NSEC5: Updated Specification & Implementation Results

DNS-OARC Workshop, May 14th 2017
Madrid, Spain

Shumon Huque, Jan Včelák, David Lawrence, Sharon Goldberg, Dimitrios Papadopoulos, Leonid Reyzin, Moni Naor

Summary

- NSEC5 is a new proposal for authenticated denial of existence in DNSSEC:
  - Original design by cryptographers and network security researchers at Boston University and the Weizmann Institute.
  - Subsequent involvement by DNS researchers and engineers at Verisign Labs and CZ.NIC to help turn it into a full DNS protocol and implementation.
- Brief overview of the protocol.
- Implementation and performance results (documented in a new research paper/technical report.)
- Discuss some possible objections & challenges.
- Get your feedback.

NSEC5 Features

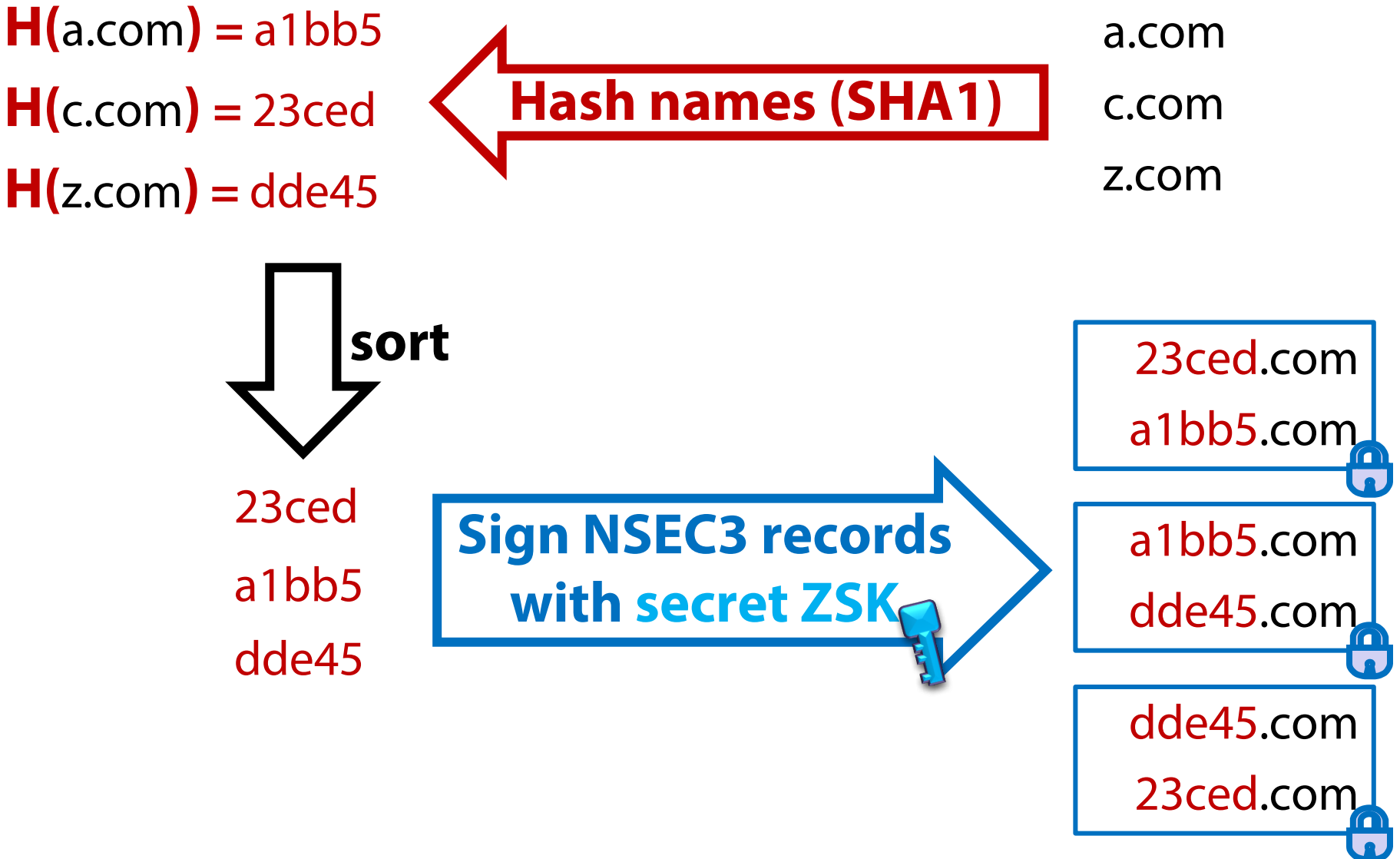
1. Prevents zone enumeration via offline dictionary attack
2. Preserves zone integrity even if nameserver is compromised

- Current denial of existence mechanisms in DNSSEC can only offer one of these properties, but not both simultaneously.
  - Precomputed NSEC3 offers the second property only, whereas existing online signing schemes offer only the first.

