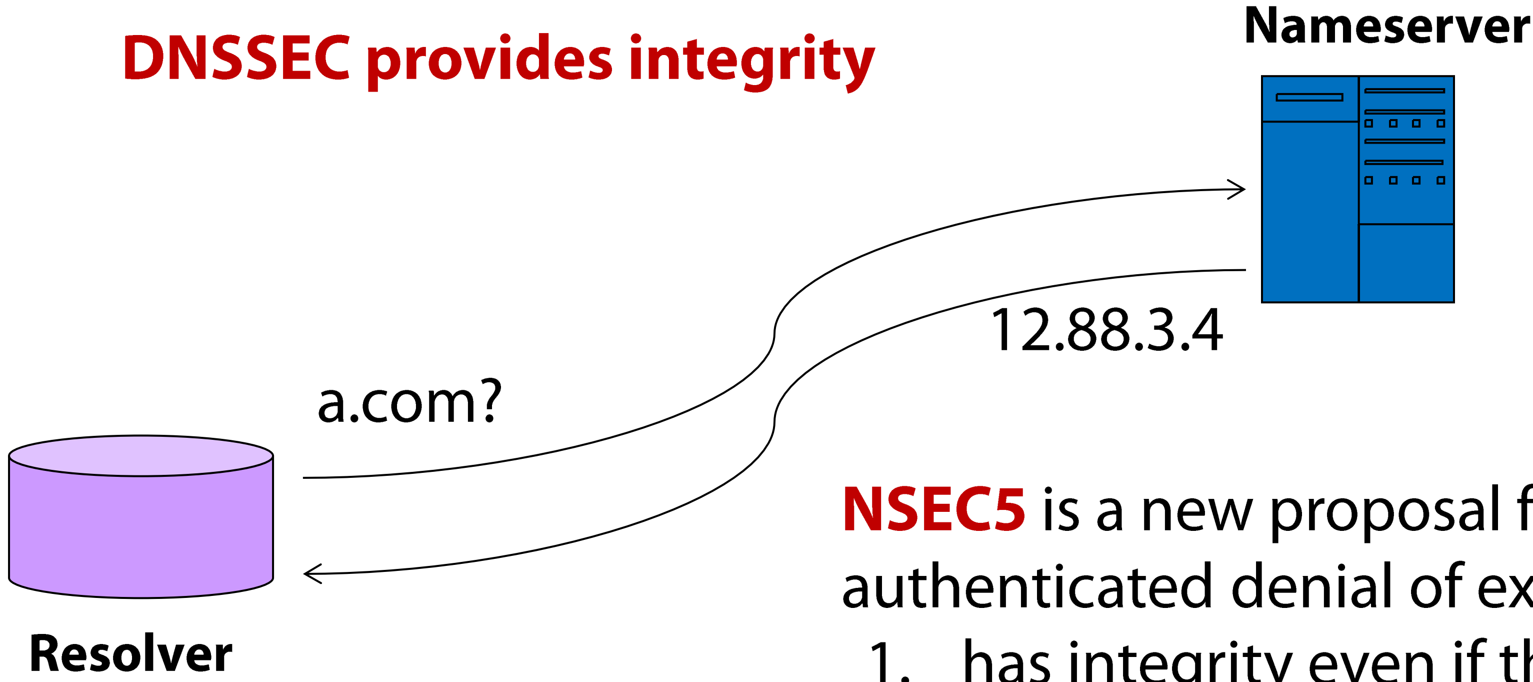
Can NSEC5 be practical for DNSSEC deployments?

Dimitrios Papadopoulos, Jan Včelák, Moni Naor, Leonid Reyzin, Sharon Goldberg

DPRIVE Workshop, San Deigo, California,
February 26 2017

BOSTON UNIVERSITY
מכון ויצמן למדע WEIZMANN INSTITUTE OF SCIENCE
UNIVERSITY OF MARYLAND
cz.nic
NS1.

DNSSEC negative responses and NSEC5

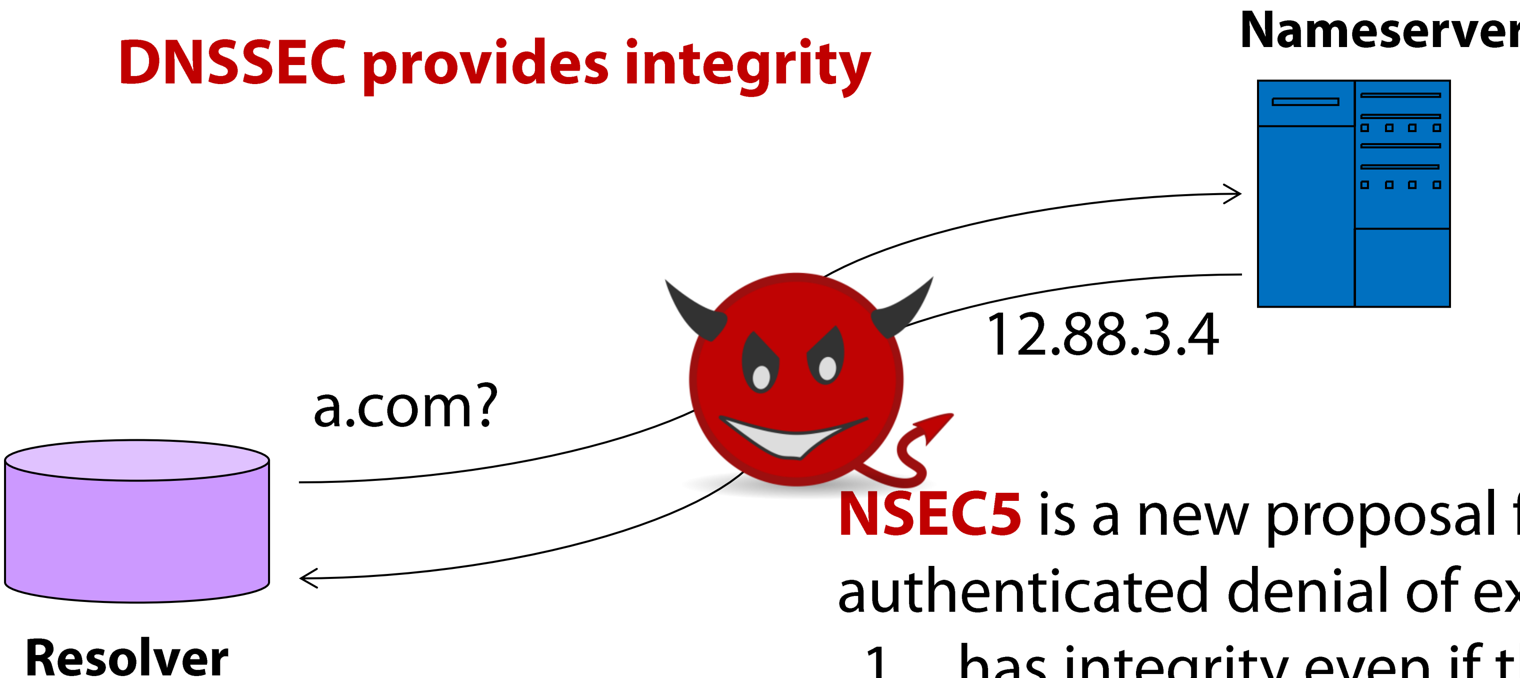


NSEC5 is a new proposal for DNSSEC authenticated denial of existence

1. has integrity even if the nameserver is compromised.
2. prevents offline zone enumeration

