PQNet

Sofía Celi

Agenda

- Introduction
- Network Protocols
- TLS
- Pitfalls

What is this and why we are here?

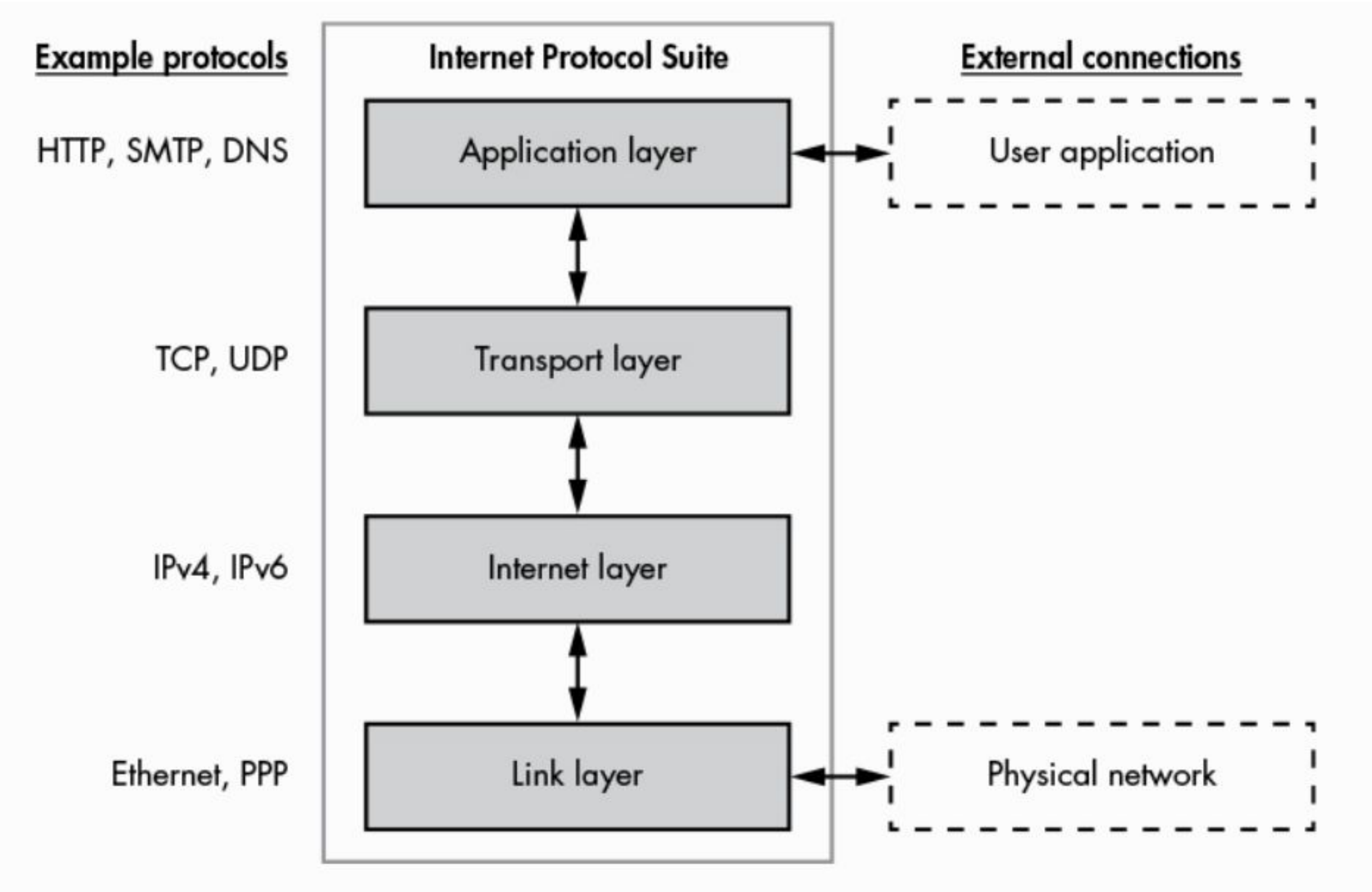
- Create a community that thinks on how to integrate post-quantum algorithms into network protocols (or how to change them)
- Is there an space for isogenies?

- Special event later

Network Protocols

- Why networks? To share information
- Why protocols? To:
  - Maintain session state
  - Identify nodes
  - Control the flow of data
  - Control de order of the flow of data
  - Define format and encoding
  - Define errors

- Protocols are stacked on top of another
- The ‘big’ Internet protocols: TCP/IP, UDP


Example protocols
Internet Protocol Suite
External connections
HTTP, SMTP, DNS
Application layer
User application
TCP, UDP
Transport layer
IPv4, IPv6
Internet layer
Ethernet, PPP
Link layer
Physical network

Securing the network protocols

- Integrating cryptography into the protocols themselves:
  - Maintain data integrity
  - Maintain confidentiality
  - Disallow impersonation
- Implementing the cryptography into the protocols themselves
- Deploying the cryptography into the protocols themselves

Integrating cryptography into the protocols themselves

- The famous case:
  - Transport Layer Security (basically, used everywhere)
- But there are more:
  - DNSSEC
  - IPSEC
  - SSH
  - Signal
  - Wireguard

TLS

- Current version: 1.3
- Basic mode: server authentication
- Two phases:
    - Handshake:
        - Key Exchange
        - Authentication
        - Confirmation
    - Record

Why TLS1.3?

- Easy transition to other algorithms
- Encrypt as much as possible
- Be efficient
- Maintain use cases

```
       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
```

PQTLS?

- Mapping post-quantum cryptography into TLS (and all other protocols) is simple in the theoretical sense; but:
    - are algorithms suitable to be used given their increased operation times?
    - are algorithms suitable to be used given their increased size?
    - are algorithms suitable to be used is they add extra round-trips?
    - are algorithms suitable to be used if they have decryption failures?

Implementing cryptography into the protocols themselves

- Bugs/incorrectness of the algorithms used can cause protocol attacks:
  - They get filtered out into lots of machines
  - It can take a lot of time to remove them once deployed
- Post-quantum cryptography can be more expensive:
  - TLS, as it is currently, is not computationally expensive for the majority of cases
  - Handshake (itself) does not seem to be the problem
  - Certificate transmission can be the problem:
    - At least it is needed a root certificate and an intermediate certificate, plus the leaf certificate
    - Intermediate certificates can be cached or looked up (which creates cost)
    - If certificates are too big, they can cost in an extra TCP round-trip

Pitfalls

- Once it is there, it is difficult to ‘un-deploy’ cryptography or to update it
- In the case of certificates or public values: more difficulty
  - TLS: unclear certificate revocation to this day
- Gigantic certificates might not be the best thing:
  - Can we cached them?
  - Can we compress them?
  - Are there already certificates with a big size, due to different extensions added?
- How to achieve 0-RTT?
- How to do ECH?

The path forward for algorithms

- Learn from past mistakes:
  - Handwritten security proofs
  - Peer-reviewed specifications
  - Formal verification
  - Formal implementation (take into account memory constraints and constant-time)
  - Careful analysis of compiler optimizations
  - Binary and symbolic analysis of code
  - User-tested APIs

A winning situation

- The deployment of TLS 1.3: success story
- ‘Tracking the deployment of TLS 1.3 on the web: a story of experimentation and centralization’: https://dl.acm.org/doi/abs/10.1145/3411740.3411742
- ‘A Comprehensive Symbolic Analysis of TLS 1.3’: https://acmccs.github.io/papers/p1773-cremersA.pdf

Integrate isogeny-based cryptography into network protocols

- Are there one-to-one mappings?
- Will protocols have to change?
- Will we be relying more in cache/storage and public lists?

Thank you!