NSEC3 Refresher

- How NSEC3 works and why it is vulnerable to zone enumeration by offline dictionary attack …

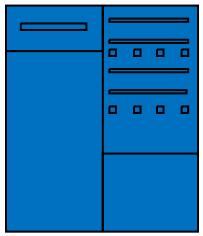
offline signing with NSEC3 [RFC5155]



NSEC3 in action [RFC5155]

Public Zone Signing Key (ZSK):

H(q.com**)** = c987b



a.com

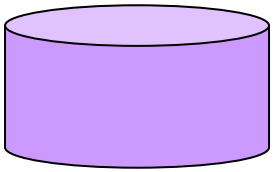
c.com

z.com

q.com?

a1bb5.com

dde45.com



To verify

Does NSEC3 cover query hash?

a1bb5 < c987b < dde45

23ced.com

a1bb5.com

a1bb5.com

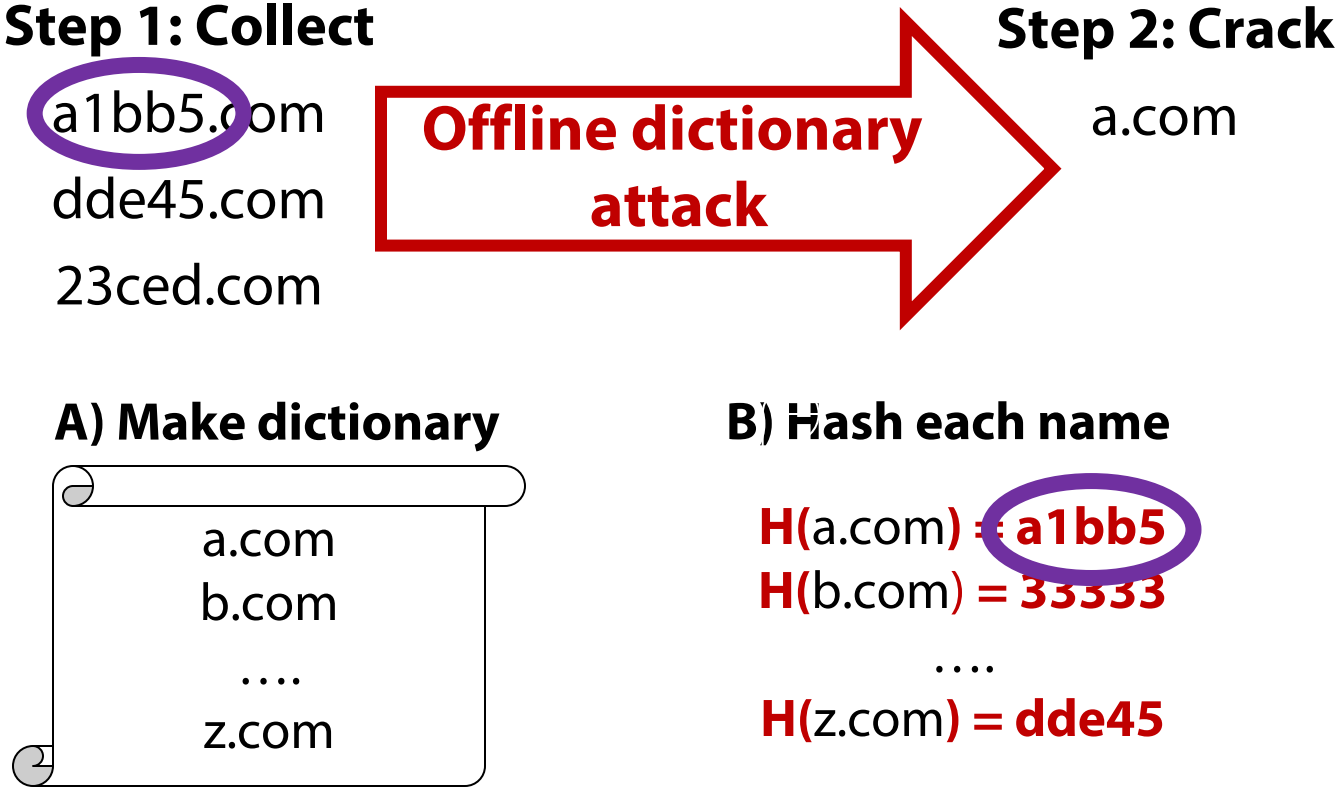
