systems that combine classical and quantum-resistant approaches, reflecting how organizations will transition their infrastructure. Assessment is designed around implementation projects, technical evaluations, and migration scenario analysis. The modules can fit into current security courses without major curriculum changes, giving instructors ready-to-use materials for teaching the latest cryptographic algorithms that students will need in practice as the transition to post-quantum systems accelerates.

Introduction

RSA and elliptic curve cryptography are foundational to data encryption in current IT infrastructures. They keep bank logins safe, medical records private, and emails from being read by the wrong people. The math behind them, large number factorization, is hard enough that regular computers can't crack it in any reasonable timeframe. But quantum computers play by different rules; quantum computers are very good at quickly processing large number factorization problems. Run Shor's algorithm on a big enough quantum machine and RSA falls apart [4]. What's worse, attackers don't have to wait. They can grab encrypted data now, sit on it, and decrypt it later once the hardware catches up. Security researchers call this "harvest now, decrypt later" [6], and it means the threat isn't hypothetical even if we debate the easy availability of quantum computers; the threat is real.

NIST foresaw this threat and spent years running a competition to find quantum-proof replacements. In August 2024, NIST finally published the winners: ML-KEM (built on CRYSTALS-Kyber) handles key exchange, ML-DSA (built on CRYSTALS-Dilithium) handles signatures [1, 3]. These aren't optional upgrades. Every company, every government agency, every system that uses public-key cryptography will eventually need to adopt them [2].

Here's the problem: most higher education programs are not preparing students for this new set of technologies and skill sets. Most cybersecurity programs teach RSA and ECC taught core material, but they do not cover the technologies related to a post-quantum world. Barely

mentioned. Students finish their degrees having never touched the algorithms they'll amost likely deploy and operate in a few years. That's a knowledge gap we can't afford.

Teaching post-quantum cryptography isn't straightforward. Lattice-based cryptography works differently than the large numbers factorization theory students learned for RSA. Instructors can't just add incremental instruction material. They need materials that make these ideas accessible to students who aren't math majors, and labs that provide real implementation experience rather than just theory.

The goal of this paper is to explore approaches to bridging the instructional gap between pre and post quantum world encryption instruction. We developed five labs that walk students through post-quantum cryptography (PQC) from the ground up. They start with why quantum computing threatens the current cryptography principles, move through hands-on coding with Kyber and Dilithium, and end with a realistic migration exercise. Everything runs on the Open Quantum Safe library, which is free and doesn't need special hardware. We focused heavily on hybrid approaches, mixing classical and post-quantum encryption, since that's how most organizations will actually make the transition.

Can structured laboratory modules effectively teach post-quantum cryptography concepts and practical implementation skills to undergraduate cybersecurity students without requiring advanced mathematics and quantum mechanics prerequisites? This is the research question of our paper. We believe they can. The labs described here test whether students gain both conceptual understanding and hands-on skills when working directly with PQC implementations, even without a strong math background. We also look at how students weigh tradeoffs between classical and post-quantum methods, and whether they can take what they learn and apply it to messy, real-world migration problems.

Methodology

Curriculum Design Principles

We developed five labs using backward design: starting from the skills students needed to acquire, then building the path to get there. Two problems shaped the design more than any others. First, post-quantum cryptography concepts are abstract. Lattice mathematics and learning-with-errors problems are not intuitive. Second, undergraduate cybersecurity students are not typically math majors, and an approach that leads with proofs loses them before they even start to learn the technological concepts.

To address both, we structured each module around a three-phase loop: Concept, Experiment, and Reflection. Theory is kept to short bursts, typically three to five sentences, before students encounter the idea directly through code or a structured exercise. Abstract concepts are grounded in concrete analogies before formal definitions are introduced. The modules then close with reflection questions that ask students to synthesize what they observed, not just what they were told.

