6.5610 Applied Cryptography Project Report

Hybrid Post-Quantum Variation of OpenID

Zitong Chen Shreya Thipireddy
Chon Hou (Jophy) Ye
{zitongc, shreyat, jophyyjh}@mit.edu

May 14, 2025

Abstract

With the dawn of quantum computing, widely deployed cryptographic authentication systems like OpenID Connect (OIDC) face increasing risks due to their reliance on classical public-key algorithms. As organizations move to adopt post-quantum secure cryptographic schemes, they must confront challenges related to interoperability, backward compatibility, and the weakening of assumptions such as factoring and discrete logarithms.

To address these challenges, our team built a hybrid post-quantum OIDC authentication system that integrates both classical and quantum-resistant cryptographic primitives. Our implementation combines ECDH (P-256) with Kyber512 for hybrid key exchange and employs a dual-signature scheme—ES256 and Falcon512—to secure JSON Web Tokens (JWTs). Built using the open-source libraries liboqs and oqs-provider, the system adheres to current cryptographic standards while enhancing OIDC's resilience against quantum attacks.

We tested the full authentication flow locally using Flask-based Relying Party (RP) and OpenID Provider (OP) servers. The RP's authorization endpoint responded in approximately 41ms, and the OP's token endpoint—performing a full hybrid TLS handshake—completed in around 27ms. These results demonstrate that integrating post-quantum cryptography into OIDC is both feasible and efficient under realistic conditions.

By open-sourcing our implementation and performance benchmarks, we aim to provide a practical foundation for organizations seeking to adopt quantum-safe identity systems without compromising existing infrastructure or user experience.

1 Motivation

1.1 Why OpenID Connect?

OpenID Connect is a universally accepted third-party authentication protocold. It is based on *Transport Layer Security* (TLS) for network traffic protection between the application, the third-party identity provider, and the user, and *JSON Web Token* (JWT), as a proof of authentication of the user and the authorization of user resources for the application, which the application can later use to access the user's resources. The choice

of OpenID Connect for this project is two-fold: it is a critical, widely adopted authentication protocol, and its multi-round network communication nature makes it a natural candidate for analysis.

1.2 Why hybrid schemes instead of purely post-quantum schemes?

NIST has been running the Post-Quantum Cryptography Standardization program for almost a decade now and has had several rounds that the proposed algorithms need to make it through during which they will be publicized so that they can be researched and eliminated in case of any vulnerabilities found. After 4 rounds, they have standardized CRYSTALS-Kyber and HQC for key encapsulation mechanism (KEM) algorithms and CRYSTALS-Dilithium, FALCON and SPHINCS+ for signature algorithms.

A pre-print paper "Quantum Algorithms for Lattice Problems" [1], released last year in 2024, alarmed the research community as it proposed that the LWE problem could be solved. As the hardness of the LWE problem forms the basis of many lattice-based cryptographic schemes, this could have upended much of the previous research in the field and set back post-quantum standardization significantly.

Up to that point, most of the promising algorithms that had passed several rounds in the NIST standardization program and were marked for standardization were dependent on lattice-based assumptions, and having one of the core lattice-based assumptions, hardness of the LWE problem, being questioned would have suggested vulnerabilities in the algorithms proposed for standardization.

The initial winners of CRYSTALS-Kyber for key encapsulation mechanism (KEM) algorithms and CRYSTALS-Dilithium, FALCON and SPHINCS+ for signature algorithms were announced in 2022 and are all lattice-based except for SPHNCS+. Therefore, for the next round, NIST called for algorithms that came from assumptions other than lattice-based to account for the case that future research could break some types of assumptions, so widening their search of algorithms to different bases could come to be prudent.

The quantum algorithm proposed by that paper, Quantum Algorithms for Lattice Problems, was found to have a bug in one of the steps that was not easy to fix, so the LWE problem has not been solved in polynomial time as of yet. However, this emphasized that cryptographic research is ever changing, and depending on algorithms based on one still-to-be-tested assumption is not reliable. Further, we need more time to keep studying both the proposed post-quantum schemes and the assumptions these schemes are based on to improve trust in purely post-quantum systems. Even if the assumptions hold, cryptographers have not studied implementations long enough to catch major side-channel attacks.

While the amount of time and effort spent on standardization of post-quantum algorithms might seem long, it is no match to the time spent by the field in general on studying classic schemes, which comes to upwards of 50 years. [2] It follows that post-quantum schemes have not had had enough time to be fully studied and therefore to be trusted. Furthermore, post-quantum alternatives have not stood the test of time as factoring-/discrete-log- based algorithms.

In the process of upgrading to post-quantum secure cryptographic schemes, interoperability and backward compatibility will be important issues. In the meantime, transitioning towards a post-quantum secure future will still involve systems relying on widely-researched classical schemes as a basis to affirm that the system will be at least as secure as the guarantees provided by the classical scheme. Yet, as the need of transition work

is pressing, hybrid schemes have been brought into play. Hybrid schemes secure systems using specific combinations of classical and post-quantum schemes to assure that even if the chosen post-quantum crypto-system ongoing research faces any obstacles, the system and the encryption can still rely on the classical schemes' security before the advent of cryptographically relevant quantum computers.

