**VIETNAM NATIONAL UNIVERSITY HOCHIMINH CITY**

**UNIVERSITY OF INFORMATION TECHNOLOGY**

**ADVANCED PROGRAM IN INFORMATION SYSTEMS**

**VÕ MINH TÙNG – LÊ BÁ PHÚC**

**POST QUANTUM CRYPTOGRAPHY**

**IN SECURE SHELL PROTOCOL**

**ĐỒ ÁN MÔN MẬT MÃ HỌC**

**NT219.O21**

**HO CHI MINH CITY,** **2024**

**NATIONAL UNIVERSITY HOCHIMINH CITY**

**UNIVERSITY OF INFORMATION TECHNOLOGY**

**ADVANCED PROGRAM IN INFORMATION SYSTEMS**

**VÕ MINH TÙNG – 22521623**

**LÊ BÁ PHÚC – 22521115**

**POST QUANTUM CRYPTOGRAPHY**

**IN SECURE SHELL PROTOCOL**

**ĐỒ ÁN MÔN MẬT MÃ HỌC**

**NT219.O21**

**GIẢNG VIÊN HƯỚNG DẪN**

**PhD NGUYỄN NGỌC TỰ**

**HO CHI MINH CITY, 2024**

**/\*Assessment Committee**

The Assessment Committee is established under the Decision . ........................, date .......................by Rector of the University of Information Technology.

1 .................................................. - Chairman.

2 .................................................. - Secretary.

3 .................................................. - Member

4 .................................................. - Member.\*/

# GIỚI THIỆU

Đồ án này nói về việc ứng dụng các hệ mã kháng lượng tử vào ngữ cảnh Remote Login qua giao thức Secure Shell trước các mối nguy mà máy tính lượng tử kết hợp với Shor’s algorithm tạo ra cho các hệ mã truyền thống.

# Mục lục

🙢🕮🙠

[GIỚI THIỆU i](#_Toc169435611)

[Mục lục ii](#_Toc169435612)

[1 Ngữ cảnh 1](#_Toc169435613)

[2 Rủi ro bảo mật 4](#_Toc169435614)

[3](#_Toc169435615) [Giải pháp 6](#_Toc169435616)

[4](#_Toc169435617) [DEMO 10](#_Toc169435618)

[REFERENCES 11](#_Toc169435619)

# 1 Ngữ cảnh

Một người dùng sử dụng OpenSSH remote log in tới một SSH server thông qua SSH Client.

Các bên liên quan:

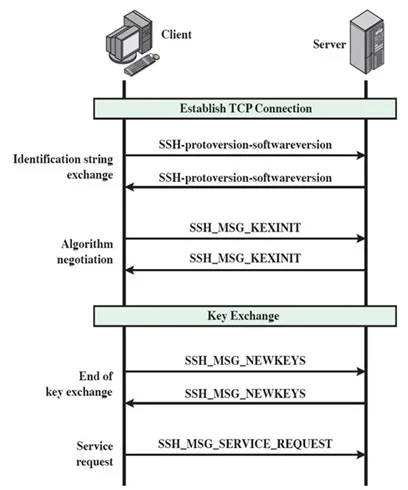
Client machine, Server machine, OpenSSH ở cả hai bên.

Sơ lược về cách hoạt động của giao thức:

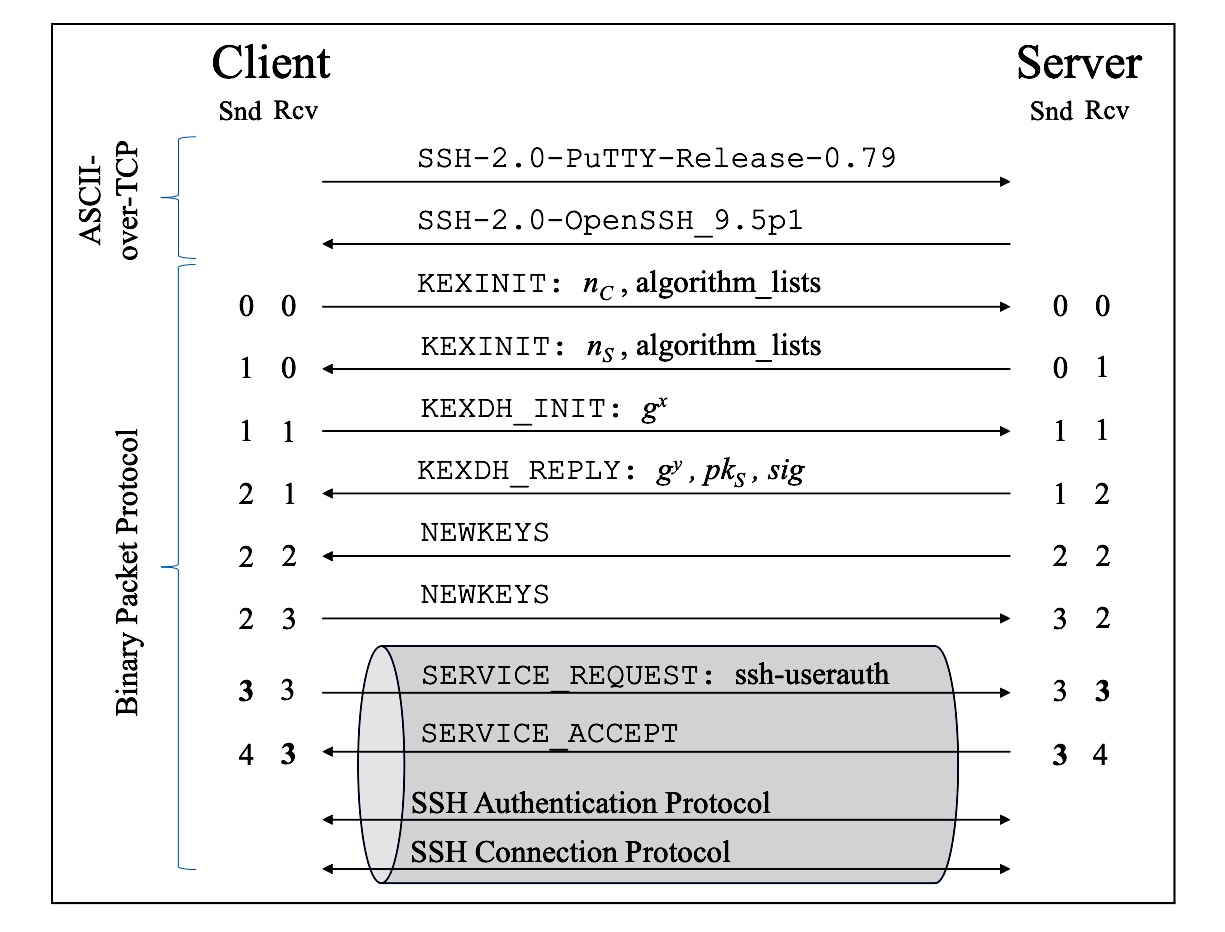
Khi người dùng khởi tạo kết nối SSH, hai bên trao đổi các thông tin như SSH version, các thuật toán mã hóa, MAC,... và tiến hành chọn ra các thuật toán bao gồm thuật toán trao đổi khóa (KX). Đây là bước handshake trong SSH. Cụ thể hơn, client là bên gửi SSH version, các thuật toán hỗ trợ cho server,... đây gọi là Client hello. Server tiến hành chọn ra các thuật toán được ưu tiên cao nhất mà hai bên hỗ trợ, kèm theo SSH version,... và trả về cho client. Đây là Server hello.

Sau đó, server xác thực với client: Server khởi tạo và gửi public key (host key) của nó cho client. Client lưu host key ở ~/.ssh/known\_hosts (Ở những lần tiếp theo, việc xác thực server chỉ yêu cầu client kiểm chứng các thông tin đã lưu trong ~/.ssh/known\_hosts ).

