POST-QUANTUM TLS 1.3
PQNET 2022

Sofía Celi

PQNET?

- Workshop on the challenges/opportunities of putting post-quantum cryptography into the real-world
- Meant to create collaboration

https://sofiaceli.com/PQNet-Workshop/

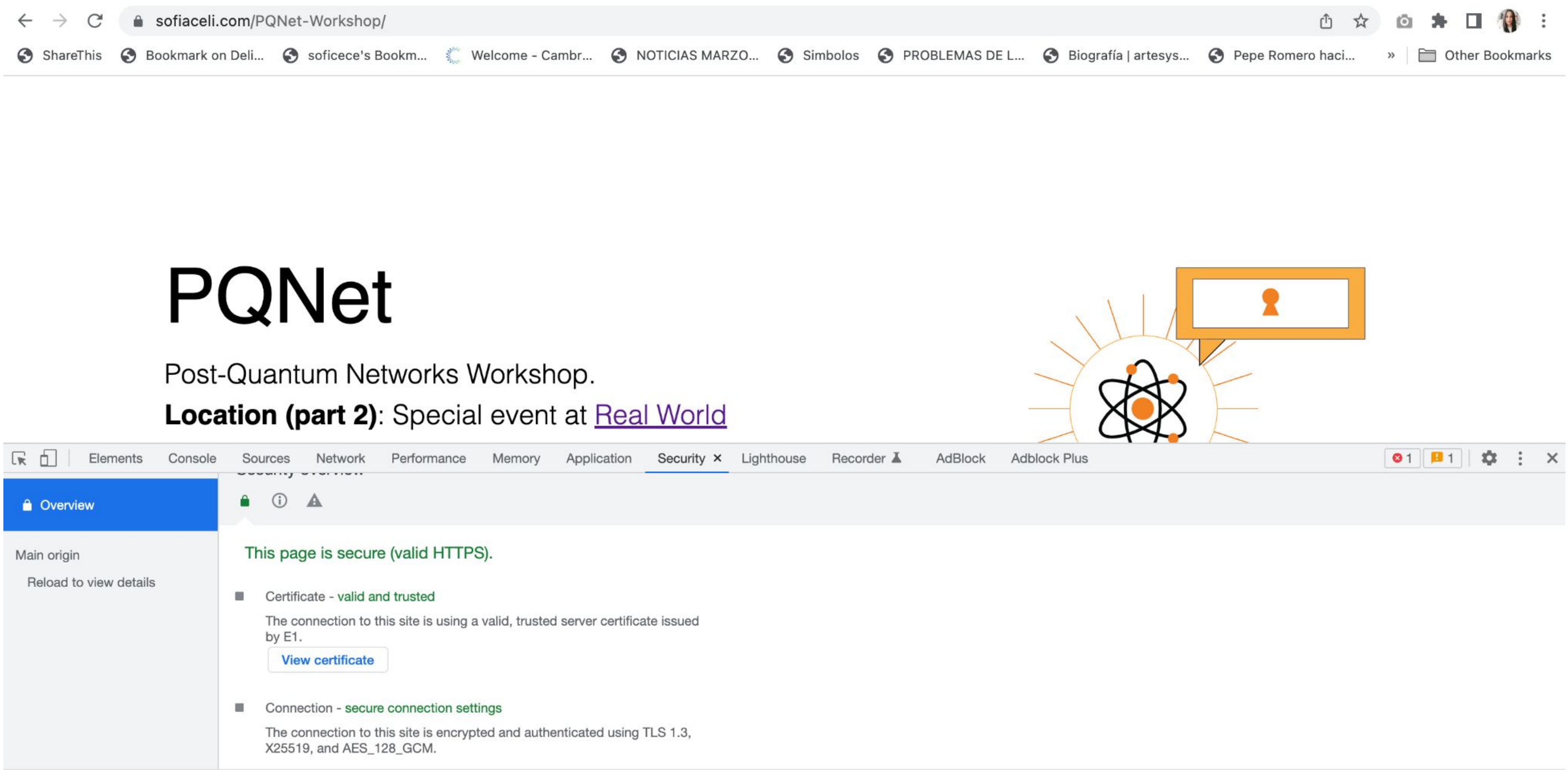
WHO AM I?

- Cryptography researcher and implementer
- Works at Post-quantum algorithms and formal verification and more!

https://sofiaceli.com/

AGENDA

- TLS 1.3
- PQ TLS
- Limitations
- Open problems
- The future



PQNet

Post-Quantum Networks Workshop.

