Mock Discovery Agent PQC Handshake Report Update: Hybrid Key Exchange and Separating Cryptographic Functions

Instruction

The example is largely like the original implementation in terms of user-level operation: Discovery_Server.py is run first, then Discovery_Client.py. This time, however, Discovery_Client.py will take an input to either choose creating a pure PQC shared secret key with ML-KEM, or a Hybrid shared secret key with ML-KEM and X25519. The program will continue to simulate 5 clients concurrently connecting to the server and completing the handshake with the server before sending JSON-formatted system information encrypted with the shared secret key.

Hybrid Implementation

If the client chooses the hybrid key exchange option, there will be additional communications to/from the client and server for sending their X25519 public keys (Note: This program does not create a signature when sending public keys since it is assumed that it should be verified by a Certificate Authority when in a production environment). When both parties receive the other's public key, they each derive a 128-bit X25519 shared secret from that and their respective private key. The previously generated ML-KEM shared secret will then be split in half to 128-bits and concatenated with the X25519 secret to form the new hybrid 256-bit secret for AES encryption/decryption of client system info. X25519 was chosen as the classical means of key exchange to more closely resemble TLS 1.3's hybrid key exchange option, X25519Kyber768, which combines the confidence of current scrutinized standards and resistance against future quantum threats (Shor's Algorithm). This does mean, however, that if one algorithm was compromised, the hybrid shared secret would have an effective security level of 128-bits.

Function Classes

Discovery_Server.py and Discovery_Client.py now calls functions from 3 different files, ML_KEM_Functions.py, ML_DSA_Functions.py, and X25519_Functions.py. This is so that the main files do not need to handle the logic of handling the format of communications when transmitting public keys, ciphertext, and system information during

the handshake process. The AES encryption/decryption when sending and receiving system info is also included in ML_DSA_functions.py.

The program also prints the keys used in the exchange depending on pure or hybrid PQC

```
Client ('127.0.0.1', 7976) connected.
Received client's ML_KEM public key. Generating ML_KEM secret and sending ciphertext to client...

ML_KEM (PQC) shared secret key: b'W\x96\xf8\x1f\x7f\x91\xcb\xa9\xb4\xfbK\xc3N\xaf\xe4\x17P\xb7[\x19\xad\xc2~\xfe\x1a\xa3\x03X9\x03\x90\x87'

Client ('127.0.0.1', 7976) successful handshake. System info: {"hostname": "mockdevice_4", "os": "Windows 10", "ip": "127.0.0.1"}
Client ('127.0.0.1', 7976) disconnected.
```

```
Client ('127.0.0.1', 8064) connected.
Received client's ML_KEM public key. Generating ML_KEM secret and sending ciphertext to client...
Received client's X25519 public key. Deriving X25519 secret...
Sending X25519 public key to client...

ML_KEM (PQC) half: b'\xb1nn\xfaA\xf1\x90)\x98XS\xea\x8a\xcd2:'
X25519 (Classical) half: b'*\xa0\xc9\xc6\xd1\x1f\xa6x\xfc\xaa\x02\xc0\x8d\x00\x12&'
New Hybrid shared secret key: b'\xb1nn\xfaA\xf1\x90)\x98XS\xea\x8a\xcd2:*\xa0\xc9\xc6\xd1\x1f\xa6x\xfc\xaa\x02\xc0\x8d\x00\x12&'

Client ('127.0.0.1', 8064) successful handshake. System info: {"hostname": "mockdevice_1", "os": "Windows 11", "ip": "127.0.0.1"}
Client ('127.0.0.1', 8064) disconnected.
```