dde45.com

dde45.com

23ced.com

why is offline zone enumeration possible with NSEC3?

Because resolvers can compute hashes offline.



People have done this

[Wander, Schwittmann, Boelmann, Weis 2014] reversed 64% of NSEC3 hashes in the .com in less than a day with one GPU.

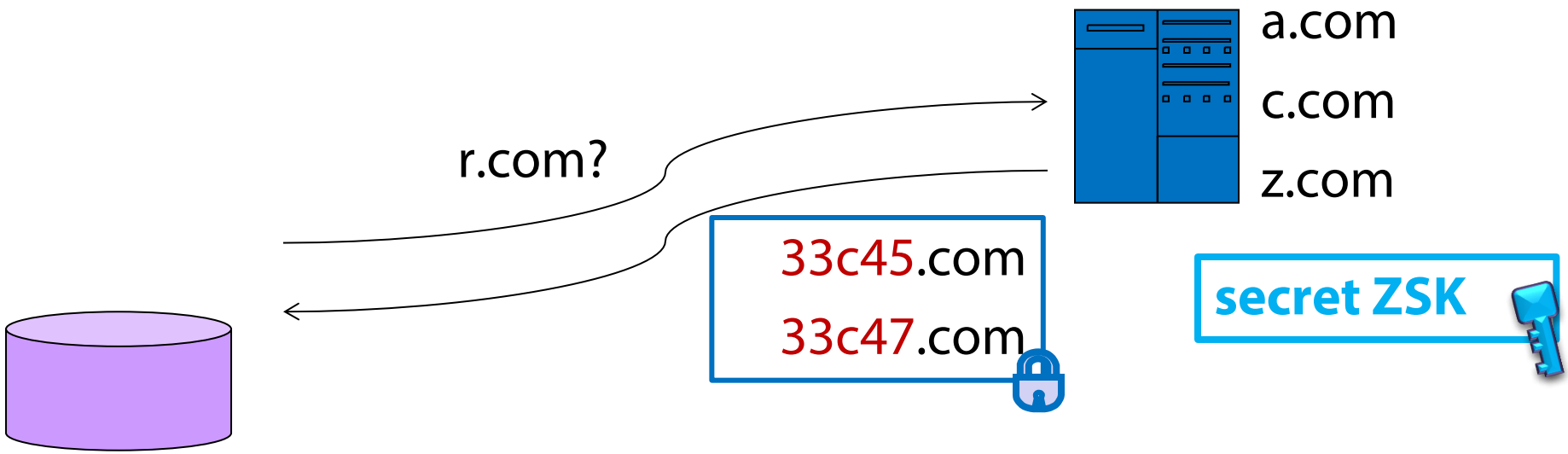
See also [nmap] & [jack-the-ripper] plugins.

Online Signing Schemes

online signing stops offline zone enumeration!

Public Zone Signing Key (ZSK):