Location (part 2): Special event at Real World

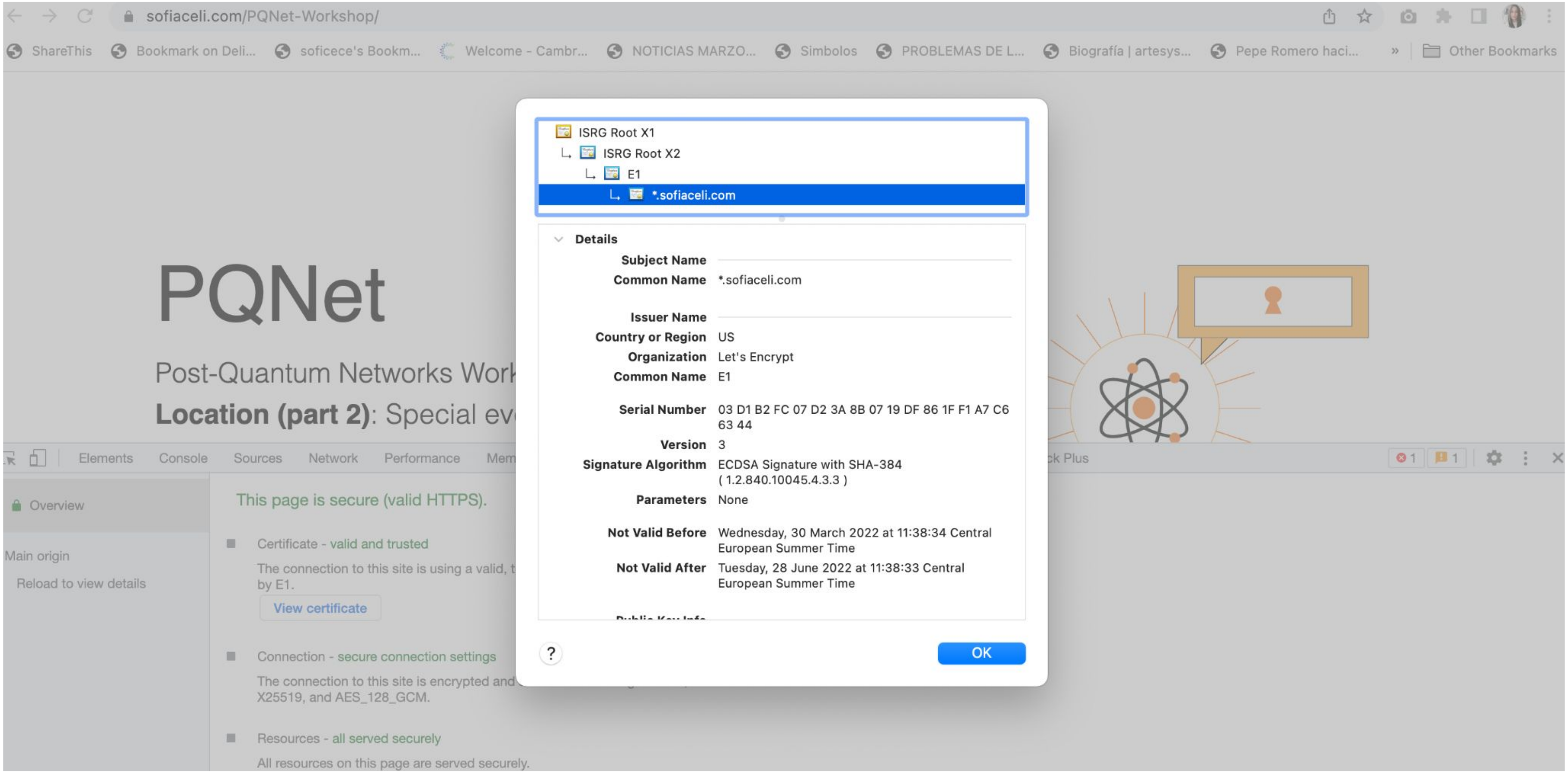
This page is secure (valid HTTPS).

- Certificate - valid and trusted

  The connection to this site is using a valid, trusted server certificate issued by E1.

- Connection - secure connection settings

  The connection to this site is encrypted and authenticated using TLS 1.3, X25519, and AES_128_GCM.



ISRG Root X1
└ ISRG Root X2
  └ E1
    └ *.sofiaceli.com

Details

Subject Name
Common Name *.sofiaceli.com

Issuer Name
Country or Region US
Organization Let's Encrypt
Common Name E1

Serial Number 03 D1 B2 FC 07 D2 3A 8B 07 19 DF 86 1F F1 A7 C6 63 44
Version 3
Signature Algorithm ECDSA Signature with SHA-384 (1.2.840.10045.4.3.3)
Parameters None

Not Valid Before Wednesday, 30 March 2022 at 11:38:34 Central European Summer Time
Not Valid After Tuesday, 28 June 2022 at 11:38:33 Central European Summer Time

This page is secure (valid HTTPS).

Certificate - valid and trusted

The connection to this site is using a valid, t
by E1.

Connection - secure connection settings

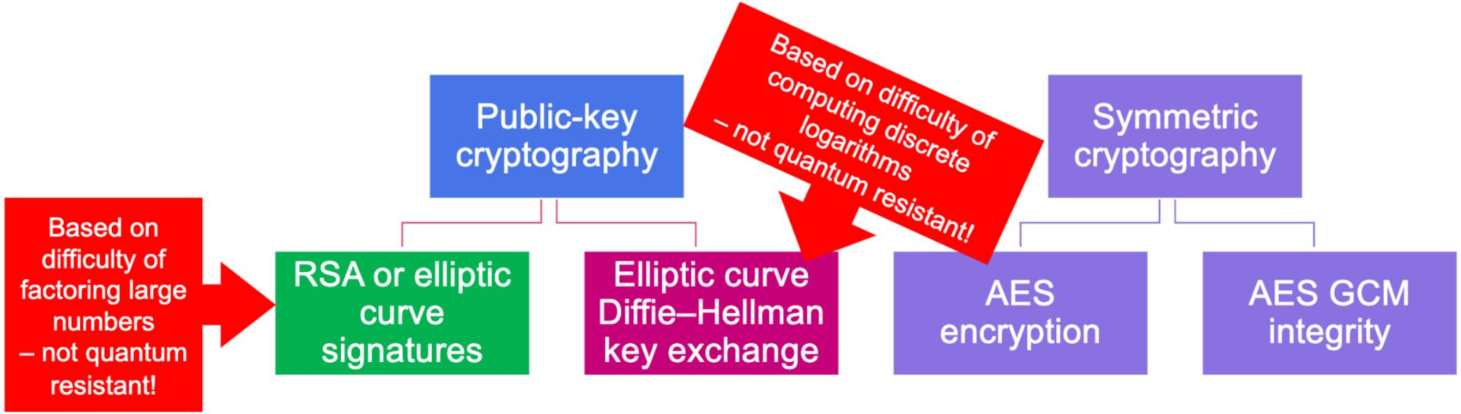
The connection to this site is encrypted and
X25519, and AES_128_GCM.

Resources - all served securely

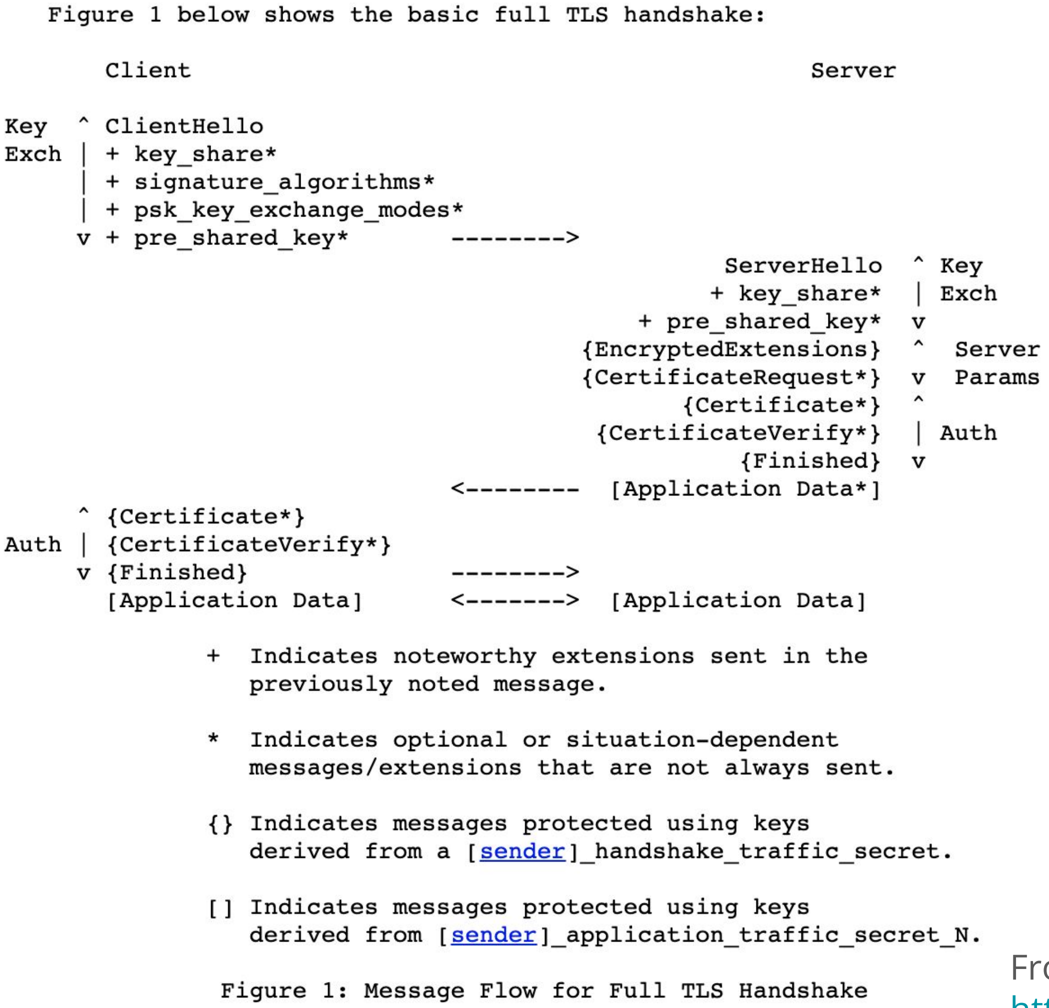
All resources on this page are served securely.

TLS 1.3

- Encrypted and authenticated using TLS 1.3:
  - X25519
  - AES-GCM-128
  - ECDSA


Public-key cryptography
Symmetric cryptography
RSA or elliptic curve signatures
Elliptic curve Diffie–Hellman key exchange
AES encryption
AES GCM integrity
Based on difficulty of factoring large numbers – not quantum resistant!
Based on difficulty of computing discrete logarithms – not quantum resistant!

Credit to Douglas Stebila



```
   Figure 1 below shows the basic full TLS handshake:

       Client                                           Server

Key  ^ ClientHello
Exch | + key_share*
     | + signature_algorithms*
     | + psk_key_exchange_modes*
     v + pre_shared_key*       -------->
                                                  ServerHello  ^ Key
                                                 + key_share*  | Exch
                                            + pre_shared_key*  v
                                        {EncryptedExtensions}  ^  Server
                                        {CertificateRequest*}  v  Params
                                               {Certificate*}  ^
                                         {CertificateVerify*}  | Auth
                                                   {Finished}  v
                               <--------  [Application Data*]
     ^ {Certificate*}
Auth | {CertificateVerify*}
     v {Finished}              -------->
       [Application Data]      <------->  [Application Data]

              +  Indicates noteworthy extensions sent in the
                 previously noted message.

              *  Indicates optional or situation-dependent
                 messages/extensions that are not always sent.

              {} Indicates messages protected using keys
                 derived from a [sender]_handshake_traffic_secret.

              [] Indicates messages protected using keys
                 derived from [sender]_application_traffic_secret_N.

               Figure 1: Message Flow for Full TLS Handshake
```

From: https://datatracker.ietf.org/doc/html/rfc8446

TLS 1.3

- Properties:
  - Authentication (certificates, password, pre-shared)
  - Confidentiality
  - Integrity

- Two parts:
  - A record protocol
  - A handshake protocol

PQ TLS 1.3

- Properties:
  - **Authentication (certificates, password, pre-shared)**
  - **Confidentiality (key exchange)**
  - Integrity