DNSSEC negative responses and NSEC5

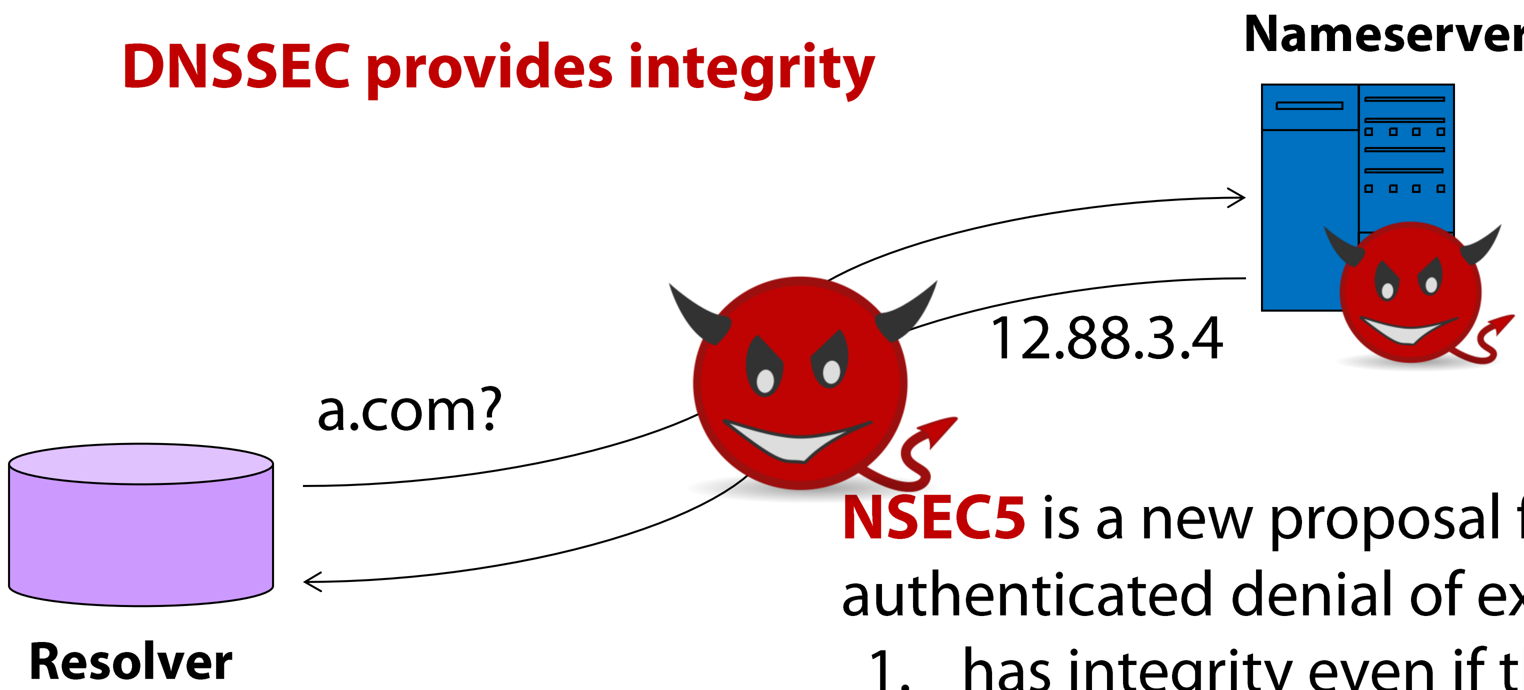
DNSSEC provides integrity



NSEC5 is a new proposal for DNSSEC authenticated denial of existence

1. has integrity even if the nameserver is compromised.
2. prevents offline zone enumeration

DNSSEC negative responses and NSEC5

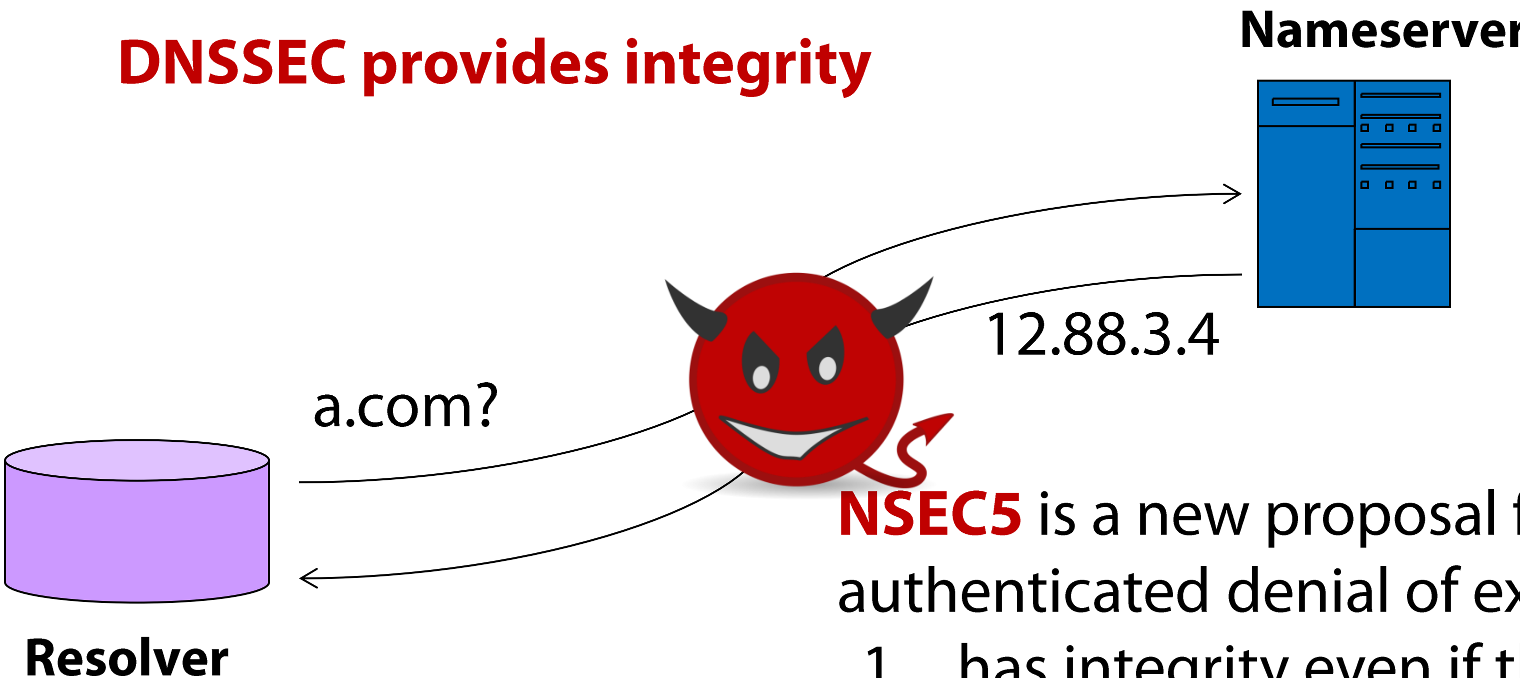


NSEC5 is a new proposal for DNSSEC authenticated denial of existence

1. has integrity even if the nameserver is compromised.
2. prevents offline zone enumeration

DNSSEC negative responses and NSEC5

DNSSEC provides integrity



NSEC5 is a new proposal for DNSSEC authenticated denial of existence

1. has integrity even if the nameserver is compromised.
2. prevents offline zone enumeration

New contributions:

- Elliptic curve NSEC5
- Full specification
- Full implementation
- Prelim performance results

offline signing with NSEC3 [RFC5155]

a.com

c.com

z.com

offline signing with NSEC3 [RFC5155]

H(a.com) = a1bb5

H(c.com) = 23ced

H(z.com) = dde45

a.com

c.com

z.com

offline signing with NSEC3 [RFC5155]

H(a.com) = a1bb5

H(c.com) = 23ced

H(z.com) = dde45

Hash names

a.com

c.com

z.com

sort

23ced

a1bb5

dde45

offline signing with NSEC3 [RFC5155]

a.com
c.com
z.com

Hash names

H(a.com) = a1bb5
H(c.com) = 23ced
H(z.com) = dde45

sort

23ced
a1bb5
dde45

Sign NSEC3 records with secret ZSK

23ced.com
a1bb5.com

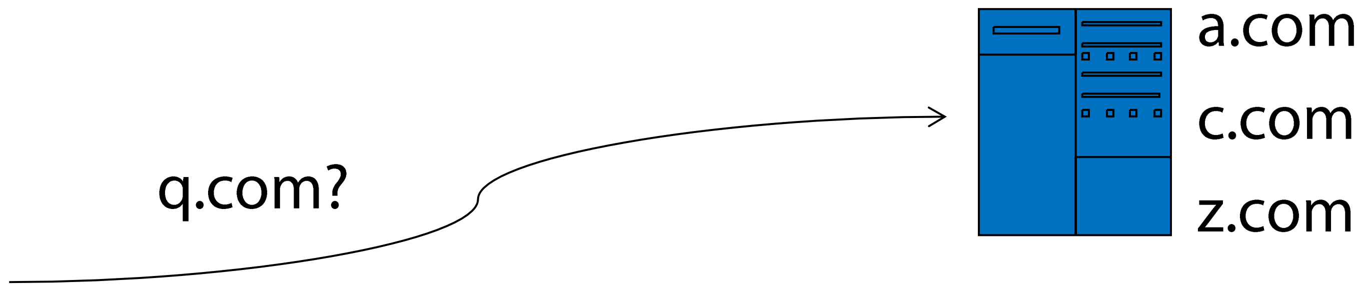
a1bb5.com
dde45.com

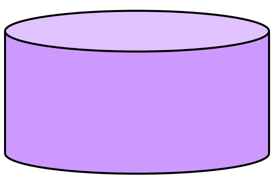
dde45.com
23ced.com

NSEC3 in action [RFC5155]

Public Zone Signing Key (ZSK): 