H(r.com**)** = 33c46



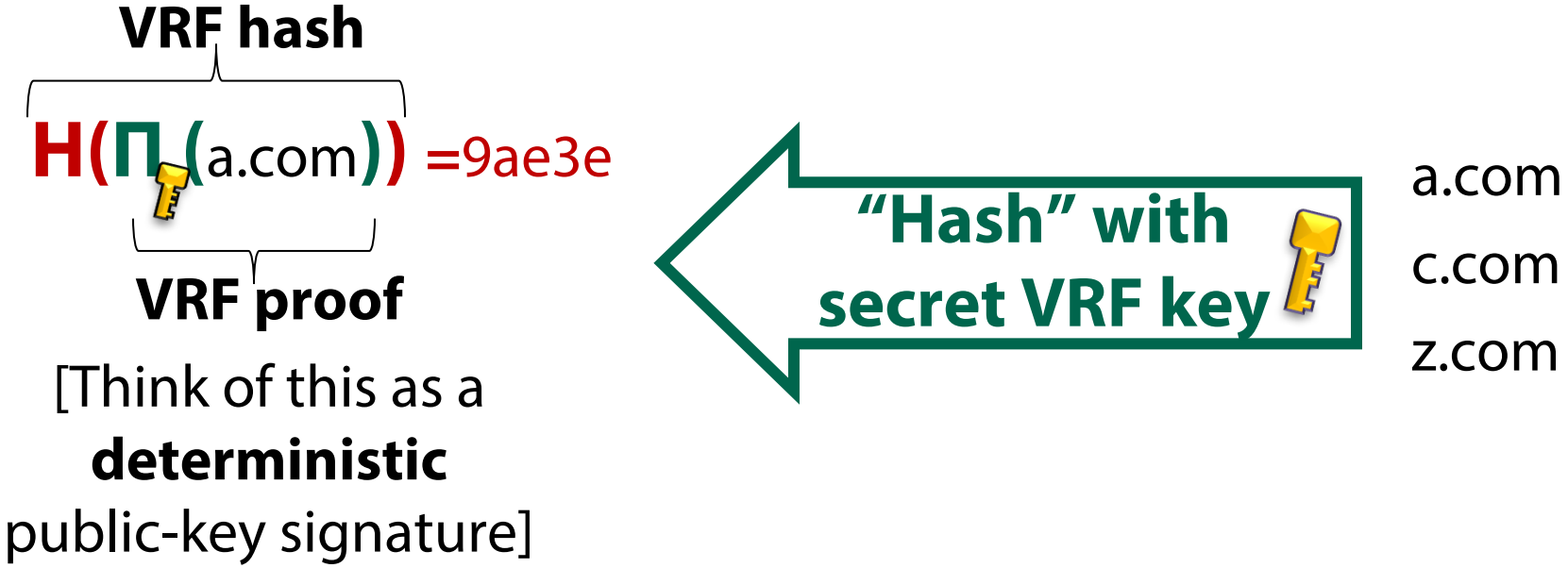
“NSEC3 White Lies”

How NSEC5 works

- NSEC5 replaces the hash (SHA1) used in NSEC3 with a hash computed by a **Verifiable Random Function (VRF)** that resolvers cannot compute offline.
- The VRF has two outputs:
  - The hash output
  - Proof value – that can be used to verify that the hash is correct

- Roughly, the proof value is like a deterministic public key signature.
- NSEC5 uses a separate public/private key pair for the VRF
  - Authoritative server has access to the VRF private key
  - Uses it to pre-compute offline the hashes for existing names that are signed into NSEC5 records with the ZSK private key
  - Uses it to dynamically compute the VRF proofs for non-existent names.

offline signing with NSEC5



* **NSEC5-ECC:** VRF based on elliptic curves

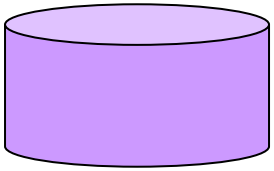
- **[draft-goldbe-vrf-00]**.
- Has a formal cryptographic security proof.
- For 256-bit elliptic curves, Π gives 641-bit outputs.

answering queries with NSEC5

Public Zone Signing Key (ZSK):

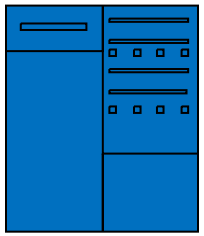
Public VRF Key:

q.com?



PROOF
aa8678

NSEC5 3cd91.com 8cb67.com



a.com
c.com
z.com

Π(q.com)= PROOF aa8678

H(aa867)=7a89b

secret VRF key

NSEC5 3cd91.com 8cb67.com

NSEC5 8cb67.com 9ae3e.com

NSEC5 9ae3e.com 3cd91.com

To verify:

Does NSEC5 cover PROOF?

3cd19 < H(aa8678) < 8cb67

Does PROOF match query?

