Algorithm Negotiation in DNSSEC?

Shumon Huque, Haya Shulman, Shane Kerr

July 20th 2017

IETF99, Prague, Czech Republic

draft-huque-dnssec-alg-nego-01

Assumptions/Use case

- A zone operator wants to incrementally deploy a new algorithm

    - And new algorithms are appearing or will be in the pipeline (Ed25518, Ed448, NSEC5, PQC, etc.)

- It will sign using both old and new algorithms for a transition period

- Then withdraw the old algorithm once it’s confident that all or most of its client population of resolvers understand the new algorithm

Assumptions/Use case

- This is possible today with DNSSEC - double sign the zone, and return signatures with the multiple algorithms in responses

- But zone operators often don’t want to unnecessarily bloat the size of responses with multiple signatures because of potential operational issues:

Assumptions/Use case

- .. operational issues (cont):

  - Response might exceed the path MTU (or the IPv6 minimum MTU) and be fragmented - fragments often don’t work reliably on the Internet - blocked by middleboxes and network security devices

  - To avoid this, zone operator could truncate messages that exceeded path MTU, forcing retry over TCP - introduces additional latency and processing costs

  - Wants to avoid large scale use of TCP (or perhaps QUIC in the future) before it has scaled up infrastructure to handle that

Assumptions/Use case

- Why don’t you just deploy the new algorithm only? After all DNSSEC fails open (several people to me this week):

  - Obviously, I lose the benefit of DNSSEC protection for a lot of the resolver population

  - Furthermore, if I have DANE applications critically dependent on DNSSEC authentication, then this is an unacceptable security risk:

  - DANE applications can’t really afford to fail open, unless they are entirely opportunistic or optional, so they will be even more dissuaded from new DNSSEC algorithm adoption

Algorithm Negotiation Proposal

- A new EDNS0 option that allows a DNS client to specify a list of DNSSEC algorithms, in preference order, that the client desires to use.
- DNS server, upon receipt of this option:
  - chooses the strongest algorithm that it supports in common with the client.
  - selectively delivers DNSSEC signatures using that algorithm only

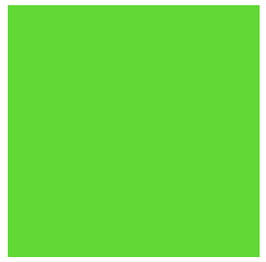
Algorithm Negotiation Proposal

- For more technical details, read the draft:
    - https://tools.ietf.org/html/draft-huque-dnssec-alg-nego