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





<u>To verify</u>

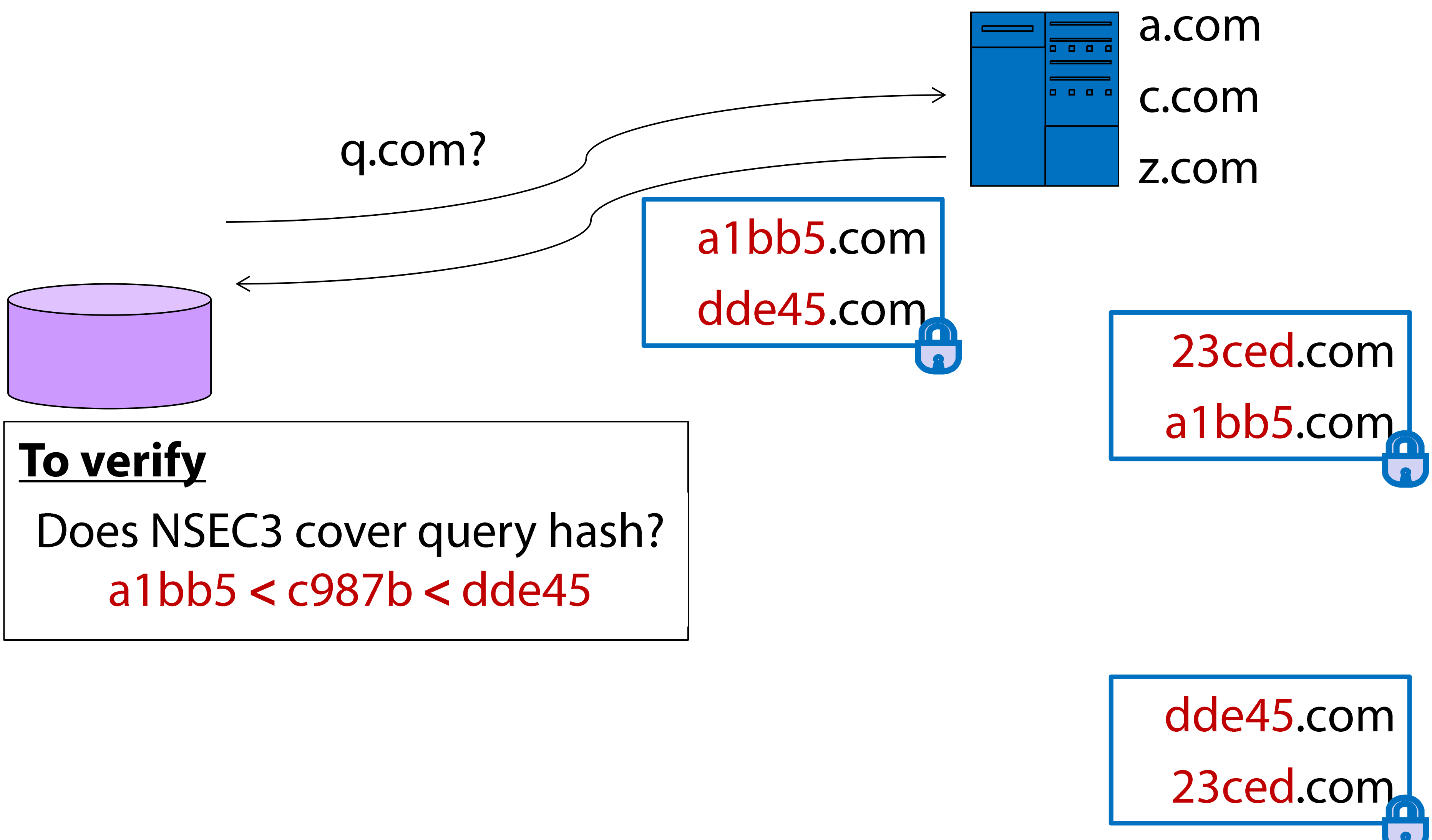
Does NSEC3 cover query hash?

a1bb5 < c987b < dde45

NSEC3 in action [RFC5155]

Public Zone Signing Key (ZSK):

H(q.com) = c987b



To verify

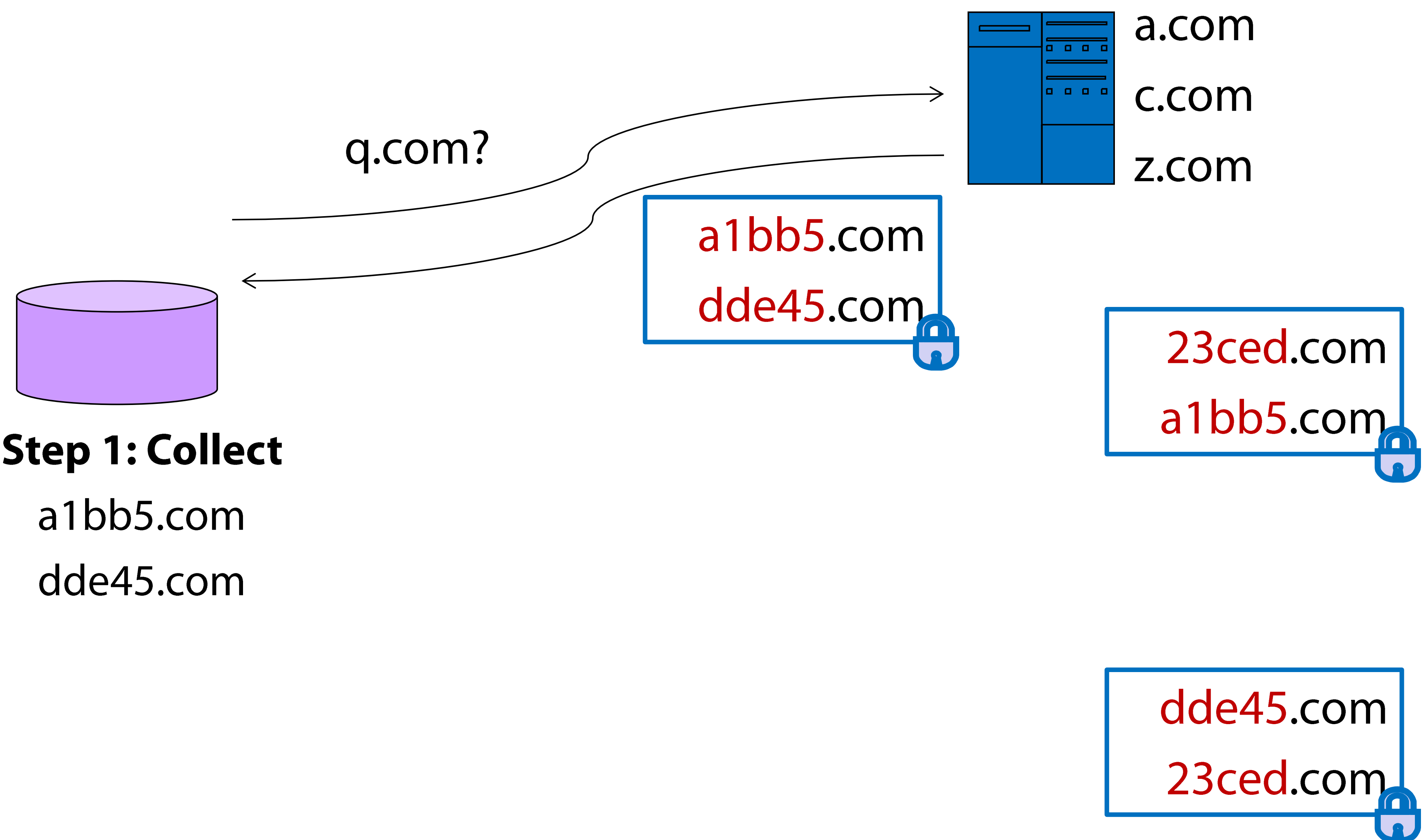
Does NSEC3 cover query hash?

a1bb5 < c987b < dde45

NSEC3 offline zone enumeration attack

Public Zone Signing Key (ZSK):

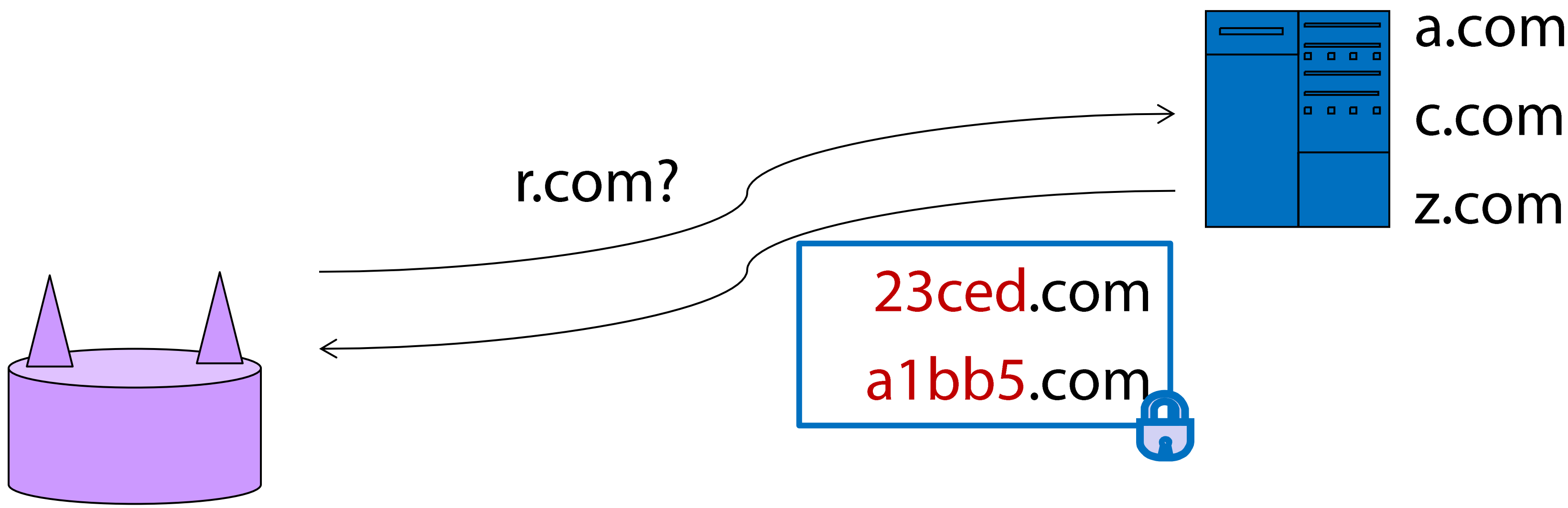
H(q.com) = c987b



NSEC3 offline zone enumeration attack

Public Zone Signing Key (ZSK): 

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



Step 1: Collect

a1bb5.com

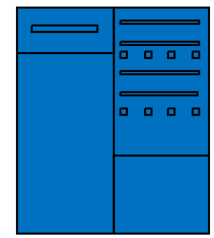
dde45.com

23ced.com

NSEC3 offline zone enumeration attack

Public Zone Signing Key (ZSK): 

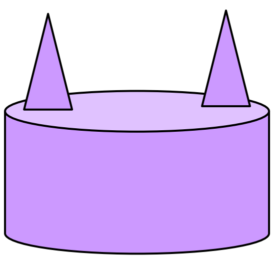
H(r.com) = 33c46



a.com

c.com

z.com



Step 1: Collect

a1bb5.com

dde45.com

23ced.com

Offline dictionary attack

Step 2: Crack

a.com

z.com

c.com

[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.

why is offline zone enumeration possible with NSEC3?