Moreover, we hypothesize that the overhead of including classical algorithms to post-quantum algorithms will be minimal because post-quantum schemes have much larger key sizes. In fact, a hybrid key exchange combining X25519 with post-quantum key encapsulation mechanism (KEM) ML-KEM-768 has a hybrid public key less than 3% larger than ML-KEM-768 alone. Similarly, a hybrid signature scheme combining ECDSA with ML-DSA-44 has a hybrid signature size less than 3% larger. [3]

2 Background

2.1 Post-quantum Cryptography

With the rise of computing power and the dawn of quantum computing, National Institute of Standards and Technology (NIST) has made calls for post-quantum cryptographic scheme submissions in their work towards standardization of post-quantum algorithms. After several rounds over the years, they have standardized CRYSTALS-Kyber and HQC for key encapsulation mechanism (KEM) algorithms and CRYSTALS-Dilithium, FALCON and SPHINCS+ for signature algorithms. However, industry adoption of post-quantum cryptography (PQC) into all their products will be an arduous task in the coming years. In the process of upgrading to post-quantum secure cryptographic schemes, interoperability and backward compatibility will be important issues. Furthermore, post-quantum alternatives have not stood the test of time as factoring-/discrete-log- based algorithms.

2.2 OpenID Connect

OpenID Connect (OIDC) is the standard third-party authentication and authorization protocol in the web. Building on top of the OAuth 2 protocol, it enables the end application, i.e., the *relying party* (RP) (or *client*), to verify the authenticity/identity of a user, i.e., the *resource owner*, through a third-party *identity provider* (IdP) such as Google, Apple, or Github, and access a restricted set of authorized resources on behalf of the user, such as the profile information of the user. [4]

In a typical *authorization code grant flow*, as shown in Figure 1, when the RP requires verification of user identity, a user is redirected to the IdP's authentication endpoint. After successful authentication, the browser receives a temporary authorization code from the IdP, which it can then forward to the RP. The RP then exchanges the authorization code and a client secret for an access token and an ID token, which are used to access protected resources on behalf of the user.

The only public-key cryptographic systems in the flow are Transport Layer Security (TLS) and JSON Web Tokens (JWT). JWTs are asymmetrically signed tokens given to the RP for future access to the protected resources. It is signed by the IdP with a private key and is verifiable by any *resource server* (and even the RP) with the public key. All network traffic is protected by TLS, which starts with a *handshake* to perform *key establishment* using an asymmetric key suite to obtain a shared secret key to be used