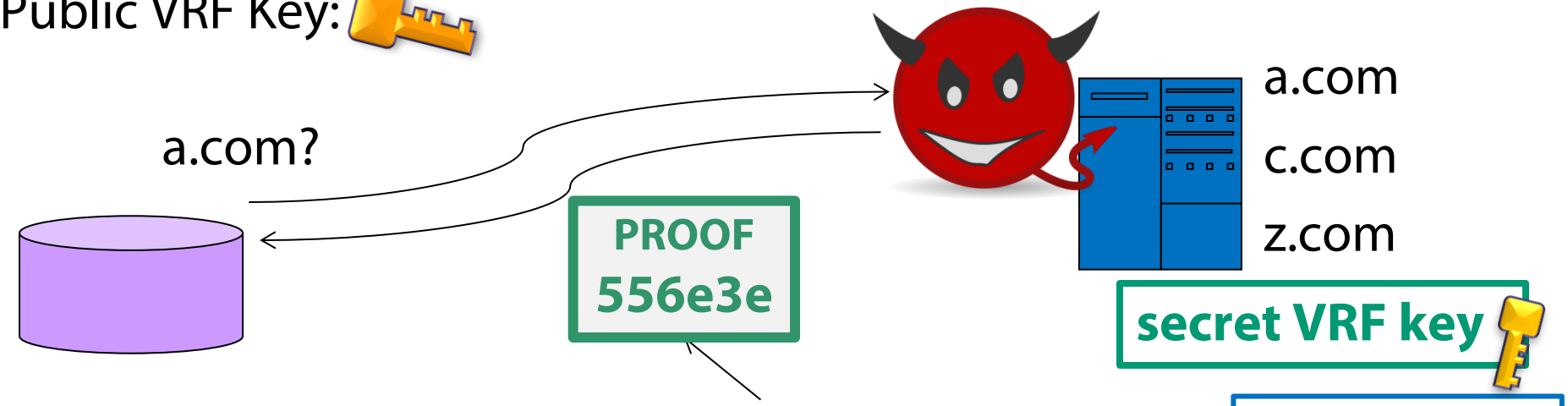
VER (q.com, aa8678)

[Think of this as a signature verification]

why NSEC5 has integrity even if secret VRF key is lost

Public Zone Signing Key (ZSK):

Public VRF Key:



The proof is unique given the public VRF key. It must be correct b/c resolvers validate it!

! Don't know secret ZSK, so can't forge NSEC5s

! There is no covering NSEC5 to replay, since **H(556e3e)=9ae3e**

3cd91.com
8cb67.com

8cb67.com
9ae3e.com

9ae3e.com
3cd91.com

DNSSEC Authenticated Denial of Existence

	No offline zone enumeration	Integrity vs outsiders	Integrity vs compromised nameserver	No online crypto
DNS (legacy)	✔	X	X	✔
NSEC or NSEC3	X	✔	✔	✔
Online Signing ("NSEC3 White Lies")	✔	✔	X	X
NSEC5	✔	✔	✔	X

Because resolvers cannot compute VRF hashes offline

Because the nameserver doesn't know the zone-signing key

In **[NDSS'15]** we proved this is **necessary** to prevent zone enumeration & have integrity

