Incorporating Post-Quantum Cryptography in a microservice architecture

Research Project 2

R. van der Gaag, D. Weller
Why think about post-quantum cryptography

W. Buchanan et. al concluded

- Gate-based quantum computers pose a significant threat to a-symmetrical encryption (which is used in PKI)
  - Shor’s algorithm
- Likely theoretical $\rightarrow$ practical <10 years

A-symmetric keys are used by:

- (D)TLS
- SSH
- WPA & WPA2
- DNSSec
- IKEv2 (IPSec & VPN)
- S/MIME
Research questions

What are the implications of transitioning to post-quantum cryptography in many-to-one microservice architectures where certificates are used for both encryption and mutual authentication?

Two sub questions:
1. Suitable algorithms
2. Practical feasibility
Related work

National Institute of Standards and Technology (NIST)
- 2nd round with Post Quantum Cryptography (PQC)
  - 17 different Post Quantum Key Exchange Algorithms
  - 9 different Post Quantum Signature Algorithms

E. Crockett et. al - OpenQuantum Safe
- Forked OpenSSH
- Forked OpenSSL
  - 8 different Post Quantum Key Exchange Algorithms
  - 3 different Post Quantum Signature Algorithms
J. Kreps et. al - detailed insight about inner workings of Kafka

K. Sheykh Esmaili et. al - important aspects of microservices:
- Correctness - Delivery guarantees & Ordering guarantees
- Availability - Maximize its uptime
- Transactions - Group messages into units
- Scalability - Evolve with growing amount of tasks
- Efficiency
  - Latency of a packet / message
  - Throughput (number / bytes of packets per time unit)

Incorporating post-quantum cryptography in a microservice architecture
Background

- What is Kafka?
  - Publish / subscribe mechanism
  - Developed by LinkedIn
  - Stands out in bulk messaging
  - Passive and stateless
    - Publisher (delivers data) pushes data
    - Consumer (requests data) pulls data

- What is Post Quantum Cryptography?
  - Classical key exchange relies on factorization (e.g. RSA) or logarithmic (e.g. DH and ECC) mathematical problems
  - PQC relies on other mathematical problems
    - Not yet solvable by quantum computers
## Open Quantum Safe OpenSSL fork

<table>
<thead>
<tr>
<th>Level</th>
<th>Post Quantum Key Exchange Mechanisms</th>
<th>Post Quantum Digital Signature Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>bike1l1cpa, bike1l1fo, frodo640aes, frodo640shake, Kyber512, newhope512cca, ntru_hps2048509, lightsaber, sidhp434, sikep434</td>
<td>dilithium2, picnicl1fs, qteslapi</td>
</tr>
<tr>
<td>II</td>
<td>Sidhp503, sikep503</td>
<td>dilithium3</td>
</tr>
<tr>
<td>III</td>
<td>Bike1l3cpa, bike1l3fo, frodo976aes, frodo976shake, ntru_hps2048677, ntru_hrss701, Saber, Sidhp610, sikep610</td>
<td>dilithium4, qteslapiii</td>
</tr>
<tr>
<td>IV</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>V</td>
<td>frodo1344aes, frodo1344shake, kyber1024, newhope1024cca, Ntru_hps4096821, Firesaber, Sidhp751, sikep751</td>
<td>None</td>
</tr>
</tbody>
</table>
## Open Quantum Safe OpenSSL fork

### Hybrid Algorithms

<table>
<thead>
<tr>
<th>Level</th>
<th>Hybrid Post Quantum Key Exchange Mechanisms</th>
<th>Hybrid Post Quantum Digital Signature Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>p256_bike1l1cpa, p256_bike1l1fo, p256_frodo640aes, p256_frodo640shake, p256_kyber512, p256_newhope512cca, p256_ntru_hps2048509, p256_lightsaber, p256_sidhp434, p256_sikep434.</td>
<td>rsa3072_dilithium2, p256_dilithium2, rsa3072_picnicl1fs, p256_picnicl1fs, rsa3072_qteslapi, p256_qteslapi</td>
</tr>
<tr>
<td>II</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>III</td>
<td>None</td>
<td>p384_dilithium4, p384_qteslapiii</td>
</tr>
<tr>
<td>IV</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>V</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Methodology:

- What are the handshake differences (elapsed time, peak heap memory) between
  - Classical cryptography
  - Post-Quantum Cryptography
  - Hybrid-Post-Quantum Cryptography
- Divide the algorithms per security level (provided by NIST)

<table>
<thead>
<tr>
<th>Level</th>
<th>Security Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>At least as hard to break as AES128 (exhaustive key search)</td>
</tr>
<tr>
<td>II</td>
<td>At least as hard to break as SHA256 (collision search)</td>
</tr>
<tr>
<td>III</td>
<td>At least as hard to break as AES192 (exhaustive key search)</td>
</tr>
<tr>
<td>IV</td>
<td>At least as hard to break as SHA384 (collision search)</td>
</tr>
<tr>
<td>V</td>
<td>At least as hard to break as AES256 (exhaustive key search)</td>
</tr>
</tbody>
</table>

(NIST, 2019)
Results

Classical Cryptography algorithms

Incorporating post-quantum cryptography in a microservice architecture
Results

Handshake Level 1 - PQC

Incorporating post-quantum cryptography in a microservice architecture
Results

Handshake Level 1 - Hybrid PQC

Incorporating post-quantum cryptography in a microservice architecture
Results
Handshake Level 2 - PQC
Results

Handshake Level 3 - PQC
Preliminary conclusions

What are the implications of transitioning to post-quantum cryptography in many-to-one microservice architectures where certificates are used for both encryption and mutual authentication?

- Suitable algorithms
  - L1
    - Dilithium2 - Kyber512 / Lightsaber / NewHope512cca
    - Picnic1fs - Kyber512 / Lightsaber / NewHope512cca
    - qTeslapi - Kyber512 / Lightsaber / NewHope512cca
  - L2
    - Dilithium3 - SiDHp503
  - L3
    - Dilithium4 - Saber / Frodo / NTRU
    - qTeslapiii - Saber / Frodo / NTRU
Preliminary conclusions (cont)

What are the implications of transitioning to post-quantum cryptography in many-to-one microservice architectures where certificates are used for both encryption and mutual authentication?

- Practical feasibility
  - Kafka relies on Java
    - PQC not yet implemented in Java Security stack
    - Using the OpenSSL fork for Kafka requires additional customization
  - Using the OpenSSL fork
    - Using Hybrid for transitioning
    - Handshake time is not that much longer
Discussion

- Algorithms still in development
  - NIST Round 2 still in progress

- We did not test these algorithms in a microserver environment
  - CPU measurements not taken into account
  - Our setup was optimal, we did not test multiple concurrent sessions
Future work

- Experiment with Java Security stack
  - development of general interface for third party libraries

- Experiment with liboqs algorithms in the OpenSSL fork
  - Still in development
  - Not all are available for proper testing