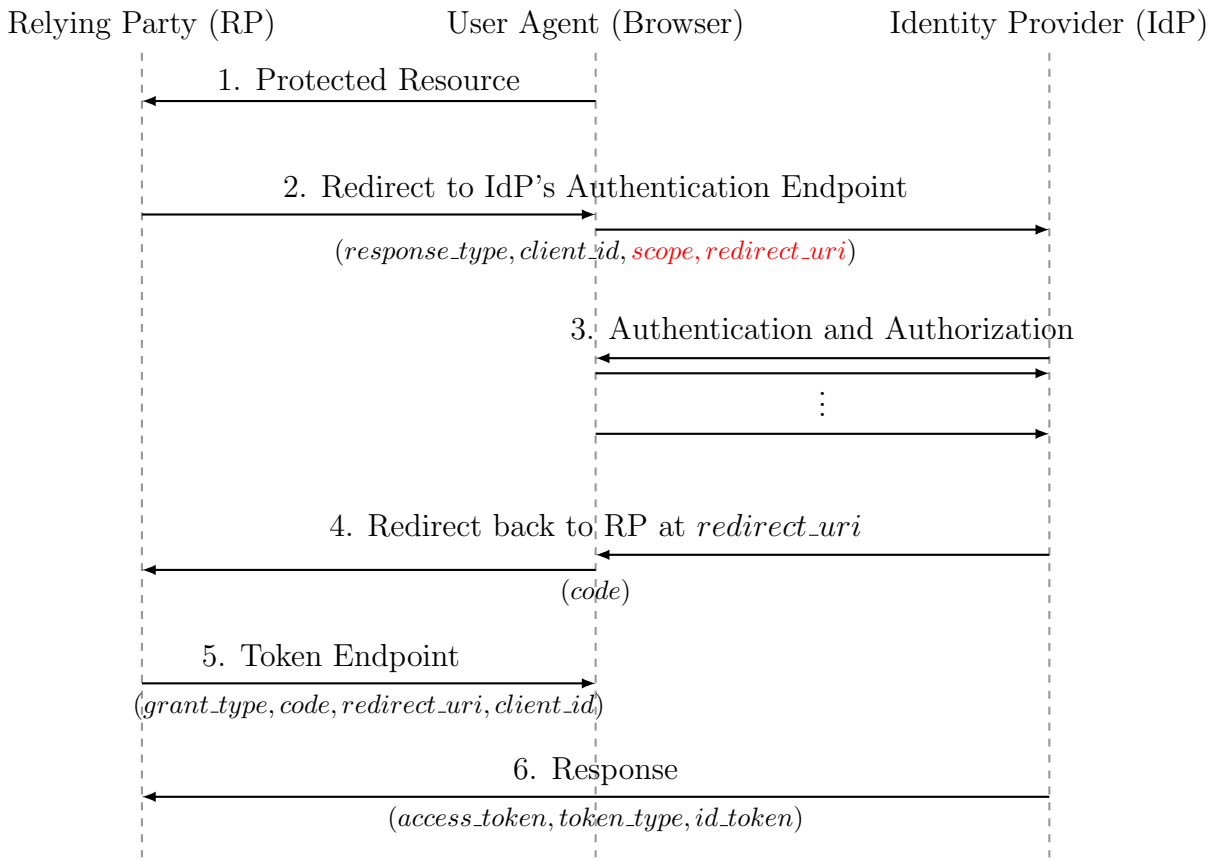

Figure 1: OIDC Authentication Flow

with a symmetric cipher for actual communication. [4] Symmetric ciphers are considered quantum-safe as long as keys are of at least 128 bits.

For post-quantum key exchange, people often study *key encapsulation mechanisms* (KEMs), first introduced by Shoup (2000) [5]. The motivation is that public-key encryption is rarely used to encrypt every message due to its larger key and ciphertext size, but is often used for session key establishment in the beginning. KEMs abstract the use of public-key encryption for key establishment. Mathematically, KEM is a triple of algorithms $(KeyGen, Encap, Decap)$, where

- $KeyGen()$ generates (pk, sk), a public key and a private key pair;
- $Encap(pk)$, a randomized algorithm run by a client, takes pk and outputs a shared symmetric session key k to be kept and a ciphertext c of k to be sent to the server;
- $Decap(sk, c)$, run by the server, outputs the shared key k. [6]

Although a public-key encryption scheme can be trivially used to construct a KEM (by public-key encrypting a random session key k to send to the server), it is common in applied cryptography to build KEMs for constructing post-quantum public-key encryptions.

2.3 Post-Quantum Hybridization and Security Guarantees

Hybrid cryptographic schemes in the post-quantum context refer to the use of both a classical algorithm in combination with a post-quantum algorithm. The combination of

classical and post-quantum algorithms could follow two main methods, a layered approach or a composition approach. As for the security of these two approaches, in basic terms, when a post-quantum algorithm is discovered to be broken by either classical or quantum computers, the inner classical signature can still guard against classical adversaries.

2.3.1 Layered Hybrid Schemes

A layered approach works by somehow nesting a classical algorithm within a post-quantum algorithm.

- KEM schemes: One proposed method for a nesting type of combination of KEM schemes is to have the resulting ciphertext be the concatenation of the outer KEM's ciphertext with an inner KEM's ciphertext that is encrypted with the "outer" KEM's secret key derived from its shared secret. And the final shared secret is the result of the shared secrets of the "inner" and "outer" KEMs being passed through a split-key PRF. [7]

- Signature schemes: One possible nesting is using a post-quantum algorithm to sign the resulting classical signature of a message. A stronger nesting method for hybridization is for the post-quantum algorithm to sign both the classical signature along with the original message. [8]

The following outlines the security of layered approach for KEM and signature schemes.

- KEM: CCA security is achieved since a split-key PRF is used to create the final shared secret. The ciphertext will be indistiguishable from random as long as the outer KEM uniform guarantee holds; since the outer KEM ciphertext is itself uniform and so is the inner KEM ciphertext since it is encrypted with the outer KEM's shared secret. [7]

- Signature: The strong nesting method can get guaranteed strong unforgeability as long as the outer signature scheme (the one that signs both the message and the first classically signature) has strong unforgeability. This holds even if the first signature is only existentially unforgeable. [9]

2.3.2 Composite Hybrid Schemes

A composition approach works by combining the objects from the classical and the post-quantum algorithm.

- KEM: Independently generate the key suites (pk_1, sk_1) and (pk_2, sk_2) from each type of KEM and pass these results to a *key derivative function* (KDF). As an example, [10] takes the exclusive or (XOR) of the two session keys and a concatenation of the two ciphertexts for encapsulation:

$$\begin{gathered}(c_1, k_{KEM,1} \| k_{MAC,1}) \leftarrow Encap_1(pk_1), \quad (c_2, k_{KEM,2} \| k_{MAC,2}) \leftarrow Encap_2(pk_2), \\ k_{KEM} = k_{KEM,1} \oplus k_{KEM,2}, \qquad \text{(shared symmetric key)} \\ c = (c_1, c_2), \quad k_{MAC} = (k_{MAC,1}, k_{MAC,2}). \qquad \text{(sent to server)}\end{gathered}$$

  A server may XOR $Decap(sk_1, c_1)$ and $Decap(sk_2, c_2)$ to obtain k_{KEM}. [10] showed that the scheme is IND-CCA secure if either KEM is IND-CCA secure. The addition

of a one-time unforgeable MAC ensures security against an *active adversary* (with access to the decapsulation oracle) cannot forge a k_{KEM}-k_{MAC} pair by mixing-and-matching multiple ciphertexts.

Alternative constructions rely on dual-PRFs, which satisfy PRF security when either the key or the message has enough entropy. [11, 12]

- Signature: Independently sign using the classic scheme and post-quantum scheme and send both to verifier; verifier only accepts if both signatures are valid.