NSEC5 implementation

Knot DNS

authoritative nameserver

&

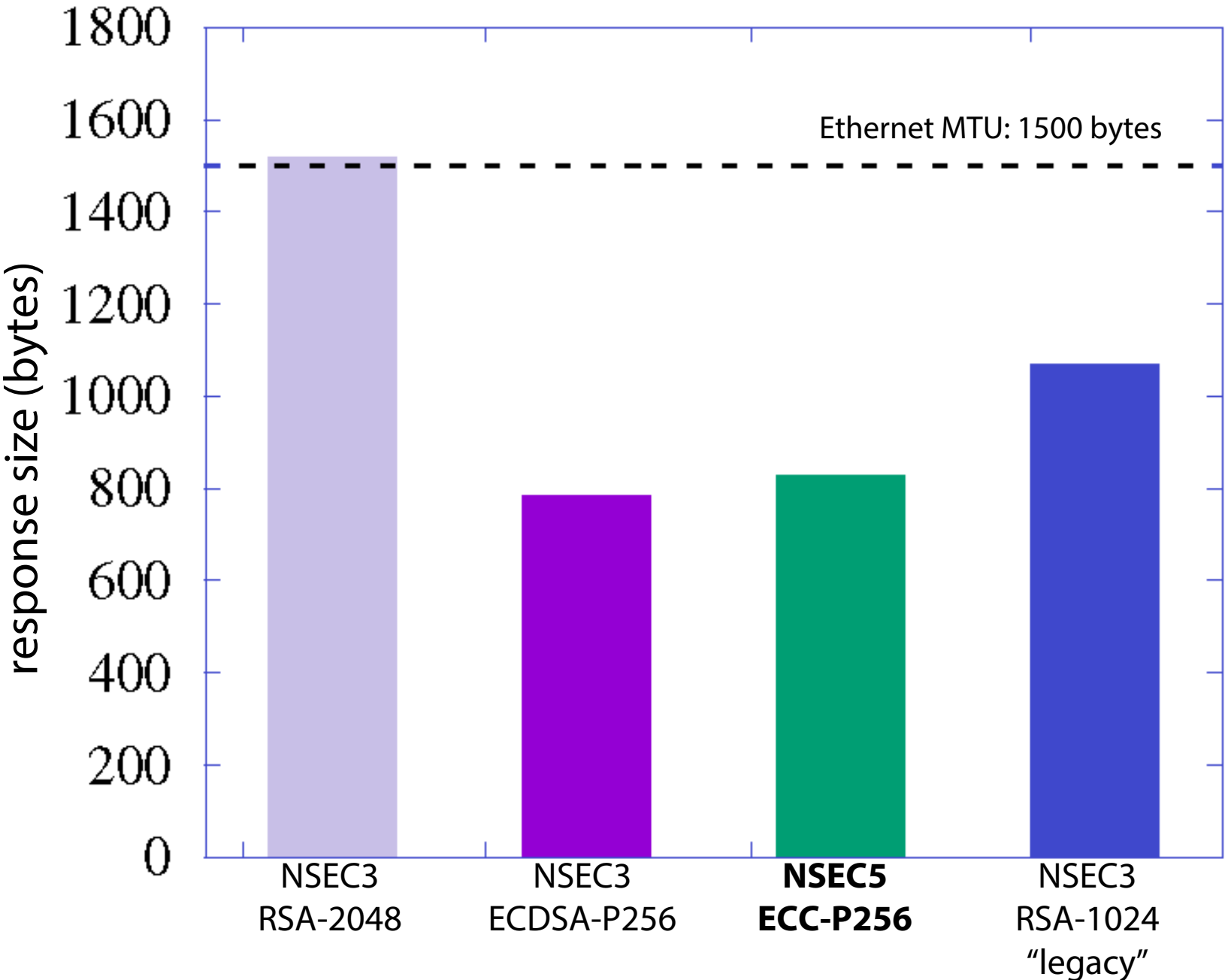
Unbound

recursive server

9K Lines of Code, no new libraries or system optimizations

Current implementations support P-256 curve.
Could be faster with Ed25519 curve included in the -04 draft

empirical measurement of NXDOMAIN response sizes



nameserver query throughput (pure NXDOMAIN traffic)

Machine specs: 2 x 10-core Intel(R) Xeon(R) CPU E5-2660 v3 @ 2.60GHz Dual Mode
(Total 24 threads on 40 virtual CPUs) 256GB RAM running CentOS Linux 7.1

More discussion re: performance

- Intuitively we would expect NSEC5 performance to fall between precomputed NSEC3 (fastest) and NSEC3 White Lies
  - And that is what we observe
  - NSEC3 – everything is precomputed
  - NSEC5 with our additional protocol optimizations does 1 online asymmetric signing per negative response
  - NSEC3 White Lies does multiple (2 to 3) online asymmetric signings per negative response.
- In the real world, there will be a mix of queries for existing and non-existing names, so NSEC5 performance is further improved and is closer to that of precomputed NSEC3.
  - More details of such testing is in the NSEC5 research paper.

Latest Protocol Specification

- NSEC5: DNSSEC Authenticated Denial of Existence
  - **https://tools.ietf.org/html/draft-vcelak-nsec5-04**

- Removed RSA
- Elliptic Curves: 2 defined: NIST P256, and Ed25519
- DNS level optimizations:
  - Wildcard bit from draft-gieben-nsec4
  - Precomputed closest encloser proofs

3 New DNS Record Types

- NSEC5KEY
  - Contains the VRF algorithm and Public Key.
  - The Public Key part of the RDATA is the same format as in DNSKEY.

- NSEC5
  - Like NSEC3 but contains the precomputed VRF hash a name rather than the SHA1 hash of the name.
  - Precomputed and signed (RRSIG) by the zone signing key.

- NSEC5PROOF
  - Contains the VRF proof output for the non-existent name being queried for.
  - Dynamically generated and not signed (no accompanying RRSIG).

Example dig/kdig output (DNSKEY)

```
$ kdig +dnssec example.com. DNSKEY

;; new algorithm number (temporarily using private# 253) that are aliases
;; to existing ones like ecc p256 used to signal the zone is using nsec5.

;; ANSWER SECTION:
example.com.            3600 IN DNSKEY 256 3 253 (
                                +f4VijH2siRemoL1y8leU0T4/YF15D9Vso+K0luy
                                Pj+Tsixc9VcI5UcTbB9sQIGg/NpPqm0ThN6pv2aW
                                63moAQ==
                                ) ; ZSK, alg = NSEC5_ECP256SHA256, id = 5137
example.com.            3600 IN DNSKEY 257 3 253 (
                                TrUFT4wFWtVxRhApIBowUu6DekUxZqjRQJvqMMTZ
                                Y1kvu5PBjRfW07cVjw/1nn9gFm/H6aMOVD4iUNtp
                                nA6oZA==
                                ) ; KSK, alg = NSEC5_ECP256SHA256, id = 41260
example.com.            3600 IN RRSIG DNSKEY 253 2 3600 20170530171847 (
                                20170430171847 41260 example.com.
                                Hiqqje1BmCmeZJjbry9eDpoFKUUA+GkL8H5rjN2D
                                mEHL4ybhciAkQLR2/K+lOlcYP2YjEl0Lgw+CAwuX
                                VD+llA==                                )
```

Example dig/kdig output (NSEC5KEY)

```
$ kdig +dnssec example.com. NSEC5KEY

;; New RRtype, NSEC5KEY, that contains the NSEC5 algorithm and associated
;; VRF public key.

;; ANSWER SECTION:
example.com.            3600 IN NSEC5KEY 253 (
                                16uluxDTop/7xAKAN9y/4xW/CqnjHJ6wA+RmXM32
                                GjDzwOV+dr65G7TvuG9vH2Nds3lUx5TiBJdtRjuB
                                ImXlYQ==
                                )
example.com.            3600 IN RRSIG NSEC5KEY 253 2 3600 20170530171847 (
                                20170430171847 41260 example.com.
                                f3Xp4HLH2pCzJRGiZdPj/5JNF+vNx0QQF3oo62sZ
                                lDayahmtwYdWeETiV7g4cr+BFdYTwc1VeJmZFPIc
                                nitWZg==                                )
```

Example dig/kdig output (NXDOMAIN)

```
$ kdig +dnssec doesntexist.example.com. A

;; AUTHORITY SECTION:
;; Following NSEC5PROOF corresponds to the closest encloser of the qname.
;; This is a signature produced by the VRF private key.
;; Note: this can be precomputed and cached by the authoritative server.
;; SHA256 hash: EC2I1K1ADN16BB9SBH1K5QJBODGNTAB96P39RMJ30H1OKMMEDOUG

example.com.            86400 IN NSEC5PROOF 48566 (
                                 AiZnaTPduKWyigRmOOohGGaxBXlGnNmttEsQ5HSj
                                 tHF1ePiphu6zkIgSPTcWL5WO7y2qKtX6/3L/FY5W
                                 0ZvyekQGvTv3/NrlsSW/+3pjvy15
                                 )

;; Hash(NSEC5PROOF(closest encloser) corresponds to the following
;; NSEC5 record. The absence of the wildcard flag in the NSEC5 record
;; shows that wildcard synthesis was not possible.

ec2i1k1adn16bb9sbh1k5qjbodgntab96p39rmj30h1okmmedoug.example.com. 86400 IN
NSEC5 48566 0 FRHPR3K6ATTMM20F2N38SIAIV947A2N7RALADUE2GQKNTN44FQJ0 (
                                 NS SOA MX RRSIG DNSKEY NSEC5KEY
                                 )
[Precomputed RRSIG for above NSEC5 record omitted for brevity.]
```