Tiếp đến trong quá trình KX, hai bên tạo ra shared secret (master key). Sau đó, master key kết hợp với các dữ liệu khác tạo ra session key và khởi tạo đường truyền được mã hóa. Bước cuối cùng trong quá trình remote login là User Authentication, server tiến hành bước xác thực này trong kênh truyền bảo mật trên.

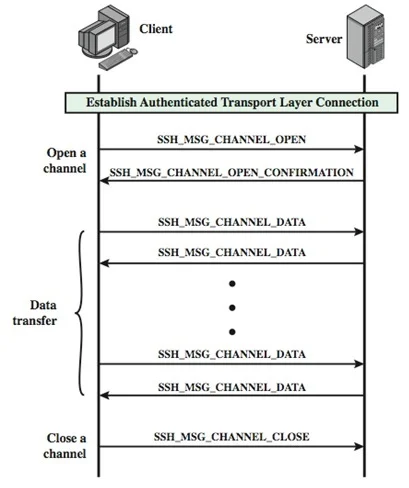


*Hình 1: Minh họa SSH Handshake (RF Wireless World)*



*Hình 2: SSH Authentication sau bước Handshake (Ars Technica*)

Trong bước xác thực, client cần tạo keypair (dùng thuật toán bất đồng bộ như RSA) và gửi public key của nó tới ~/.ssh/authorized\_keys của server. Server yêu cầu người dùng chứng minh rằng họ sở hữu private key. Điều này được thực hiện khi server dùng public key của client để mã hóa một đoạn text ngẫu nhiên. Client chứng minh mình sở hữu private key bằng cách giải mã đoạn text trên và trả về cho server ở dạng băm. Server băm chính đoạn mã tự tạo và so với bản băm của client. Nếu hai bản băm giống nhau vậy thì client đã được xác thực. Server trả về thông báo thành công và tiến hành tạo một phiên làm việc mới:



*Hình 3: Minh họa một phiên làm việc trong SSH protocol (RF Wireless World)*

# 2 Rủi ro bảo mật

Các bước xác thực và trao đổi khóa trên vẫn còn sử dụng các thuật toán dựa trên ECC và RSA. Chẳng hạn như RSA-3072 trong bước xác thực và ECDHE trong bước key exchange. Mặt khác, một máy tính lượng tử với đủ số lượng qubit và kỹ thuật chỉnh lỗi (QECC) tối ưu có thể sử dụng Shor’s algorithm để phá các hệ mã nói trên trong một khoảng thời gian nhanh hơn máy tính thường rất nhiều lần. [1]

Cụ thể hơn trong kết nối Secure Shell, các rủi ro bảo mật có thể xảy ra bao gồm (từ cao đến thấp):

* Trong bước xác thực, client gửi public key cho server để tiến hành bước challenge-response. Attacker có thể bắt lấy public key nói trên, tìm ra private key. Với cặp khóa này, attacker có thể mạo danh người dùng, từ đó login vào SSH server và thực hiện các tác vụ ngang quyền với client có cặp khóa bị phá.
* Ở bước handshake, hai bên trao đổi public key (Diffie-Hellman based KX). Attacker với sự hỗ trợ từ máy tính lượng tử có thể bắt được các public key trong quá trình này và tiến hành phá key, tìm ra private key tương ứng. Với private key, attacker có thể dễ dàng tính ra master key. Từ đó, attacker có thể tìm ra được session key và giải mã toàn bộ phiên làm việc.

Như vậy, ở mức rủi ro cao nhất, dữ liệu ở SSH Server bị lộ và attacker có thể thực hiện các lệnh tùy ý và mức độ nghiêm trọng lệ thuộc vào quyền của compromised SSH Client trong cấu hình của Server.

