Post-Quantum-Ready Prototype for Standardized Certificate Creation and Verification Model

Praveenraj J
Master of Technology in Cyber Security
The Northcap University
Gurugram – 122017, Haryana, India
praveenraj25csp002@ncuindia.edu

Dr. Swati Gupta
Assistant Professor, Department of Computer Science
The NorthCap University
Gurugram – 122017, Haryana, India
Email ID: swatigupta@ncuindia.edu

Abstract **– Certificate forgery is a major problem in academic, professional, governmental, and industrial domains. Classical digital signature algorithms like RSA and ECC are weak against emerging quantum attacks which compromises the authenticity and integrity of the documents. To address this challenge, this paper proposes a Standardized Post-Quantum-Ready Prototype for Certificate Creation and Verification using NIST-approved CRYSTALS-Dilithium2 post-quantum digital signature algorithm. Furthermore, each physical certificate has QR code which links to its corresponding digital version, enables the authenticity and integrity checks to all formats. Not limited only for certificates, can be applicable for all forms of document creation and verification.**

Keywords – Post-Quantum Cryptography, Digital Signature, Certificate Security, Lattice Cryptography, CRYSTALS-Dilithium.

I. Introduction

Certificates are widely used in academic, professional, governmental, and industrial domains to prove identity and document achievements. Traditionally, certificates were issued in physical form, but with increasing digitalization, digital certificates are common. However, both physical and digital certificates remain vulnerable to certificate forgery.

Traditionally, digital certificates were protected against forgery by using classical cryptographic techniques like RSA and ECC [8]. For decades, these techniques provided the trust that the institutions can rely on. With the rise of quantum computing, algorithms like Shor's Algorithm can break RSA and ECC [4]. Researchers estimate that within 10-15 years practical quantum computers may feasible and place the classical certificate systems at risk [7].

To address this challenge, this paper presents a standardized post-quantum-ready prototype for certificate creation and verification to address the above gap. The proposed prototype uses CRYSTALS-Dilithium2 digital signature algorithm to generate, sign, and verify certificates. This ensures certificate authenticity, integrity, and tamper detection against unauthorized modifications.

The system also supports printed physical certificates by including QR codes which link to their corresponding digital version. This ensures authenticity and integrity verification for both physical and digital documents.

II. Background and Related Work

A. Classical Digital Signatures and Certificate Infrastructure

Physical and Digital certificates are used to validate identity and confirm achievements. These digital certificates are verified through the Public Key Infrastructure (PKI) which uses asymmetric cryptography for signing and verification. Classical algorithms like RSA and ECC are commonly used to ensure certificate authenticity and integrity.

In PKI-based systems, a private key is used to generate a digital signature and its corresponding public key is used to verify it. RSA and ECC depend on mathematical problems like integer factorization and elliptic curve discrete logarithm, but both can be solved by Shor's Algorithm on a large-scale quantum computer [4]. Therefore, current digital signature mechanisms are not quantum-resistant and they are vulnerable in a post-quantum era.

B. Post-Quantum Cryptography Developments

To address the threat caused by quantum attacks, the NIST started the PQC Standardization Project in 2016 [5]. The primary goal was to identify and standardize algorithms which can be secure against both classical and quantum attacks.

In 2024, NIST standardized two post-quantum algorithms, CRYSTALS-Kyber for encryption (FIPS 203) and CRYSTALS-Dilithium for digital signatures (FIPS 204) [1], [2]. Both these algorithms are based on lattice-based cryptography (Module Learning with Errors and Module Short Integer Solution problems), these are believed to be hard to crack even for quantum computers [1].

In these two algorithms, Dilithium offers a good balance between security and performance, so it is widely recommended for post-quantum secure digital signature applications.

C. Related Implementations and Research

Many studies and implementations have adopted post-quantum algorithms, but mostly focuses on network-level protocols rather than document-centric systems.

For Example,

1) Hulsing et al. [9] used hash-based signatures to achieve long-term security in firmware updates.
2) Bos et al. [1], [11] implemented lattice-based signatures for Internet of Things (IoT) devices.
3) NIST's PQC project reports [5], [11] studied Dilithium integration in secure communication protocols such as TLS 1.3 and SSH.

However, document centric systems such as PDF and XML signatures or certificate creations are still depending on RSA/ECC, but they are not quantum-resistant which makes them vulnerable in the post-quantum era.

D. Research Gap and Motivation

Most studies highly focus on secure key exchange and network-layer encryption [7], [10]; it leaves a significant gap in document creation and verification mechanisms which can be post-quantum ready. This paper addresses that gap by implementing post-quantum ready framework document verification.

