Building post-quantum cryptography for the cloud

Alex Weibel
Senior Software Engineer, AWS Cryptography
AWS
Agenda

• Why do we need post-quantum cryptography?
• What are the interfaces that we care most about?
• What have we done so far?
• Post-quantum TLS performance numbers
• Demo – how to use it yourself
• Future work
Why do we need post-quantum crypto?

Key agreement algorithms in use today will break given a quantum adversary and we don’t know precisely when (or if) that will happen.

We know it takes many years to go through design, analysis, implementation, testing, and deployment for new cryptography.

We need time to get it right, build agility, and potentially modify system architectures.
## Best-known attack by a quantum adversary

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Invented</th>
<th>Standardized</th>
<th>Widely used</th>
<th>Best-known algorithm</th>
<th>Post-quantum security</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECC</td>
<td>1985</td>
<td>1999</td>
<td>2000s</td>
<td>Shor’s</td>
<td>Broken</td>
</tr>
<tr>
<td>FFDHE</td>
<td>1976</td>
<td>1991</td>
<td>1990s</td>
<td>Shor’s</td>
<td>Broken</td>
</tr>
<tr>
<td>RSA</td>
<td>1978</td>
<td>1998</td>
<td>1990s</td>
<td>Shor’s</td>
<td>Broken</td>
</tr>
<tr>
<td>SHA2</td>
<td>2001</td>
<td>2002</td>
<td>2005</td>
<td>Birthday attack / Grover’s</td>
<td>Weakened</td>
</tr>
<tr>
<td>SHA3</td>
<td>2008</td>
<td>2015</td>
<td>?</td>
<td>Birthday attack / Grover’s</td>
<td>Weakened</td>
</tr>
</tbody>
</table>
## Best-known attack by a quantum adversary

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Invented</th>
<th>Standardized</th>
<th>Widely used</th>
<th>Best-known algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECC</td>
<td>1985</td>
<td>1999</td>
<td>2000s</td>
<td>Shor’s</td>
</tr>
<tr>
<td>FFDHE</td>
<td>1976</td>
<td>1991</td>
<td>1990s</td>
<td>Shor’s</td>
</tr>
<tr>
<td>RSA</td>
<td>1978</td>
<td>1998</td>
<td>1990s</td>
<td>Shor’s</td>
</tr>
<tr>
<td>SHA2</td>
<td>2001</td>
<td>2002</td>
<td>2005</td>
<td>Birthday attack / Grover’s</td>
</tr>
<tr>
<td>SHA3</td>
<td>2008</td>
<td>2015</td>
<td>?</td>
<td>Birthday attack / Grover’s</td>
</tr>
</tbody>
</table>
Timeline for quantum computing

- **Quantum computing conceived**
- **Simon's algorithm**
- **2-qubit computer**
- **12-qubit computer**
- **Grover's algorithm**
- **17-qubit computers**
- **50-qubit computer quantum advantage**
- **100-qubit computer quantum advantage**
- **IBM claims 1000-bit quantum computer by 2023**
Timeline for quantum computing

- Quantum computing conceived
- Simon's algorithm
- 2-qubit computer
- 12-qubit computer
- Grover's algorithm
- 17-qubit computers
- 100-qubit computer quantum advantage
- 50-qubit computer quantum advantage
- IBM claims 1000-bit quantum computer by 2023
- Quantum computer to factor RSA 2048?
NIST post-quantum standardization process

• In 2016, NIST began the process of evaluating new post-quantum algorithms for standardization

• NIST standardization process consists of multiple rounds of evaluation, elimination, and algorithm improvements

• Two NIST post-quantum algorithm submissions (BIKE and SIKE) have Amazon employees as co-authors

• Standardization process expected to continue until 2022 / 2024
Hybrid key agreement

Alice

(a, A) = generateECDH()
(sk, pk) = generatePQ()

ss₁ = a * B
ss₂ = decapsulatePQ(sk, ct)

k = KDF(ss₁ || ss₂)

Bob

(b, B) = generateECDH()

ss₁ = b * A
(ss₂, ct) = encapsulatePQ(pk)

k = KDF(ss₁ || ss₂)
Hybrid post-quantum standardization efforts

1) **NIST** post-quantum cryptography standardization process
   NIST SP800-56C Rev 2 affordance for hybrid key derivation

2) **ETSI CYBER** quantum safe cryptography technical committee
   TS103 744 V.0.0.10 (2020-06) hybrid key exchange