Throughout, each module uses a narrative framing. Rather than presenting cryptographic protocols as isolated technical exercises, each lab opens with a scenario that places students in a

specific professional role confronting a real problem. This framing is maintained throughout the lab. Alice and Bob are not just the variable names typically used in covering quantume computing concepts; they are characters in an unfolding situation that raises the stakes of each implementation decision. Research in technical education consistently shows that narrative context improves retention and transfer of abstract concepts, and cryptography, with its inherent story of secrets, adversaries, and trust, lends itself particularly well to this approach.

Each module also includes at least two "Break It" challenges: adversarial exercises in which students deliberately attempt to defeat, circumvent, or find the limits of the cryptographic systems they have just implemented. These challenges were included because students learn more durably from observing failure than from observing success. Watching a tampered ciphertext fail to decapsulate, or seeing a forged signature rejected, makes the security properties of an algorithm concrete in a way that reading about them does not.

The overall module design consists of approximately 20% conceptual instruction, 60% active experimentation, and 20% reflection, a distribution consistent with laboratory design literature on technical skill acquisition [9]. Formative micro-checks are embedded at the end of each conceptual section for students to identify misconceptions before beginning the corresponding implementation exercise.

Target Audience and Prerequisites

The target audience for this content is undergraduate students at East Carolina University enrolled in cybersecurity courses. They typically completed introductory networking and security coursework but had no prior exposure to quantum computing or advanced abstract algebra. Every student completed at least one cryptography course, so they arrived familiar with RSA, symmetric encryption, and public-key concepts at an introductory level. No quantum physics or linear algebra background was assumed or required.

The curriculum comprises five sequential modules, each designed as a two-hour lab session with additional time for preparatory reading and a follow-up assignment. The modules progress from foundational concepts through practical implementation to an applied capstone scenario. All implementation work uses the Open Quantum Safe (OQS) library and its Python bindings, which are freely available and require no special hardware [7].

Laboratory Modules

The five modules are described below. Each description includes the motivating scenario, the conceptual framing, the core implementation exercises, and the challenge activities that push students beyond guided procedure.

Module 1: The Quantum Threat

Module 1 establishes the foundational threat model and motivates the rest of the curriculum. It is the only module with no implementation component; instead, students work entirely with analysis and structured reasoning exercises.

> *Scenario: It is 2031. A government agency announces that a cryptographically relevant quantum computer now exists. Overnight, every RSA-encrypted archive becomes readable to anyone with access to the machine. Students are asked: how did this happen, and what should have been done?*

The conceptual section introduces Shor's algorithm at an intuitive level, specifically why factoring large integers and solving discrete logarithms are efficiently solvable on quantum hardware while lattice problems are not. A comparison table presented at the start of the lab makes this distinction concrete before any formal explanation is attempted (see Table 1):

Problem Type	Classical Computer	Quantum Computer	Post-Quantum Safe?
Factoring large integers (RSA)	Computationally infeasible	Efficient (Shor's)	No
Discrete logarithm (ECC)	Computationally infeasible	Efficient (Shor's)	No
Lattice problems (ML-KEM)	Computationally infeasible	No known speedup	Yes
Symmetric key search (AES-256)	Infeasible	Hard (Grover's)	Yes (double key size)

Table 1. *Comparison of Cryptographic Problem Types by Computing Model*

A formative micro-check following the conceptual section asks students to identify which of the four problem types Shor's algorithm specifically targets, ensuring the key distinction is understood before students proceed to the analysis exercise.

The visualization section introduces the "harvest now, decrypt later" attack model. Students map an attack timeline from data interception today to a decryption event post-quantum-computer-availability against the sensitivity and required confidentiality period of various asset types. This timeline makes the urgency of migration concrete: an organization that encrypts patient records with RSA-2048 today is already in the harvest window.

