Post-Quantum DNSSEC with Faster TCP Fallbacks

Aditya S. Rawat, Mahabir P. Jhanwar

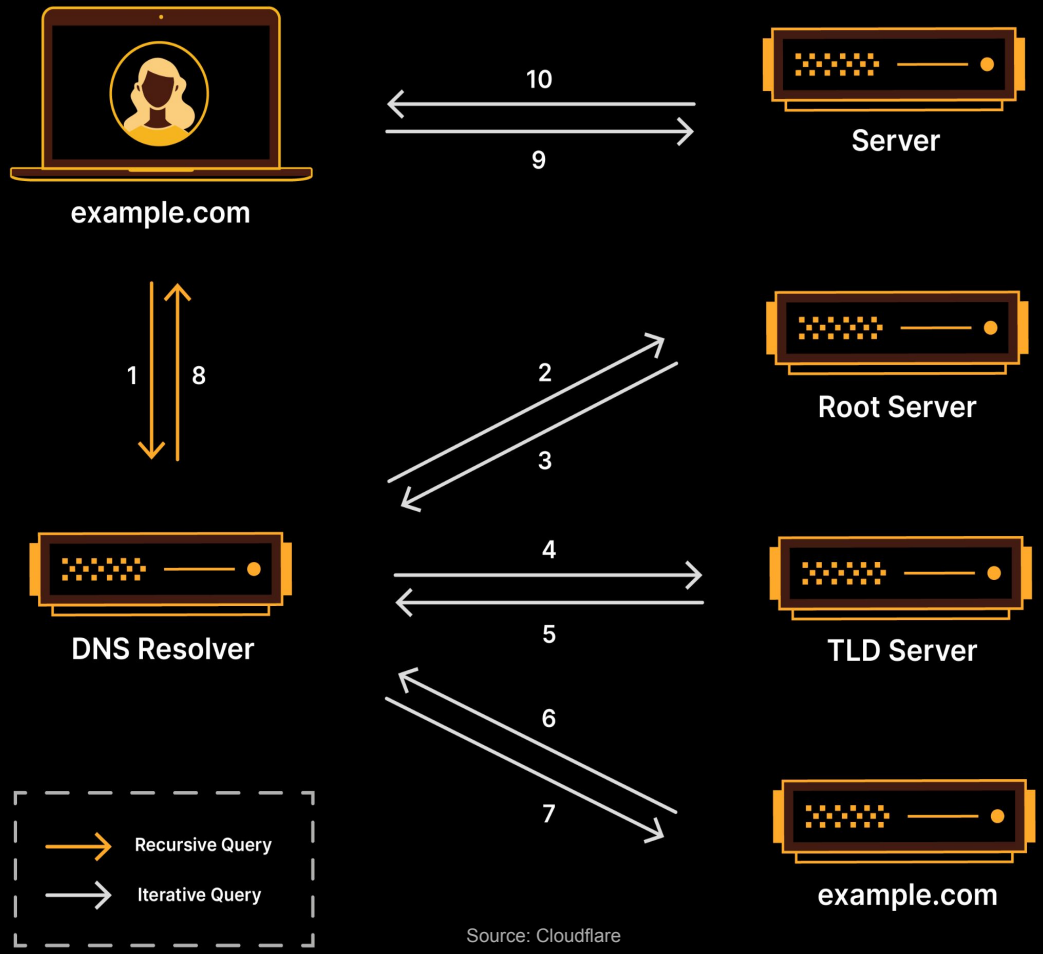
Ashoka University

Outline

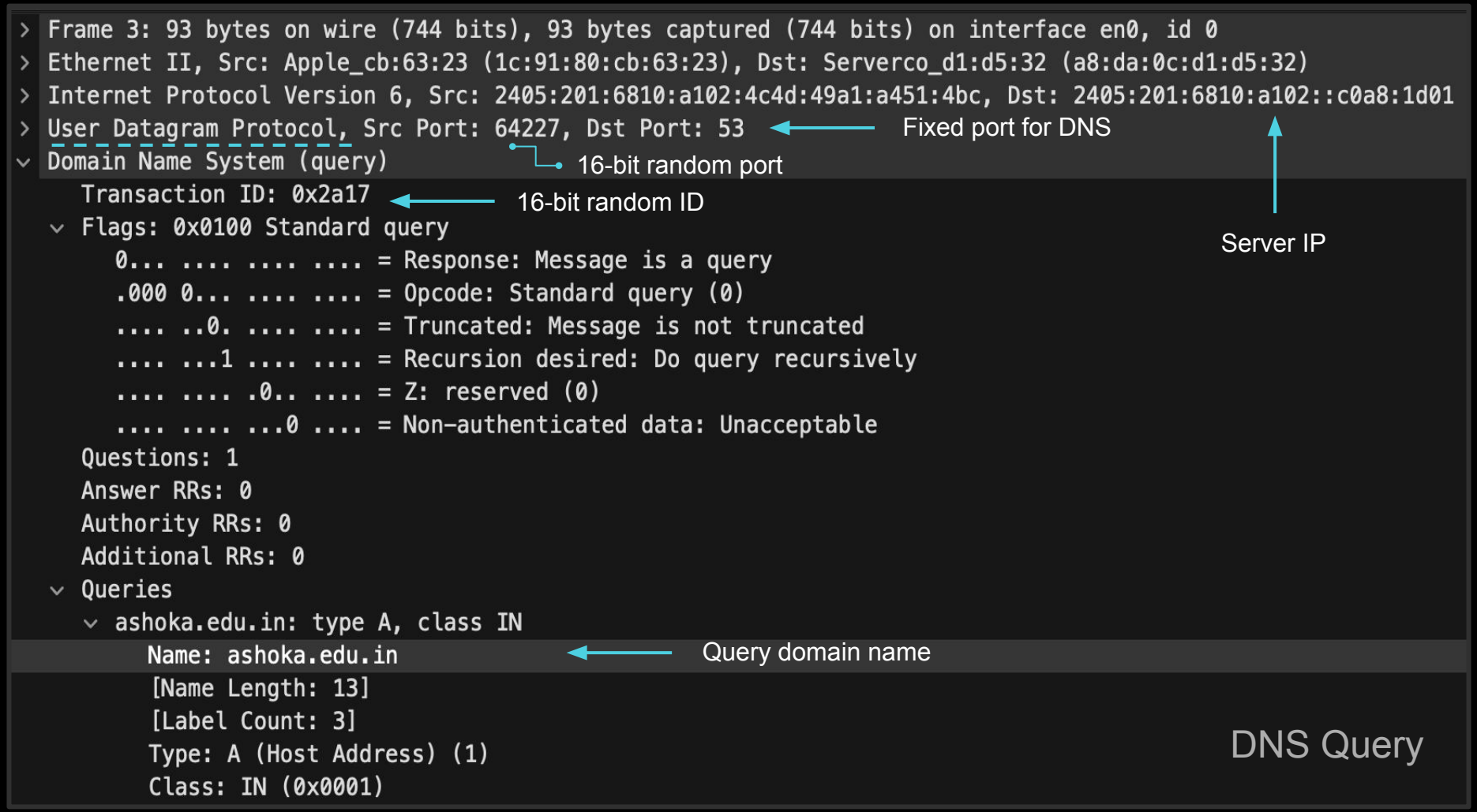
- Preliminaries
  - DNS, DNSSEC
- DNSSEC PQC migration
  - 2x slower DNS performance (vs RSA / ECDSA)
- TurboDNS
  - PQ-DNSSEC *as fast as* classical DNSSEC

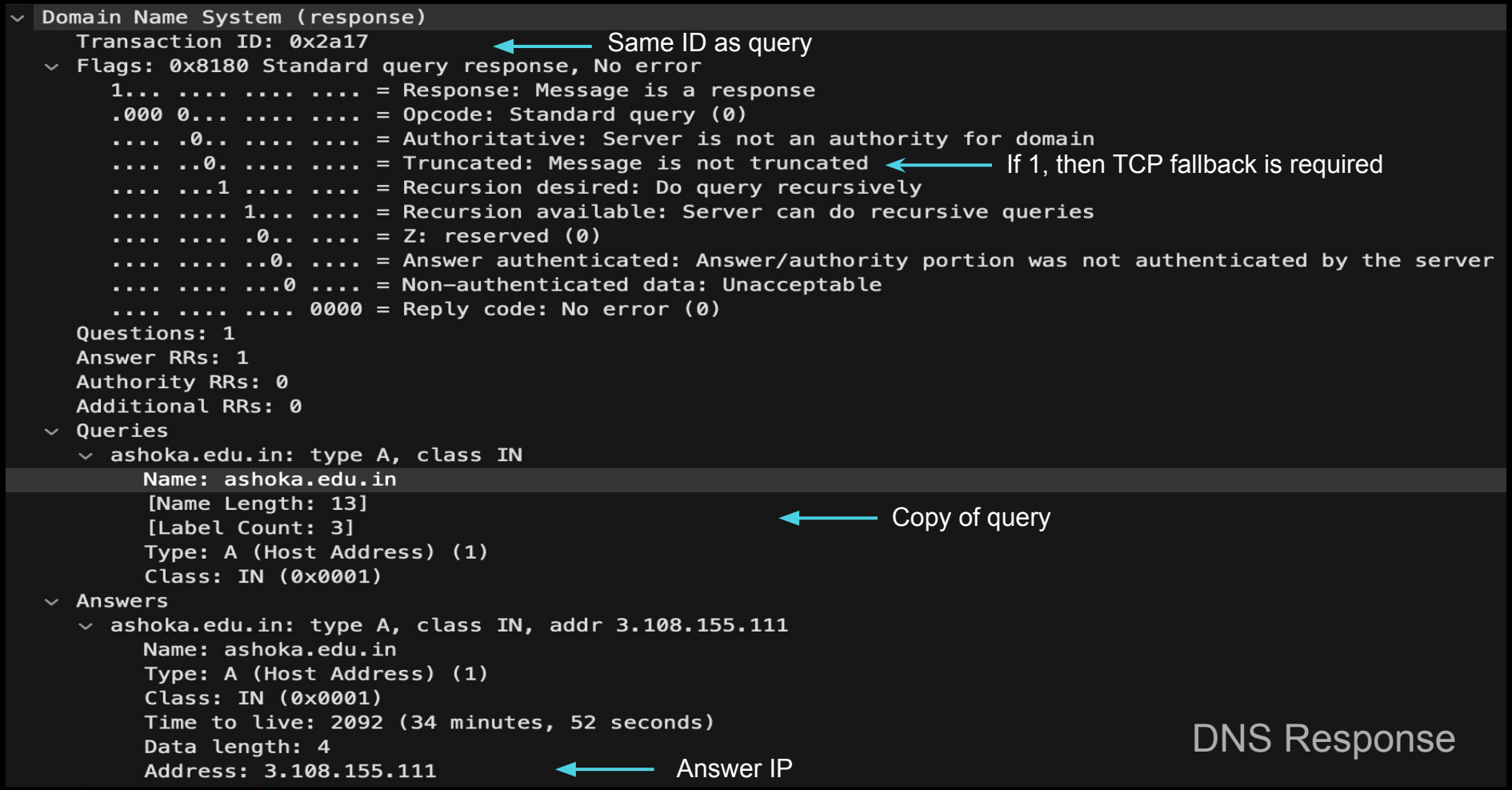
DNS

example.com → 1.2.3.4



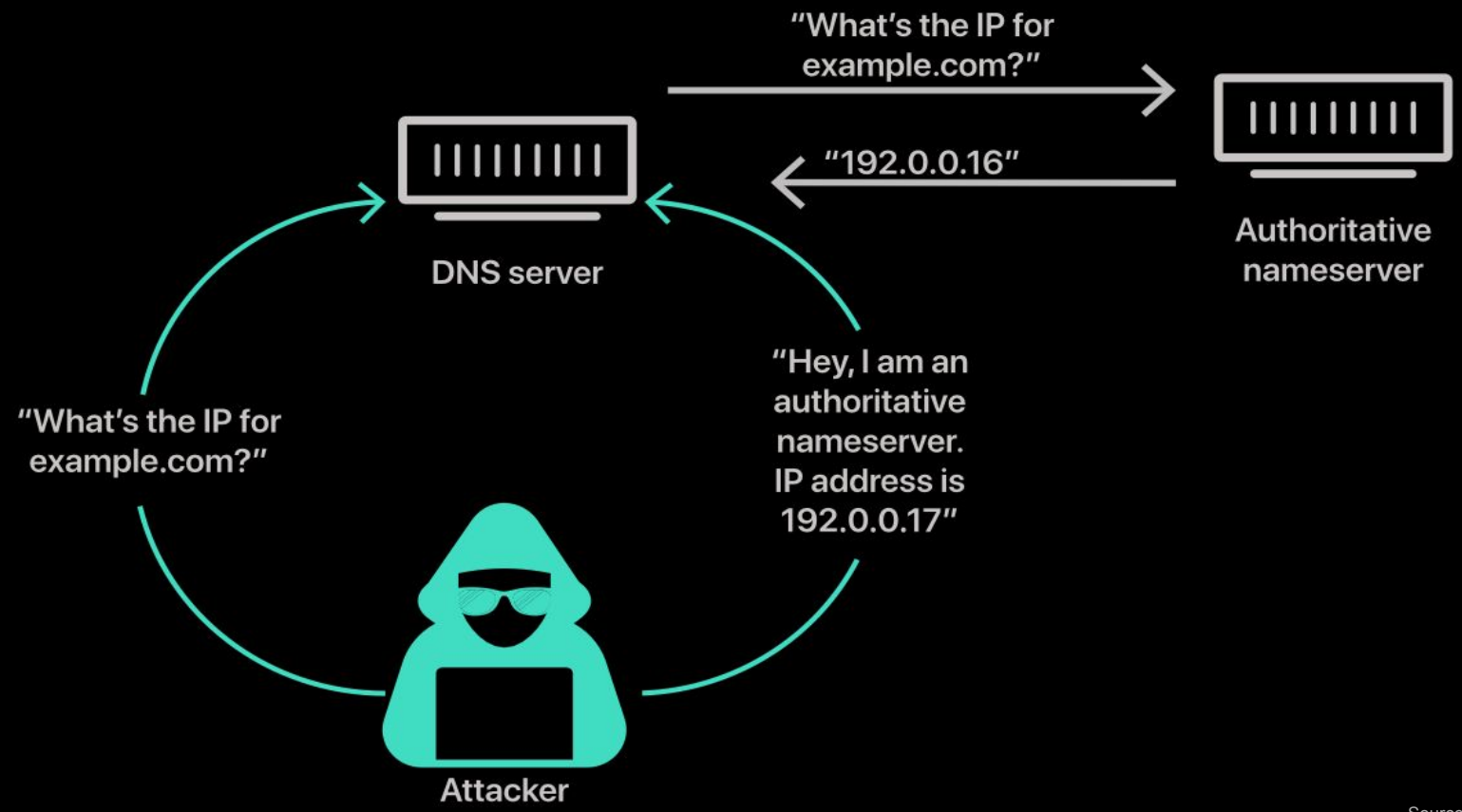
Source: Cloudflare


Frame 3: 93 bytes on wire (744 bits), 93 bytes captured (744 bits) on interface en0, id 0
Ethernet II, Src: Apple_cb:63:23 (1c:91:80:cb:63:23), Dst: Serverco_d1:d5:32 (a8:da:0c:d1:d5:32)
Internet Protocol Version 6, Src: 2405:201:6810:a102:4c4d:49a1:a451:4bc, Dst: 2405:201:6810:a102::c0a8:1d01
User Datagram Protocol, Src Port: 64227, Dst Port: 53
Fixed port for DNS
16-bit random port
Server IP
Domain Name System (query)
Transaction ID: 0x2a17
16-bit random ID
Flags: 0x0100 Standard query
0... .... = Response: Message is a query
.000 0... .... = Opcode: Standard query (0)
.... ..0. .... = Truncated: Message is not truncated
....1 .... = Recursion desired: Do query recursively
.... .0.. .... = Z: reserved (0)
....0 .... = Non-authenticated data: Unacceptable
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
ashoka.edu.in: type A, class IN
Name: ashoka.edu.in
Query domain name
[Name Length: 13]
[Label Count: 3]
Type: A (Host Address) (1)
Class: IN (0x0001)
DNS Query

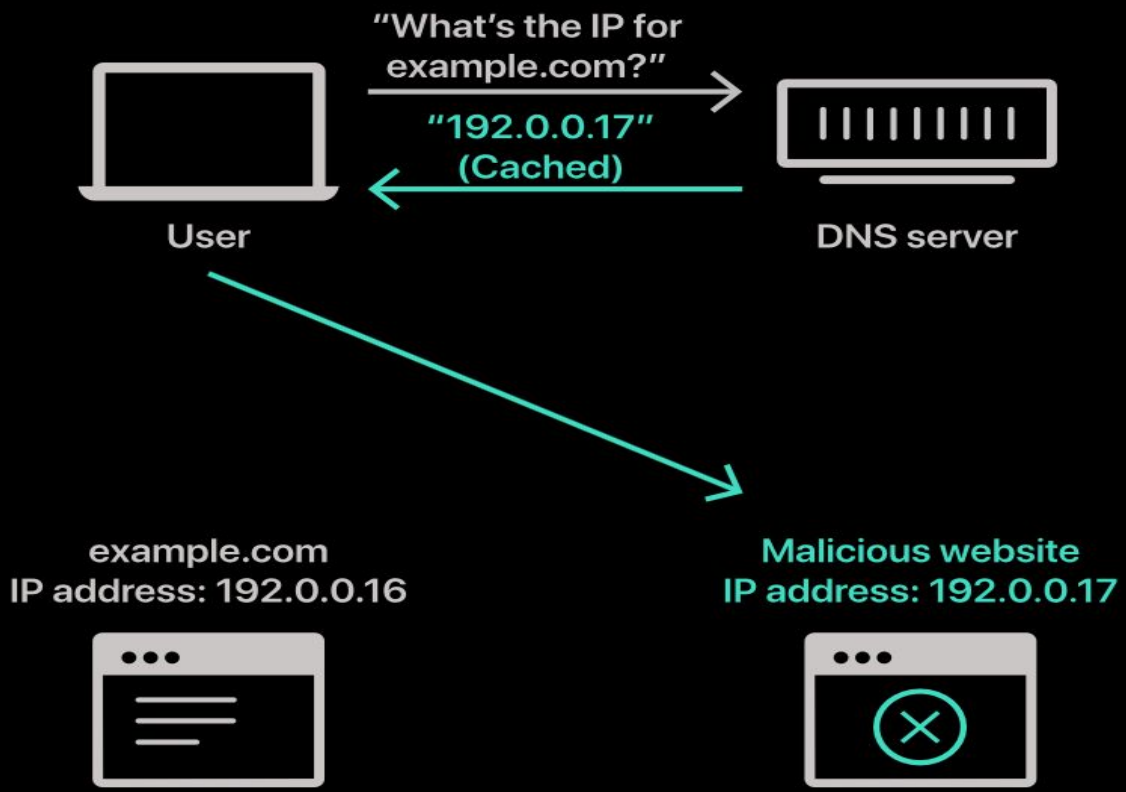

Domain Name System (response)
Transaction ID: 0x2a17
Same ID as query
Flags: 0x8180 Standard query response, No error
1... .... = Response: Message is a response
.000 0... .... = Opcode: Standard query (0)
.... .0.. .... = Authoritative: Server is not an authority for domain
.... ..0. .... = Truncated: Message is not truncated
If 1, then TCP fallback is required
....1 .... = Recursion desired: Do query recursively
.... 1... .... = Recursion available: Server can do recursive queries
.... .0.. .... = Z: reserved (0)
....0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
....0 .... = Non-authenticated data: Unacceptable
.... 0000 = Reply code: No error (0)
Questions: 1
Answer RRs: 1
Authority RRs: 0
Additional RRs: 0
Queries
ashoka.edu.in: type A, class IN
Name: ashoka.edu.in
[Name Length: 13]
[Label Count: 3]
Copy of query
Type: A (Host Address) (1)
Class: IN (0x0001)
Answers
ashoka.edu.in: type A, class IN, addr 3.108.155.111
Name: ashoka.edu.in
Type: A (Host Address) (1)
Class: IN (0x0001)
Time to live: 2092 (34 minutes, 52 seconds)
Data length: 4
Address: 3.108.155.111
Answer IP
DNS Response

There’s only one problem, though.

Cache Poisoning Attack



Source: Cloudflare



Source: Cloudflare

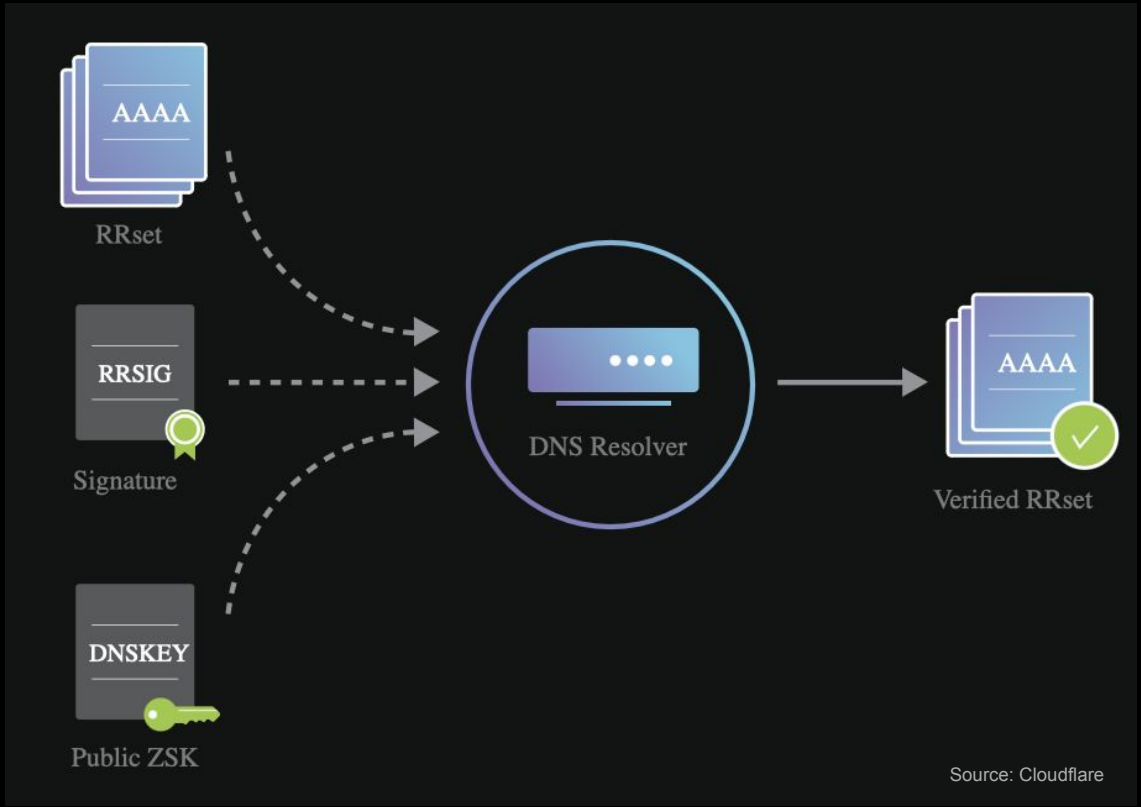
Need to validate responses.

Authenticate server + Check msg integrity

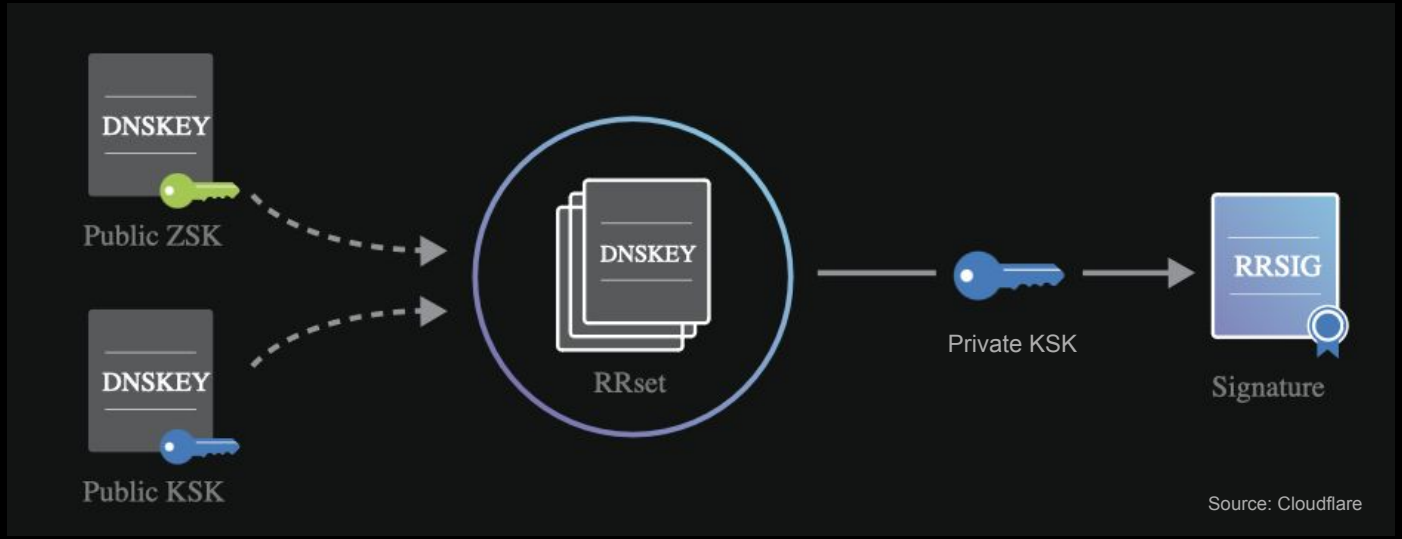
DNSSEC

DNS + Signatures

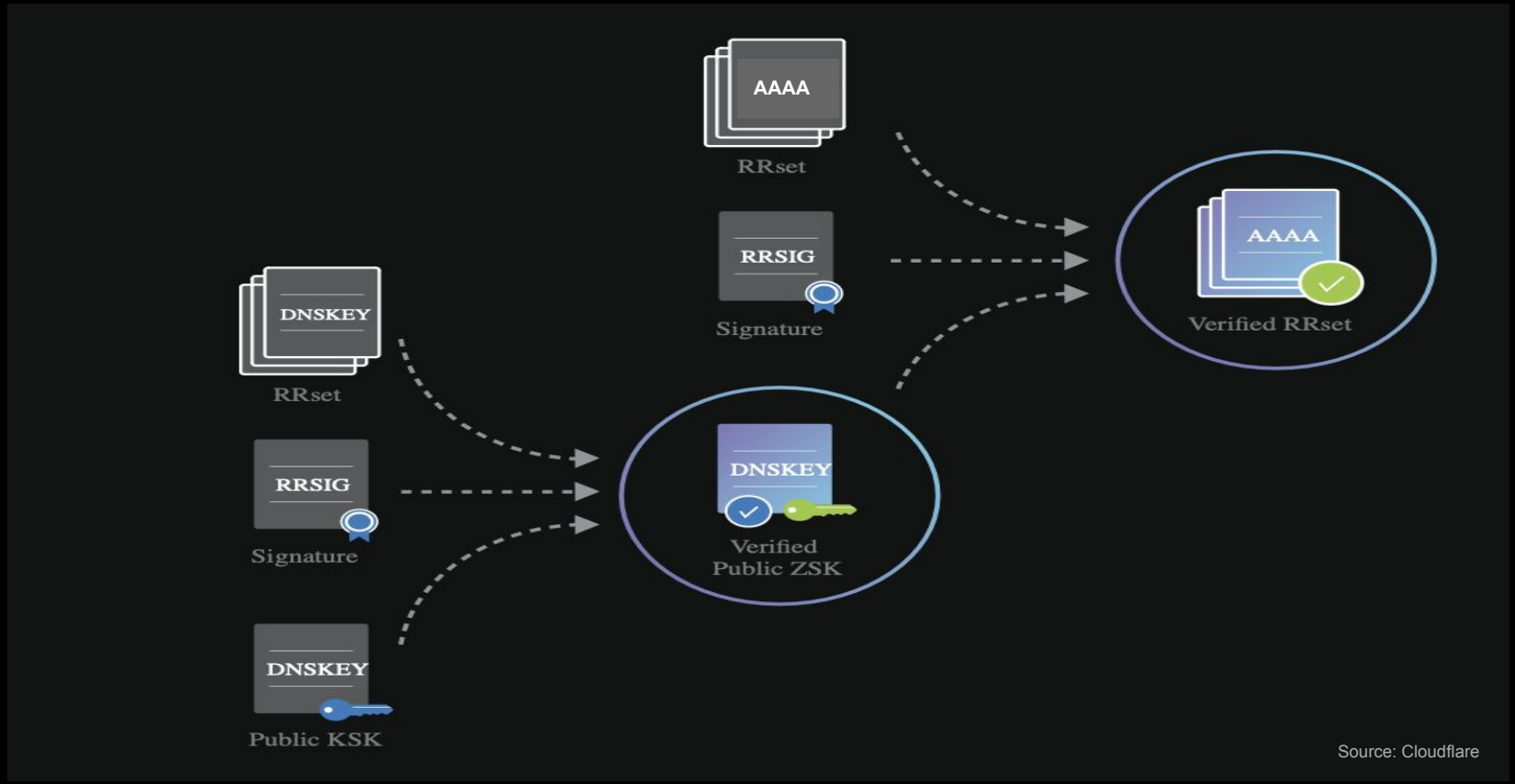
DNSSEC Validation



Key Signing

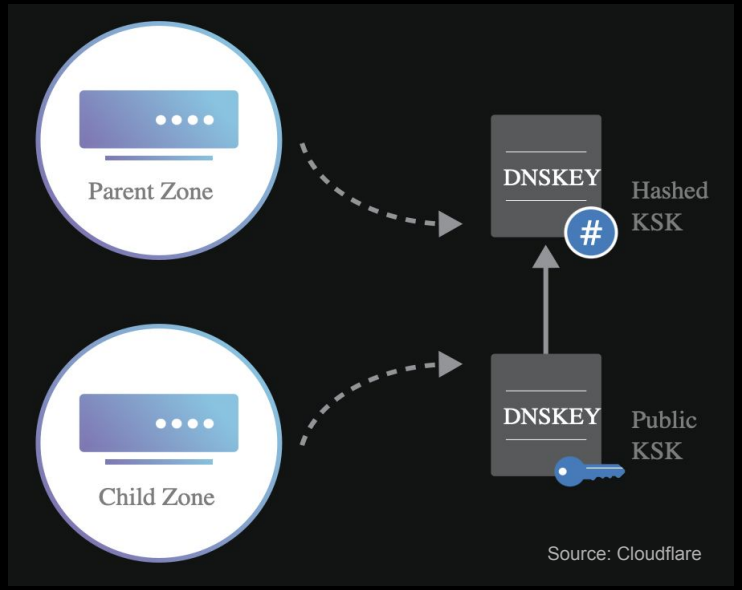

DNSKEY
Public ZSK
DNSKEY
Public KSK
DNSKEY
RRset
Private KSK
RRSIG
Signature
Source: Cloudflare

Full DNSSEC Validation

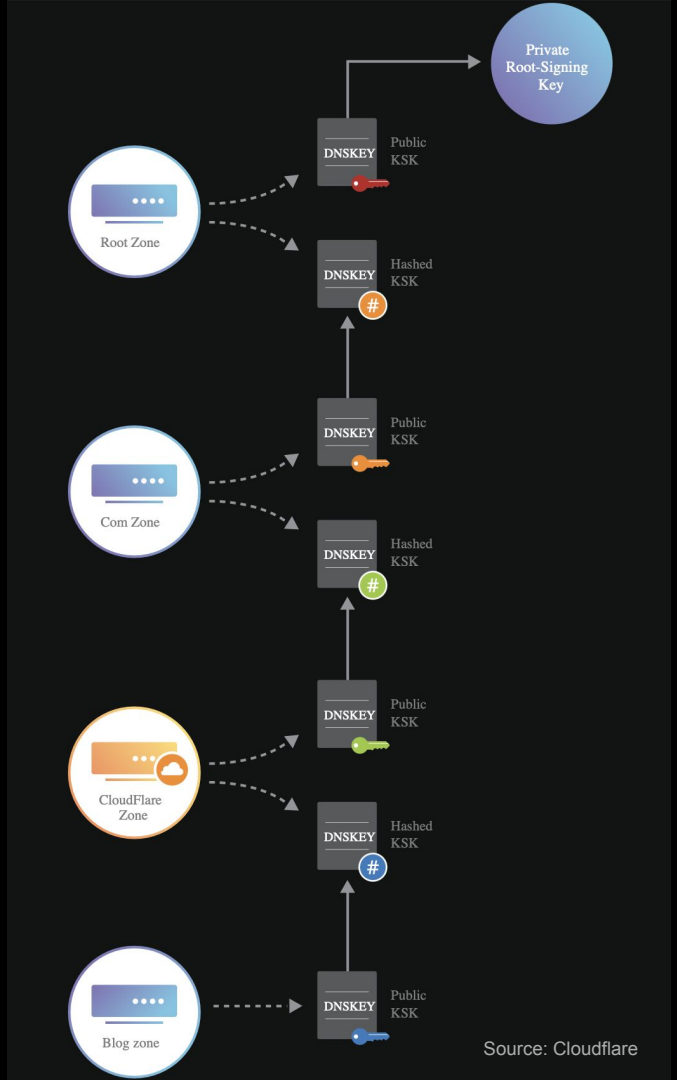


DNSSEC PKI

DS Record


Parent Zone
DNSKEY
Hashed
KSK
#
Child Zone
DNSKEY
Public
KSK
Source: Cloudflare

Chain of Trust

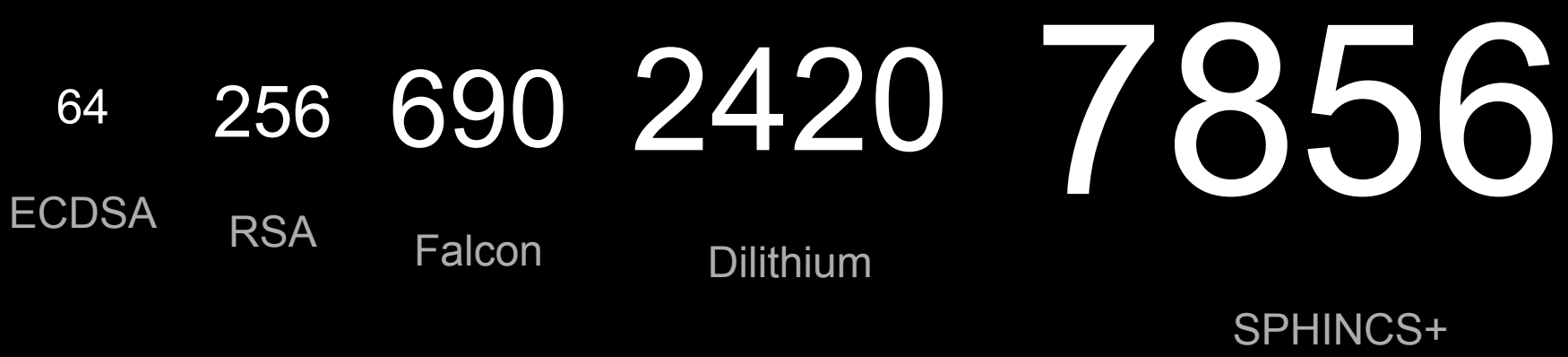


So far, so good.

But there’s a caveat.

signatures ⇒ bigger DNS messages

Size in Bytes



If size > 1232 bytes, then two things may happen.

1232 = 1280 (IPv6 MTU) – 40 (IPv6 Header) – 8 (UDP Header)

1. DNS message is fragmented by routers.

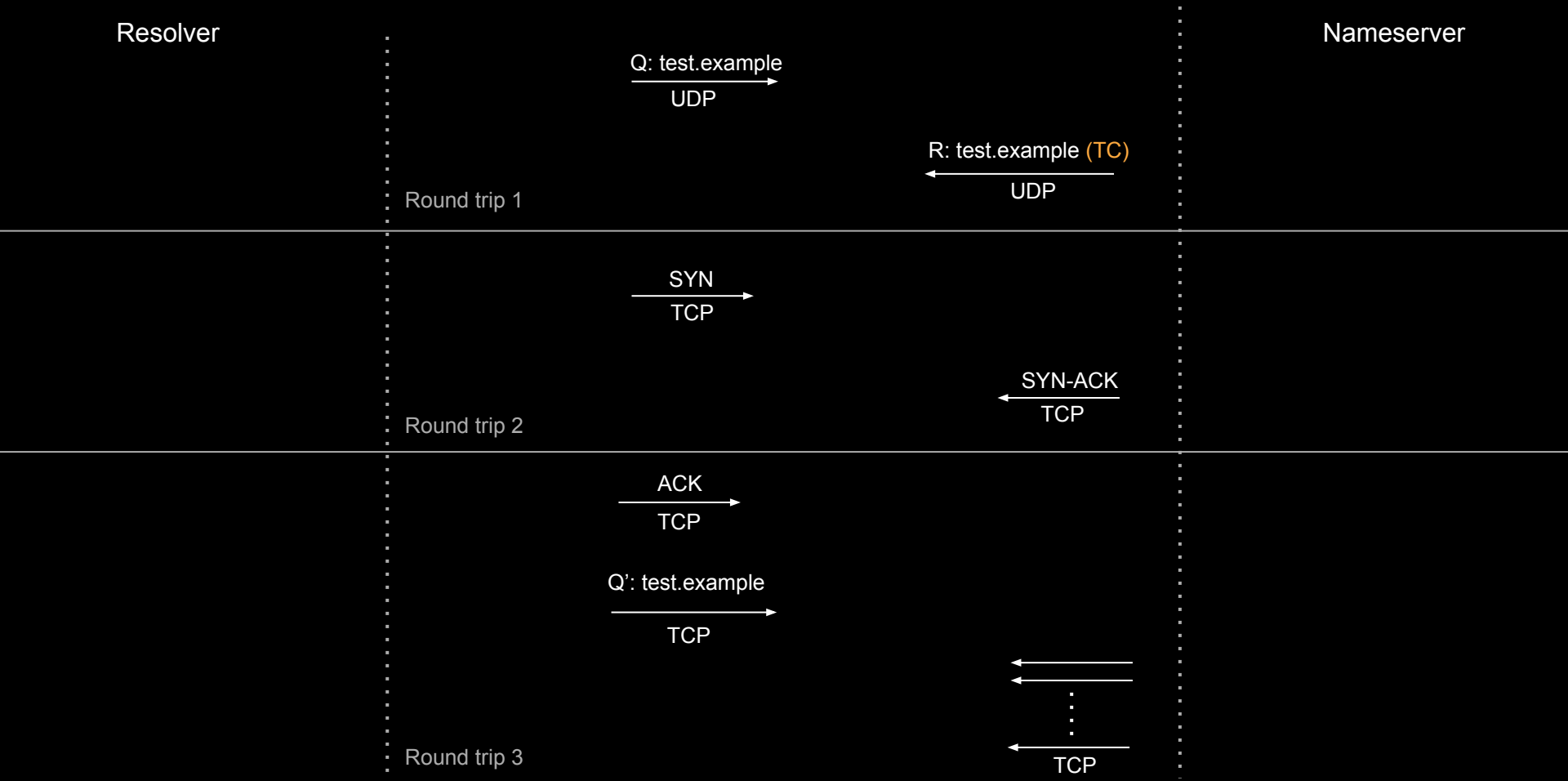
“IP Fragmentation Considered Fragile”

— RFC 8900

2. DNS message is marked truncated (TC) by server.

TC messages require a TCP retransmit.

Standard DNS with TCP Fallback



3 round trips

DNS Resolution Time*

*Type A
10 ms latency
100 Mbps bandwidth

Can we do better?

Naive way

Bypass UDP-trip and directly start with TCP

DNS Protocol – RFC 1035

Send initial query over UDP
Fallback to TCP in case of truncated (TC) response

Truncation (TC) depends upon

Response size

- Namserver’s signature algorithm
- whether minimal responses are enabled or not
- NOERROR or NXDOMAIN
- No. of RRs in RRsets
- …

Nameserver’s EDNS0 send size (can range from 512 – 4096)

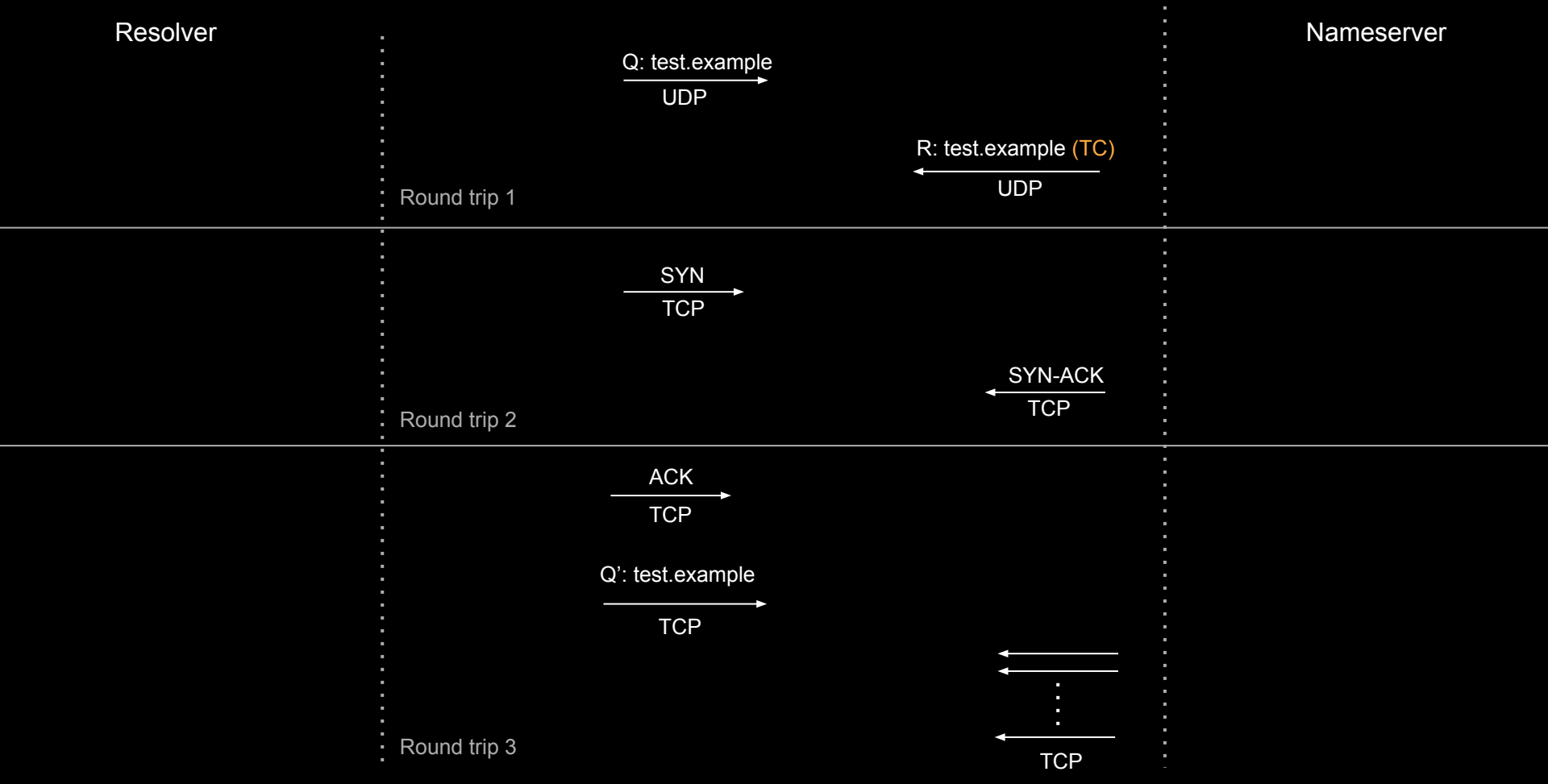
Not straightforward to predict whether TCP will be required.

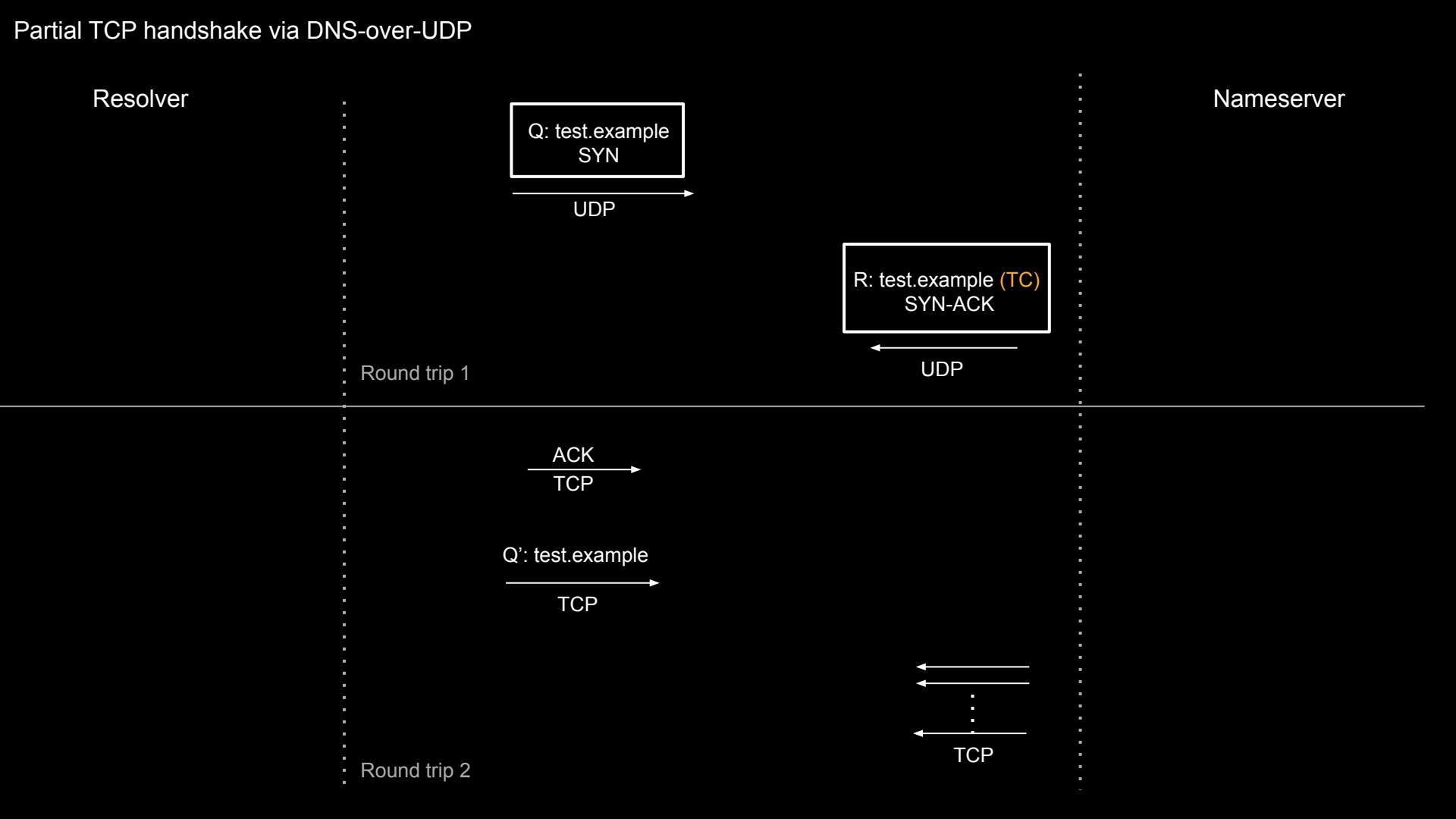
Can we reduce round trips while still adhering to the RFC?

TurboDNS

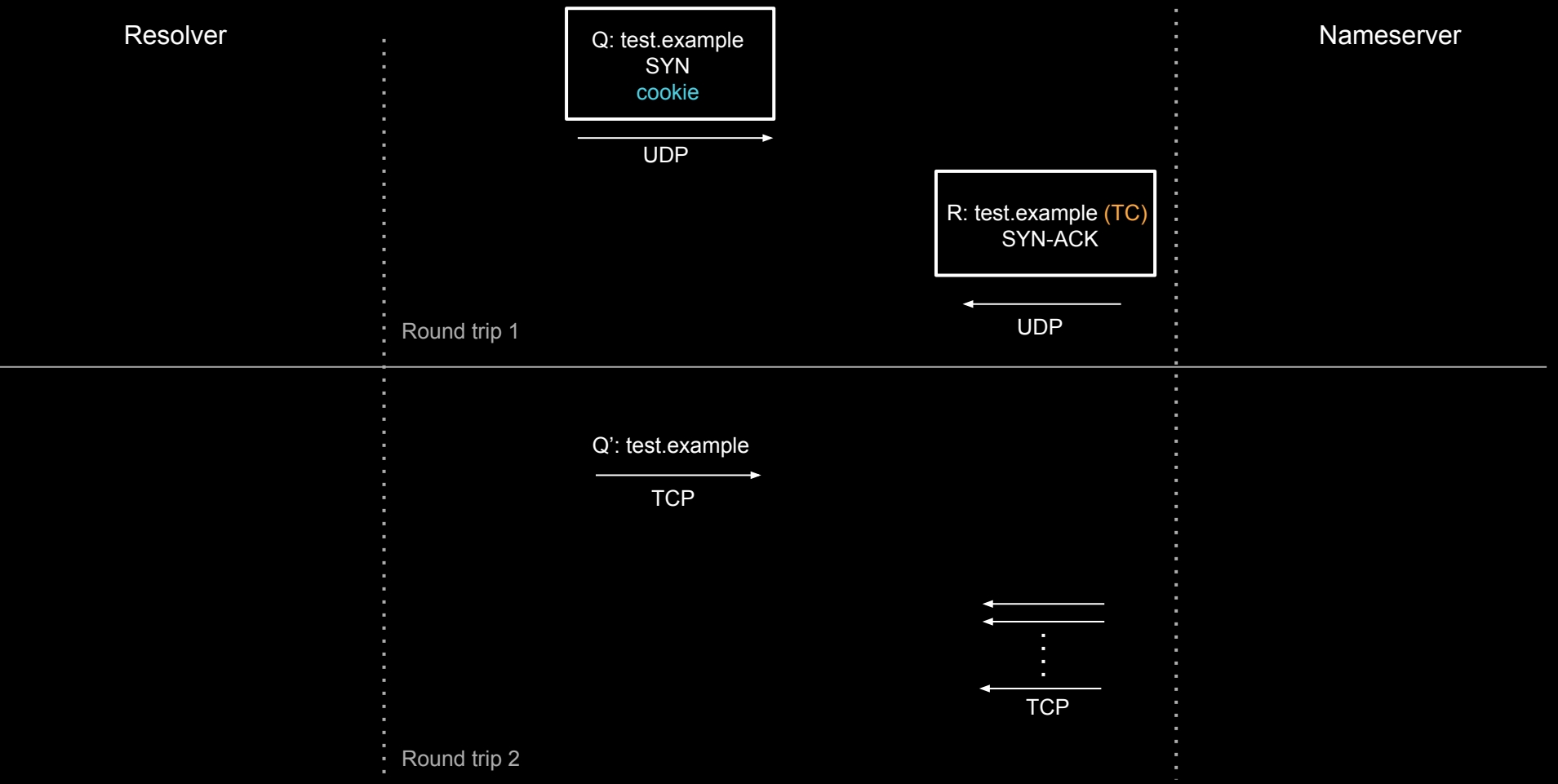
3 → 1 round trip

Standard DNS with TCP Fallback

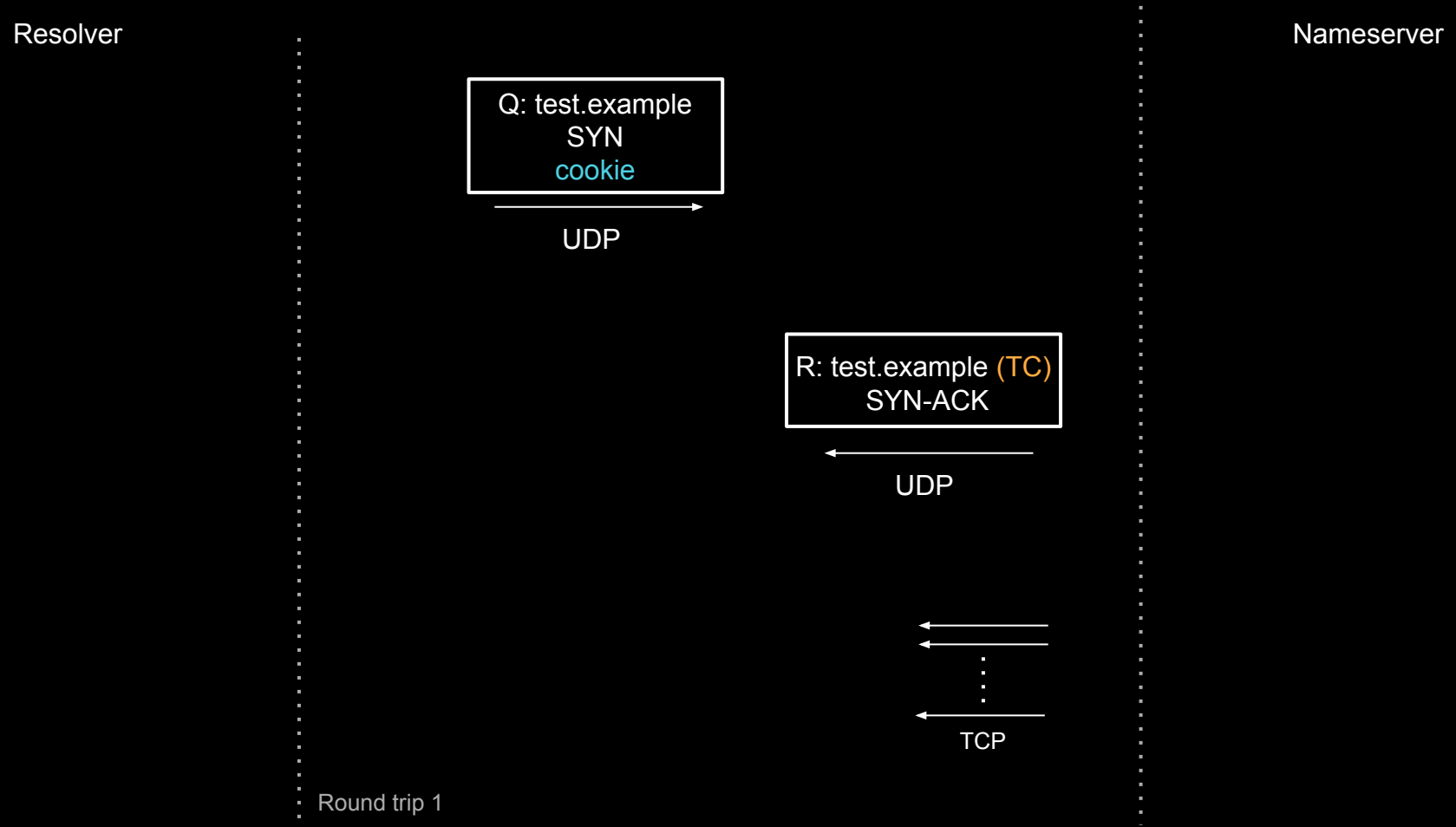



Partial TCP handshake via DNS-over-UDP
Resolver
Nameserver
Q: test.example
SYN
UDP
R: test.example (TC)
SYN-ACK
UDP
Round trip 1
ACK
TCP
Q': test.example
TCP
TCP
Round trip 2

Eliminating ACK with cookie



Eliminating redundant TCP query



TD-Cookie

TD-Cookie

- Verifies client IP ownership (similar to TFO — RFC 7413)
- Doesn’t require server to maintain per-client state
- Prevents attacks such as server resource exhaustion, amplification & reflection
- TD-Cookie = HMAC(sk, client IP); sk is held by server
- Client obtains TD-Cookie in its 1st interaction with server (over UDP or TCP)

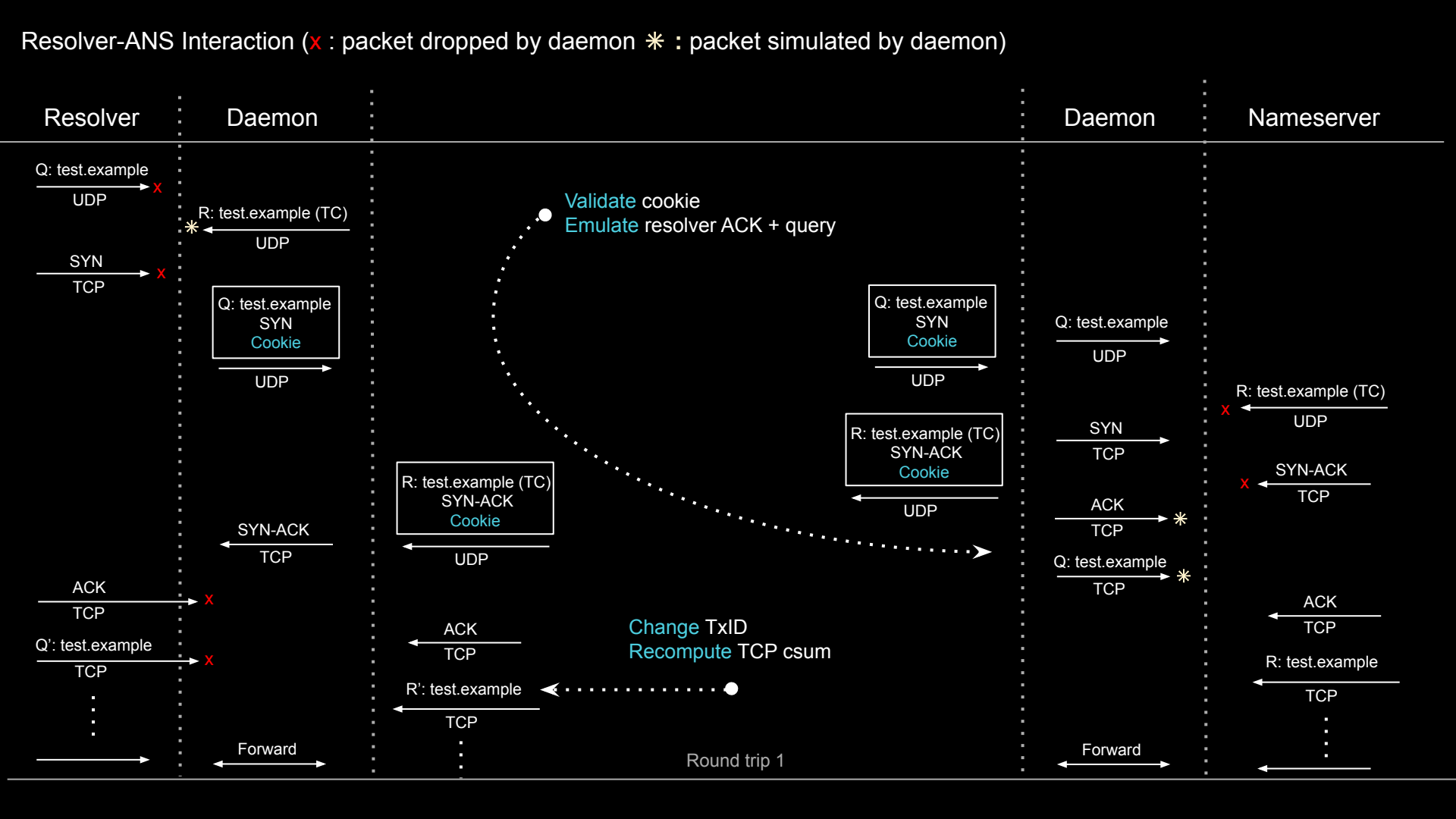
Implementation

Implementation

- Docker containers: client, resolver, root ANS, example ANS
- Host machine: Macbook Air M1 with 8 GB RAM
- DNS software: BIND9 (can be swapped with different one)
- Crypto libs: openssl, liboqs, oqs-provider
- Packet handling lib: libnetfilter_queue
- TurboDNS daemon (written in C) installed on resolver & example ANS
- No changes to DNS or TCP stack required
- 100 Mbps bandwidth, 10 ms latency, 0% loss
- Resolver has pre-fetched DNSKEY and NS records of all zones
- Resolver has TD-Cookie from example ANS
- Measure average resolution time of 10 Type A queries of client

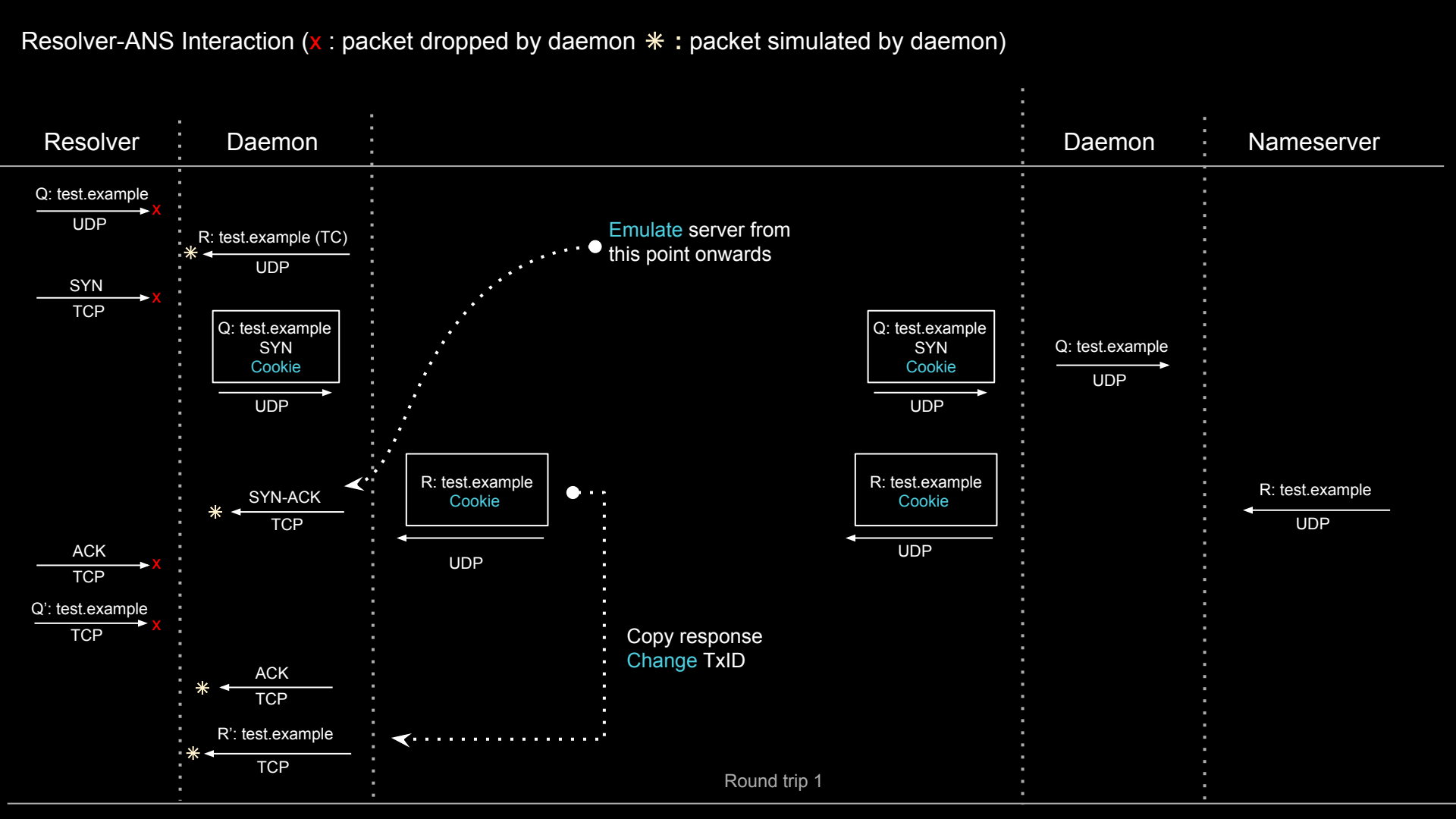
Scenario 1

Response is marked TC


Resolver-ANS Interaction (x : packet dropped by daemon ✳ : packet simulated by daemon)
Resolver
Daemon
Daemon
Nameserver
Q: test.example
UDP
x
R: test.example (TC)
UDP
SYN
TCP
x
Q: test.example
SYN
Cookie
UDP
Validate cookie
Emulate resolver ACK + query
Q: test.example
SYN
Cookie
UDP
Q: test.example
UDP
R: test.example (TC)
x
UDP
SYN
TCP
R: test.example (TC)
SYN-ACK
Cookie
UDP
SYN-ACK
x
TCP
ACK
TCP
R: test.example (TC)
SYN-ACK
Cookie
UDP
SYN-ACK
TCP
ACK
TCP
x
Q': test.example
TCP
x
Q: test.example
TCP
ACK
TCP
ACK
TCP
Change TxID
Recompute TCP csum
R': test.example
TCP
R: test.example
TCP
Forward
Round trip 1
Forward

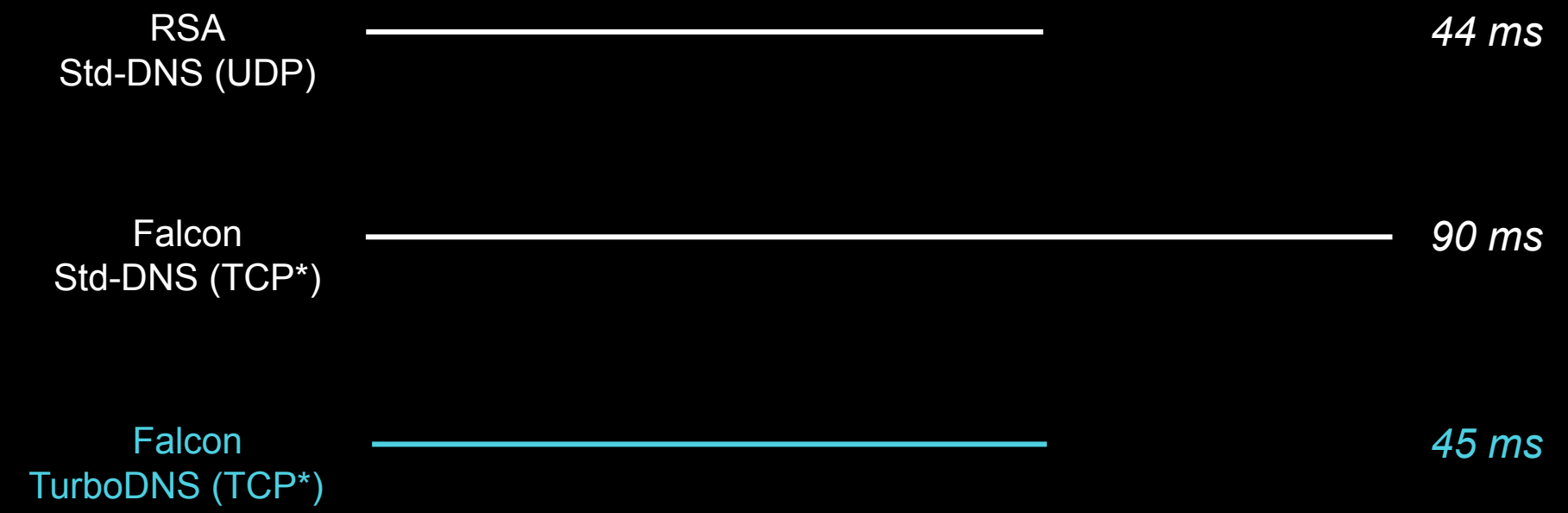
Scenario 2

Response is not marked TC


Resolver-ANS Interaction (x : packet dropped by daemon ✳ : packet simulated by daemon)
Resolver
Daemon
Daemon
Nameserver
Q: test.example
UDP
x
R: test.example (TC)
UDP
SYN
TCP
x
Q: test.example
SYN
Cookie
UDP
Emulate server from
this point onwards
Q: test.example
SYN
Cookie
UDP
Q: test.example
UDP
SYN-ACK
TCP
R: test.example
Cookie
UDP
R: test.example
Cookie
UDP
R: test.example
UDP
ACK
TCP
x
Q': test.example
TCP
x
Copy response
Change TxID
ACK
TCP
R': test.example
TCP
Round trip 1

Benchmarks

Resolution Time*

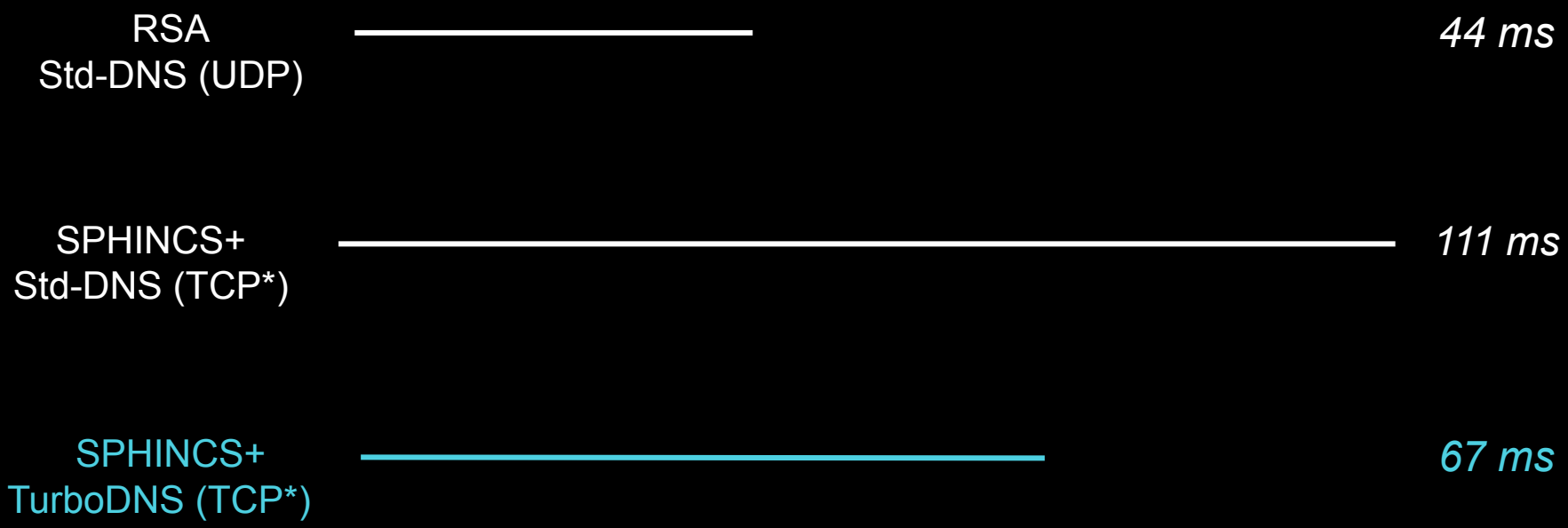


*Same trend with ECDSA & Dilithium
Std: Standard
TCP*: TCP Fallback

2x

Faster resolution vs Standard DNS (Falcon & Dilithium)

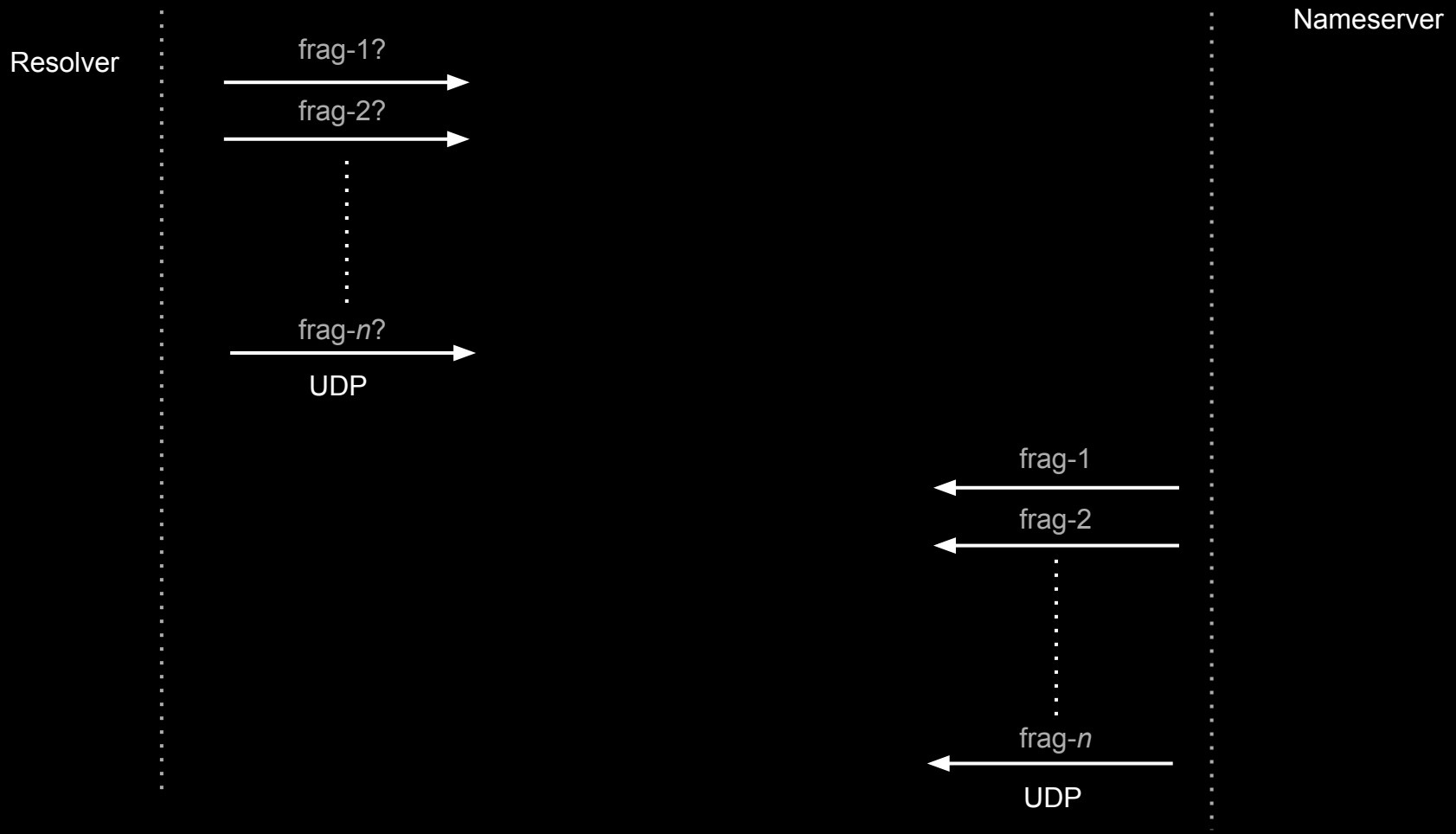
Resolution Time



SPHINCS+ requires an extra round trip because of exceeding TCP initcwnd of ~12.2 KB

TurboDNS vs Fragmentation Schemes

Fragmentation Schemes (DNS-over-UDP)

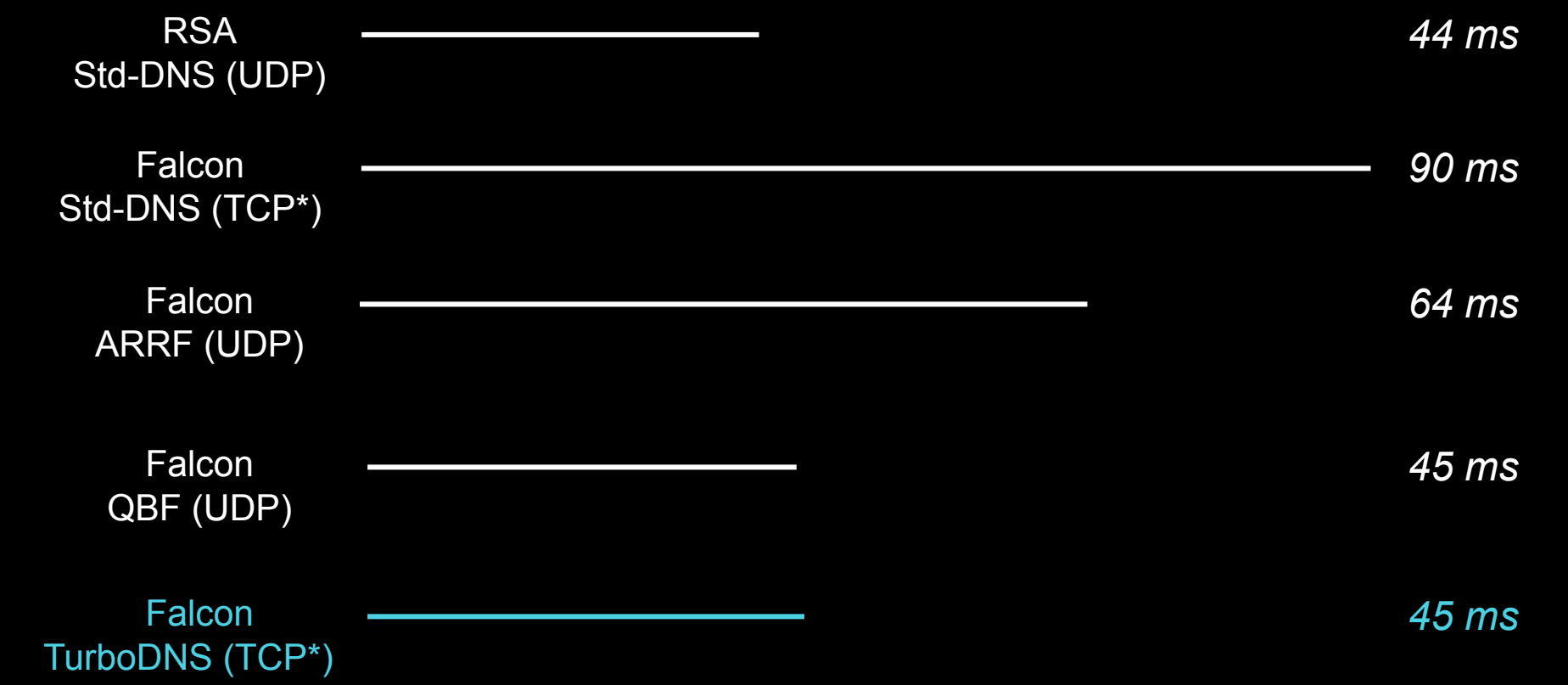


Fragmentation Schemes

1. ARRF. Goertzen and Stebila. PQCrypto 2023.
    a. Requires 2 round-trips
    b. Uses a custom DNS record: RRFRAG

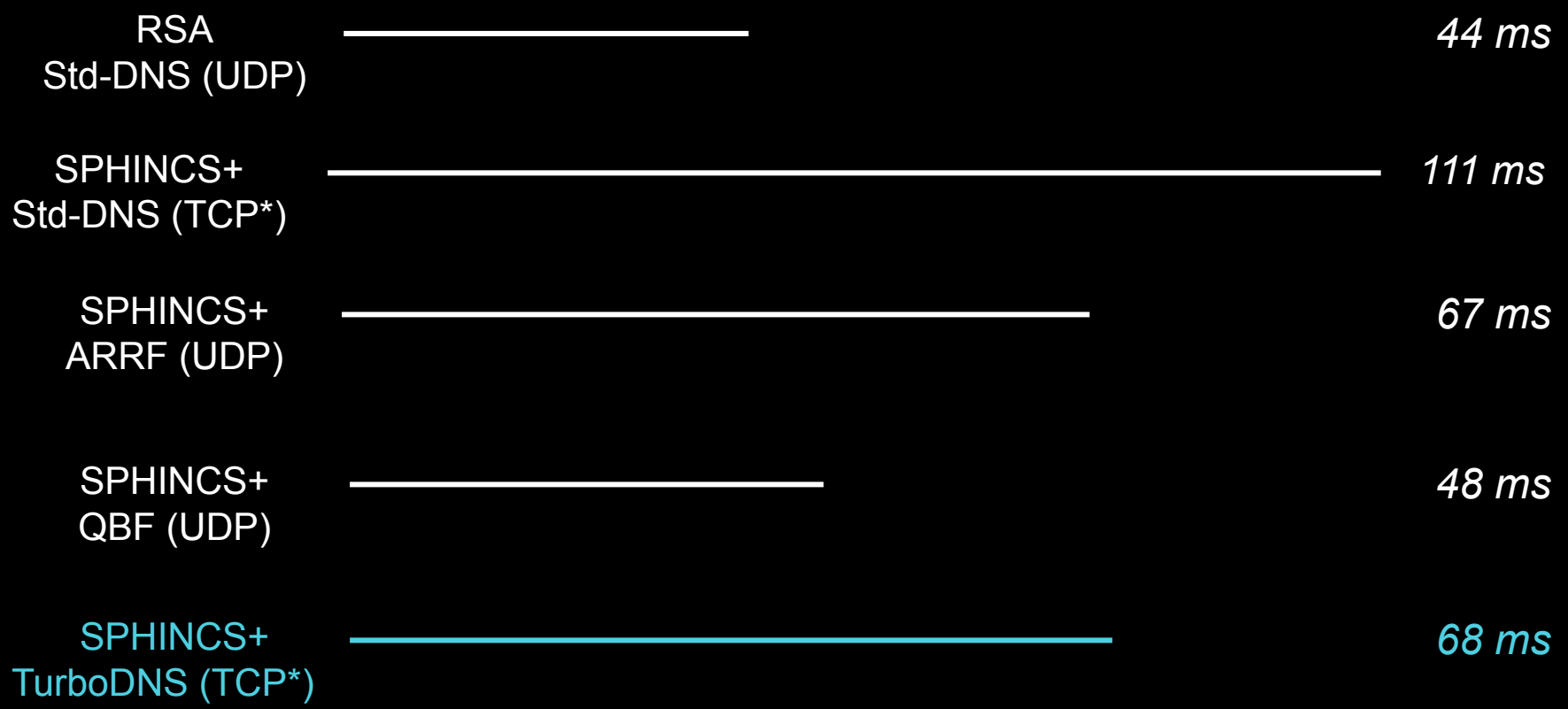
2. QBF. Rawat and Jhanwar. SPACE 2023.
    a. Requires 1 round-trip
    b. Uses standard DNS records

Resolution Time*



*Same trend with ECDSA & Dilithium
0% packet loss

Resolution Time



SPHINCS+ requires an extra round trip because of exceeding TCP initcwnd of ~12.2 KB

Why use TurboDNS over ARRF / QBF?

ARRF / QBF

Parallel DNS/UDP packets

- Could exhaust bandwidth of busy servers
- Could overwhelm middle-boxes/routers
- DNS/UDP messages may get lost
- Could be misused as a DDoS amplifier & reflector (unless paired with EDNS0 cookies)

vs

TurboDNS

DNS/TCP*

- Uses TCP flow control
- Uses TCP congestion control
- Uses TCP retransmit mechanism
- Uses TD-Cookie by design

Code available on GitHub.

Questions?