- Two parts:
  - A record protocol
  - **A handshake protocol**

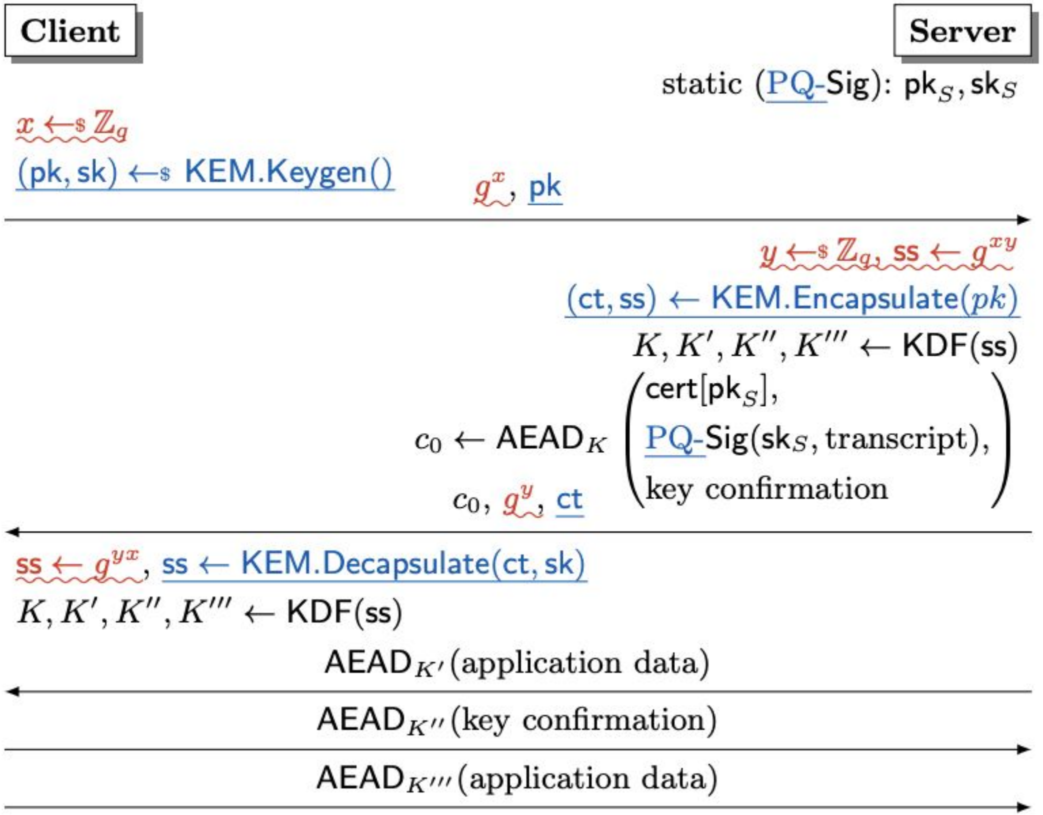

Client
Server
static (PQ-Sig): $\mathsf{pk}_S, \mathsf{sk}_S$
$x \leftarrow_\$ \mathbb{Z}_g$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow_\$ \mathsf{KEM.Keygen}()$
g^x, pk
$y \leftarrow_\$ \mathbb{Z}_g$, $\mathsf{ss} \leftarrow g^{xy}$
$(\mathsf{ct}, \mathsf{ss}) \leftarrow \mathsf{KEM.Encapsulate}(pk)$
$K, K', K'', K''' \leftarrow \mathsf{KDF}(\mathsf{ss})$
$c_0 \leftarrow \mathsf{AEAD}_K \begin{pmatrix} \mathsf{cert}[\mathsf{pk}_S], \\ \text{PQ-}\mathsf{Sig}(\mathsf{sk}_S, \text{transcript}), \\ \text{key confirmation} \end{pmatrix}$
c_0, g^y, ct
$\mathsf{ss} \leftarrow g^{yx}$, $\mathsf{ss} \leftarrow \mathsf{KEM.Decapsulate}(\mathsf{ct}, \mathsf{sk})$
$K, K', K'', K''' \leftarrow \mathsf{KDF}(\mathsf{ss})$
$\mathsf{AEAD}_{K'}(\text{application data})$
$\mathsf{AEAD}_{K''}(\text{key confirmation})$
$\mathsf{AEAD}_{K'''}(\text{application data})$

Fig. 1. Comparison between TLS 1.3 handshake (black and red) and PQTLS (black and blue). Note that both protocols require the same number of rounds and messages.

LIMITATIONS

- Is it just as simple as swapping classical algorithms for post-quantum ones?
  - Depends on which protocol/property of the handshake
  - Depends on the network, system and overall environment
  - Measurements and experiments need to be done

Studies:

- Bos et al.[1][2] measured the impact of post-quantum key-exchange schemes on the performance of an Apache server with TLS 1.2
- Kampanakis et al. [3] and Sikeridis et al. [4] measured the impact of post-quantum signatures in TLS 1.3 and when handshake failures will occur
- Paquin et al. [5] measured the effect of network latency and packet loss rate on handshake completion time

[1] Bos, Costello, Naehrig, Stebila. IEEE S&P 2015. https://eprint.iacr.org/2014/599
[2] Bos, Costello, Ducas, Mironov, Naehrig, Nikolaenko, Raghunathan, Stebila. ACM CCS 2016. https://eprint.iacr.org/2016/659
[3] Kampanakis, Sikeriis. https://eprint.iacr.org/2019/1276
[4] Sikeridis, Kampanaokis, Devetsikiotis. NDSS 2020. https://eprint.iacr.org/2020/071
[5] Paquin, Stebila, Tamvada. PQCrypto 2020. https://eprint.iacr.org/2019/1447

Experiments:

- Google: NewHope in TLS 1.2: https://www.imperialviolet.org/2016/11/28/cecpq1.html https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html
- An update from Google: https://www.imperialviolet.org/2018/12/12/cecpq2.html
- Google and Cloudflare: https://csrc.nist.gov/Presentations/2019/measuring-tls-key-exchange-with-post-quantum-kem

Conclusions:

- Seems like candidates based on lattice cryptography perform well for the key exchange of TLS 1.3 (due to their sizes and operations time)
- Seems like isogeny-based cryptography is not so usable at the moment due to the time of their operations
- Not so many experiments focused on the code-based cryptography
- Mainly focused on the key exchange part

Highlights:

- **A hybrid approach:** use classical and post-quantum cryptography simultaneously in the key exchange to reduce risk during migration:
    - Allows the possibility of early migration
    - Allows for FIPS compliance
    - Hybrid key exchange in TLS 1.3: draft-ietf-tls-hybrid-design-04 by Douglas Stebila, Scott Fluhrer, Shay Gueron, https://datatracker.ietf.org/doc/html/draft-ietf-tls-hybrid-design-04

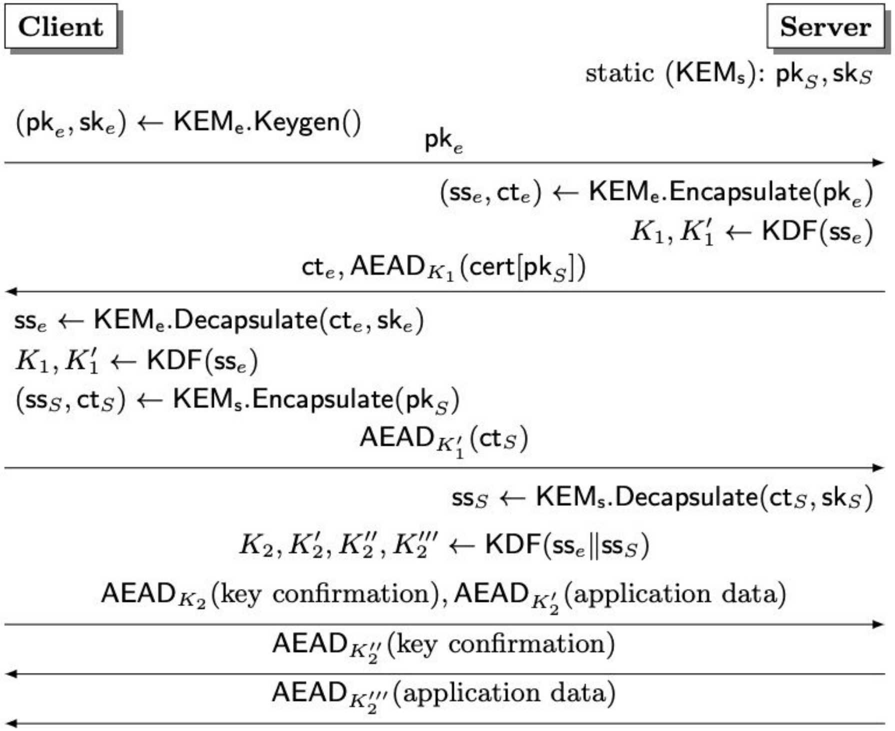
What about authentication in TLS 1.3?

- The harder process to migrate/experiment as it requires coordination
- Post-quantum signatures have the bigger sizes and/or bigger computational times
- Westerbaan measured the impact of various sizes by using dummy certificates on real connections: https://blog.cloudflare.com/sizing-up-post-quantum-signatures/

Conclusions:

- It seems like handshake and certificate signatures could be replaced for post-quantum ones for reliable connections, but it comes with a cost
- Not so many ‘real-world’ experiments of post-quantum authentication in TLS 1.3
- Not so many understanding on how the environment will react when the certificate chain will be migrated
- Not so many experiments with caching public keys
- Does this open the way to new forms of authentication?

New protocol: KEMTLS



Client

Server

static $(\mathsf{KEM_s})$: $\mathsf{pk}_S, \mathsf{sk}_S$

$(\mathsf{pk}_e, \mathsf{sk}_e) \leftarrow \mathsf{KEM_e.Keygen}()$

pk_e

$(\mathsf{ss}_e, \mathsf{ct}_e) \leftarrow \mathsf{KEM_e.Encapsulate}(\mathsf{pk}_e)$

$K_1, K_1' \leftarrow \mathsf{KDF}(\mathsf{ss}_e)$

$\mathsf{ct}_e, \mathsf{AEAD}_{K_1}(\mathsf{cert}[\mathsf{pk}_S])$

$\mathsf{ss}_e \leftarrow \mathsf{KEM_e.Decapsulate}(\mathsf{ct}_e, \mathsf{sk}_e)$

$K_1, K_1' \leftarrow \mathsf{KDF}(\mathsf{ss}_e)$

$(\mathsf{ss}_S, \mathsf{ct}_S) \leftarrow \mathsf{KEM_s.Encapsulate}(\mathsf{pk}_S)$

$\mathsf{AEAD}_{K_1'}(\mathsf{ct}_S)$

$\mathsf{ss}_S \leftarrow \mathsf{KEM_s.Decapsulate}(\mathsf{ct}_S, \mathsf{sk}_S)$

$K_2, K_2', K_2'', K_2''' \leftarrow \mathsf{KDF}(\mathsf{ss}_e \| \mathsf{ss}_S)$

$\mathsf{AEAD}_{K_2}(\text{key confirmation}), \mathsf{AEAD}_{K_2'}(\text{application data})$

$\mathsf{AEAD}_{K_2''}(\text{key confirmation})$

$\mathsf{AEAD}_{K_2'''}(\text{application data})$

Fig. 2. Overview of KEMTLS with server-only authentication.

- Post-quantum signatures are big and have costly operations:
  - solution: use KEMs for authentication

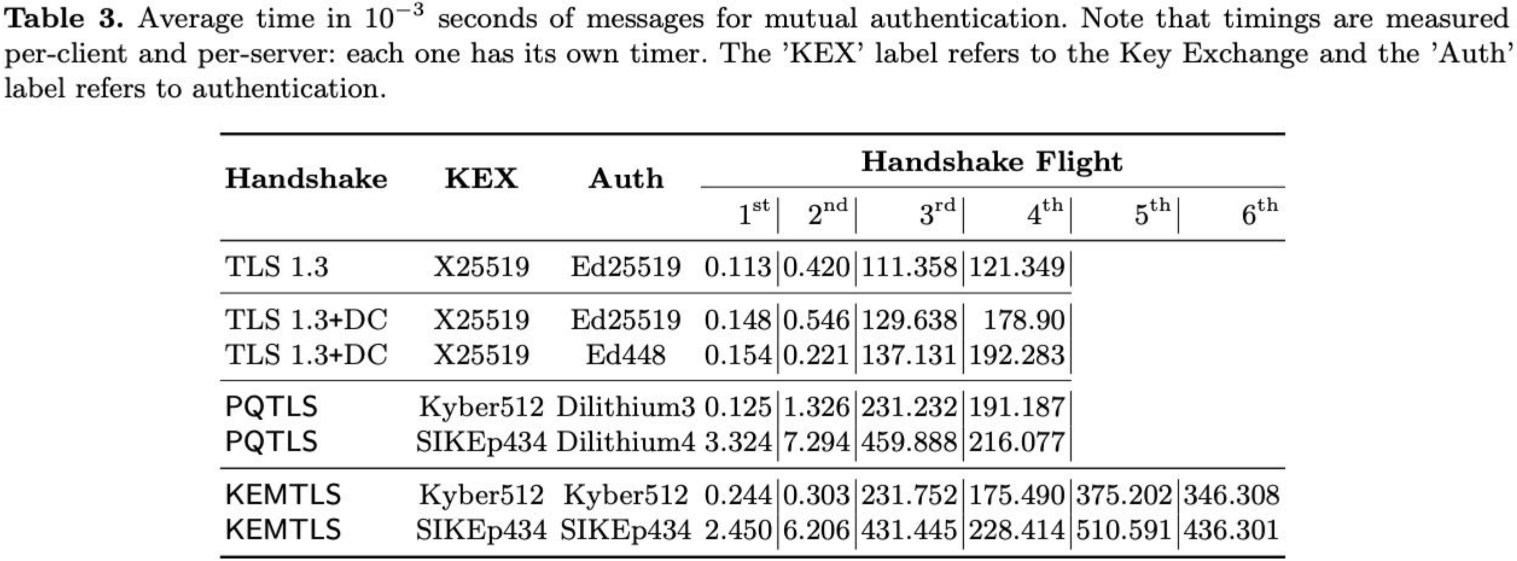


Table 3. Average time in 10^{-3} seconds of messages for mutual authentication. Note that timings are measured per-client and per-server: each one has its own timer. The 'KEX' label refers to the Key Exchange and the 'Auth' label refers to authentication.

Handshake	**KEX**	**Auth**	**Handshake Flight**					
			1st	2nd	3rd	4th	5th	6th
TLS 1.3	X25519	Ed25519	0.113	0.420	111.358	121.349		
TLS 1.3+DC	X25519	Ed25519	0.148	0.546	129.638	178.90		
TLS 1.3+DC	X25519	Ed448	0.154	0.221	137.131	192.283		
PQTLS	Kyber512	Dilithium3	0.125	1.326	231.232	191.187		
PQTLS	SIKEp434	Dilithium4	3.324	7.294	459.888	216.077		
KEMTLS	Kyber512	Kyber512	0.244	0.303	231.752	175.490	375.202	346.308
KEMTLS	SIKEp434	SIKEp434	2.450	6.206	431.445	228.414	510.591	436.301

- Peter Schwabe, Douglas Stebila, Thom Wiggers
- ACM CCS 2020: https://eprint.iacr.org/2020/534
- ESORICS 2021: https://eprint.iacr.org/2021/779
- Sofía Celi, Peter Schwabe, Douglas Stebila, Nick Sullivan, Thom Wiggers: https://datatracker.ietf.org/doc/html/draft-celi-wiggers-tls-authkem-01
- Measuring and Implementing KEMTLS: https://eprint.iacr.org/2021/1019.pdf

Highlights:

- **Anonymity:** can KEMTLS-PDK be used with ECH?
- **Deniability**: a form of offline deniability can be achieved:
    - Are the quantum attacks to deniability?

OPEN PROBLEMS

- What about other variants of authentication?
    - Password-based authentication:
        - PAKEs can be considered quantum-annoyant
- What about different “variants” of TLS:
    - DTLS
    - QUIC
- PKI and PQ signatures/KEM algorithms
- What about TLS usage for video/audio?