This paper proposes a standardized post-quantum-ready prototype which is capable of

1) Generating digital certificates,
2) Embedding secure digital signatures based on CRYSTALS-Dilithium2, and
4) Validating and detecting any tampering within the certificate file.

This approach ensures document authenticity, integrity and non-repudiation in the quantum era and bridges the gap between theoretical PQC research and real-world document verification system.

III. Proposed Framework

This Prototype ensures certificate authenticity, integrity, and non-repudiation of certificates even in the post-quantum era. It uses NIST-approved CRYSTALS-Dilithium2 digital signature algorithm for signing and verification [1], [2].

This model has four major parts:

1) *Key Generation* – Generates post-quantum key pairs using the Dilithium2 algorithm. A centralized authority (like government or regulatory committee) manages key creation to avoid key duplication. This authority provides a secure portal where certificate issuers can register themselves to get unique ID and key pairs. Certificate verifier can use the creator's id to get its corresponding public key to verify the certificate.
2) *Certificate Creation* – Generates digital certificate with embedded metadata and unique identifiers.
3) *Hashing and Signing* – Creates a hash of the certificate content and embeds the digital signature in the PDF metadata for tamper detection.
4) *Verification and Tamper Detection* – Extracts and verifies the embedded signature to confirm the document's integrity.

Fig. 1, shows the work flow of key generation, database, certificate creation and certificate verification works together through secure portal.

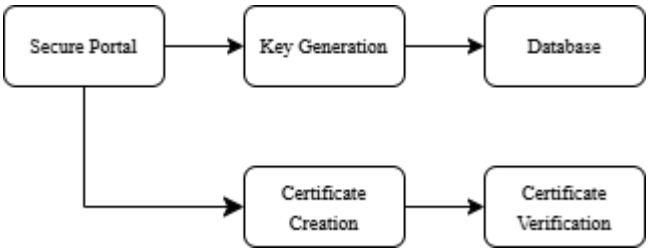

Fig. 1. Overall Process Flow

A. Key Generation

Key Generation is the foundation of this framework. It uses the CRYSTALS-Dilithium2 algorithm from Open Quantum Safe (OQS) library to create key pairs [3].

Private Key is used by the certificate issuer (like university or organization) to sign certificates and Public Key is used to verify the signature.

Algorithm 1: Key Generation using CRYSTALS-Dilithium2

1) Choose Dilithium2 parameter set,
2) Use OQS API to generate key pair,
3) Encrypt and Save keys,
4) Assign unique issuer ID,
5) Share the private key and issuer ID with the certificate creator and share the corresponding public key with the certificate verifiers,
6) Maintain encrypted backups of the key pairs.

The generated key pair gives 128-bit quantum-resistant security.

B. Certificate Creation

Certificate Creation generates digital certificates in Portable Document Format (PDF) using FPDF and PyPDF2 libraries in Python.

Each certificate includes key attributes like

1) Issuer name and organization,
2) Recipient name and ID,
3) Course details,
4) Date of issuance,
5) Unique certificate identifier.

Metadata fields like Payload, Signature, Fingerprint are added to PDF.

This design helps to detect any tampering in the visual elements, or metadata during the verification process.

C. Hashing and Signing

Once the certificate is created, its visual content is hashed using SHA-256 algorithm. Generated hash will be placed in a payload. Payload is then signed using the issuer's Dilithium2 private key to produce a post-quantum digital signature. The signature, along with the payload and algorithm identifier will be embedded into the PDF's metadata fields.

Algorithm 2: Certificate Signing and Embedding

1) Generate hash of the certificate visual content,
2) Generate a payload with the content and hash,
3) Load issuer's private key,
4) Generate digital signature of that payload,
5) Embed payload and signature into the PDF metadata,
6) Save the signed certificate file.

D. Verification and Tamper Detection

Verification and Tamper Detection allow verifiers to validate the authenticity and integrity of the certificate.

The verifier extracts the embedded signature, payload, and metadata from the received PDF. First, Signature is verified using the issuer's public key and payload bytes, Secondly, New hash is generated and compared with the hash stored in the payload. A valid signature confirms the payload has not been modified. Therefore, the hash stored inside the payload can be trusted. If both the newly generated hash and hash stored in payload hash are equal then the certificate is valid else not.

Algorithm 3: Certificate Verification and Tamper Detection

1) Extract signature, and Payload from the certificate,
2) Load issuer's public key,
3) Verify signature with the payload bytes, and issuer's public key,
4) Generate new hash of the certificate visual content,
5) Comparing the newly generated hash and the hash stored in the payload
6) If both, the signature and newly generated hash are valid then the certificate is authentic, otherwise the certificate is tampered.

This process ensures tamper detection and authenticity validation. Any changes in visual content, or metadata will lead to verification failure.

IV. Implementation and Results

The Post-Quantum-Ready Prototype for Standardized Certificate Creation and Verification is built using Python 3.10 on a Windows 11, 64-bit system. It uses the OQS library for Dilithium2 signatures and standard Python libraries like FPDF, PyPDF2, and hashlib for certificate creation, PDF handling, and hashing [3]. This prototype is designed to simulate real world certificate issuance and verification process workflow securely.

A. Implementation Environment Details

Table I presents the implementation environment used to develop and evaluate the prototype.

TABLE I
IMPLEMENTATION ENVIRONMENT DETAILS

Component	Specification
Programming Language	Python 3.10
Operating System	Windows 11 (64-bit)
Hardware	Intel i7 & 16 GB RAM
Cryptographic Library	OQS
Post-Quantum Algorithm	CRYSTALS-Dilithium2
Hash Algorithm	SHA-256
PDF Libraries	FPDF, PyPDF2

B. System Architecture and Workflow

The system architecture and workflow in Fig.2, shows how the secure portal manages certificate creation and certificate verification, and how the individual modules interact.

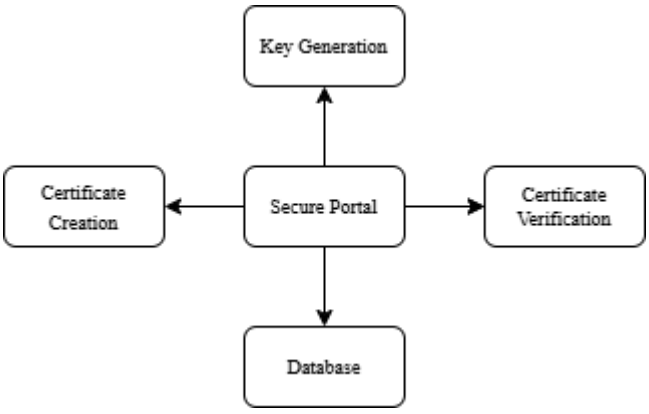

Fig. 2. System architecture and workflow

C. Functional Demonstration

This section demonstrates the complete workflow of the proposed prototype using real execution outputs and certificate samples. Demonstration includes key generation, certificate creation, verification of genuine certificates, tampering, and verification results of tampered certificates.

1. *Key Generation* – Key generation module generates a unique Dilithium2 keypair for the certificate issuer. The following files are generated:
   a. Public Key (.pub),
   b. Secret Key (.sk),
   c. PEM encoded public key (.pub.pem).

Fig. 3. Files Generated by Key Generation

2. *Certificate Generation* – Certificate Generation module produces a digital certificate with:
    a. Complete visual content,
    b. Metadata fields,
    c. Visual text-hash,
    d. Payload,
    e. Dilithium2 digital signature.

Fig. 4. Files Generated by Certificate Generation

Fig. 5. Certificate Generated by Certificate Generation

3. *Verification of Genuine Certificate* – The signed certificate is validated using the issuer's public key. As expected, the verification process confirms the authenticity and integrity of the certificate.
    a. Signature verification: **Successful**
    b. Payload hash: **Matched**
    c. Verdict: **VALID**

Fig. 6. Terminal Output of Verifying a Genuine Certificate

4. *Tampering a Genuine Certificate* – An attacker may attempt various forms of tampering:
    a. Modifying only the visual text,
    b. Modifying both visual text and metadata,
    c. Removing embedded payload/signature metadata.

Fig. 7. Tampered Certificate

5. *Verification of Tampered Certificate* – For all tampering attempts like,
    a. Visual text only tampering,
    b. Visual and metadata tampering,
    c. Removing required embedded metadata.

The verification system detects inconsistence in signature, payload hash, or missing metadata. For those tampered certificates,

    a. Signature verification: **Failed**,
    b. Payload has: **Mismatched,**
    c. Missing metadata: **Error reported,**
    d. Verdict: **INVALID**

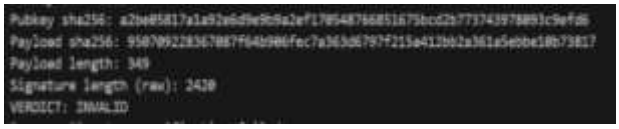

Fig. 8. Terminal Output of Verifying a Tampered Certificate

D. QR-Based Physical-Digital Link

Each certificate can be issued in both physical and digital formats. For physical certificates, the QR code in the certificate will point to the digital copy stored in the issuer's portal. The verification can be performed by the same process to check authenticity and integrity of those certificates. This approach creates a bidirectional trust link between physical and digital format certificates.

E. Performance Evaluation

This framework is evaluated based on execution time, file size, and verification accuracy.

TABLE II
PERFORMANCE DETAILS

Parameter	RSA-2048	ECDSA-P256	Dilithium2 (Proposed)
Key Size (Public/Private)	256 bytes / 1190 bytes	64 bytes / 96 bytes	1312 bytes / 2528 bytes
Signature Size	256 bytes	64 bytes	2420 bytes
Signing Time	1.45 ms	0.95 ms	1.73 ms

Verification Time	1.02 ms	0.88 ms	1.21 ms
Quantum Resistance	No	No	Yes
Tamper Detection	Yes	Yes	Yes
Tamper Detection Post-Quantum	No	No	Yes
Final signed PDF size	4.8-5.0 MB	4.8 MB	4.8-5.2 MB

F. Experimental Results

The experiment results confirm that the proposed prototype successfully:

1) Generates post-quantum key pairs using Dilithium2 algorithm,
2) Creates digital certificates with verifiable metadata,
3) Embeds payload, hash and post-quantum signature in PDF,
4) Detects any modification or alteration of content, or metadata accurately.

Verification accuracy is 100% and tampered certificates are marked as **INVALID** in verification.

CONCLUSION

The proposed Post-Quantum-Ready prototype provides secure and scalable framework for certificate creation and verification in the quantum era. It uses Dilithium2 signature algorithm and ensures long term resistance against quantum threats [1], [2].

This prototype achieves full document integrity and authenticity by implementing modules like key generation, certificate creation, hashing and signing, and certificate verification with tamper detection.

The system detects any form of tampering in the certificate content, or metadata and ensures authenticity, and integrity. Experimental results also confirm that the proposed system is quantum resistant. The use of SHA-256 for hashing and Dilithium2 for signature generation strengthens security and follows FIPS 204 and NIST PQC standards [2], [5], [6].

This makes it a step forward toward quantum secure certification systems. As quantum computing is growing rapidly, frameworks like this help to preserve the authenticity, integrity, and trust in the certification ecosystem.

The QR code in the certificate connects the physical and digital versions of each certificate, allows quick and unified verification across formats.

REFERENCES

[1] C. Dang, J. Hoffstein, T. Poppelmann, et al, "CRYSTALS-Dilithium: Digital Signatures from Module Lattices," *Proceedings of the NIST Post-Quantum Cryptography Standardization Conference, 2020.*
[2] National Institute of Standards and Technology (NIST), "FIPS 204: CRYSTALS-Dilithium Digital Signature Algorithm (Draft)," *Federal Information Processing Standard (FIPS), Aug. 2023.*
[3] Open Quantum Safe (OQS) Project, "liboqs: Open Quantum Safe Library," 2024.
[4] P.W. Shor, "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," *Proceedings 35th Annual Symposium on Foundations of Computer Science,* Santa Fe, NM, USA, 1994, pp.124-134.
[5] National Institute of Standards and Technology (NIST), "Post-Quantum Cryptography Standardization Process," *NIST Computer Security Resource Center (CSRC),* 2022.
[6] G. Bertoni, J. Daeman, M. Peeters, and G.Van Assche, "The Keccak Reference: SHA-3 Hash Algorithm," *NIST Computer Security Division,* 2015.
[7] M. Mosca and D. Stebila, "Quantum-Safe Cryptography and Security: An Introduction, Benefits, and Impact," *ETSI White Paper No, 8,* June 2015.
[8] C. Adams and S. Lloyd, "Understanding PKI: Concepts, Standards, and Deployment Considerations," 2nd ed, Addison-Wesley Professional, 2003.
[9] A. Hulsing, J. Rijneveld, F. Schwabe, and D. Butin, "XMSS: Extended Hash-Based Signatures," *IETF RFC 8391, May 2018.*
[10] B. Kaliski and Y. Liao, "Public Key Infrastructure and Certificate Management in the Post-Quantum Era," *IEEE Security & Privacy,* vol. 20, no. 4, pp. 56-65, July-Aug. 2022.
[11] P. Schwabe, T. Oder, and S. Gueron, "Efficient Implementation of Post-Quantum Signatures: A Case Study on CRYSTALS-Dilithium," *IEEE Transactions on Computers,* vol. 69, no. 11, pp. 1659-1670, Nov. 2020.