The primary hands-on exercise presents an asset triage table. Students receive a list of organizational assets with varying sensitivity levels and confidentiality requirements and must assign migration priorities. The exercise deliberately includes items that appear sensitive but have short confidentiality windows (for example, press release drafts) alongside items with low apparent sensitivity but long windows (for example, long-term research strategy documents). The goal is to teach students that migration urgency involves both sensitivity and time horizon, not sensitivity alone. This is in line with the cybersecurity best practice to start by defining the risk profile of managed assets such as data in this case.

The Break It challenge presents an accelerated threat scenario: a quantum computer is assumed to exist by 2028 rather than 2035. Students must recalculate which assets are already in the

exposure window, compute the vulnerable data window under different migration timelines, and produce a one-paragraph briefing for a non-technical executive. This challenge tests students' ability to translate a technical threat model into organizational communication, a skill that cryptography courses rarely address explicitly.

Reflection questions close the module by asking students to articulate why waiting for quantum computers to "actually exist" is a flawed risk mitigation strategy, and to construct a single-sentence argument for immediate migration action aimed at a non-technical decision-maker.

Module 2: ML-KEM Key Encapsulation

Module 2 introduces CRYSTALS-Kyber (ML-KEM), NIST's standardized key encapsulation mechanism. Students install the OQS library, implement key generation, encapsulation, and decapsulation, and benchmark ML-KEM against RSA and ECDH across key size and performance dimensions.

> *Scenario: Alice and Bob need to establish a shared secret. Diffie-Hellman and ECDH are broken by Shor's algorithm. Students must implement the replacement and measure what it costs in key size and computation time.*

The conceptual section explains the Learning With Errors (LWE) problem that underlies ML-KEM's security. Rather than introducing the mathematical model directly, the lab presents LWE through an analogy: a system of linear equations with small random noise added, where recovering the hidden secret from the noisy equations is computationally hard. This framing is sufficient for students to understand why LWE resists both classical and quantum attacks without requiring formal proof.

A formative check asks students why Shor's algorithm cannot be applied to lattice problems, ensuring they understand that Shor's speedup is specific to the algebraic structure of factoring and discrete logarithm, not a general-purpose attack on all public-key cryptography.

The implementation section is structured in two exercises. In the first, students implement a complete ML-KEM key exchange using the OQS Python bindings: generating a keypair, performing encapsulation on the public key, and recovering the shared secret through decapsulation. Students record key sizes and confirm that both parties derive the same shared secret. The following code structure is provided as a starting framework:

```
import oqs

kem = oqs.KeyEncapsulation('ML-KEM-768')
public_key = kem.generate_keypair()
ciphertext, shared_secret_bob = kem.encap_secret(public_key)
shared_secret_alice = kem.decap_secret(ciphertext)
print(f'Secrets match: {shared_secret_alice == shared_secret_bob}')
```

Figure 1. *ML-KEM key exchange implementation framework (Python, using OQS bindings)*

In the second exercise, students benchmark all three ML-KEM security levels (ML-KEM-512, ML-KEM-768, ML-KEM-1024) against RSA-2048 and ECDH P-256, measuring key generation time, encapsulation and decapsulation time, and key and ciphertext sizes across 100 iterations. Results are organized in a recording table and then visualized as a relative size comparison, illustrating that ML-KEM keys are substantially larger than RSA or ECC keys while performance is competitive or superior.

The Break It challenges extend beyond the guided benchmarking. In the first challenge, students modify the ciphertext by flipping a single bit before decapsulation and observe whether ML-KEM fails loudly with an exception or silently returns an incorrect but structurally valid shared secret. This exercise teaches an important protocol design lesson: cryptographic primitives that fail silently require authentication wrappers to be safe in practice. In the second challenge, students select a security level and justify their choice for a specific scenario, a 20-year intelligence archive, weighing security equivalence against performance overhead.

Reflection questions ask students to articulate the difference between key exchange and key encapsulation as design patterns, and to explain why organizations might run RSA and ML-KEM simultaneously during a transition period rather than switching entirely.