Proposal 1

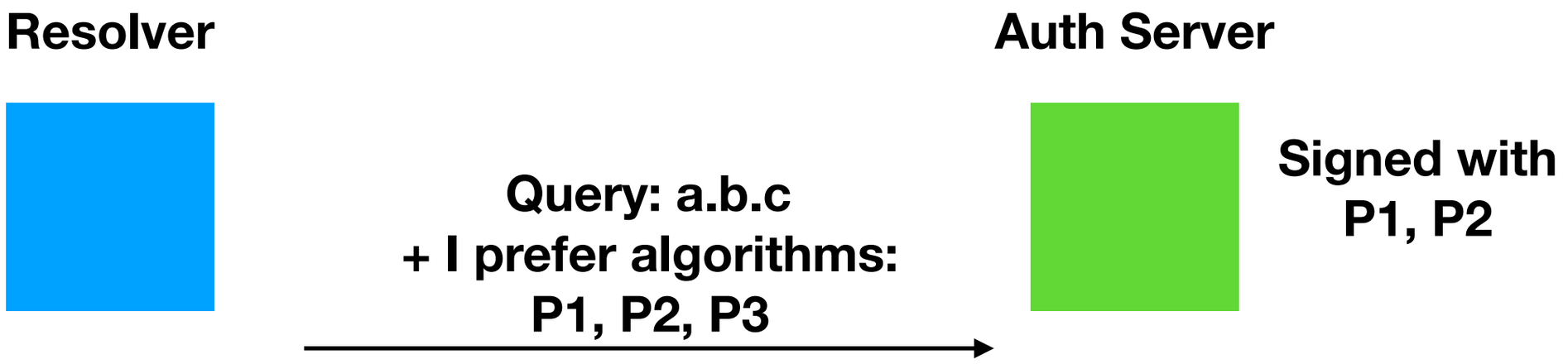
Resolver

Auth Server

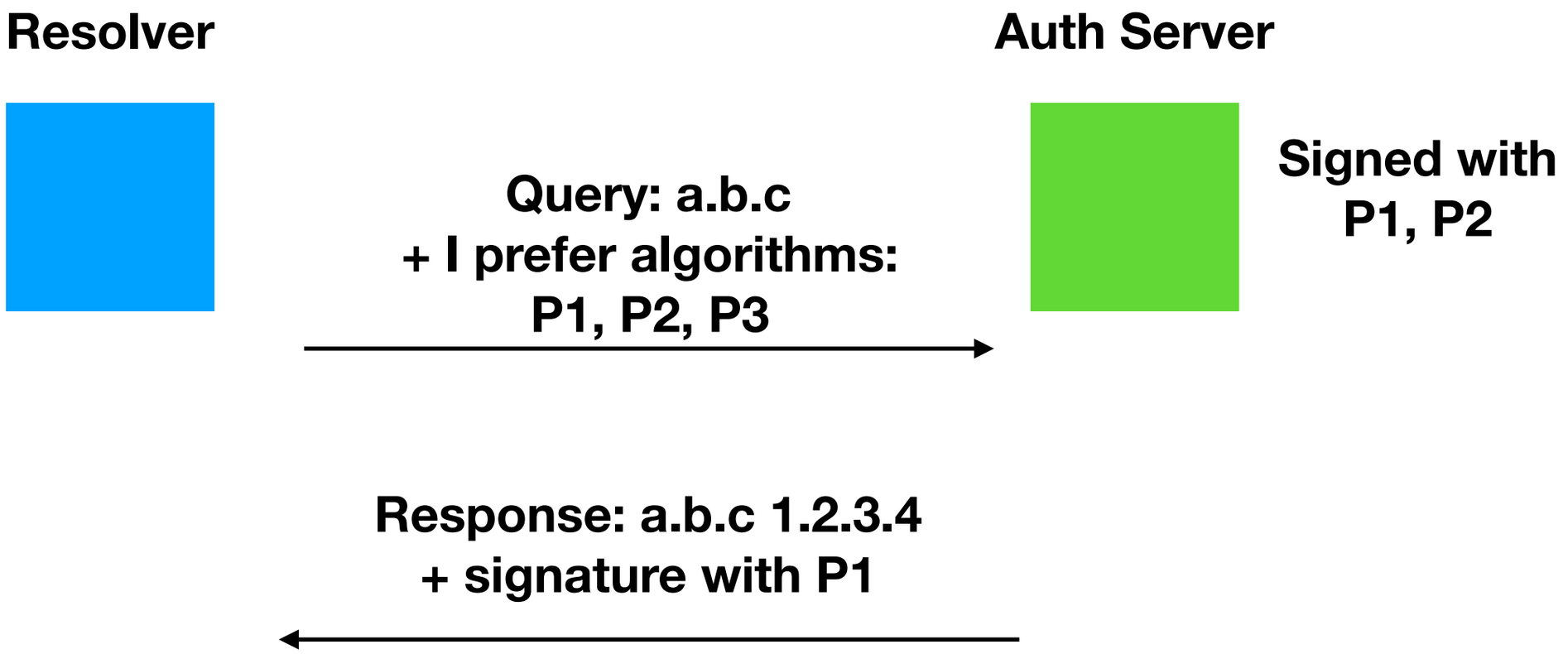


Signed with P1, P2

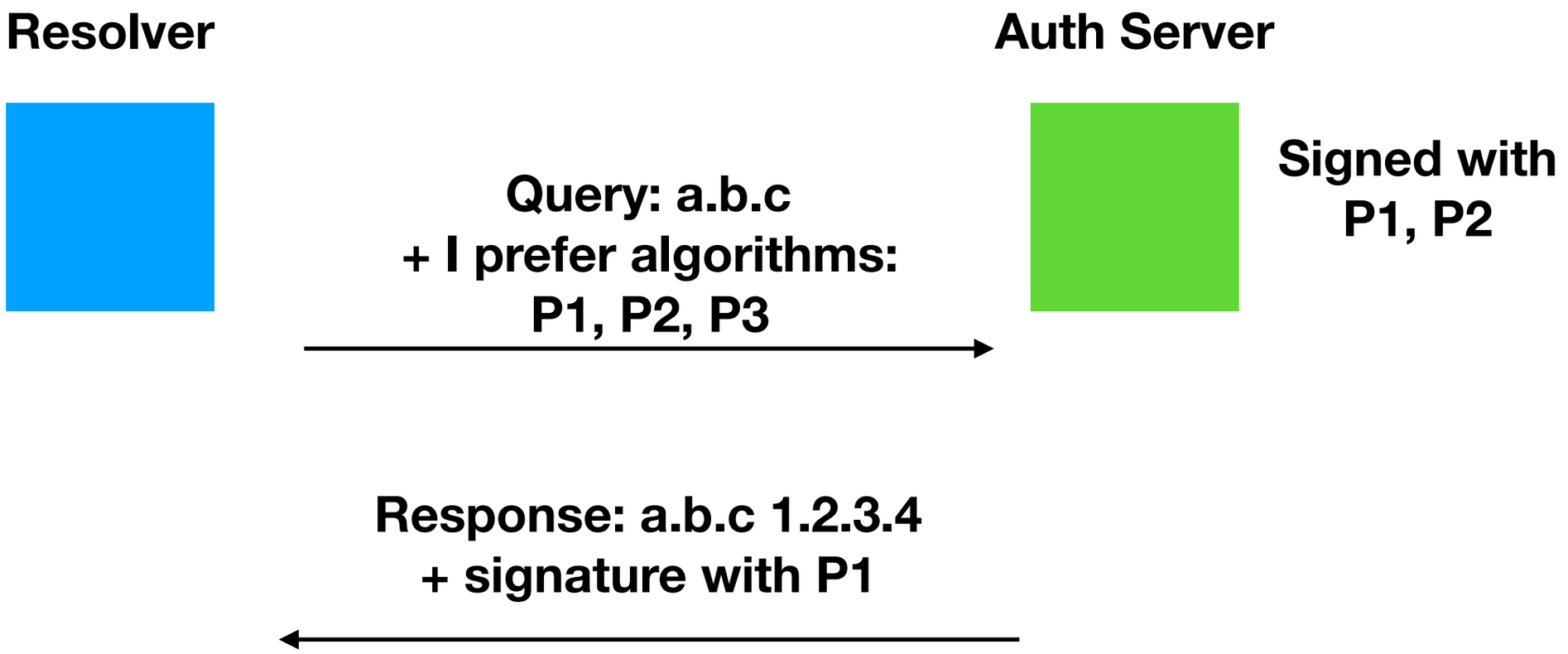
Proposal 1


Resolver
Auth Server
Query: a.b.c
+ I prefer algorithms:
P1, P2, P3
Signed with
P1, P2

Proposal 1


Resolver
Auth Server
Signed with
P1, P2
Query: a.b.c
+ I prefer algorithms:
P1, P2, P3
Response: a.b.c 1.2.3.4
+ signature with P1

Proposal 1



Downgrade protection: EDNS option is not protected. MITM attacker could strip, P1 from the query, and downgrade the authentication to happen using algorithm P2. To protect against, this Resolver compares the signature algorithm in the response to the list of known algorithms in the (authenticated) DNSKEY RRset that it already has from the zone.