Because resolvers can compute hashes offline.

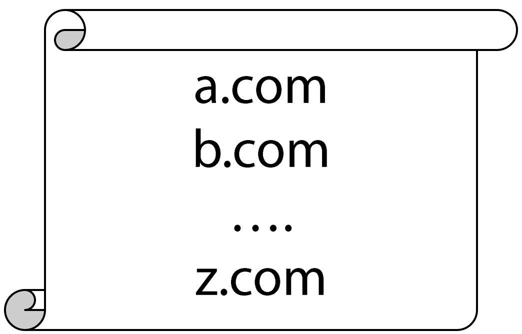
Step 1: Collect

a1bb5.com

dde45.com

23ced.com

A) Make dictionary



B) Hash each name

H(a.com**) = a1bb5**
H(b.com**) = 33333**
....
H(z.com**) = dde45**

NSEC5 replaces the hash **H** with a **Verifiable Random Function (VRF)** that resolvers cannot compute offline.

why is offline zone enumeration possible with NSEC3?

Because resolvers can compute hashes offline.

Step 1: Collect

a1bb5.com

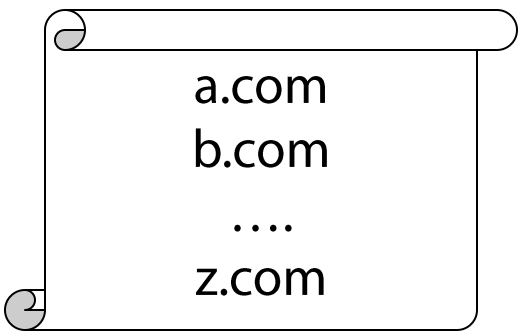
dde45.com

23ced.com

Step 2: Crack

a.com

A) Make dictionary



B) Hash each name

H(a.com**) = a1bb5**
H(b.com**) = 33333**
....
H(z.com**) = dde45**

NSEC5 replaces the hash **H** with a
Verifiable Random Function (VRF)
that resolvers cannot compute offline.

why is offline zone enumeration possible with NSEC3?

Because resolvers can compute hashes offline.

Step 1: Collect

a1bb5.com
dde45.com
23ced.com

Offline dictionary attack

Step 2: Crack

a.com

A) Make dictionary

a.com
b.com
....
z.com

B) Hash each name

H(a.com**) = a1bb5**
H(b.com**) = 33333**
....
H(z.com**) = dde45**

NSEC5 replaces the hash **H** with a **Verifiable Random Function (VRF)** that resolvers cannot compute offline.

why is offline zone enumeration possible with NSEC3?

Because resolvers can compute hashes offline.

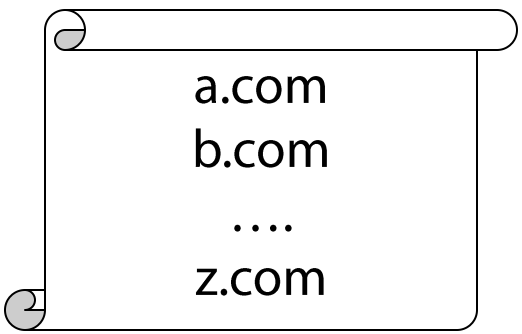
Step 1: Collect

a1bb5.com
dde45.com
23ced.com

Step 2: Crack

a.com
z.com
c.com

A) Make dictionary



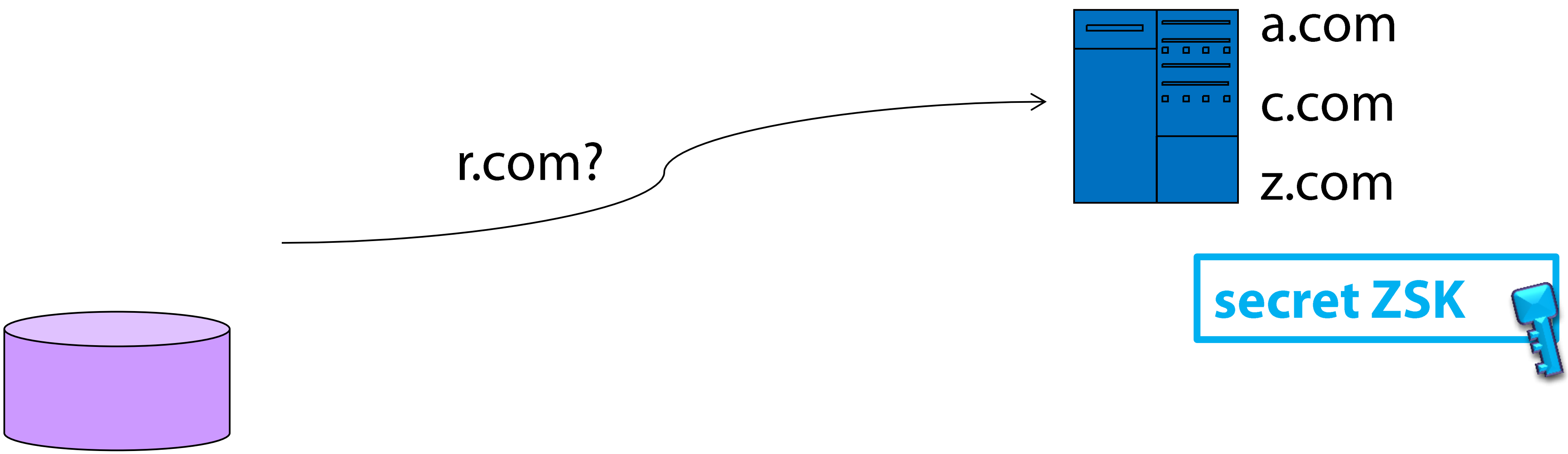
B) Hash each name

H(a.com**) = a1bb5**
H(b.com**) = 33333**
....
H(z.com**) = dde45**

NSEC5 replaces the hash **H** with a
Verifiable Random Function (VRF)
that resolvers cannot compute offline.

online signing stops offline zone enumeration!

Public Zone Signing Key (ZSK): 

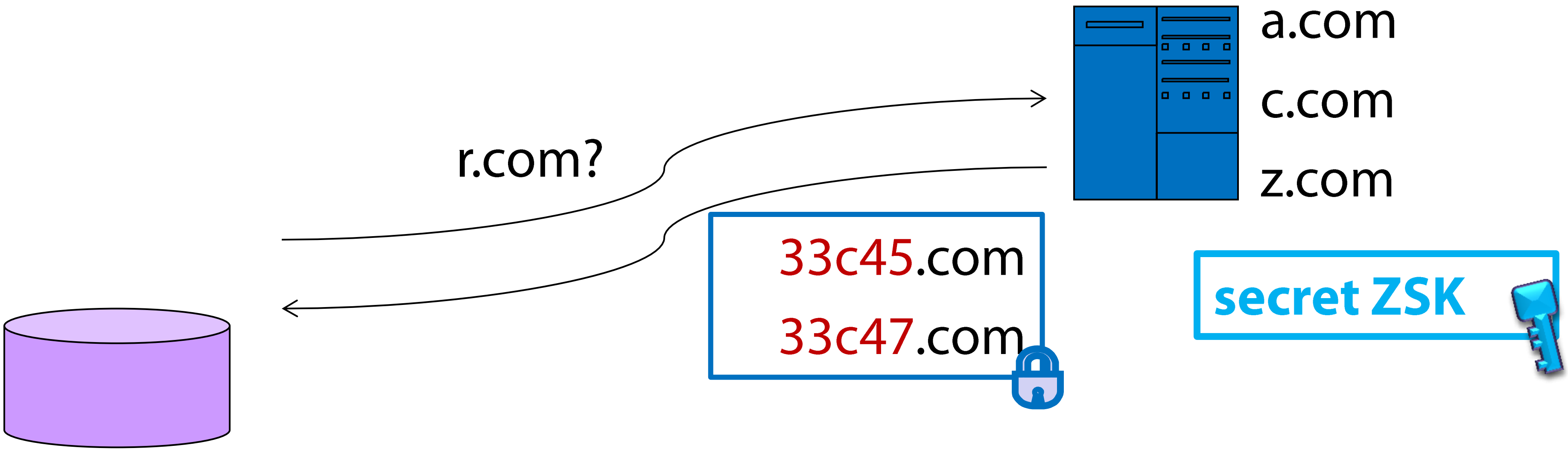


“NSEC3 White Lies”

online signing stops offline zone enumeration!