The following outlines the security of composite approach for KEM and signature schemes.

- KEM: security depends on the method of combination used. For example, if simple concatenation is used, and the ciphertexts of each type of KEM are just concatenated to each other, the indistinguishability from randomness can be broken if either one portion can be identified as not random with probability greater than $\frac{1}{2} + \epsilon$ which is possible if just one of the portions is recalculated. [7]

- Signature: composite signature is deemed invalid if the verification of either part of the composite signature fails, so it follows that the security of the composite approach is at least as strong as the weaker of the two types of signature algorithms. For example, if concatenating signatures from Dilithium (a strongly unforgeable scheme) with one from ECDSA (existentially unforgeable scheme), the resulting hybrid will only have the weaker security guarantee of existential unforgeability. [9]

2.3.3 Deployed Hybrid Schemes

Hybridization in use in current systems follows a form of composition, a simple concatenation. This type of composition is also used in the construction of hybrid schemes that are provided for use through the open-source libraries that we utilized in our project implementations.

For example, hybrid KEM in TLS 1.3, as proposed in an internet draft, looks like a simple concatenation of the different algorithms' results and transmitting these as one. [12] The client side would receive a concatenation of the public keys generated from the KeyGen functions, and the server side would keep the concatenation of the ciphertext outputs of the encapsulation method of the KEMs. Lastly, the shared secrets are also concatenated together and run through a HKDF function (HMAC based key derivative function). One aspect of the security for this approach is using only constant length values for all public keys, ciphertexts and shared secrets. Variable length of the keys could lead to timing side channel attacks, and variable length of the shared secrets could detract from the collision resistant properties of the HKDF. Given that the shared secrets are of fixed length, however, this method was shown to be secure when the HKDF utilized the dual-PRF combiner. [12] This hybrid version of TLS has been supported in Chrome. [13]

3 Other Related Works

A prior work that inspired our idea for this project came from the paper Post-Quantum Electronic Identity: Adapting OpenID Connect and OAuth 2.0 to the Post-Quantum Era.[14] This 2022 paper outlines research on making OpenID and OAuth 2.0 secure

for post-quantum usage. OpenID is built on top of OAuth 2.0 with TLS and JWT forming major components of both, so this research also largely involved making these more ubiquitous protocols capable of using post-quantum cryptographic schemes, since at the time of the paper, deployed TLS and JWT protocol usage was still based entirely on classical cryptography.

After describing their literature survey on "identify[ing] existing solutions to [the] problem... of a record-now-decrypt-later attacker retrieving and subsequently impersonating users with these tokens", they detail the changes made for their post-quantum secure implementation. They used NIST standardized PQC KEX algorithm (Crystal-KYBER) to update the classical TLS and PQC signature algorithms (Dilithium, Falcon, SPHINCS+) for JWT. In addition, they outlined a testing method using a reproducible containerized testing environment built as part of their implementation.

4 Benchmarking Various Hybrid Scheme Pairs

While open source libraries like liboqs offer hybrid schemes, we created Golang scripts to benchmark KEM hybrids and signature hybrids to test the performance and memory usage of more pairs of classic and post-quantum algorithms. This manual hybridization script focused on using the concatenation approach for both KEM and signature algorithms.

The list of various statistics can be found in the Appendix A. For the KEM pairs, times for key generation, encapsultation and decapsulation are included, and sizes for the hybrid public and private keys, shared secret adn ciphertext are also included. For the signature pairs, times for key generation, signing and verifying are included as well as the signature size and the rate of successful verification validity of both the classic and post-quantum signatures.

In the end, even though we have tested the results of various pairs of hybrid schemes, our follow-up implementation of the complete hybrid OpenID only uses one pair type for each of the KEM and signature hybrid schemes due to the increased complexity of the full protocol implementation.

5 Implementation and Hybrid Communication Flow

In this section, we introduce the architecture and communication flow of our hybrid OpenID Connect (OIDC) authentication system. Our design aims to ensure secure authentication that is resilient against both current classic adversaries and future post-quantum adversaries.

5.1 Libraries

Our implementation adopts the Open Quantum Safe (OQS) project's open source libraries to integrate post-quantum cryptography with classic security schemes. In particular, we used:

liboqs[15]: An open-source C library providing implementations of quantum-safe key encapsulation mechanisms and digital signature algorithms. It offers a common API for these algorithms, and we used the liboqs implementation of the hybrid ES256 + Falcon512 hybrid signature scheme and the hybrid TLS KEM P-256 + Kyber512 scheme.

oqs-provider[16]: A provider module based on OpenSSL 3.0 that allows the modular use of quantum-safe algorithms from liboqs within the standard OpenSSL framework. Integrating oqs-provider modules allowed us to deploy hybrid key exchange and signature schemes in TLS 1.3, X.509, and S/MIME protocols.

Utilising these tools, our system achieves a balance between current operational performance and long-term security sustainability, providing a practical path forward for organizations preparing for emerging quantum threats without sacrificing usability or interoperability.

5.2 Hybrid OIDC Authentication Flow

This section details the structure of our hybrid OIDC authentication system from the perspective of a user's journey through the authentication process. A diagram depicting the flow of information in the system is shown in the diagram below Fig. 2.

Upon initiating the client application, the user's browser establishes TLS sessions with both the OpenID Provider (OP) and the Relying Party (RP). While traditional TLS configurations employ classical elliptic-curve Diffie–Hellman (ECDH) to perform key exchange, our system adopt a hybrid key-exchange mechanism combining ECDH over curve P-256 with the Kyber512 Key Encapsulation Mechanism (KEM). This hybrid scheme combines the classic and efficient ECDH with the quantum-resistant properties of Kyber512, and the shared secret secure generated is secure against both classical and quantum threats.

Following TLS establishment, the user can now initiate the login process, prompting the RP to redirect the browser to the OP's authorization endpoint over the hybrid-secured channel. The OP then authenticates the user through standard methods (such as credentials or multi-factor authentication). The process remains secure within the hybrid TLS framework.

Upon successful authentication, the OP issues an authorization code and redirects the user back to the RP. The RP exchanges this authentication code for user ID tokens at the OP's token endpoint. Again, all communications between RP, OP and client application are secured via the hybrid TLS channel. Confidentiality and integrity are maintained throughout the process.

When the token is successfully exchanged, the OP generates an ID token in the form of a JSON Web Token (JWT) that is signed with dual signatures: the classical ECDSA (ES256) and the post-quantum Falcon512 algorithm. The RP retrieves the OP's public keys through the JSON Web Key Set (JWKS) endpoint and verifies both signatures. Finally, access is granted to the user only upon successful verification of both signatures, ensuring the token's authenticity and integrity against both classical and quantum adversaries.

5.3 Hybridization of Classical and Post-Quantum Schemes

As our implementation combines classical cryptographic algorithms with post-quantum cryptographic (PQC) algorithms, it is important to discuss how two different schemes are hybridized to safeguard security promises. Our hybridization method follows a straightforward concatenation strategy, aligning with relevant IETF Internet-Drafts [12] and supported by Open Quantum Safe project's libraries liboqs[15] and oqs-provider[16].

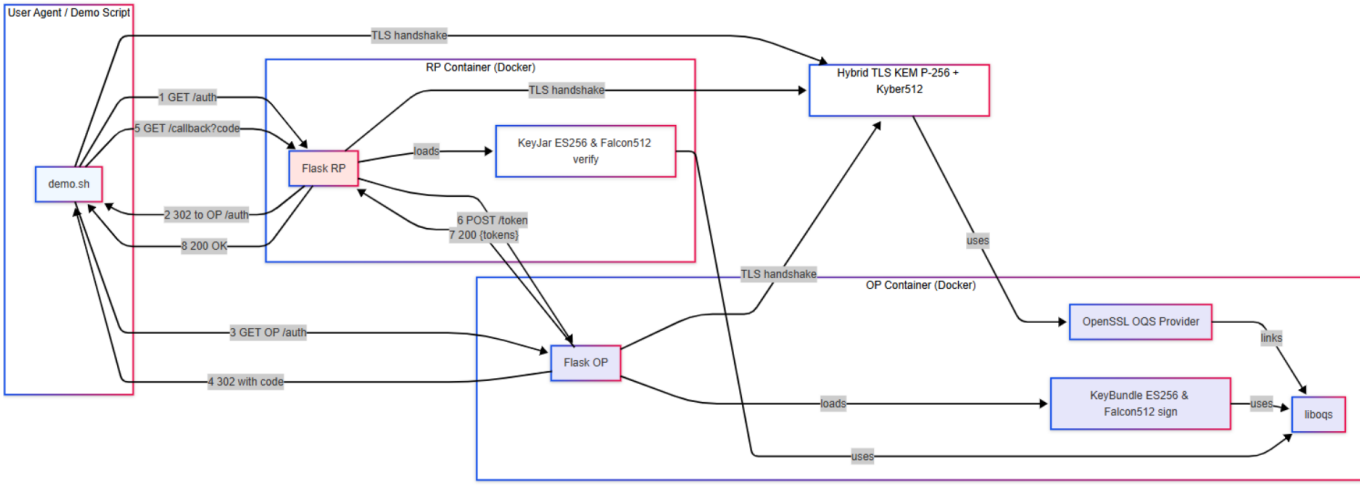

Figure 2: Structure Diagram of Hybrid OIDC Authentication Flow

5.3.1 Hybrid Key Exchange (KEX)

Our TLS 1.3 handshake employs hybrid key exchange by concatenating classical elliptic-curve Diffie–Hellman (ECDH, P-256) and post-quantum Kyber512 Key Encapsulation Mechanism (KEM) public keys. The resulting shared secret concatenates the classical components and the PQ [16].

5.3.2 Hybrid Digital Signatures

The JWT signatures are generated using a hybrid combination of classical ECDSA (ES256) and the Falcon512 PQ algorithm. Following the standards for composite signatures, classical and PQ signatures are simply concatenated. The verifying party independently validates two signatures using corresponding public keys obtained via JWKS [16].