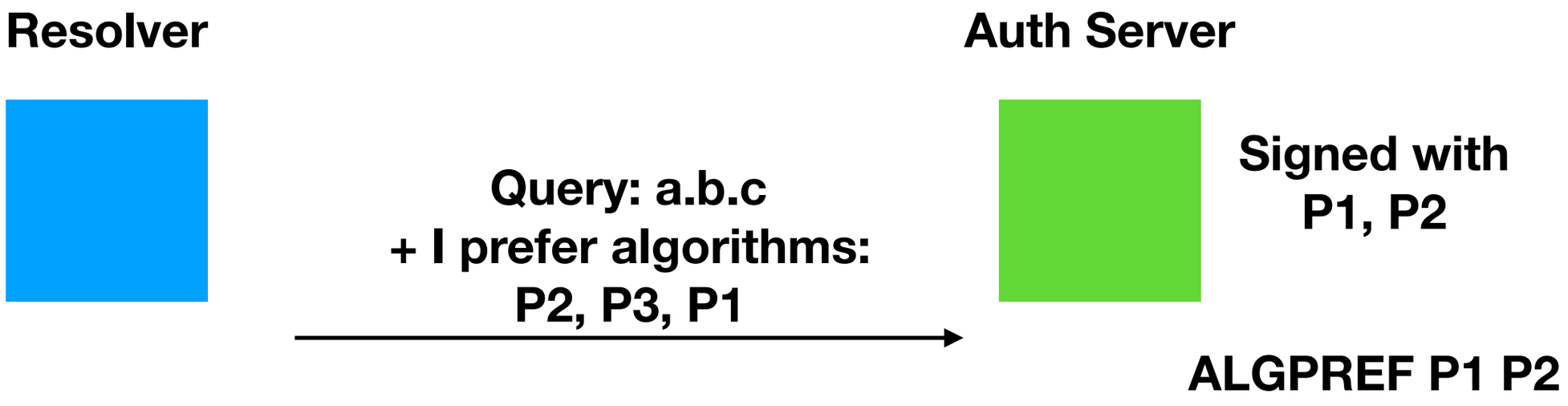
Proposal 2

* The zone operator, not the client, should really be dictating algorithm preference.

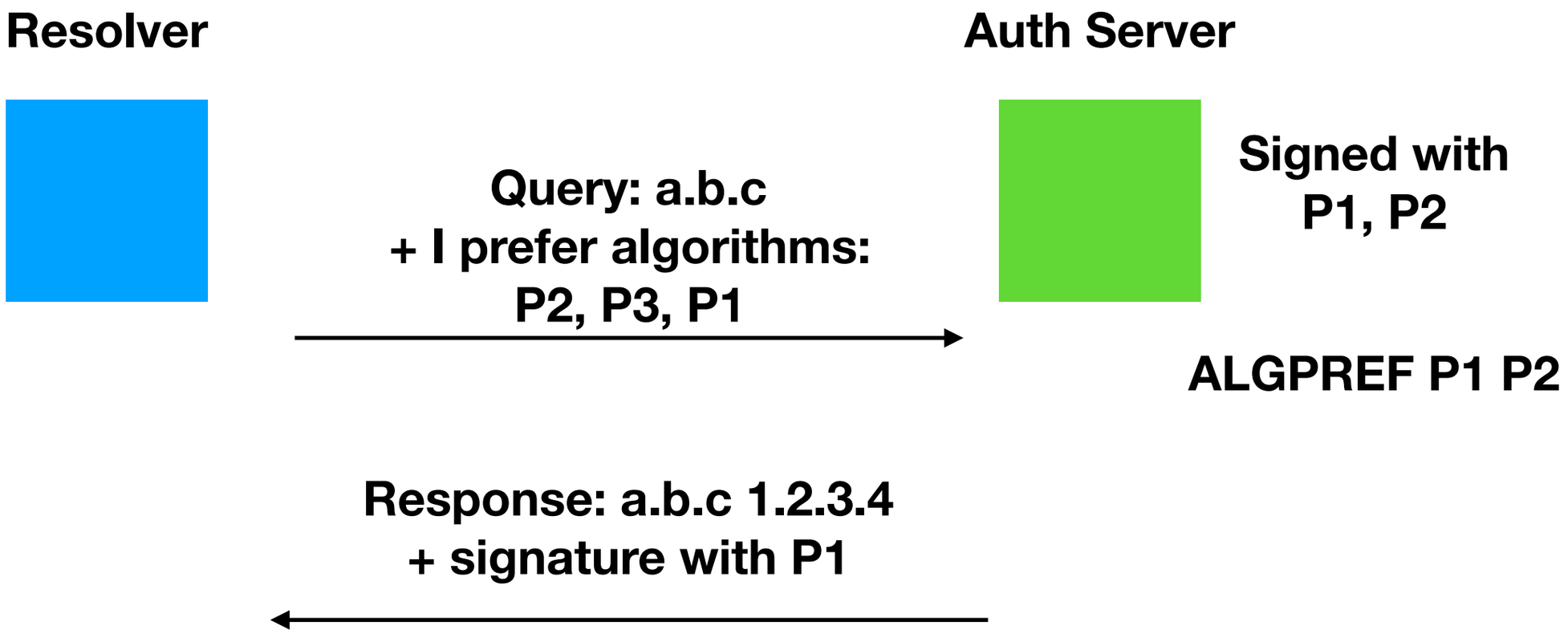
* So client sends its list of algorithms, but server picks the strongest of the list that it supports.