Module 3: ML-DSA Digital Signatures

Module 3 covers CRYSTALS-Dilithium (ML-DSA), NIST's standardized digital signature algorithm. The module follows the same structure as Module 2 but shifts the threat model: here, the consequence of quantum vulnerability is not privacy loss but forgery, the ability to issue fraudulent software updates, contracts, or certificates with no detectable difference from legitimate ones.

> *Scenario: A hospital receives a software update signed by the vendor. The signature validates correctly. But if quantum computers can forge ECDSA (Elliptic Curve Digital Signature Algorithm) signatures, the entire trust chain is broken, and the hospital may have just installed malware.*

The conceptual section opens with a comparison table presenting RSA, ECDSA, and ML-DSA across security basis, quantum safety, signature size, and verification speed. The key point students must internalize is that Shor's algorithm can recover an ECDSA private key from a public key, enabling an attacker to forge any future signature as that entity. The formative check presents this as a concrete attack scenario: an attacker who harvested a signed software update in 2024 and obtains a quantum computer in 2032 can impersonate the original vendor indefinitely.

Implementation Exercise A asks students to sign a realistic message, a software deployment authorization, using ML-DSA-65, then attempt to verify it against a tampered message. Students observe that even a one-character modification in the message causes verification to fail, and that

modifying any single bit in the signature itself also causes rejection. This empirical confirmation of signature binding is more memorable than a textual description of the same property.

Implementation Exercise B benchmarks ML-DSA-44, ML-DSA-65, and ML-DSA-87 against RSA-2048 PSS and ECDSA P-256 across signing time, verification time, and signature size. Students record results in a structured table and then apply them to a specific question: at what document size does the ML-DSA signature overhead become negligible as a percentage of payload? This calculation connects the raw benchmark numbers to practical deployment decisions.

Exercise C asks students to sign an actual file from the filesystem rather than a fixed test message. Students measure the ratio of signature size to file size across several file sizes and identify the threshold below which signature overhead is a meaningful concern. This exercise grounds the size comparison in real-world deployment contexts such as firmware images, configuration files, and legal documents.

The Break It challenges include an attempted forgery exercise, in which students systematically modify message and signature bits to find any combination that verifies, and a security level selection exercise requiring written justification for a 30-year medical software signing certificate versus a standard TLS certificate.

Module 4: Hybrid Cryptography

Module 4 addresses the organizational reality that no migration happens overnight or in a greenfield scenario. Legacy systems, compliance requirements, and embedded hardware mean that classical and post-quantum algorithms will coexist in production infrastructure for years. The security architecture must address not only the operational considerations of each eky exchange mechanism but also the security considerations of their co-existence. Students implement a hybrid X25519 + ML-KEM-768 key exchange that maintains security against both classical and quantum adversaries.

> *Scenario: A bank cannot immediately replace millions of RSA certificates embedded in hardware tokens and compliance systems. But it needs quantum-resistant security now. Students must build a system that is safe under both classical and quantum attack models simultaneously.*

The conceptual section introduces the security guarantee of hybrid key exchange: the combined key is broken only if both component algorithms are simultaneously compromised. This guarantee is presented in a comparison table across three approaches, classical only, post-quantum only, and hybrid, rating each against classical attack, quantum attack, and transition complexity. The formative check presents a scenario in which a vulnerability is found in ML-KEM and asks what the security impact is on a properly implemented hybrid system. Students must recognize that the hybrid falls back to X25519 security, not zero security.

The implementation exercise builds a complete hybrid key exchange using X25519 for the classical component and ML-KEM-768 for the post-quantum component, combining the two shared secrets using HKDF (SHA-256). Students confirm that both parties derive an identical hybrid key, then measure the total bytes exchanged in the handshake compared to classical-only and post-quantum-only variants. A discussion prompt connects this overhead calculation to TLS: a hybrid TLS 1.3 handshake adds approximately 5,565 bytes compared to a classical handshake, which is negligible on broadband connections but potentially significant for constrained IoT devices.