Example dig/kdig output (NXDOMAIN)

```
[continued from previous page]

;; AUTHORITY SECTION:

;; Following NSEC5PROOF corresponds to the next closer name (which in
;; this case is the same as the qname). It is the VRF proof output of
;; the next closer name generated on-the-fly using the NSEC5 private key.
;; VRF hash output: HHH2PQNQ5M6RB06U1TLER4I0CHH0LN6F4NVD3BKO3TT661VLI2HG

doesntexist.example.com.        86400 IN NSEC5PROOF 48566 (
                                AorfNogAbm5EJzrrrj9jTTm6iP7MfUY0kfhKPkAU
                                MCvx/zpUxEgoEmBYi+DBA77JYN0avEwsEiXQqbz6
                                JT5D3dVAO7Oh1NnMsGtC6xmNGOYB
                                )

;; NSEC5 record covering next closer name.
;; VRF hash(next closer) falls within the following NSEC5 record span

h4ettrt2rnlvqa2du6hmpjdkmcavq69gh67nui2hskvd9rcjt9r0.example.com. 86400 IN
NSEC5 48566 0 IN7MUIR4VTSQGKLF6HR2VD5LL9Q6KOEO1ICU6J6G2TRDU2VH0AU0 (
                                A AAAA RRSIG
                                )
[Precomputed RRSIG(NSEC5) and also SOA + RRSIG(SOA) omitted for brevity]
```

Addressing some possible objections

- Is Zone Enumeration prevention needed?
  - Yes, several European ccTLDs, many enterprises, universities, etc
  - DNS data are public, but see RFC 7626 (DNS Privacy Considerations)
  - You have to know what to query for, and mechanisms that allow mass leakage of zone data should be avoided.

- Is NSEC3 good enough?
  - Evidence from folks relatively easily cracking nsec3 zones and availability of nsec3 zone enumeration tools suggests not.

Addressing some possible objections

- Do Passive DNS databases make the zone enumeration prevention goal unrealistic?
  - We don't think so.
  - Passive DNS services see a lot, but they don't have anywhere near a comprehensive view of the DNS.
  - Many large providers will not contribute data to PDNS services for privacy reasons.
  - Most smaller resolvers are under the radar.
  - Privacy conscious users likely won't use a resolver that participates in PDNS collection.
  - Regardless of the existence of such systems, the DNS protocol itself should not provide easy mechanisms to leak masses of zone data.

Addressing some possible objections

- Are the Performance costs too high?
  - Our performance testing results indicate that NSEC5 is well within the reach of modern hardware.

- Are the Transition costs too high?
  - They are certainly high, but …

Reducing Transition Costs

- Algorithm transitions in DNSSEC are very painful today.
- But we have several waiting in the pipeline:
  - EdDSA
  - Post Quantum algorithms
  - NSEC5, if the community adopts it
  - Other proposals: SHA3 and RSASSA-PSS
- We should figure out how to make this less painful, and in particular how to efficiently transition to new algorithms
  - Lessons from RSA->ECDSA transition?
  - Do we lump together multiple transitions?
  - Will alternative transports (like DNS over TLS/DTLS/QUIC) make it more palatable for zone operators to sign their zones with multiple algorithms simultaneously?
  - Does the protocol need an algorithm selection mechanism?

Questions?

- Research paper with performance numbers & crypto proofs:
  http://ia.cr/2017/099
- NSEC5 Project page:
  https://www.cs.bu.edu/~goldbe/papers/nsec5.html
- NSEC5 Protocol Specification:
  https://tools.ietf.org/html/draft-vcelak-nsec5-04

dnsreactions…

Hearing about NSEC5

When I finally grasp NSEC5

Extra Slides

VRF versus Public Key signature schemes

- They are very similar. A VRF can be thought of as a public key version of a keyed cryptographic hash, which also satisfies some specific properties:

1. VRFs have two outputs: the VRF hash output, and the VRF proof, that could be delivered separately. The proof is constructed such that anyone with the VRF public key can verify that the hash is correct for the given input.
2. Pseudo-randomness: The VRF hash output is indistinguishable from random by anyone who does not know the VRF private key.
3. Trusted Uniqueness: each VRF input corresponds to a unique output value.

VRF versus Public Key signature schemes

- Note: in the VRF construction used for NSEC5, we effectively use one output, since the VRF hash output can be derived from the proof. For DNSSEC, we do not need to separate the two, and thus delivering one quantity is more efficient.

- Other applications using VRFs:
  - Google Key Transparency Project

- References:
  - "Verifiable Random Functions", 1999, MRV
    - https://people.csail.mit.edu/silvio/Selected%20Scientific%20Papers/Pseudo%20Randomness/Verifiable_Random_Functions.pdf
  - Internet-Draft: Verifiable Random Functions:
    - https://tools.ietf.org/html/draft-goldbe-vrf-00