Yêu cầu bảo mật:

Dễ dàng nhận ra các rủi ro trên chỉ tồn tại khi mà attacker sở hữu máy tính lượng tử và các hệ mã sử dụng cho bước KX và Authentication còn là các hệ mã truyền thống, dựa vào ECC và RSA. Thêm vào đó, kiểu tấn công Harvest now, Decrypt later đòi hỏi ta phải thay mới các hệ mã nói trên bằng các hệ mã tiên tiến hơn, kháng lượng tử.

Dĩ nhiên, các hệ mã này vẫn phải đáp ứng sự an toàn đối với máy tính thường và hiển nhiên phải đáp ứng về mặt hiệu năng. Ta biết rằng Shor’s algorithm kết hợp với máy tính lượng tử phá được RSA và ECC, do đó toàn bộ các hệ mã dựa vào vấn đề RSA và ECC không còn an toàn. Vậy để trở nên ‘kháng lượng tử’, các hệ mã cần dựa vào các vấn đề toán học mà Shor’s algorithm không được tạo ra để giải quyết.

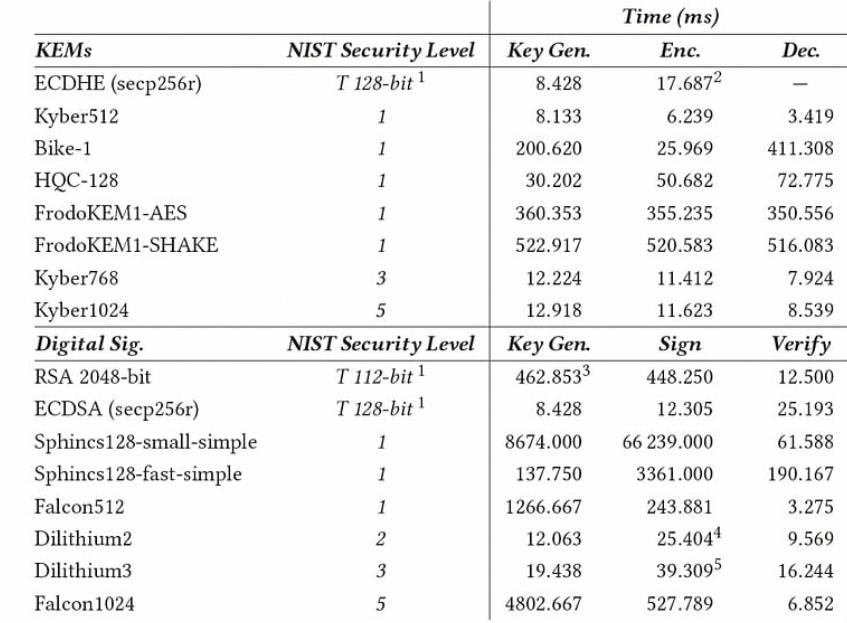
# 3

# Giải pháp

****Trong số các hệ mã Post Quantum, CRYSTALS-Kyber (KEM) và CRYSTAL-Dilithium (Authentication & Sigining) là các hệ mã được NIST chọn làm tiêu chuẩn[II].