Public Zone Signing Key (ZSK):

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



"NSEC3 White Lies"

comparison of different schemes

	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

Theorem [NDSS'15]: For ANY denial of existence scheme that

1. prevents offline zone enumeration, and
2. provides integrity against outsiders

nameservers must compute a public-key signature for each negative response.

comparison of different schemes

	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

Theorem [NDSS’15]: For ANY denial of existence scheme that

1. prevents offline zone enumeration, and
2. provides integrity against outsiders

nameservers must compute a public-key signature for each negative response.

NSEC5: precomputing records

a.com

c.com

z.com

NSEC5: precomputing records

H(Π(a.com)) =9ae3e

H(Π(c.com)) =8cb67

H(Π(z.com)) =3cd91

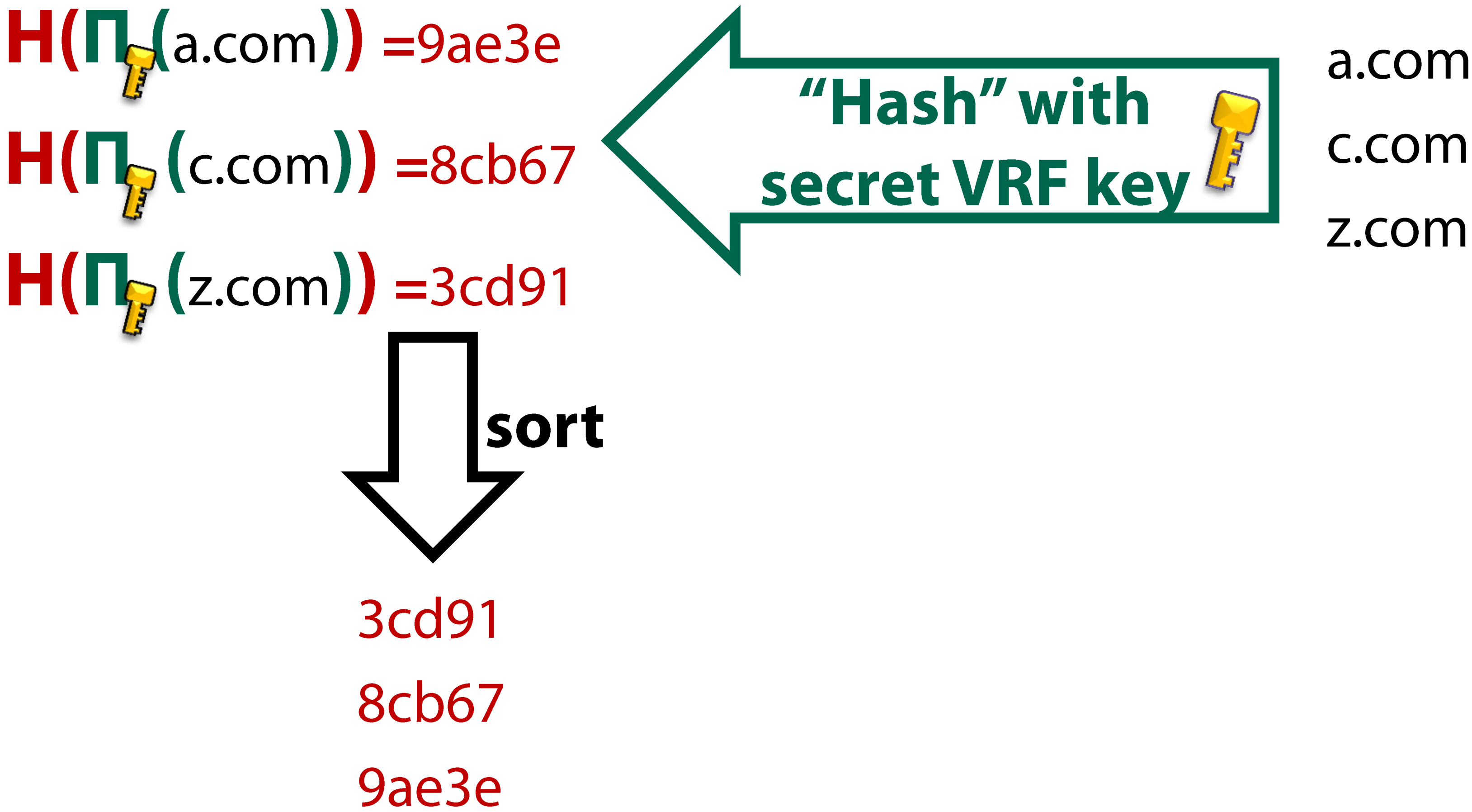
"Hash" with secret VRF key

a.com

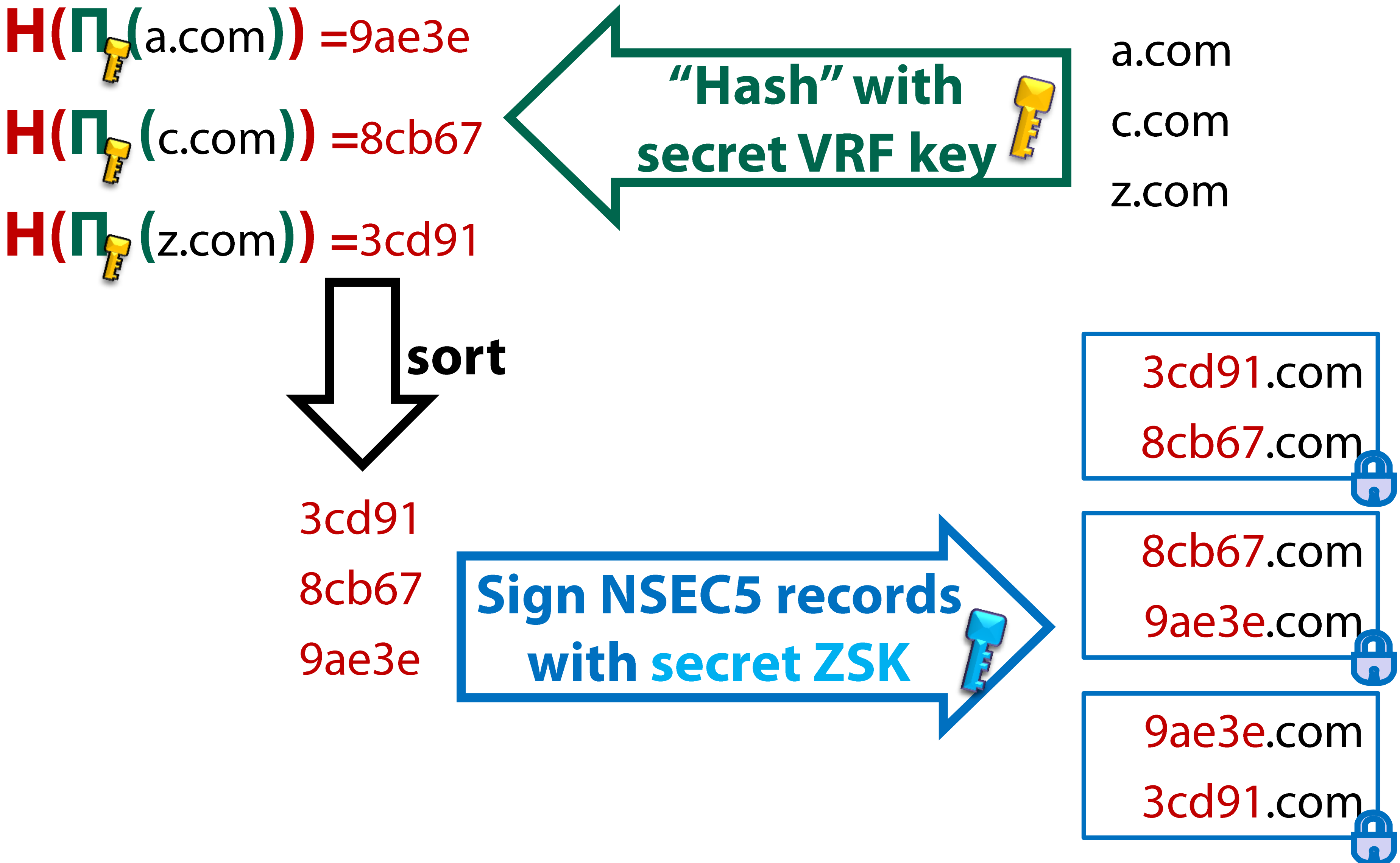
c.com

z.com

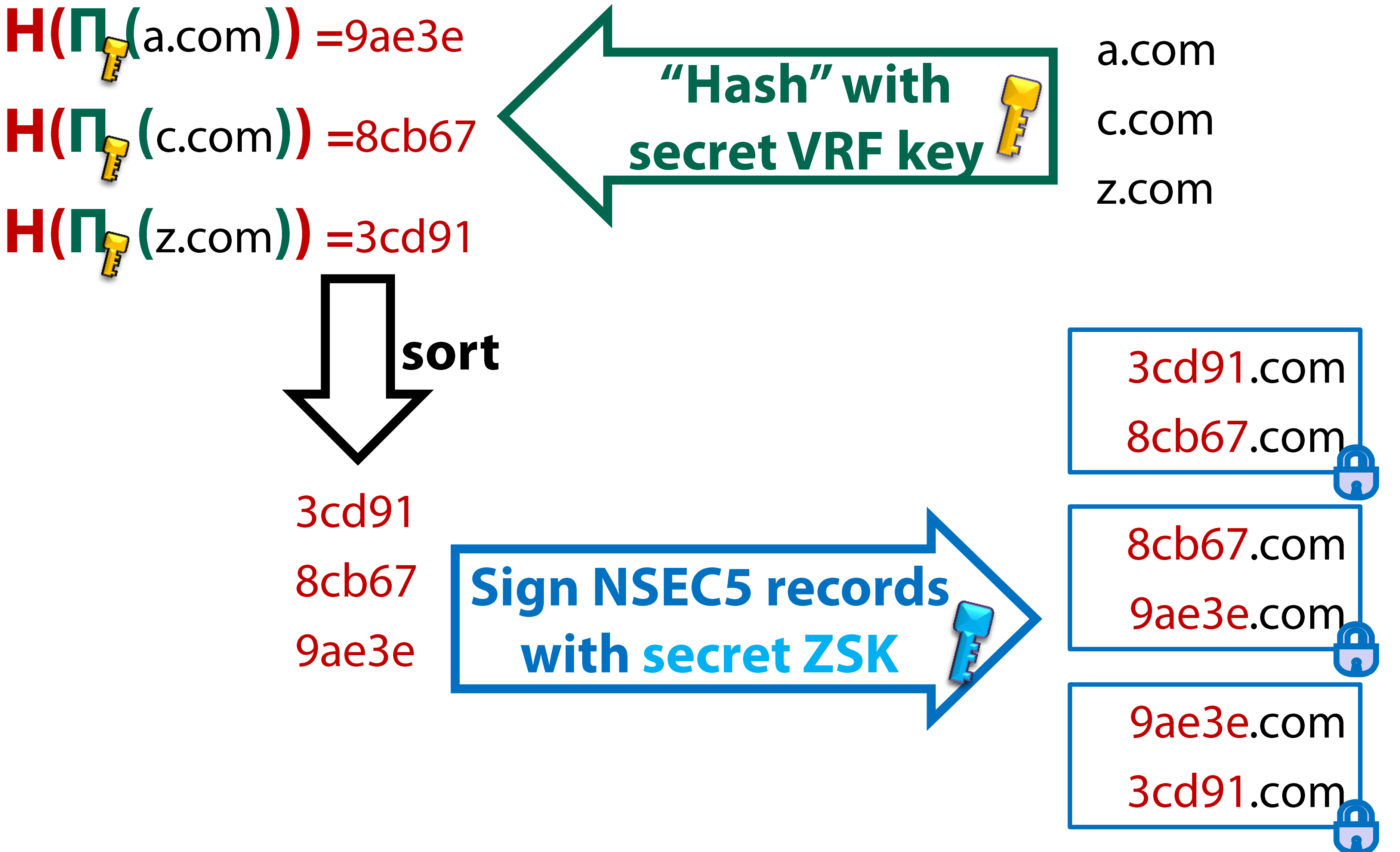
NSEC5: precomputing records



NSEC5: precomputing records



NSEC5: precomputing records



NSEC5: precomputing records

* **NSEC5-RSA:** **Π** is a deterministic RSA signature

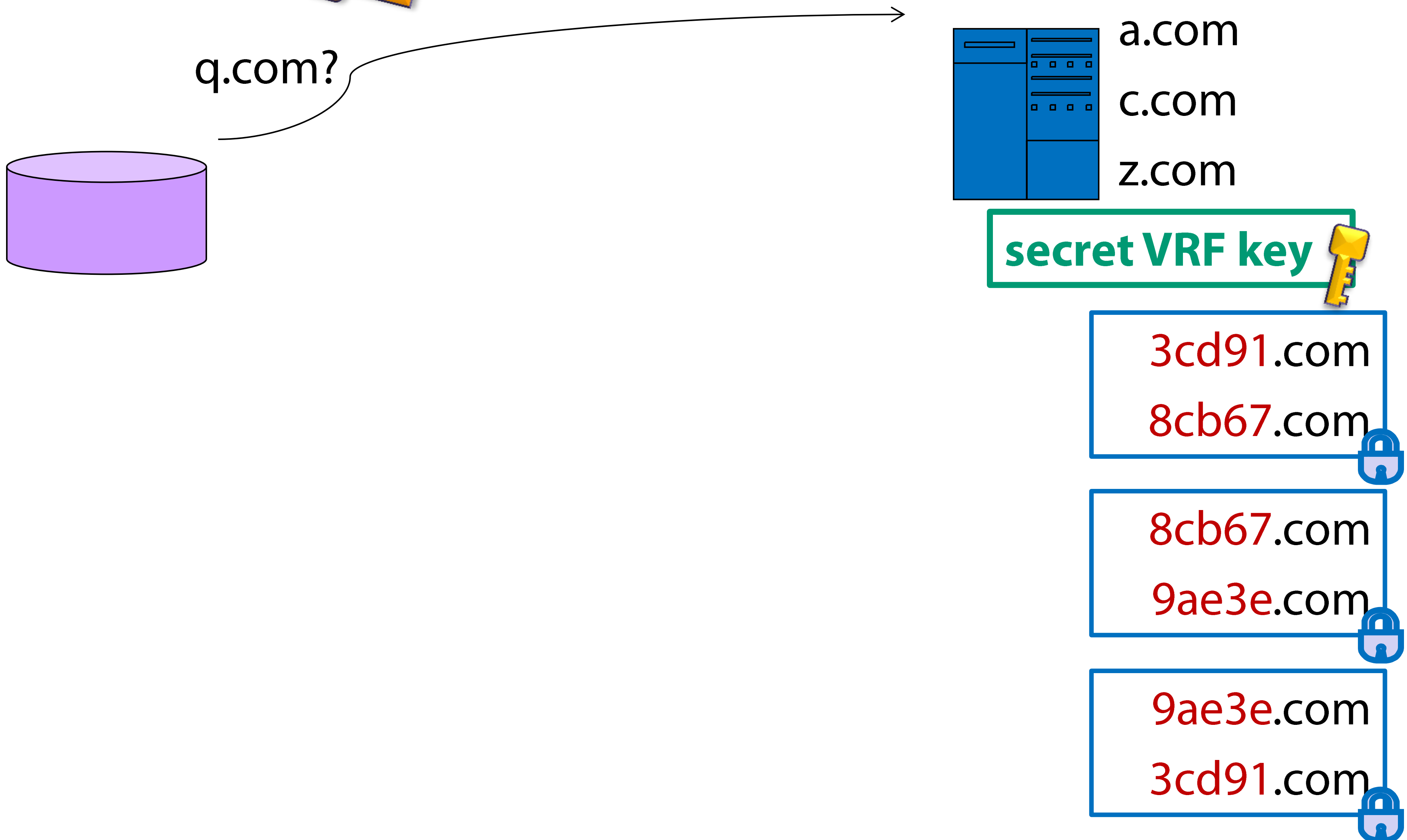
* **NSEC5-ECC:** new construction based on elliptic curves

- **Π** is implicit in **[Goh-Jareki'02][FranklinZhang'13]**
- We prove it's a VRF.
- For 256-bit elliptic curves, **Π** gives 641-bit outputs.

NSEC5 in action

Public Zone Signing Key (ZSK):

Public VRF Key:



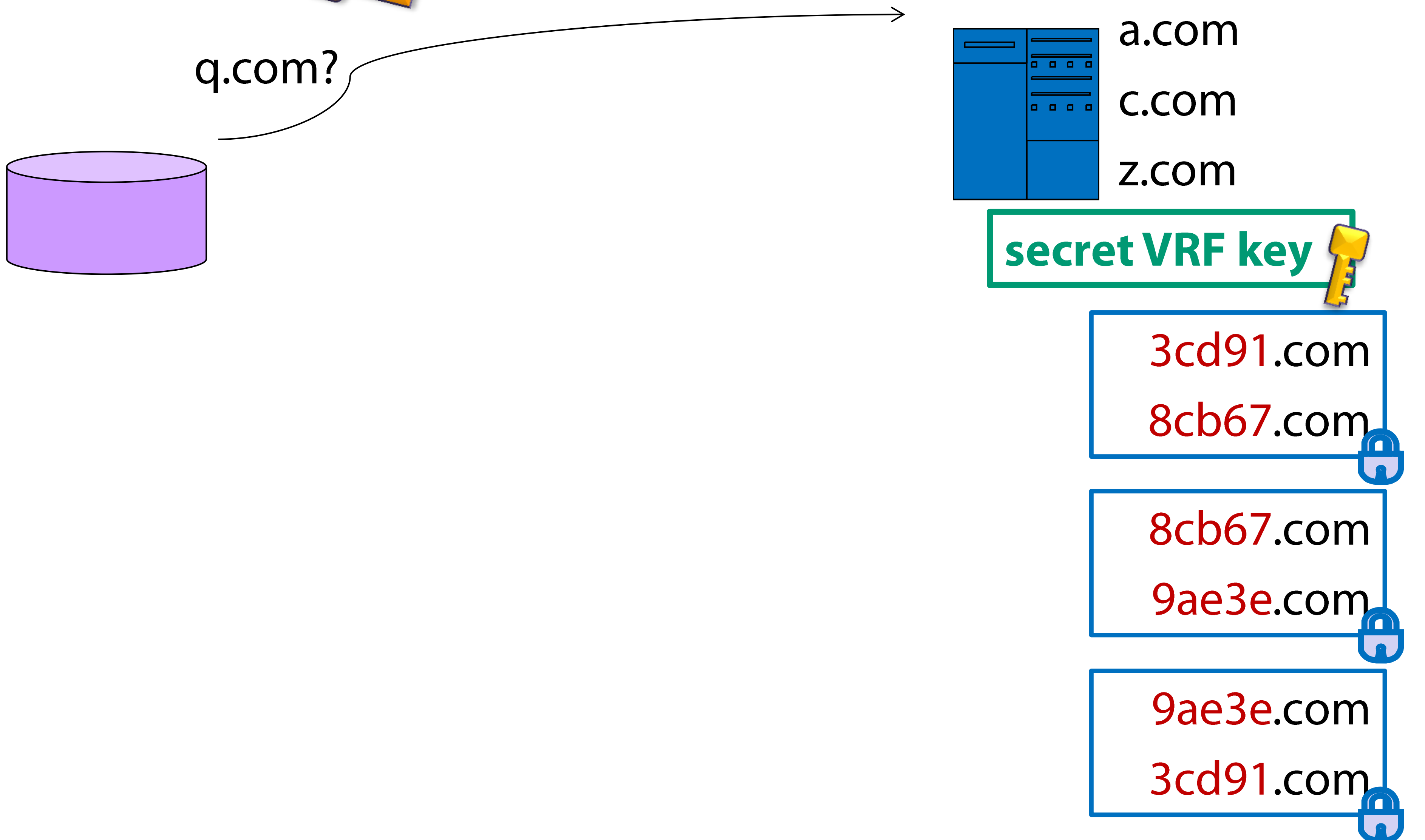
NSEC5 in action

Public Zone Signing Key (ZSK):

Public VRF Key:

$\Pi(\text{q.com}) = \text{aa8678}$

$H(\text{aa867}) = \text{7a89b}$



NSEC5 in action

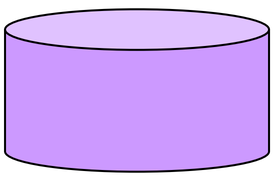
Public Zone Signing Key (ZSK):

Public VRF Key:

Π(q.com)=aa8678

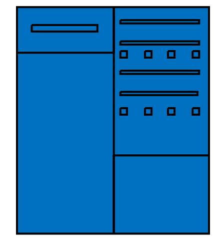
H(aa867)=7a89b

q.com?