* This means that for downgrade protection, the client will need a way to know in an authenticated manner the algorithm preference list of the server.
  * Server has a new (signed) RR at the zone apex which lists this order (contains an ordered list of algorithm numbers)

Proposal 2


Resolver
Auth Server
Query: a.b.c
+ I prefer algorithms:
P2, P3, P1
Signed with
P1, P2
ALGPREF P1 P2

Proposal 2


Resolver
Auth Server
Signed with
P1, P2
Query: a.b.c
+ I prefer algorithms:
P2, P3, P1
ALGPREF P1 P2
Response: a.b.c 1.2.3.4
+ signature with P1

Cache Implications

- Resolvers that have downstream validators potentially supporting different algorithms

- Resolver will have to keep track of which cache entries have a signatures from a subset of algorithms supported at the authoritative server, and re-query upstream as needed

- Or perhaps fetch all algorithms from upstream and delivery algorithms selectively downstream

Validating Stub1
P1

Validating Stub2
P2, P1

Query 1

Resolver

Q/R

Auth Server
(ALGPREF P1 P2)

Cache P1

Validating Stub3
P2

Validating Stub4
[]

Validating Stub1
P1

Validating Stub2
P2, P1

Resolver

Auth Server
(ALGPREF P1 P2)

Cache P1

Validating Stub3
P2

Query 2

Validating Stub4
[]

Validating Stub1
P1

Validating Stub2
P2, P1

Validating Stub3
P2

Validating Stub4
[]

Query 2

Resolver

Cache P1
Cache P2

Q/R

Auth Server
(ALGPREF P1 P2)

Validating Stub1
P1

Validating Stub2
P2, P1

Validating Stub3
P2

Resolver

Auth Server
(ALGPREF P1 P2)

Cache All Algs

Run alg nego between
validating stubs and resolver

Run vanilla (all algs)
between resolver and
authority

RFC 6975 Redux

- This protocol could also subsume the functionality proposed in RFC 6975 (which hasn't yet seen deployment) to signal DNSSEC algorithm understanding.

- This would give authoritative server operators a way to gauge when a sufficient threshold of the resolver population understands a new algorithm, that it could deploy the new one, or withdraw older ones.

TODO

- Continue refining the draft
    - Fine tune and more fully specify caching behavior for various scenarios and configurations
    - Add Direction bit (to detect broken devices that blindly reflect EDNS options)
- Write prototype code

Some Alternatives

- Mass migrate DNS ecosystem to alternate transports, like TCP, TLS, or QUIC

- Fix the entire Internet so that large fragmented UDP payloads can transit successfully
    - (although fragments can still cause issues: see Haya Shulman's "Fragmentation Considered Poisonous" paper from a few years ago, and more recent attempts to deprecate fragments altogether in the IETF)

- Design an application layer message framing protocol for the DNS

Should we work on this?