3) **IETF** draft specifications for hybrid post-quantum TLS draft
   specification on post-quantum hybrid TLS 1.2 and TLS 1.3
Long-lasting data confidentiality requirements

- Hours – temporary security credentials
- Months – earnings reports
- Years – credit card and banking information
- Couple decades – corporate trade secrets
- Many decades – classified information, sealed records
- Lifetimes – personal information, DNA data
Long-lasting data confidentiality requirements

- Hours – temporary security credentials
- Months – earnings reports
- Years – credit card and banking information
- Couple decades – corporate trade secrets
- Many decades – classified information, sealed records
- Lifetimes – personal information, DNA data
Why are we investing in post-quantum cryptography?

• We are primarily concerned in the long-term requirement of confidentiality of our users’ data

• When we say PQ Cloud we mean we want to make sure our users’ data and privacy are secure against a quantum adversary

• Given that data largely flows over public networks it is subject to store and harvest-style threats

• An adversary could collect the encrypted data today and when they have access to a quantum computer recover the plaintext communication in the future
Motivations to use post-quantum cryptography
Motivations to use post-quantum cryptography

Retain encrypted copy

Client → TLS → Internet → TLS → AWS Cloud
Motivations to use post-quantum cryptography

Plaintext data

Retain encrypted copy + New capabilities

AWS Cloud
What interfaces are we most concerned about?

• Our primary interface to users is over TLS; it is often the most challenging interface that we are concerned about

• We plan to build out solutions here knowing it is both the most challenging and impactful

• Working in open forum contributing to current research and standardization efforts

• We want to update our other interfaces to use post-quantum algorithms
Cloud networking

<table>
<thead>
<tr>
<th>Region</th>
<th>Availability Zone</th>
<th>Zone</th>
<th>Availability Zone</th>
<th>Zone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corporate</td>
<td>VPC</td>
<td>Amazon EC2 instance</td>
<td>Application</td>
<td></td>
</tr>
<tr>
<td>AWS Direct Connect</td>
<td></td>
<td>Amazon S3</td>
<td>AWS Outposts</td>
<td></td>
</tr>
<tr>
<td>Corporate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Network Protocols:
- TLS
- SSH
- MACSec
- IPSec
- DWDM
- MACSec
- IPSec
- DWDM
What have we done so far?

• Worked on and promoted the use of hybrid key agreement in ETSI (European Telecommunication Standards Institute) and advocating for it in forums with NIST

• Specified TLS handshake updates in IETF drafts for TLS 1.2 and TLS 1.3 to get interoperability with others (Open Quantum Safe)

• Deployed hybrid post-quantum TLS to our most security critical services, like AWS Key Management Service (AWS KMS)
2019: IETF draft for hybrid post-quantum TLS

- Published IETF draft for hybrid post-quantum TLS 1.2

- Combines classical security of ECDH with conjectured post-quantum security of NIST candidates

- Neither ECDH or PQ candidates are single points of failure
s2n: TLS library

- s2n is an AWS open source library for TLS
- Designed to be simple, small, fast and with security as a priority
- Aims to be smaller, safer alternative to libssl
- Currently handles all of Amazon Simple Storage Service (Amazon S3) traffic
s2n: TLS library

- s2n is an AWS open source library for TLS
- Designed to be simple, small, fast and with security as a priority
- Aims to be smaller, safer alternative to libssl
- Currently handles all of Amazon Simple Storage Service (Amazon S3) traffic
2019: Round 1 PQ TLS 1.2 in s2n

BIKE
Bit-flipping key encapsulation

https://bikesuite.org

SIKE
Supersingular isogeny key encapsulation

https://sike.org

s2n

libcrypto
2019: Developer preview of post-quantum AWS client java SDK

- Getting post-quantum crypto deployed is not just crypto work
- Need to integrate it client-side into open source libraries used by customers
- Need to integrate it server side into AWS services

- `aws-java-sdk-v2` (AWS service Java API’s)
- `aws-crt-java` (Java-to-C JNI translation)
- `aws-c-http` (HTTP protocol)
- `s2n` (TLS protocol)
2019: Round 1 PQ TLS in AWS KMS

AWS Security Blog

Post-quantum TLS now supported in AWS KMS
by Andrew Hopkins | on 04 NOV 2019 | in Advanced (300), AWS Key Management Service, Security, Identity, & Compliance | Permalink | Comments | ➔ Share

AWS Key Management Service (AWS KMS) now supports post-quantum hybrid key exchange for the Transport Layer Security (TLS) network encryption protocol that is used when connecting to KMS API endpoints. In this post, I’ll tell you what post-quantum TLS is, what hybrid key exchange is, why it’s important, how to take advantage of this new feature, and how to give us feedback.

Round 1 hybrid post-quantum TLS launched in AWS KMS
Round 2 Hybrid Post-Quantum TLS Benchmarks

AWS Cryptography has completed benchmarks of Round 2 Versions of the Bit Flipping Key Encapsulation (BIKE) and Supersingular Isogeny Key Encapsulation (SIKE) hybrid post-quantum Transport Layer Security (TLS) Algorithms. Both of these algorithms have been submitted to the National Institute of Standards and Technology (NIST) as part of NIST’s Post-Quantum Cryptography standardization process.

Round 2 hybrid post-quantum TLS benchmarks released
Updated BIKE and SIKE algorithms to round 2 versions in s2n

Added Kyber algorithm to s2n

All three algorithms (Kyber, BIKE, SIKE) entered round 3 of NIST evaluation process

2020: Round 2 PQ TLS 1.2 in s2n

Kyber
https://pq-crystals.org/kyber
### 2020: Round 2 PQ TLS 1.2 Handshake Time

<table>
<thead>
<tr>
<th>PQ Impl</th>
<th>Bandwidth (bytes)</th>
<th>50th-percentile latencies from 2,500 TLS handshakes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ECDHE (Classic)</strong></td>
<td><strong>–</strong></td>
<td>3,574</td>
</tr>
<tr>
<td><strong>ECDHE + Kyber</strong></td>
<td><strong>C99</strong></td>
<td>5,898</td>
</tr>
<tr>
<td><strong>ECDHE + BIKE</strong></td>
<td><strong>C99</strong></td>
<td>12,456</td>
</tr>
<tr>
<td><strong>ECDHE + SIKE</strong></td>
<td><strong>ASM</strong></td>
<td>4,628</td>
</tr>
</tbody>
</table>
2020: Round 2 PQ TLS 1.3 in s2n

• Collaborated externally on TLS 1.3 hybrid post-quantum IETF draft

• Multiple TLS 1.3 key shares can be sent opportunistically in order to save 1 round trip

• Interoperability with other PQ TLS implementations (OQS)
2020: Public preview for hybrid post-quantum AWS SDK client

Introducing AWS Common Runtime HTTP Client in the AWS SDK for Java 2.x

We are pleased to announce the preview release of AWS Common Runtime (CRT) HTTP Client – a new HTTP client supported in the AWS SDK for Java 2.x. AWS CRT HTTP Client is an asynchronous, non-blocking HTTP client built on top of the Java bindings of the AWS Common Runtime. You can use the CRT HTTP client to benefit from features such as improved performance, connection health checks, and post-quantum TLS support. It is the second first-party asynchronous HTTP client supported by the SDK for Java after Netty NIO HTTP client.

Java build artifacts available with post-quantum TLS support
How to use post-quantum TLS with AWS SDK for Java 2.x
Confirm Access to Post-Quantum TLS Ciphers:
1. TLS_CIPHER_PREF_KMS_PQ_TLSv1_0_2020_07 is available on current platform.
2. Hybrid post-quantum cipher are supported and will be used.

Post-Quantum KMS Key Import Example:
1. Created CMK ID: a3ea01ac-9f58-4e0b-94b4-dc7eda3b37de
2. Received Public RSA Wrapping Key from KMS for CMK import
3. Generated local secure AES Key
4. Wrapped local AES Key with public KMS Wrapping Key
5. Imported AES key into KMS with CMK ID:a3ea01ac-9f58-4e0b-94b4-dc7eda3b37de. Used PQ TLS to protect RSA-wrapped AES key in transit.

Clean Up after Demo:
1. CMK a3ea01ac-9f58-4e0b-94b4-dc7eda3b37de is scheduled to be deleted at 2020-11-19T00:00:00Z.
Work not yet done

- We want to hear from you!

- What AWS services beyond AWS KMS do you think need post-quantum support?

- Plans to evaluate how to integrate post-quantum crypto into more protocols – SSH, VPN, MACsec, etc.
Thank you!

Alex Weibel
aweibel@amazon.com
Please complete the session survey