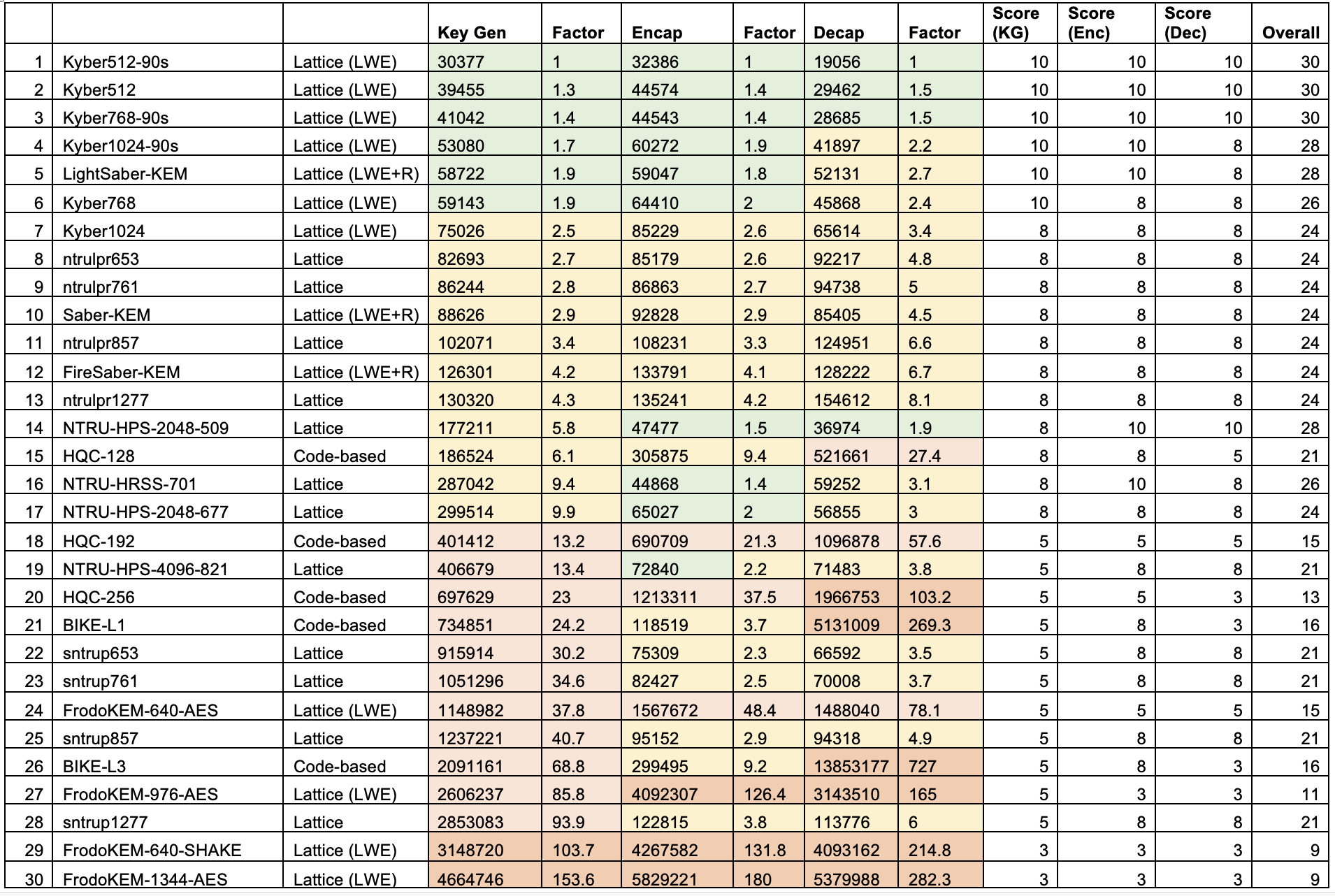
***Hình 4: NIST 2022 Selection for standardization (Wikipedia)***

Về ưu điểm, CRYSTALS-Kyber và CRYSTALS-Dilithium nhìn chung nhanh hơn các thuật toán khác:

****

***Hình 5: So sánh các hệ mã CRYSTALS với các hệ mã khác***

Trong các thuật toán KEM, Kyber với keysize ở NIST level cao nhất (1024) tiêu tốn thời gian tính toán vẫn nhỏ hơn rất nhiều so với hệ mã HQC hay FrodoKEM. Kyber vẫn nhanh hơn khi so với NTRU và SABER:

****

Ngoài ra, NTRU là thuật toán có các keysize nhỏ nhất trong số 3 NIST Finalists (Kyber, NTRU, SABER)

Tương tự, trong Digital Signing:

* Dilithium vẫn có tốc độ nhanh hơn khi so với các thuật toán Digital Signing được NIST chọn như Falcon hay Rainbow nhưng có chữ ký khá dài (2,420 bytes/Dilithium2).
* Falcon cho ra keysize có độ dài ngắn hơn Dilithium (897/ 1,281/falcon512) và chữ ký ngắn hơn (690 bytes/falcon512)
* SPHINCS+ không được ưu tiên như hai thuật toán trên do hiệu năng thấp hơn, nhưng nó dựa vào một vấn đề toán học khác (Hash-based so với Lattice-based) nên được NIST khuyến khích làm hệ mã dự phòng. [II]

Ta có các thống kê key size của các thuật toán trên bên dưới (đơn vị bytes):

Method PKsize SK size Signature size Security level

------------------------------------------------------------------------------------------------------

Crystals Dilithium 2 (Lattice) 1,312 2,528 2,420 1 (128-bit) Lattice

Crystals Dilithium 3 1,952 4,000 3,293 3 (192-bit) Lattice

Crystals Dilithium 5 2,592 4,864 4,595 5 (256-bit) Lattice

Falcon 512 (Lattice) 897 1,281 690 1 (128-bit) Lattice

Falcon 1024 1,793 2,305 1,330 5 (256-bit) Lattice

Sphincs SHA256-128f Simple 32 64 17,088 1 (128-bit) Hash-based

Sphincs SHA256-192f Simple 48 96 35,664 3 (192-bit) Hash-based

Sphincs SHA256-256f Simple 64 128 49,856 5 (256-bit) Hash-based

Nguồn: asecuritysite.com

Kèm theo thống kê hiệu năng, (đơn vị CPU ops/sec):

Method Key Gen Sign Verify Security level

------------------------------------------------------------------------------------------------------

Crystals Dilithium 2 1,400,412 6,157,001 1,461,284 1 (128-bit) Lattice

Crystals Dilithium 3 2,282,485 9,289,499 2,228,898 3 (192-bit) Lattice

Crystals Dilithium 5 3,097,421 8,469,805 3,173,500 5 (256-bit) Lattice

Falcon 512 (Lattice) 197,793,925 38,090,446 474,052 1 (128-bit) Lattice

Falcon 1024 480,910,965 83,482,883 977,140 3 (256-bit) Lattice

Sphincs SHA256-128f Simple 16,552,135 521,963,206 20,850,719 1 (128-bit) Hash-based

Sphincs SHA256-192f Simple 24,355,501 687,693,467 35,097,457 3 (128-bit) Hash-based

Sphincs SHA256-256f Simple 64,184,968 1,554,168,401 36,182,488 5 (128-bit) Hash-based

Nguồn: asecuritysite.com

Từ các số liệu nói trên, ta thấy trong các môi trường khác nhau thì việc lựa chọn hệ mã phù hợp sẽ có sự khác biệt.

Trong môi trường Secure Shell, các thuật toán PQ nên được kết hợp với các thuật toán phổ biến, tạo thành một hybrid-scheme (scheme lai). Cụ thể:

Đối với KX, ta có các hybrid scheme như: x25519\_Kyber (đã được hỗ trợ trên Google Chrome), ECDH\_x25519\_NTRUprime (default trong OpenSSH 9.0+).

Do OpenSSH đã hỗ trợ một hệ mã Post Quantum cho trao đổi khóa, nhóm em sẽ tập trung trình bày cũng như demonstrate về hệ mã cho bước xác thực. Đối với trao đổi khóa nhóm chúng em cũng sẽ lựa chọn cấu hình x25519\_Kyber512 trong phần demonstration.

Cụ thể hơn, hệ mã cho bước xác thực mà nhóm em chọn là {tungminhv ...}

# 4

# DEMO

...

# REFERENCES

1. Wikipedia contributors. "Shor's algorithm." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 10 Jun. 2024. Web. 14 Jun. 2024.
2. "NIST 2022 PQC Selection." Nist.Gov, 5 Jul. 2022, www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms.