Field Experiments on Post-Quantum DNSSEC

Jason Goertzen <jason.goertzen@sandboxaq.com>
Peter Thomassen <peter@desec.io>
Nils Wisiol <nils@desec.io>

RIPE 89, Prague – Oct 29, 2024

Motivation

- Quantum computers anticipated to destroy current public key cryptography
  - Affects encrypting but **also signing**, including DNSSEC
- NIST has spent the past 7 years working with the global community to find "quantum safe" alternatives
  - Must be secure against both classical and quantum adversaries
- So far 3 signatures algorithms have been selected by NIST
  - Dilithium (ML-DSA)
  - Falcon (FN-DSA)
  - Sphincs+ (SLH-DSA)
- Aimed to be "drop in replacements" to classical cryptography

Complications for PQC DNSSEC

- DNS deployments often ossify, especially middleboxes → all kinds of problems
  - Message size limitations
  - Unusual packet contents might be dropped
- Additional considerations apply, per candidate algorithm:
  - Time needed for generating a key
  - Time needed for signing
  - Time needed for validation
  - Storage / memory footprint
- For more details, see https://datatracker.ietf.org/doc/draft-fregly-research-agenda-for-pqc-dnssec/

Algorithm Considerations

Algorithm	NIST Verdict	Approach	Private key	Public key	Signature	Sign/s	Verify/s
Crystals-Dilithium-II [29]	Finalist	Lattice	2.8kB	1.2kB	2.0kB		
Falcon-512 [31]	Finalist	Lattice	57kB	0.9kB	0.7kB	3,307	20,228
Rainbow-I_a [56]	Finalist	Multivariate	101kB	158kB	66B	8,332	11,065
RedGeMSS128 [16]	Candidate	Multivariate	16B	375kB	35B	545	10,365
Sphincs$^+$-Haraka-128s [11]	Candidate	Hash	64B	32B	8kB		
Picnic-L1-FS [17]	Candidate	Hash	16B	32B	34kB		
Picnic2-L1-FS [17]	Candidate	Hash	16B	32B	14kB		
EdDSA-Ed22519 [12]		Elliptic curve	64B	32B	64B	25,935	7,954
ECDSA-P256 [12]		Elliptic curve	96B	64B	64B	40,509	13,078
RSA-2048 [12]		Prime	2kB	0.3kB	0.3kB	1,485	49,367

Müller, M. et al.: Retrofitting post-quantum cryptography in internet protocols: a case study of DNSSEC. SIGCOMM Comput. Commun. Rev. 50, 49–57 (2020)

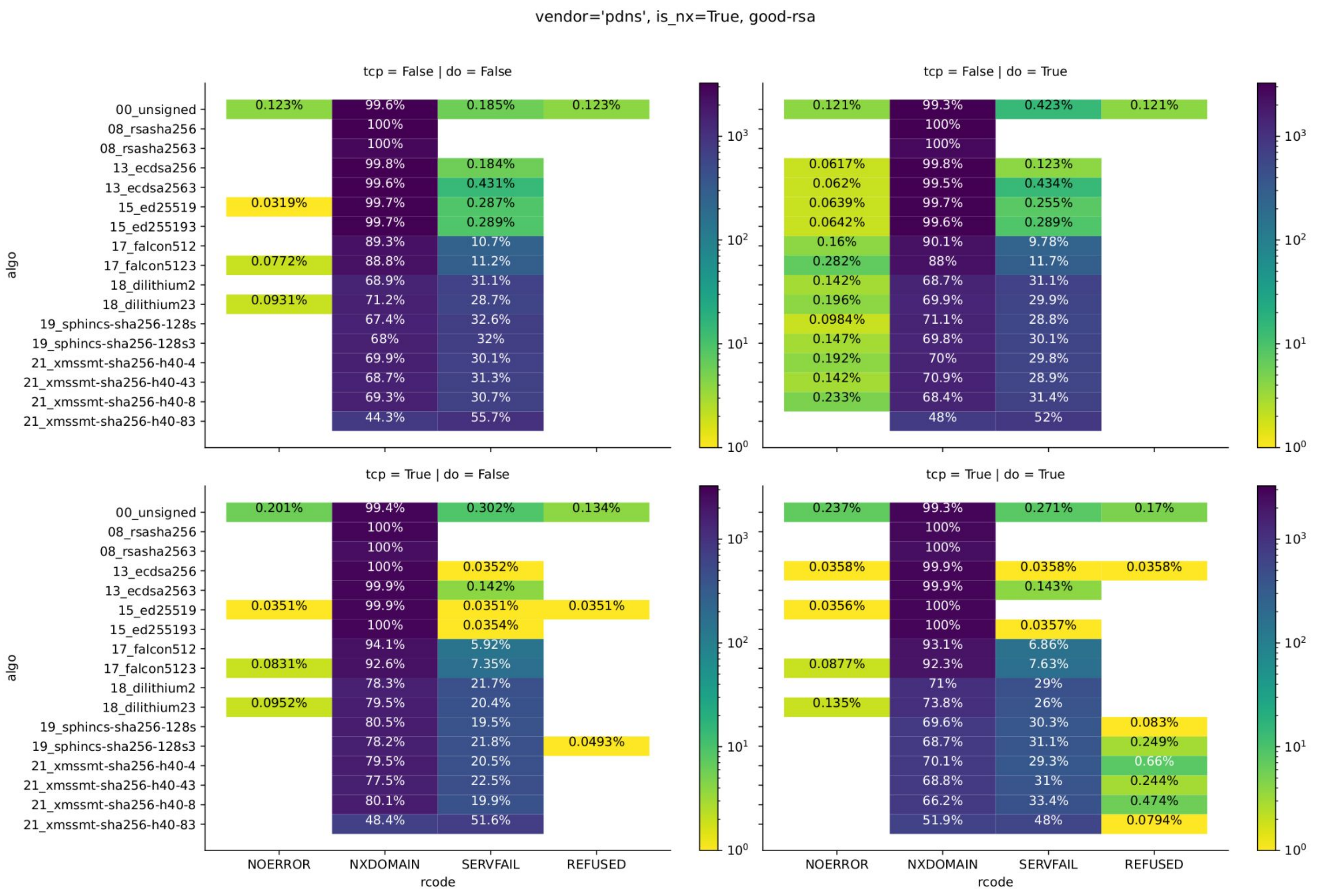
- Selected algorithms with public keys and signatures < 10 KB
- Plus: a stateful hash-based algorithm (XMSS)
- Plus: Merkle-tree based optimization (**new!**)

Our starting point

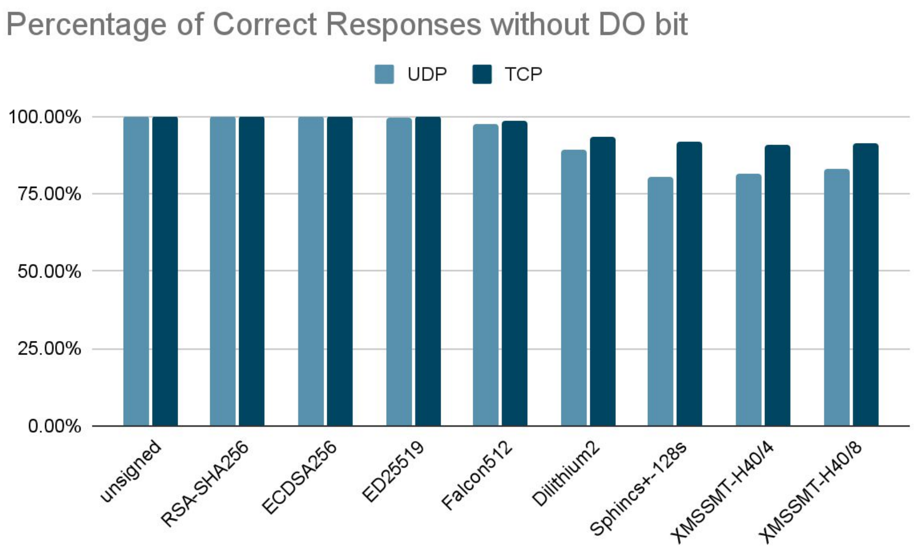
- In 2022, performed (local-only) DNSSEC study with **Falcon** in PowerDNS
  - Results: https://blog.powerdns.com/2022/04/07/falcon-512-in-powerdns
- Now: Broader experiments with **multiple PQC algorithms**
  - fast validation, short signatures, short-ish keys
- Goal: **Public deployment** on the Internet, to investigate …
  - **behavior of non-PQC-aware resolvers** typically used by clients
  - **behavior of PQC-aware resolvers**
- Parameters:
  - KSK/ZSK (BIND) vs. CSK (PowerDNS)
  - Name existence and NSEC vs. [NSEC3 conventional (BIND) vs. minimal (PowerDNS)]
  - UDP vs. TCP
  - DO bit

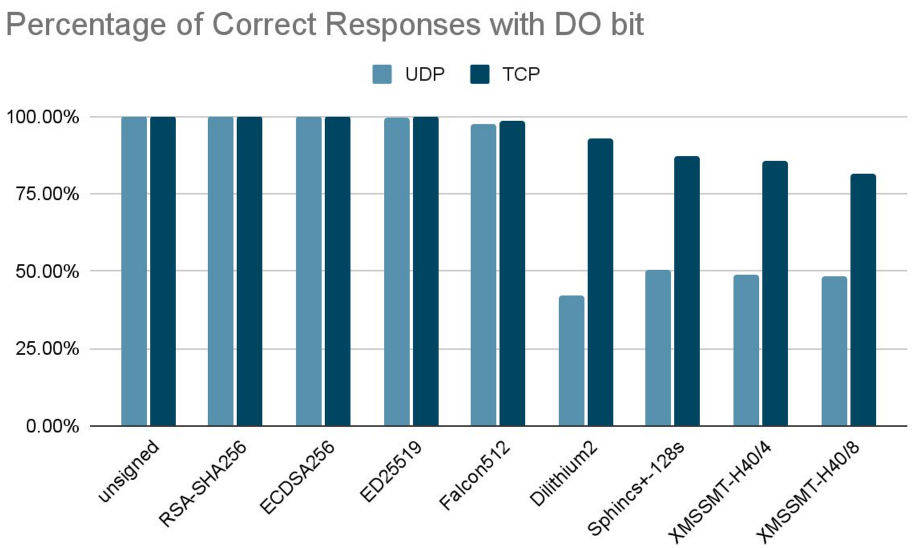
Steps Taken

- Implemented via **liboqs** (with regular unassigned algorithm numbers)
  - Falcon512
  - Dilithium2
  - SPHINCS+-SHA256-128s
  - XMSSmt-SHA256-h40-4 / XMSSmt-SHA256-h40-8 (and other parameter sets)
- Measurements using **RIPE ATLAS** (~10,000 probes, ~2M queries in May 2024)
- Deployed BIND9 and PowerDNS based zones
- Output variables: **Rcode, Correctness, AD bit, response time**
- Pre-selection: Exclude …
  - probe-resolver combinations with incorrect response for RSA-SHA256 (due to noise)
  - resolvers in private IP ranges (due to RIPE ATLAS limitation for TCP)
  - timeouts and network errors

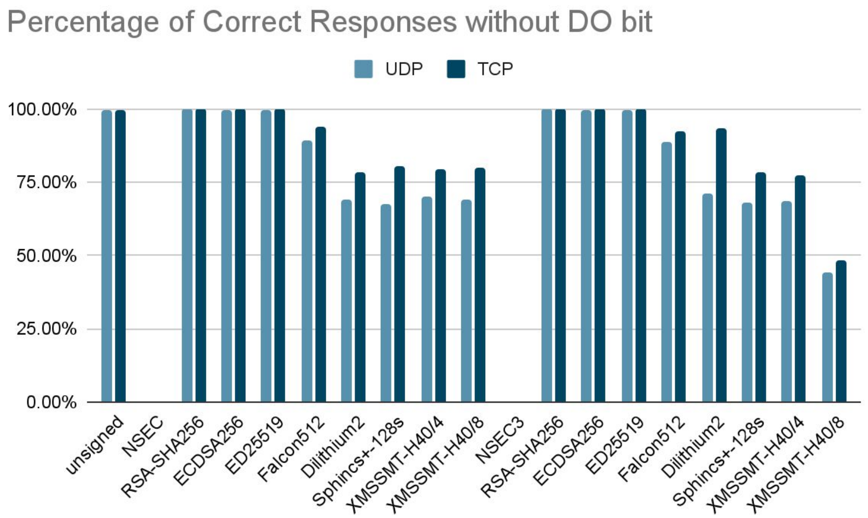

vendor='pdns', is_nx=True, good-rsa
tcp = False | do = False
tcp = False | do = True
tcp = True | do = False
tcp = True | do = True
algo
rcode
NOERROR
NXDOMAIN
SERVFAIL
REFUSED
00_unsigned
08_rsasha256
08_rsasha2563
13_ecdsa256
13_ecdsa2563
15_ed25519
15_ed255193
17_falcon512
17_falcon5123
18_dilithium2
18_dilithium23
19_sphincs-sha256-128s
19_sphincs-sha256-128s3
21_xmssmt-sha256-h40-4
21_xmssmt-sha256-h40-43
21_xmssmt-sha256-h40-8
21_xmssmt-sha256-h40-83
10^0
10^1
10^2
10^3
0.123% 99.6% 0.185% 0.123%
100%
100%
99.8% 0.184%
99.6% 0.431%
0.0319% 99.7% 0.287%
99.7% 0.289%
89.3% 10.7%
0.0772% 88.8% 11.2%
68.9% 31.1%
0.0931% 71.2% 28.7%
67.4% 32.6%
68% 32%
69.9% 30.1%
68.7% 31.3%
69.3% 30.7%
44.3% 55.7%
0.121% 99.3% 0.423% 0.121%
100%
100%
0.0617% 99.8% 0.123%
0.062% 99.5% 0.434%
0.0639% 99.7% 0.255%
0.0642% 99.6% 0.289%
0.16% 90.1% 9.78%
0.282% 88% 11.7%
0.142% 68.7% 31.1%
0.196% 69.9% 29.9%
0.0984% 71.1% 28.8%
0.147% 69.8% 30.1%
0.192% 70% 29.8%
0.142% 70.9% 28.9%
0.233% 68.4% 31.4%
48% 52%
0.201% 99.4% 0.302% 0.134%
100%
100%
100% 0.0352%
99.9% 0.142%
0.0351% 99.9% 0.0351% 0.0351%
100% 0.0354%
94.1% 5.92%
0.0831% 92.6% 7.35%
78.3% 21.7%
0.0952% 79.5% 20.4%
80.5% 19.5%
78.2% 21.8% 0.0493%
79.5% 20.5%
77.5% 22.5%
80.1% 19.9%
48.4% 51.6%
0.237% 99.3% 0.271% 0.17%
100%
100%
0.0358% 99.9% 0.0358% 0.0358%
99.9% 0.143%
0.0356% 100%
100% 0.0357%
93.1% 6.86%
0.0877% 92.3% 7.63%
71% 29%
0.135% 73.8% 26%
69.6% 30.3% 0.083%
68.7% 31.1% 0.249%
70.1% 29.3% 0.66%
68.8% 31% 0.244%
66.2% 33.4% 0.474%
51.9% 48% 0.0794%

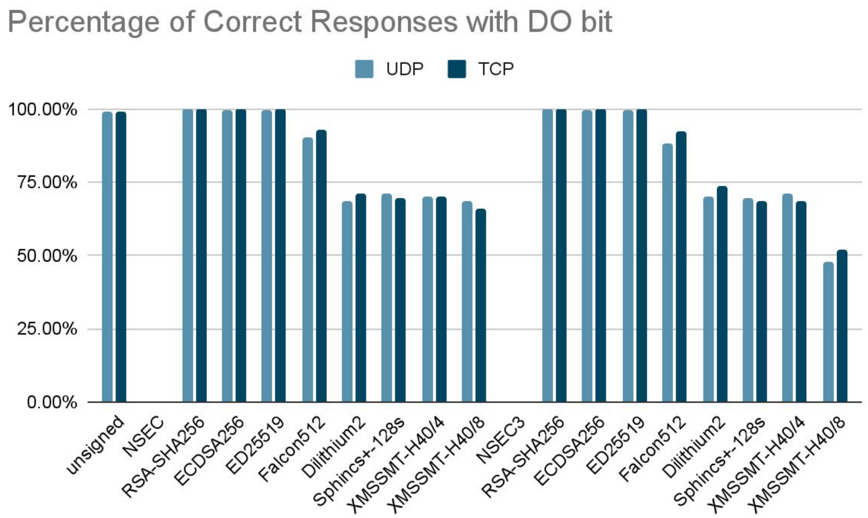
Correct responses for a valid label





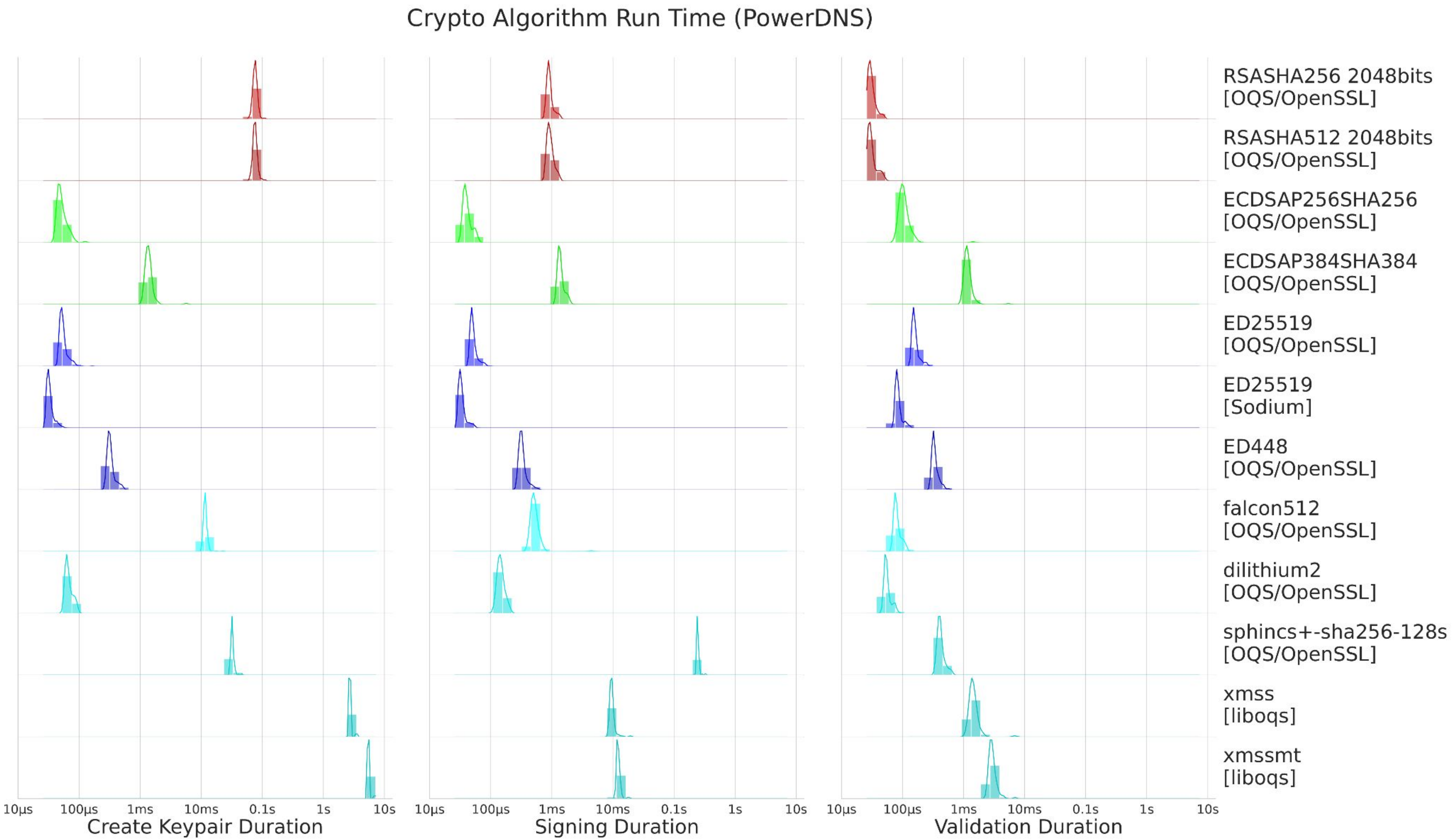
Correct responses for a nonexistent label





What we observed

- **Transmission issues are real**
  - PQC response delivery rates go down significantly as response sizes increase → Falcon leads
  - Gets worse depending on circumstances, like with DO bit or with NSEC3
- **UDP & DO=0:**
  ~70% KSK/ZSK responses correct
  ~80% CSK responses correct
  - Goes up by ~10% via TCP
- **UDP & DO=1:**
  ~50% responses correct
  - Goes up by ~20–40% via TCP
- 8.5% of probe-resolver pairs claim successfully validating Falcon



Crypto Algorithm Run Time (PowerDNS)
RSASHA256 2048bits [OQS/OpenSSL]
RSASHA512 2048bits [OQS/OpenSSL]
ECDSAP256SHA256 [OQS/OpenSSL]
ECDSAP384SHA384 [OQS/OpenSSL]
ED25519 [OQS/OpenSSL]
ED25519 [Sodium]
ED448 [OQS/OpenSSL]
falcon512 [OQS/OpenSSL]
dilithium2 [OQS/OpenSSL]
sphincs+-sha256-128s [OQS/OpenSSL]
xmss [liboqs]
xmssmt [liboqs]
10µs 100µs 1ms 10ms 0.1s 1s 10s
Create Keypair Duration
Signing Duration
Validation Duration

Queries Using a PQC-aware Resolver

```
dig +dnssec A dilithium2.pdns.pq-dnssec.dedyn.io @bind9.pq-dnssec.dedyn.io -p 5304
```

```
;; Truncated, retrying in TCP mode.

; <<>> DiG 9.18.24-0ubuntu0.22.04.1-Ubuntu <<>> +dnssec A dilithium2.pdns.pq-dnssec.dedyn.io @bind9.pq-dnssec.dedyn.io -p 5304
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22245
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 1232
; COOKIE: 8455829f86d7fb7601000000669b5d9517dfc67dff539cac (good)
;; QUESTION SECTION:
;dilithium2.pdns.pq-dnssec.dedyn.io. IN A

;; ANSWER SECTION:
dilithium2.pdns.pq-dnssec.dedyn.io. 3599 IN A  95.217.209.184
dilithium2.pdns.pq-dnssec.dedyn.io. 3599 IN RRSIG A 18 5 3600 20240801000000 20240711000000 3978 dilithium2.pdns.pq-dnssec.ded
yn.io. 19/28JXGCgGbNtEAtUOzv1/SzP+kr6vBlglWrJ/ZfYgdC1DXZHdh+xol rnZ9uhvmADCqZzJXOyOU1Tyw2sHN32Vmcv4KLR8lI7TBwfTJq6T3nGfV oQnv9
DNvPJTyb4VonYH3fLTMYeQ3/0Wy9gbv0ngy55QqRjw+ikhS0yIp ezpZYH3ArY/xxmTgM7OBW0yBg3gXgo1G2mrX97ufqrwkO/n0Vu/xXfSI npGKq+dVu7LQQR7nM
lmM3FkbaRAFyo0FjmbzXDPtyrwJekJP8dfQ5zvc pOCRfrpjRg+ZBUofhdk1PURO539JwD[...shortened...]AA AAAAAAAAAAAAAAAAAAAAAAAAAAAABAcJzc=

;; Query time: 56 msec
;; SERVER: 35.232.14.170#5304(bind9.pq-dnssec.dedyn.io) (TCP)
;; WHEN: Fri Jul 19 23:47:49 PDT 2024
;; MSG SIZE  rcvd: 2593
```

CONTACT

Post-Quantum DNSSEC Testbed with BIND and PowerDNS

Query our PQC-enabled DNS Resolvers

Send queries to our post-quantum enabled validating resolver! You can choose from a number of post-quantum (and classical) signing schemes, NSEC or NSEC3 mode, and implementations for PowerDNS (source) and BIND (source).

Zones signed accordingly are available at `{algorithm}.{vendor}.pq-dnssec.dedyn.io`, and each has a `A` and a `TXT` record configured. To query a non-existing name, prepend the `nx` label (for example).

Queries will be sent from your browser using DNS-over-HTTPS to a BIND or PowerDNS resolvers with validation support for the selected algorithm. The resolver will talk to the corresponding BIND or PowerDNS authoritative DNS server (again, with support for the selecting signing scheme), to get your response. It will then validate the signature and send the result to your browser.

All queries are send with the `DNSSEC_OK` flag (`+dnssec` in dig), so you will see `RRSIG` and `NSEC`/`NSEC3` records the the responses.

Query type	Algorithm	Authoritative vendor	Resolver vendor
	Dilithium2	PowerDNS	PowerDNS

☐ NSEC3 ☑ non-existent name

Domain name: nx.dilithium2.pdns.pq-dnssec.dedyn.io

QUERY DNSViz

Credits: Jason Goertzen (Sandbox AQ), Nils Wisiol. Funded by NLnet Foundation.

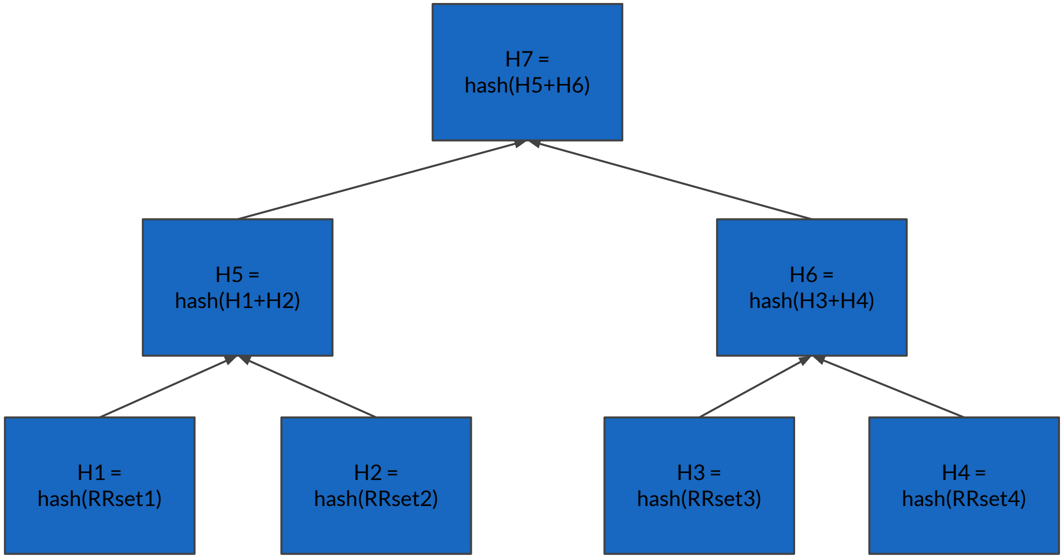
Try it yourself!

https://pq-dnssec.dedyn.io/
(also has detailed results)

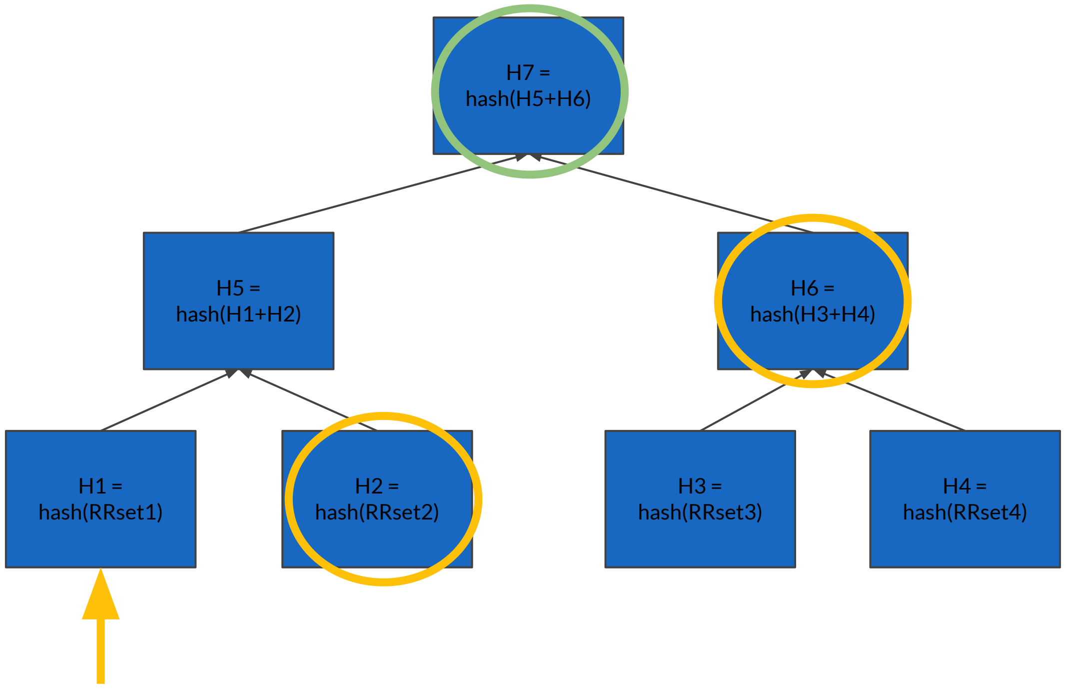
The Future? Merkle Trees

- Want to use PQC while keeping messages small
- Use Merkle trees to compress zone signing overhead
  - Signatures become authenticating paths
  - ZSK DNSKEY becomes the root hash
  - KSK is some secure algorithm with unpleasantly long signatures
    → replace with small Merkle authenticating path

What is a Merkle Tree?



What is a Merkle Tree?



Can we apply this to DNS?

Sure!

- Use a Standardized DNSSEC Algorithm for our KSK
  - Provides Authenticity and Integrity
- Define a new “Merkle Tree” algorithm and store its root hash in the ZSK rdata
  - Provides Integrity via proof of inclusion + gets Authenticity from being signed by KSK
- Signatures become the authenticating path of the Merkle tree
  - Grow logarithmically with the number of RRSets in a zone
- We can combine the work from Batched Signatures Revisited [1] to reduce hash size without reducing security (Second Preimage Resistance)

[1] https://pub.sandboxaq.com/publications/batch-signatures-revisited

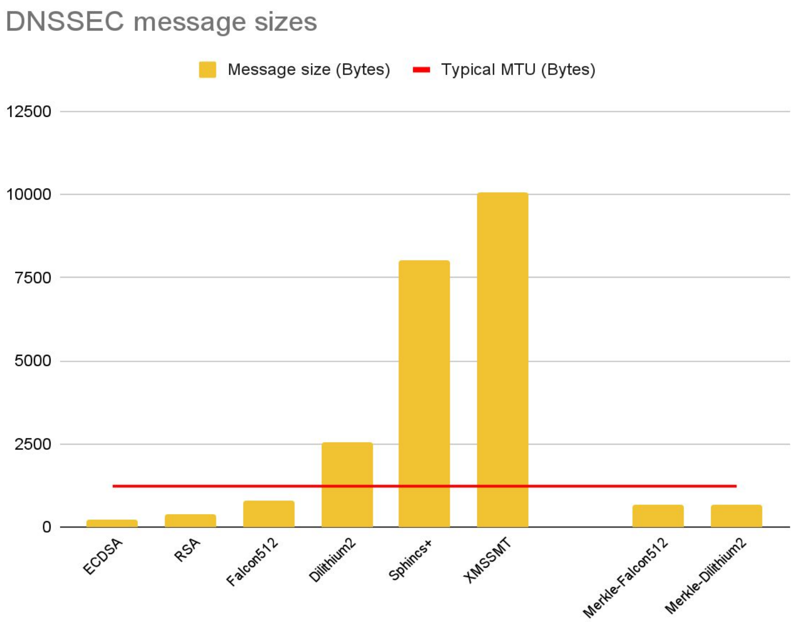
We need to change some things about DNS first…

- Circular signing is an issue
  - Everytime you sign something, the Merkle tree changes, and its root node (ZSK) changes
  - Everytime the root node changes, the keytag changes → signature's input changes
  - The DNSKEY set cannot be a part of the Merkle Tree

- We need to change how RRSets are signed when using a Merkle Tree
  - We cannot include the key tag as part of the data being signed
  - We cannot require RRSIGs for all DNSKEY algorithms
    - Instead, DNSKEY is signed with KSK algorithm, and all else signed with ZSK algorithm

We get two nice wins

DNS messages without DNSKEY set stay below line of peril!



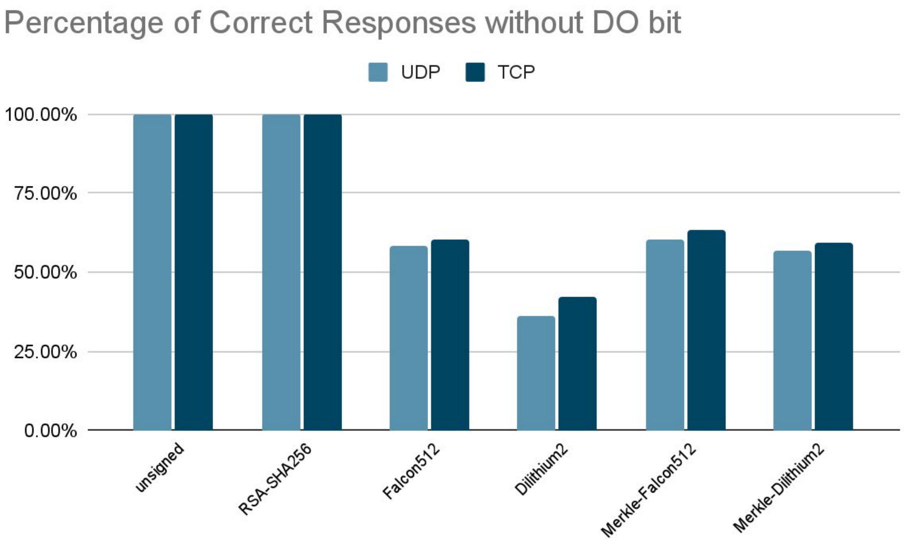
Tiny zone transfers

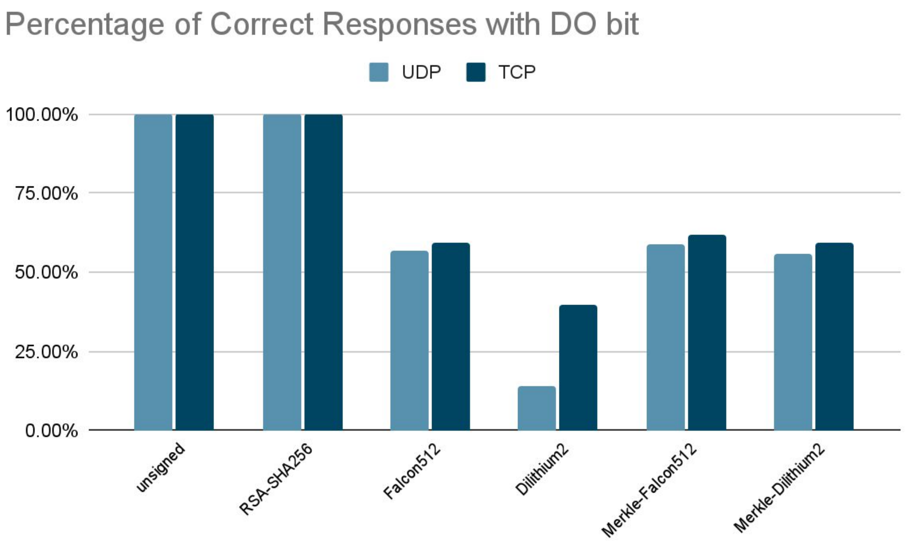
- Since a private key isn't involved, we can have all secondary servers rebuild the tree and authenticating paths
- Interesting trade-off: We can transmit empty signatures during zone transfers greatly reducing the size of the zone
    - Only one signature in zone transfer (for DNSKEY RRset)

We don't have an implementation for this

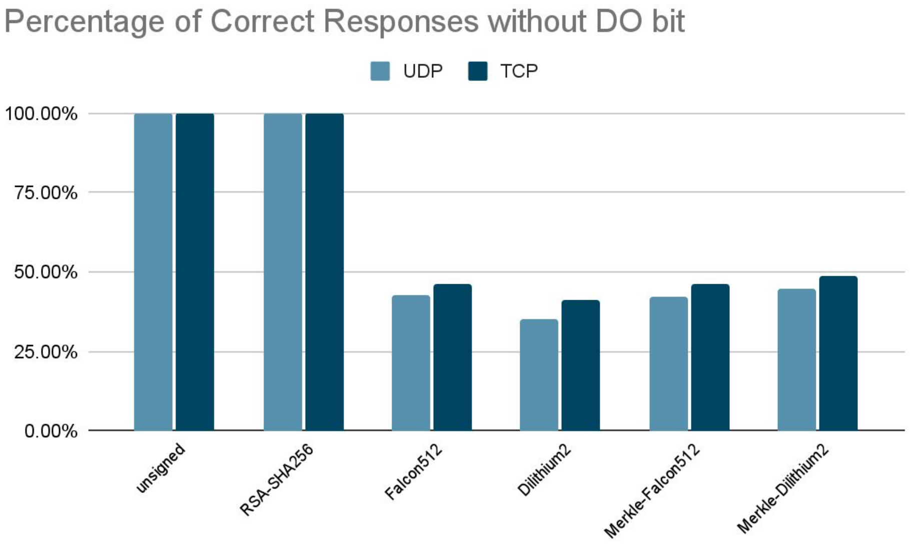
Was there a difference in the ATLAS tests?

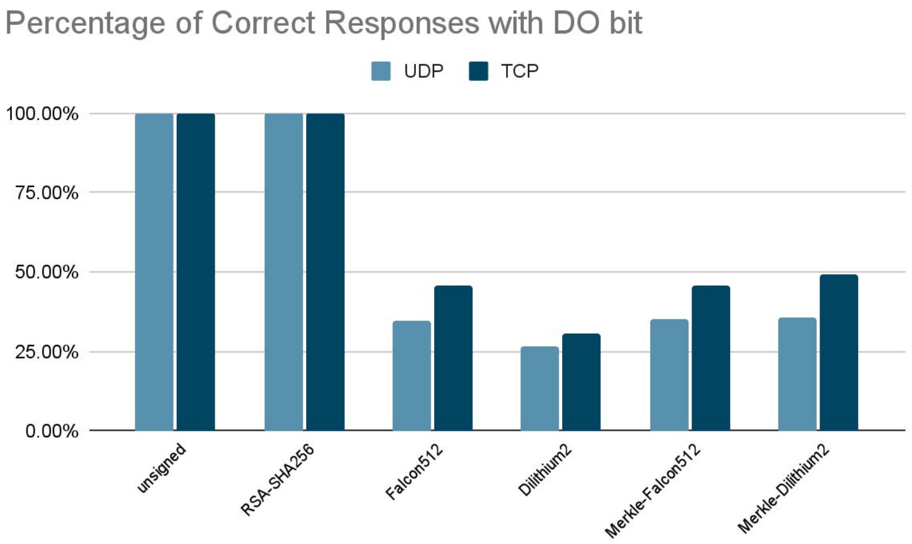
Correct responses for a valid label





Correct responses for a nonexistent label





dnssec-signzone

- Currently only supports offline signing
- Heavy modifications to BIND's dnssec-signzone
    - Iterate through all RRSets and add them to the Merkle tree
    - Finalize the Merkle tree and update keytag
    - Iterate over all RRSIGs and insert the correct authenticating path and keytag
    - Takes about half the time of signing the same zone with ECDSA
        - Additional optimization opportunities might be possible

Some takeaways for Merkle trees

- DNSSEC protocol changes would need to be made
- By defining it with its own algorithm id you can use Merkle trees with any other DNSSEC algorithm
- Zone updates are limited by the root node's (ZSK) TTL
    - Verisign's MTL might help with this?
- DNSKEY messages are not compressed
- Improve deliverability for large signature zones
- Unlike stateful hash based signatures draft no central state is required to be maintained

Biggest takeaway:
Transitioning to PQC will be **nontrivial**.
We **need** to start planning now.

Outlook

- Fixing may require **revamping signature representation** in DNS
  - ARRF?
  - Does not necessarily involve a wire format / spec change
  - Or will more robust DoT/DoH/DoQ gain enough traction?
- What would it take to **make the root quantum-safe**?
  - Further complications from double-signing – is this really needed?
- To transition, **any scalable solution will require DS provisioning automation**
- Future work needed!
  → Mailing list: pq-dnssec@ietf.org
  → Side Meeting at IETF 121 (Dublin) on Nov 7, 2024

Thank you!