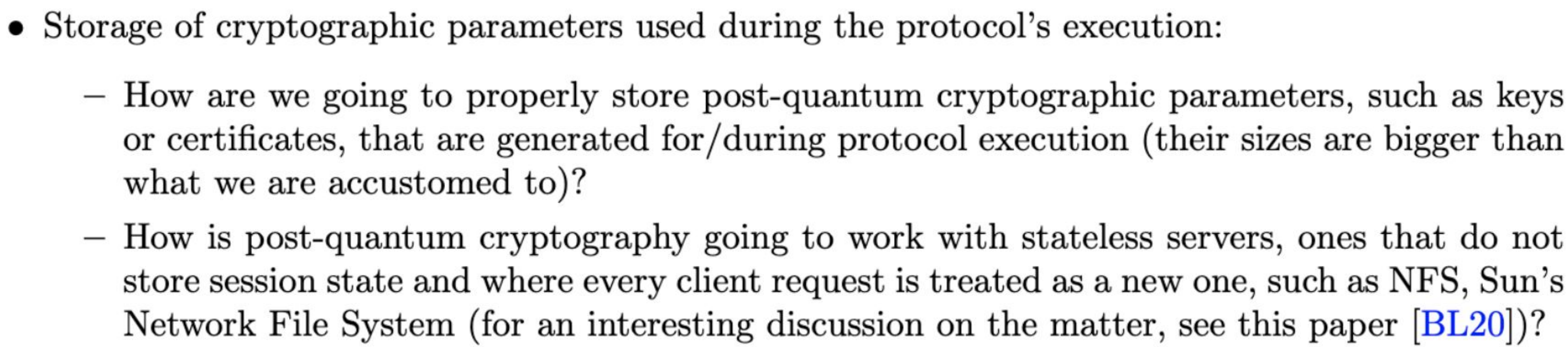
THE FUTURE

- The quantum world: VOPRFs are usually based in Diffie-Hellman (DDH) or RSA:
    - Isogeny-based proposal: *Oblivious Pseudorandom Functions from Isogenies* by Dan Boneh and Dmitry Kogan and Katharine Woo
    - Lattice-based proposal: *Round-optimal Verifiable Oblivious*
    - *Pseudorandom Functions from Ideal Lattices* by Martin R. Albrecht, Alex Davidson, Amit Deo, and Nigel P. Smart
- Quantum annoyance?: *The "quantum annoying" property of password-authenticated key exchange protocols* by Edward Eaton and Douglas Stebila:

    "a scheme is said to be "quantum-annoying" if a quantum computer can compromise the security of the scheme, but only by solving one discrete logarithm for each guess of a password (sic, for example). Considering that early quantum computers will likely take quite long to solve even a single discrete logarithm, a quantum-annoying PAKE, combined with a large password space, could delay the need for a post-quantum replacement by years, or even decades"

THE FUTURE

- There are many open questions:



- Storage of cryptographic parameters used during the protocol's execution:
  - How are we going to properly store post-quantum cryptographic parameters, such as keys or certificates, that are generated for/during protocol execution (their sizes are bigger than what we are accustomed to)?
  - How is post-quantum cryptography going to work with stateless servers, ones that do not store session state and where every client request is treated as a new one, such as NFS, Sun's Network File System (for an interesting discussion on the matter, see this paper [BL20])?

- Preservation of protocols as we know them:
  - Can we achieve the same security or privacy properties as we use them today?
  - Can protocols change: should we change, for example, the way DNSSEC or the PKI work? Can we consider this radical change?
  - Can we integrate and deploy novel ways to achieve authentication?
  - At the TLS level, can be use something like KEMTLS [SSW20]?
- Hardware (or novel alternative to hardware) usage during protocol's execution:
  - Is post-quantum cryptography going to impact network function virtualization (as used in 5G cellular networks)?
  - Will middleware, such as middleboxes, be able to handle post-quantum cryptography?
  - What will be the impacts on mobile device's connections?
  - What will be the impacts on old servers and clients?
- Novel attacks:
  - Will post-quantum cryptography increase the possibility of mounting denial of service attacks?

- Efficiency of algorithms: can we make them faster at the software, hardware (by using acceleration or FPGA-based research) or at an algorithmic level (with new data structures or parallelization techniques) to meet the requirements of network protocols and ever-fastest connections?

- Can we use new mechanisms to accelerate algorithms (such as, for example, the usage of floating point numbers as in the Falcon signature scheme)? Will this lead to portability issues as it might be dependent on the underlying architecture?

- What is the asymptotic complexity of post-quantum algorithms (how they impact time and space)?

- How will post-quantum algorithms work on embedded devices due to their limited capacity (see this paper [BZH+21], for more explanations)?

- How can we avoid attacks, failures in security proofs and misuse of APIs?

- Can we provide correct testing of these algorithms?

- Can we ensure constant-time needs for the algorithms?

- What will happen in a disaster-recovery mode: what happens if an algorithm is found to be weaker than expected or is fully broken? How will we be able to remove or update this algorithm? How can we make sure there are transition paths to recover from a cryptographical weakening?

- What are the needs of different systems? While we know what the needs of different protocols are, we don’t know exactly how all deployed systems and services work. Are there further restrictions?

- On certain systems (for example, on the PKI), when will the migration happen, and how will it be coordinated?

- How will the migration be communicated to the end-user?

- How will we deprecate pre-quantum cryptography?

- How will we integrate post-quantum cryptography into systems where algorithms are hardcoded (such as IoT devices)?

- Who will maintain implementations of post-quantum algorithms and protocols? Is there incentive and funding for a diverse set of interoperable implementations?

THE FUTURE

- Find more of these open questions: https://github.com/claucece/HACS-2022-Celi/blob/main/migration-PQ/PQ-Migration.pdf
- A need for a proper systematization of these challenges -> work in progress
- A need for a proper systematization of the challenges that isogenies face -> work in progress
- Join the slack for discussion! https://join.slack.com/t/post-quantumfuture/shared_invite/zt-16gd8po13-~i0ReTJKn3J_F6AZXlJSBA

THANK YOU!

@claucece