5.3.3 Integration into Cryptographic Standards

The OQS-provider integrates hybrid schemes into standard cryptographic formats, including TLS handshake structures used in our project, X.509 certificates, and PKCS#8 private keys. All hybrid structures use straightforward OCTET_STRING concatenation encoding, simplifying adoption and deployment [16].

Our concatenation-based hybrid approach is simple, but it provides interoperability. It is easy to deploy and creates minimal computation overhead. Overall, it is a reliable step in our transition into a post-quantum safe future.

6 Testing and Discussion

We have created a functional OIDC system built on top of hybrid KEM and hybrid JWT signature scheme. Our system works headlessly, and we tested our system's end-to-end communication using the terminal as the "client application", and 2 ports running locally as the Relying Party and the OpenID Provider.

In our preliminary benchmarks, the RP's */auth* endpoint (HTTP) consistently responded in about 41ms, reflecting only the Flask routing and Docker network overhead. The OP's */token* endpoint (HTTPS) recorded roughly 27ms per request, performing a full TLS handshake with Kyber512 KEM + ECDH (P-256) and executing the token-endpoint logic. Overall, RP redirect remained sub-50ms, and the end-to-end OP processing (including post-quantum key operations) stayed under 30ms. Further evaluating our testing results, we concluded that our hybrid-PQC layers added a tolerable small latency to the system with majority of the computation overhead spent on executing the PQC schemes.

7 Future Work

With our hybrid-PQC OpenID system, our goals for the future are to make post-quantum authentication more accessible and to validate its performance in more real-world scenarios. To that end, we will open-source both our implementation and system performance tests, allowing researchers to examine and adapt our system. We will also collect data on comprehensive performance metrics: end-to-end latency, CPU usage, token size, and memory footprint. This information would provide great insight into the trade-offs involved in deploying hybrid OIDC systems at scale. Finally, we plan to expand support for additional hybrid KEMs and signature algorithms (e.g., HQC, BIKE, Falcon, Dilithium), further enhancing the flexibility and security of our system.

From a broader perspective, our work seeks to bridge the gap transitioning from today's classical protocols to tomorrow's quantum-resistant systems. As quantum computing evolves at a steady speed, classical algorithms will become increasingly vulnerable. By combining post-quantum primitives with the familiar OIDC security schemes, we preserve interoperability and efficiency of the current OIDC implementation while upgrading it to be secure against future threats. Through open-sourcing our code, rigorous benchmarking, and supporting more PQC schemes, we hope to accelerate global adoption of hybrid post-quantum security schemes and help netizens around the world transition to a quantum-safe future without sacrificing usability or speed of the systems they use.

Lastly, it remains theoretically interesting to construct efficient hybrid schemes from the ground up which are secure when at least one of its component is secure. Although the overhead of adding classical counterparts is small, we have mainly focused on black-box hybridization in this work.

8 Member Contributions

- Zitong Chen: Completed the implementation of the hybrid OIDC authentication system; performed initial testing on the system's redirect-in latency and handshake latency; wrote the implementation (Section 5), testing (Section 6), and future work (Section 7) sections of the report; organized most team check-ins and in charge of most of written communications with our TA Katarina; kept track of the project timeline and sent reminders about deadlines and requirements.

- Shreya Thipireddy: For the report, wrote the abstract, motivation section on 'why hybrid schemes' (Section 1.2), the background sections of 'post-quantum cryptography' (Section 2.1) and 'post-quantum hybridization and security guarantees' (Section 2.3), the 'related works' section (Section 3), and the 'benchmarking various

hybrid scheme pairs' section (Section 4) with testing results listed in the appendix section 'hybrid pairs benchmarking results'. Conducted a literature review that formed most of the first half of the paper. Researched feasible libraries and integration to utilize previously implemented hybrid post-quantum schemes. Created a Golang testing scripts and benchmarked various hybrid schemes to add to our hybrid scheme analysis.

- Chon Hou (Jophy) Ye: Analyzed the communication between different parties and protocoles involved in OIDC. Completed background research on the security of different hybridization methods. Wrote Section 1.1 'Why OpenID Connect?', 2.2 'OpenID Connect', and co-wrote section 2.1 'Post-Quantum Cryptography', 2.3 'Post-Quantum Hybridization and Security Guarantees', 7 'Future Work', and the abstract. Made early attempts to set up containerized environments for the different parties involved in OIDC, which is then carried on by Zitong.

References

[1] Y. Chen, "Quantum algorithms for lattice problems," Cryptology ePrint Archive, Paper 2024/555, 2024. [Online]. Available: https://eprint.iacr.org/2024/555

[2] L. Chen and M. Scholl, "The cornerstone of cybersecurity – cryptographic standards and a 50-year evolution," 2022. [Online]. Available: https://www.nccoe.nist.gov/news-insights/cornerstone-cybersecurity-cryptographic-standards-and-50-year-evolution

[3] ETSI, "Cyber security (cyber); quantum-safe cryptography (qsc); deployment considerations for hybrid schemes," European Telecommunications Standards Institute, Technical Report TR 103 966, Oct. 2024, dTR/CYBER-QSC-0021. [Online]. Available: https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=64284

[4] F. Schardong, A. Giron, F. Müller, and R. Custódio, "Post-quantum electronic identity: Adapting openid connect and oauth 2.0 to the post-quantum era," in *Cryptology and Network Security*, ser. Lecture Notes in Computer Science, A. R. Beresford, A. Patra, and E. Bellini, Eds., vol. 13641. Cham: Springer, Nov. 2022, pp. 371–390, 21st International Conference, CANS 2022, Abu Dhabi, UAE, November 13–16, 2022, Proceedings.

[5] V. Shoup, "Using hash functions as a hedge against chosen ciphertext attack," in *Advances in Cryptology — EUROCRYPT 2000*, B. Preneel, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 275–288.

[6] S. Galbraith, *The KEM/DEM paradigm*. Cambridge University Press, 2012, ch. 23, pp. 471–478.

[7] F. Günther, M. Rosenberg, D. Stebila, and S. Veitch, "Hybrid obfuscated key exchange and KEMs," Cryptology ePrint Archive, Paper 2025/408, 2025. [Online]. Available: https://eprint.iacr.org/2025/408

[8] N. Bindel, U. Herath, M. McKague, and D. Stebila, "Transitioning to a quantum-resistant public key infrastructure," Cryptology ePrint Archive, Paper 2017/460, 2017. [Online]. Available: https://eprint.iacr.org/2017/460

[9] D. Ghinea, F. Kaczmarczyck, J. Pullman, J. Cretin, S. Kölbl, R. Misoczki, J.-M. Picod, L. Invernizzi, and E. Bursztein, "Hybrid post-quantum signatures in hardware security keys," Cryptology ePrint Archive, Paper 2022/1225, 2022. [Online]. Available: https://eprint.iacr.org/2022/1225

[10] N. Bindel, J. Brendel, M. Fischlin, B. Goncalves, and D. Stebila, "Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange," 2018, publication info: Published elsewhere. Major revision. 10th International Workshop on Post-Quantum Cryptography (PQCrypto 2019). [Online]. Available: https://eprint.iacr.org/2018/903

[11] M. Bellare, "New proofs for NMAC and HMAC: Security without collision-resistance," in *Advances in Cryptology - CRYPTO 2006*, ser. Lecture Notes in Computer Science, C. Dwork, Ed., vol. 4117. Heidelberg: Springer, Aug. 2006, pp. 602–619.

[12] D. Stebila *et al.*, "Hybrid key exchange in tls 1.3," Internet Engineering Task Force, Internet-Draft draft-ietf-tls-hybrid-design-12, Jan. 2025, expires July 18, 2025. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-tls-hybrid-design-12

[13] E. Bursztein and F. Kaczmarczyck, "Toward quantum resilient security keys," 2023. [Online]. Available: https://security.googleblog.com/2023/08/toward-quantum-resilient-security-keys.html

[14] F. Schardong, A. A. Giron, F. L. Müller, and R. Custódio, "Post-quantum electronic identity: Adapting openid connect and oauth 2.0 to the post-quantum era," in *Cryptology and Network Security*, A. R. Beresford, A. Patra, and E. Bellini, Eds. Cham: Springer International Publishing, 2022, pp. 371–390.

[15] D. Stebila, M. Mosca, and contributors, "liboqs: C library for quantum-safe cryptography," 2024, accessed: 2024-05-11. [Online]. Available: https://github.com/open-quantum-safe/liboqs

[16] D. Stebila and contributors, "oqs-provider: Quantum-safe algorithm provider for OpenSSL 3.0," 2024, accessed: 2024-05-11. [Online]. Available: https://github.com/open-quantum-safe/oqs-provider

A Hybrid Pairs Benchmarking Results

A.1 KEM

=== [X25519+ML-KEM-512] ===
KeyGen: 482.19µs
Encaps: 26.011µs
Decaps: 25.851µs

Total: 880.432µs
Hybrid key size: 832 bytes (Classical: 32, PQ: 800)
Hybrid private key size: 1664 bytes (Classical: 32, PQ: 1632)
Hybrid shared secret length: 32 bytes
Ciphertext length: 768 bytes
=== [X25519+KYBER512] ===
KeyGen: 24.422µs
Encaps: 21.998µs
Decaps: 15.481µs
Total: 387.615µs
Hybrid key size: 832 bytes (Classical: 32, PQ: 800)
Hybrid private key size: 1664 bytes (Classical: 32, PQ: 1632)
Hybrid shared secret length: 32 bytes
Ciphertext length: 768 bytes
=== [P-256+ML-KEM-512] ===
KeyGen: 17.658µs
Encaps: 14.239µs
Decaps: 16.647µs
Total: 272.745µs
Hybrid key size: 865 bytes (Classical: 65, PQ: 800)
Hybrid private key size: 1664 bytes (Classical: 32, PQ: 1632)
Hybrid shared secret length: 32 bytes
Ciphertext length: 768 bytes
=== [P-256+KYBER512] ===
KeyGen: 14.636µs
Encaps: 16.649µs
Decaps: 11.851µs
Total: 271.241µs
Hybrid key size: 865 bytes (Classical: 65, PQ: 800)
Hybrid private key size: 1664 bytes (Classical: 32, PQ: 1632)
Hybrid shared secret length: 32 bytes
Ciphertext length: 768 bytes
=== [P-384+ML-KEM-768] ===
KeyGen: 27.307µs
Encaps: 21.453µs
Decaps: 24.165µs
Total: 278.371µs
Hybrid key size: 1281 bytes (Classical: 97, PQ: 1184)
Hybrid private key size: 2448 bytes (Classical: 48, PQ: 2400)
Hybrid shared secret length: 32 bytes
Ciphertext length: 1088 bytes
=== [P-384+KYBER768] ===
KeyGen: 16.969µs
Encaps: 20.196µs
Decaps: 13.471µs
Total: 196.758µs
Hybrid key size: 1281 bytes (Classical: 97, PQ: 1184)
Hybrid private key size: 2448 bytes (Classical: 48, PQ: 2400)