The Break It challenges test students' understanding of why the key combination method matters. Students are presented with three deliberately flawed combination approaches: XOR-ing the Kyber secret with a constant, truncating and concatenating the two secrets, and selecting the longer secret as the combined key. Students must determine the actual security level achieved by each flawed approach. A second challenge simulates component failure by revealing one component's secret to a simulated attacker and asking whether the hybrid key is recoverable. Students conclude by writing one sentence explaining why HKDF is necessary rather than naive concatenation.

Module 5: Capstone Migration Planning

Module 5 is the capstone exercise for the course. Rather than introducing new cryptographic primitives, it tests whether students can apply everything they learned in Modules 1 through 4 to a realistic organizational scenario requiring prioritized decision-making under constraints.

> *Scenario: Students have been hired as cryptography consultants for MedCore Health Systems, a regional hospital network with 12 facilities, 2 million patient records, medical imaging systems from 2008, and 24/7 encrypted pharmacy communications. NIST's migration deadline is 2035. The clock is already running.*

The module opens with a migration complexity table presenting the different infrastructure layers that require post-quantum migration: TLS certificates, software signing, VPN/IPsec, hardware security modules, legacy embedded systems, and regulatory compliance, each with typical timelines and difficulty ratings. This table is a deliberate contrast with the implementation-focused modules: migration is not a technical problem alone but an organizational and prioritization challenge.

The primary exercise presents an asset inventory of eight fictional MedCore systems, each with a listed cryptographic method, data sensitivity level, and defined life cycle. Students must assign a migration priority to each asset and justify their ranking; they are defining the risk profile for the analyzed environment. The inventory is designed to force difficult tradeoffs: a 2008 MRI system using RSA-1024 is both highly sensitive and physically impossible to update via software, while a 2023 public-facing website using ECDH P-256 is low-sensitivity but easily migrated. There is

no single correct prioritization, which requires students to articulate and defend their reasoning rather than locate a right answer.

Students then draft a ten-year phased migration roadmap, assigning systems to four phases spanning 2025 through 2035. A structured table with empty cells for system assignments and migration approaches guides this exercise without over-constraining the output. A dedicated sub-exercise asks students to evaluate three options for the difficult to migrate, legacy devices: network isolation, a post-quantum-capable proxy gateway, and full hardware replacement, by completing a table of security impact and feasibility assessments for each.

The Break It challenges raise the learning stakes. In the first, intelligence suggests a quantum computer may be available by 2028 rather than 2035, collapsing the migration window by seven years. Students must revise Phase 1 to reflect this acceleration and write a board-level email communicating urgency without technical jargon. In the second challenge, a budget constraint limits migration to three systems in 2026. Students must select those three systems from their full inventory and present a justification suitable for a non-technical board audience, grounding each recommendation in a concrete risk scenario rather than technical metrics alone.

The module's deliverable is a one-to-two-page written migration plan for MedCore covering a risk profile summary, a prioritized asset list, the phased timeline, the legacy device recommendation, and one immediate low-cost action. This deliverable is explicitly framed for a non-technical audience, requiring students to translate technical analysis into organizational communication, a skill that is central to cybersecurity practice but rarely developed in implementation-focused courses.

Reflection questions close the module by asking students to assess the likelihood that most organizations will meet NIST's 2035 deadline given historical patterns in large-scale cryptographic transitions, and to articulate the ethical obligation of an organization operating a legacy system that cannot be updated but handles sensitive data.

Assessment Strategy

Assessment across all five modules follows a consistent structure. Pre- and post-delivery checks test conceptual understanding before and after each module. The pre-check is administered before students encounter any module content, providing a knowledge baseline; the post-check uses equivalent questions assessing the same concepts but with different wording and examples after the module is complete. Implementation projects are graded with rubrics that assess both correctness (does the code execute, do shared secrets match, do signatures verify) and analytical depth (do recorded benchmark results include appropriate context and interpretation).