Acknowledgments:

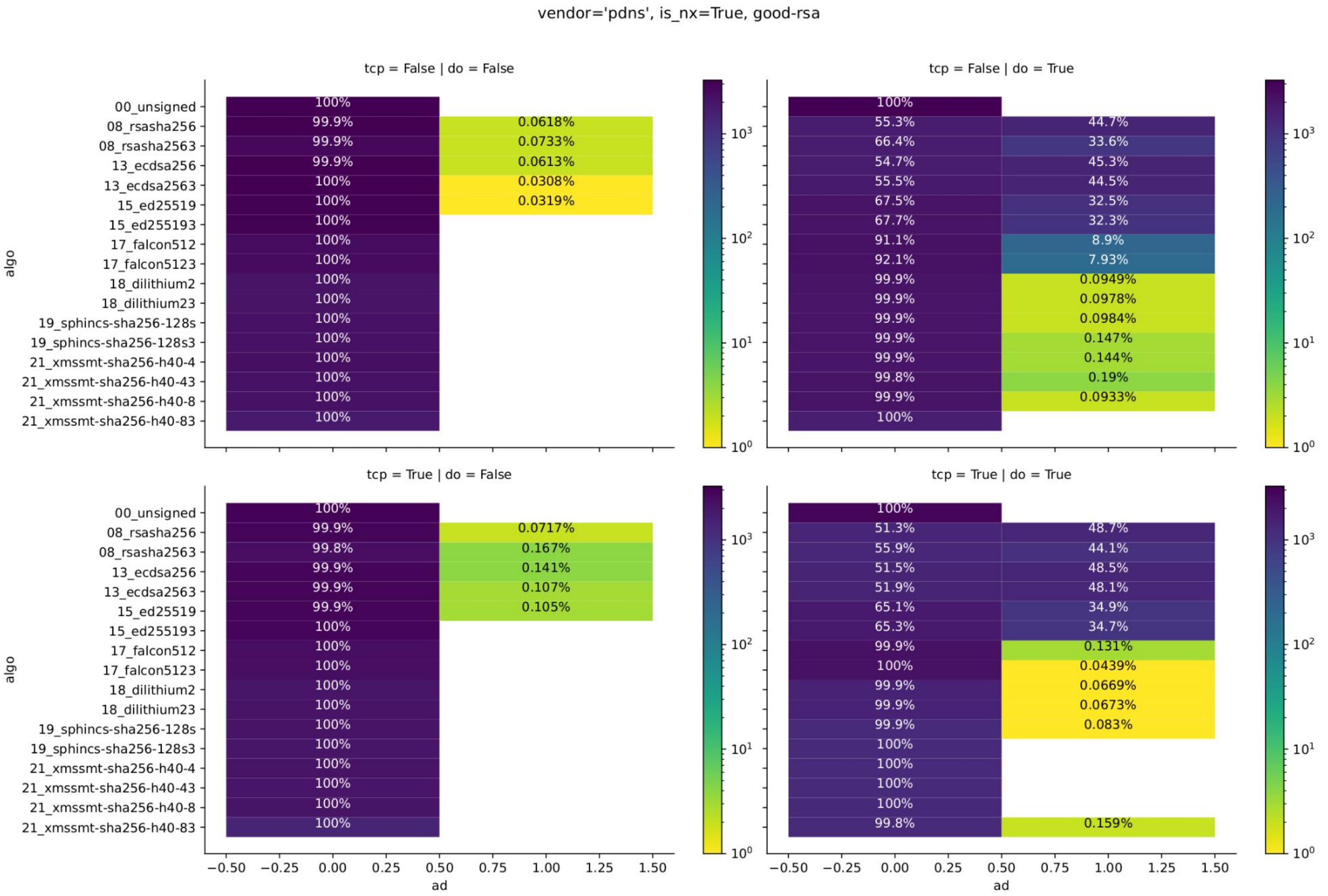
nlnet FOUNDATION

SANDBOXAQ

SSE

deSEC

Questions?


vendor='pdns', is_nx=True, good-rsa
tcp = False | do = False
tcp = False | do = True
tcp = True | do = False
tcp = True | do = True
algo
ad
00_unsigned
08_rsasha256
08_rsasha2563
13_ecdsa256
13_ecdsa2563
15_ed25519
15_ed255193
17_falcon512
17_falcon5123
18_dilithium2
18_dilithium23
19_sphincs-sha256-128s
19_sphincs-sha256-128s3
21_xmssmt-sha256-h40-4
21_xmssmt-sha256-h40-43
21_xmssmt-sha256-h40-8
21_xmssmt-sha256-h40-83
100% 99.9% 99.9% 99.9% 100% 100% 100% 100% 100% 100% 100% 100% 100% 100% 100% 100% 100%
0.0618% 0.0733% 0.0613% 0.0308% 0.0319%
100% 55.3% 66.4% 54.7% 55.5% 67.5% 67.7% 91.1% 92.1% 99.9% 99.9% 99.9% 99.9% 99.9% 99.8% 99.9% 100%
44.7% 33.6% 45.3% 44.5% 32.5% 32.3% 8.9% 7.93% 0.0949% 0.0978% 0.0984% 0.147% 0.144% 0.19% 0.0933%
100% 99.9% 99.8% 99.9% 99.9% 99.9% 100% 100% 100% 100% 100% 100% 100% 100% 100% 100% 100%
0.0717% 0.167% 0.141% 0.107% 0.105%
100% 51.3% 55.9% 51.5% 51.9% 65.1% 65.3% 99.9% 100% 99.9% 99.9% 99.9% 100% 100% 100% 100% 99.8%
48.7% 44.1% 48.5% 48.1% 34.9% 34.7% 0.131% 0.0439% 0.0669% 0.0673% 0.083% 0.159%
−0.50 −0.25 0.00 0.25 0.50 0.75 1.00 1.25 1.50
10^0 10^1 10^2 10^3