Hybrid shared secret length: 32 bytes
Ciphertext length: 1088 bytes
=== [P-384+ML-KEM-1024] ===
KeyGen: 27.362µs
Encaps: 23.667µs
Decaps: 25.179µs
Total: 1.360509ms
Hybrid key size: 1665 bytes (Classical: 97, PQ: 1568)
Hybrid private key size: 3216 bytes (Classical: 48, PQ: 3168)
Hybrid shared secret length: 32 bytes
Ciphertext length: 1568 bytes
=== [P-384+KYBER1024] ===
KeyGen: 30.069µs
Encaps: 26.504µs
Decaps: 19.415µs
Total: 1.226372ms
Hybrid key size: 1665 bytes (Classical: 97, PQ: 1568)
Hybrid private key size: 3216 bytes (Classical: 48, PQ: 3168)
Hybrid shared secret length: 32 bytes
Ciphertext length: 1568 bytes

A.2 Signature

=== [Ed25519+Dilithium2] ===
KeyGen avg time: 346.036µs
Sign avg time: 109.597µs
Verify avg time: 97.802µs
Avg signature size: 2484 bytes
Valid signatures: 100.0%
=== [Ed25519+Dilithium3] ===
KeyGen avg time: 66.834µs
Sign avg time: 144.205µs
Verify avg time: 95.709µs
Avg signature size: 3357 bytes
Valid signatures: 100.0%
=== [Ed25519+Dilithium5] ===
KeyGen avg time: 96.665µs
Sign avg time: 209.781µs
Verify avg time: 125.186µs
Avg signature size: 4659 bytes
Valid signatures: 100.0%
= = = [RSA_2048 + Dilithium2] = = =
KeyGen avg time: 122.797336ms
Sign avg time: 992.264µs
Verify avg time: 59.061µs
Avg signature size: 2676 bytes
Valid signatures: 100.0
=== [RSA_2048+Dilithium3] ===

KeyGen avg time: 117.918473ms
Sign avg time: 1.039301ms
Verify avg time: 75.954µs
Avg signature size: 3549 bytes
Valid signatures: 100.0%
=== [RSA_2048+Dilithium5] ===
KeyGen avg time: 127.850464ms
Sign avg time: 1.122242ms
Verify avg time: 107.045µs
Avg signature size: 4851 bytes
Valid signatures: 100.0%
=== [ECDSA_P256+Dilithium2] ===
KeyGen avg time: 54.433µs
Sign avg time: 131.09µs
Verify avg time: 113.72µs
Avg signature size: 2484 bytes
Valid signatures: 100.0%
=== [ECDSA_P256+Dilithium3] ===
KeyGen avg time: 71.558µs
Sign avg time: 157.294µs
Verify avg time: 136.384µs
Avg signature size: 3357 bytes
Valid signatures: 100.0%
=== [ECDSA_P256+Dilithium5] ===
KeyGen avg time: 109.096µs
Sign avg time: 232.675µs
Verify avg time: 180.79µs
Avg signature size: 4659 bytes
Valid signatures: 100.0%
=== [Ed25519+Falcon-512] ===
KeyGen avg time: 7.092674ms
Sign avg time: 286.077µs
Verify avg time: 120.069µs
Avg signature size: 719 bytes
Valid signatures: 100.0%
=== [Ed25519+Falcon-1024] ===
KeyGen avg time: 19.348759ms
Sign avg time: 498.339µs
Verify avg time: 154.808µs
Avg signature size: 1334 bytes
Valid signatures: 100.0%
=== [RSA_2048+Falcon-512] ===
KeyGen avg time: 118.558771ms
Sign avg time: 1.149987ms
Verify avg time: 75.59µs
Avg signature size: 910 bytes
Valid signatures: 100.0%
=== [RSA_2048+Falcon-1024] ===

KeyGen avg time: 140.695659ms
Sign avg time: 1.442385ms
Verify avg time: 130.651µs
Avg signature size: 1526 bytes
Valid signatures: 100.0
=== [ECDSA_P256+Falcon-512] ===
KeyGen avg time: 6.924798ms
Sign avg time: 289.829µs
Verify avg time: 135.884µs
Avg signature size: 718 bytes
Valid signatures: 98.0%
=== [ECDSA_P256+Falcon-1024] ===
KeyGen avg time: 20.115205ms
Sign avg time: 588.235µs
Verify avg time: 201.154µs
Avg signature size: 1334 bytes
Valid signatures: 100.0%