Written technical comparisons produced in Modules 2, 3, and 4 are evaluated on whether students correctly identify the tradeoffs between approaches rather than simply restating elements of algorithmics. The capstone migration plan in Module 5 is evaluated on the quality of prioritization reasoning, the feasibility of the proposed timeline, and the clarity of the executive communication.

Formative micro-checks embedded within each module are not graded but are used to identify conceptual gaps before students begin the corresponding implementation exercise, allowing instructors to intervene before misconceptions propagate into the coding work.

Results

The five modules went through a development and review process before reaching the pilot stage. The roadmap in Figure 2 shows where the project currently stands: content development and senior instructor review are complete, limited pilot delivery being planned, and will be followed by a second set of review, optimization work based on received feedback, and then a second delivery.

1. Content Development	**2. Instructor Review**	**3. Pilot Delivery**	4. Review of Results	5. Optimizations	6. Second Delivery
✓*Completed*	✓*Completed*	▶ ***Current***	*Planned*	*Planned*	*Planned*

Figure 2. Curriculum development and delivery roadmap. Green cells indicate completed stages; blue cell indicates the current stage; gray cells indicate planned future stages.

Internal Instructor Review

Two senior instructors reviewed all five modules before the limited pilot. One has a PhD in theoretical physics and teaches senior and graduate level cybersecurity courses. He focused on whether the quantum computing concepts were presented accurately and clearly. He also reviewed the modules from the perspective of cybersecurity instruction with emphasis on alignment with cybersecurity frameworks and cybersecurity best practices. The second instructor who has experience in teaching junior cybersecurity courses, reviewed the content from a cybersecurity instruction standpoint, looking at exercise sequencing, difficulty, and alignment with content. He evaluated the quality of both tests and hands-on activities in assessing the effectiveness of student learning. Both reviewers gave feedback on the Learning With Errors analogy in Module 2, the MedCore scenario in Module 5, and the Break It challenges. Both also noted the value of the embedded formative micro-checks as a mechanism for surfacing conceptual gaps before implementation exercises begin, and affirmed the scenario-based learning structure as effective for grounding abstract cryptographic concepts in professional contexts. That feedback drove the revisions from the first version of the modules to the current version.

Module Revisions: v1 to v2

The main difference between v1 and v2 is how the conceptual content is paced. In v1, the theory sections were longer and came before any hands-on work, which delayed students getting to the actual implementation. In v2 the theoretical discussion is broken into short sections that appear immediately before the exercise they relate to, matching the Concept-Experiment-Reflection structure described earlier. The narrative framing was also added in v2; in v1, Alice and Bob were just variable names, while in v2 they are part of a running scenario that gives each exercise context. The Break It challenges were expanded and given clearer instructions in v2 after

reviewers noted that the v1 versions were hard to complete without extra guidance. Figures 3 and 4 show screenshots comparing the v1 and v2 interfaces for Lab 1 and Lab 2 respectively.

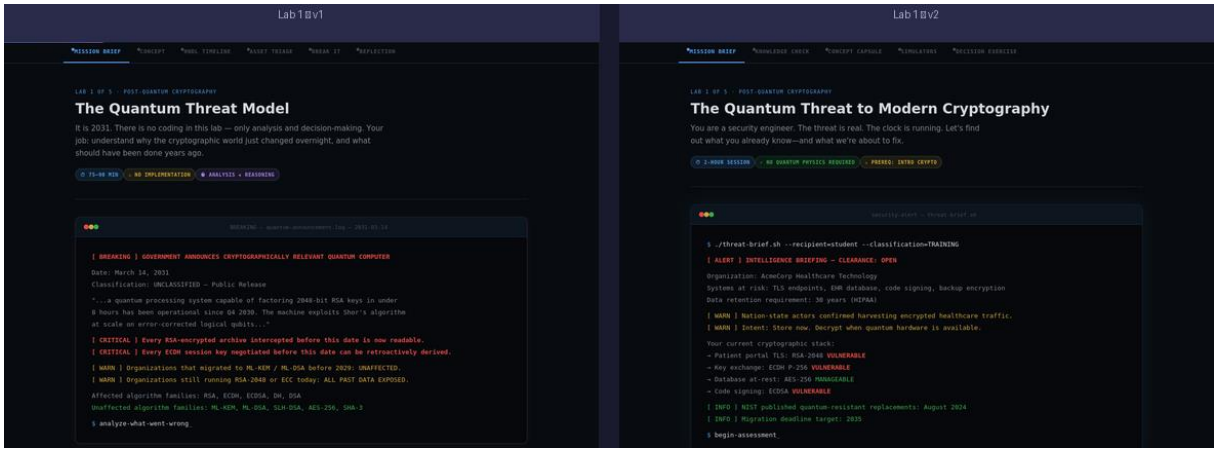

Figure 3. Lab 1 interface: v1 (left) vs. v2 (right). Tab structure, scenario framing, and badge labels all revised in v2.

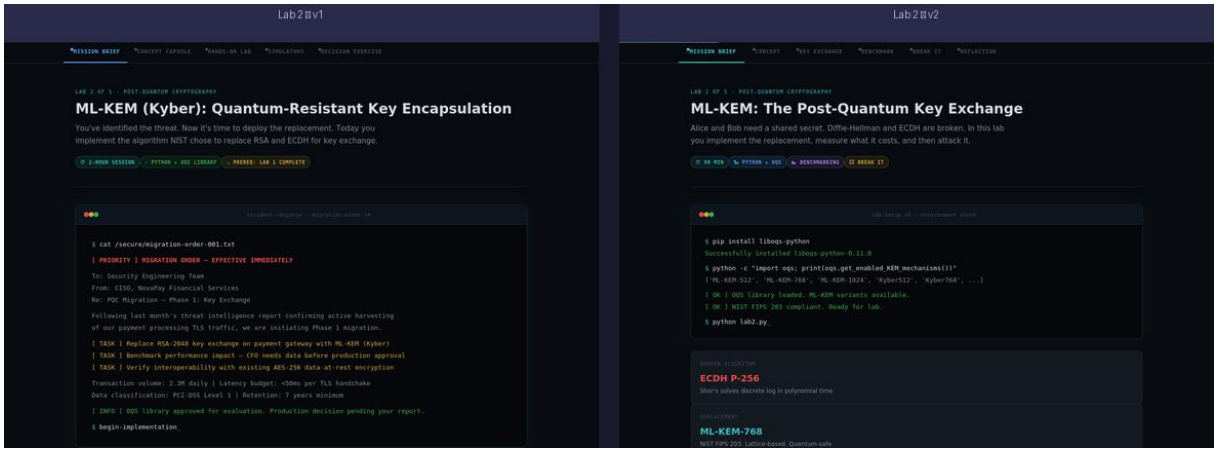

Figure 4. Lab 2 interface: v1 (left) vs. v2 (right). v2 introduces an environment check terminal and structured concept cards replacing the v1 mission order framing.

Beta Testing

Before the instructor' review, a select group of cybersecurity students worked through the modules informally to provide an early usability and content perspective. Before beginning, students were asked to describe their familiarity with quantum cryptography. Responses spanned a broad range: some students had no prior exposure at all, others had heard the term but could not define it, and a smaller number described themselves as somewhat familiar with the concept. None entered with working knowledge of post-quantum algorithms or NIST's standardization process. After completing the modules, students reported a meaningfully improved understanding of the concepts and how the algorithms functioned in practice, including how ML-KEM and ML-DSA differ from the classical systems they had previously encountered.

Student feedback also addressed the module design and presentation. Students responded positively to the visual presentation of the labs, noting that the interface looked and felt like

something they would encounter in a professional or enterprise environment rather than a standard classroom exercise. The aesthetic authenticity was a meaningful factor: students said it helped them take the material seriously and engage with it as a realistic work context. The scenario framing resonated for similar reasons. Students described the professional contexts as believable and relevant, and several noted that working through a realistic migration scenario made the content feel directly applicable to the kind of work they expected to do after graduating. These observations, centered on authenticity and relevance rather than instructional pacing or exercise structure, gave us confidence that the v1-to-v2 changes addressed the right problems.

Planned Pilot Delivery

The full pilot is planned for summer 2026, delivered as part of an existing cybersecurity course at East Carolina University. Students entering the pilot will have completed at least one prior cryptography course. Pre- and post-module checks will be used to measure conceptual gains, and implementation rubrics will evaluate hands-on skills. Those results will be included in the final version of this paper and will feed into the optimization stage shown in Figure 2.

Conclusions and Future Work

This initial version of the content represents a subset of the envisioned course delivering the technical knowledge relevant to a quantum computing powered world. New concepts will be included as NIST continues to develop and evaluate related algorithms. FN-DSA (the FALCON-based signature scheme) and HQC for key encapsulation are in the pipeline, and they will be integrated in the course. Similar to NIST' CSF profiles, we are interested in developing more advanced modules that tackle migration scenarios for specific industries: what does a hospital need to consider versus a bank versus a power utility? The authors are also exploring mechanisms to assess the impact of this type of knowledge on the professional career of graduates. The adoption of this type of content will depend on the availability of instructor resources like training workshops and lesson plans. A comprehensive strategy for driving instruction in these topics must be explored if higher-education institutions plan to support NIST' vision to have organizations migrate to quantum resilient cybersecurity by 2035. The lab redesign described in this paper reflects a broader pedagogical argument: post-quantum cryptography cannot be taught effectively through lectures and readings alone. The concepts are abstract, the stakes are high, and the timeline for deployment is already underway. Students need to generate key pairs, observe benchmark data, attempt forgeries, and reason through migration tradeoffs, not because these exercises are novel, but because the skills they develop are directly transferable to the jobs these students will hold within the next several years. The five modules described here represent one structured path toward that outcome, and the pilot's results, when complete, will help clarify what works and what requires further revision.

References

[1] National Institute of Standards and Technology. (2024, August 13). NIST Releases First 3 Finalized Post-Quantum Encryption Standards. https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards

[2] National Institute of Standards and Technology. (2024). NIST IR 8547: Transition to Post-Quantum Cryptography Standards. https://nvlpubs.nist.gov/nistpubs/ir/2024/NIST.IR.8547.ipd.pdf

[3] National Institute of Standards and Technology. (2024). FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard. https://csrc.nist.gov/pubs/fips/203/final

[4] Shor, P. W. (1994). Algorithms for quantum computation: discrete logarithms and factoring. Proceedings 35th Annual Symposium on Foundations of Computer Science, 124-134.

[5] Global Risk Institute. (2021). Quantum Threat Timeline Report. https://globalriskinstitute.org/publication/quantum-threat-timeline-report-2021/

[6] Mascelli, J., & Rodden, M. (2025). “Harvest Now Decrypt Later”: Examining Post-Quantum Cryptography and the Data Privacy Risks for Distributed Ledger Networks. Federal Reserve Board FEDS Working Paper No. 2025-93.

[7] Open Quantum Safe Project. (2024). liboqs: C library for quantum-safe cryptographic algorithms. https://openquantumsafe.org/

[8] IBM Research. (2024). NIST’s post-quantum cryptography standards are here. https://research.ibm.com/blog/nist-pqc-standards

[9] Hofstein, A., & Lunetta, V. N. (2004). The laboratory in science education: Foundations for the twenty-first century. Science Education, 88(1), 28–54.