3cd91.com

8cb67.com



a.com

c.com

z.com

secret VRF key

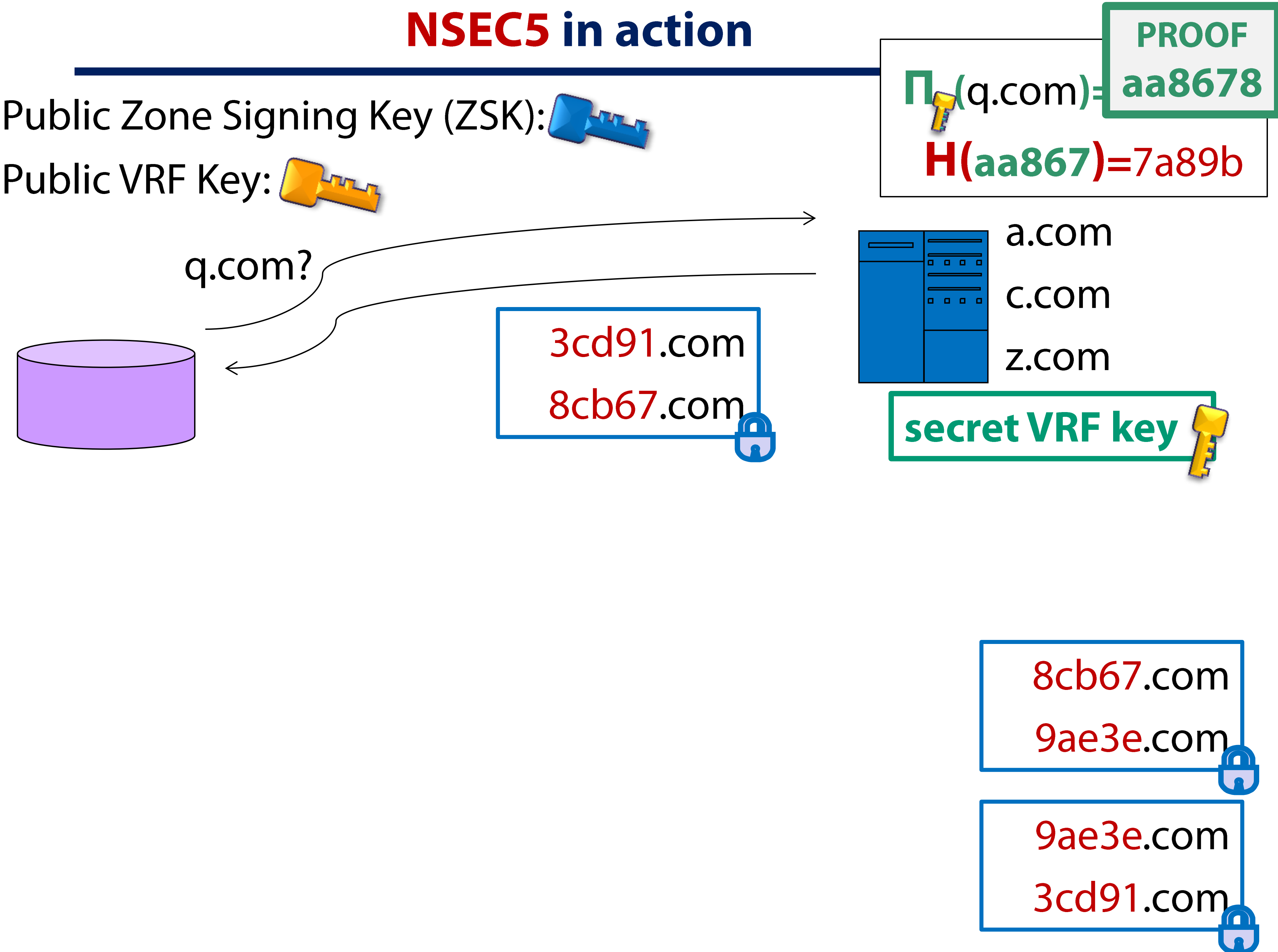
8cb67.com

9ae3e.com

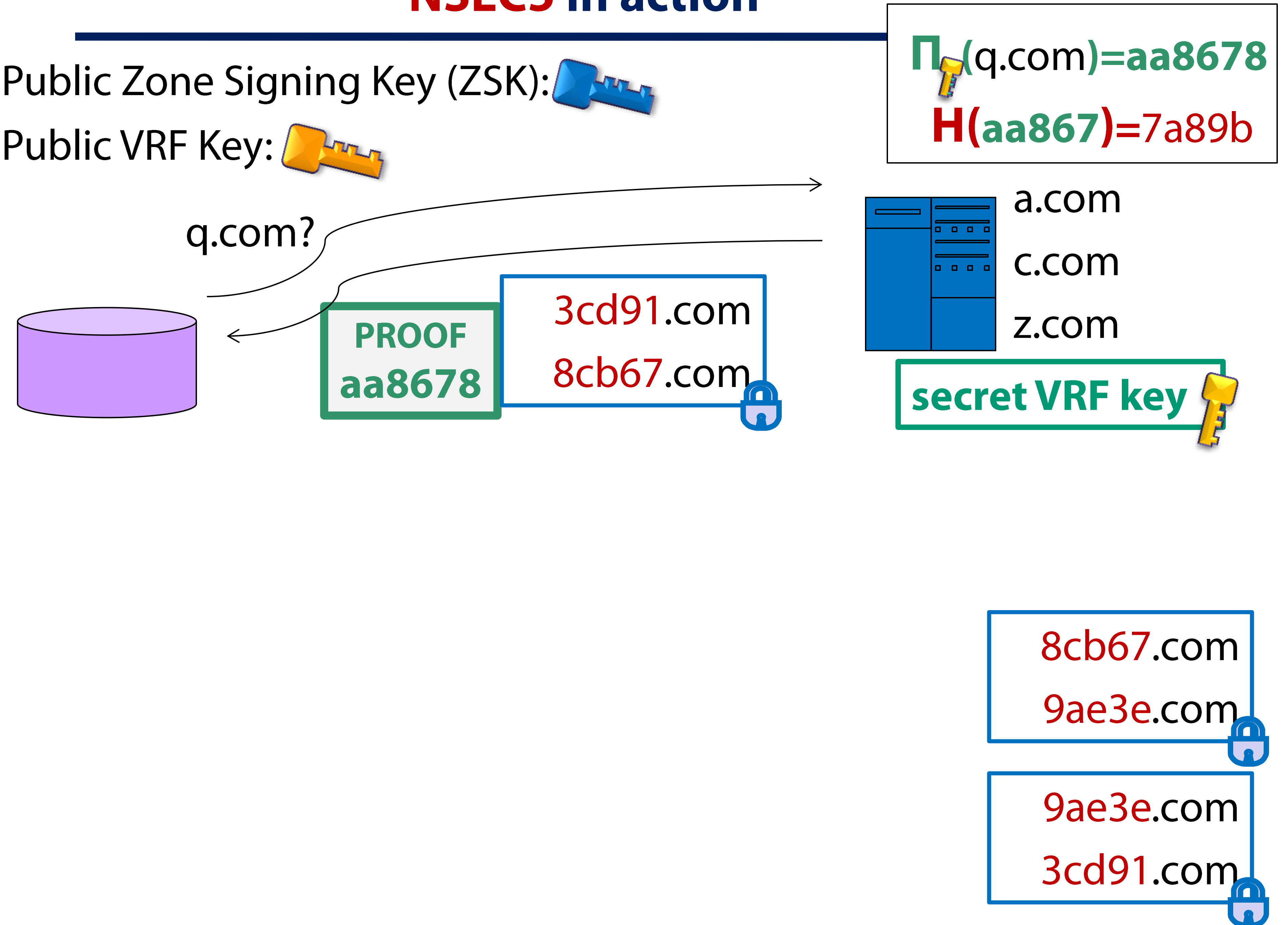
9ae3e.com

3cd91.com

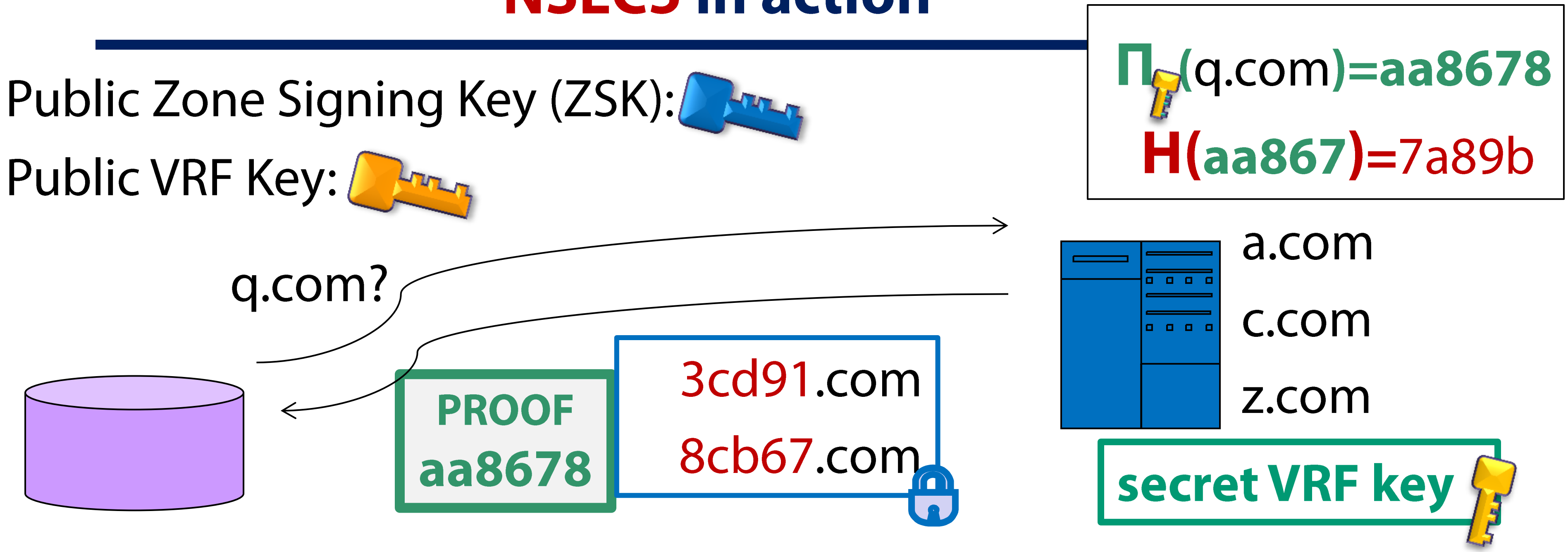
NSEC5 in action



NSEC5 in action



NSEC5 in action



To verify:

Does NSEC5 cover PROOF?

3cd19 < H(aa8678) < 8cb67

Does PROOF match query?

VER(q.com, aa8678)

With **NSEC5-RSA** this is just an RSA verification

comparison of different schemes

	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

Necessary to prevent zone enumeration & have integrity

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

Show proof

NSEC5 implementation*

Knot DNS & **Unbound**

authoritative nameserver — recursive resolver

Two versions of NSEC5:

1. NSEC5-RSA from **[NDSS'15]**
   - The VRF proof is a deterministic RSA signature (2048 bits)
2. New NSEC5-ECC:
   - For 256-bit elliptic curves, the VRF proof is 641 bits.

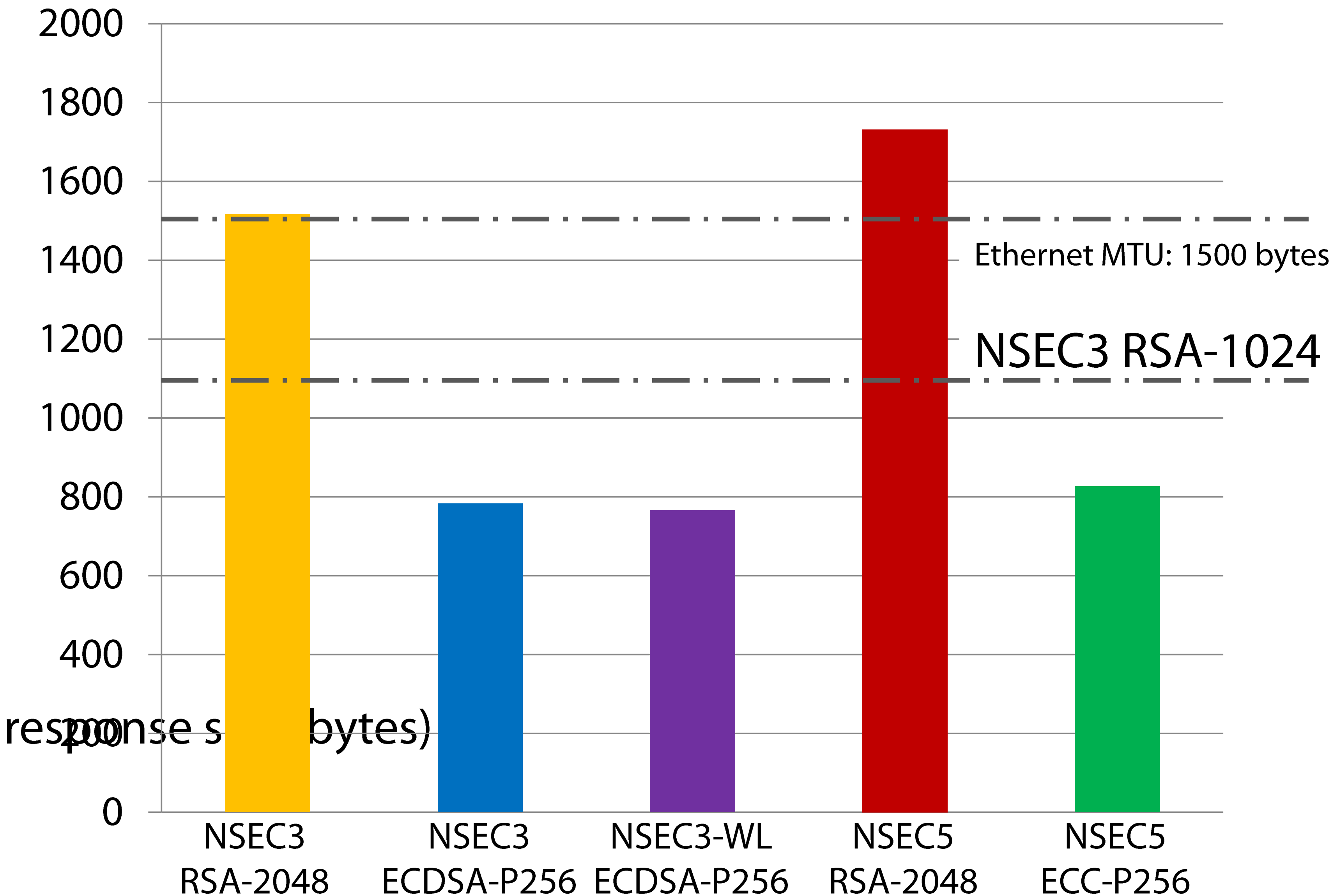
We use unstandardized optimizations developed for NSEC3

1. The wildcard bit **[GiebenMekking'12]**
2. Precomputed closest encloser proofs

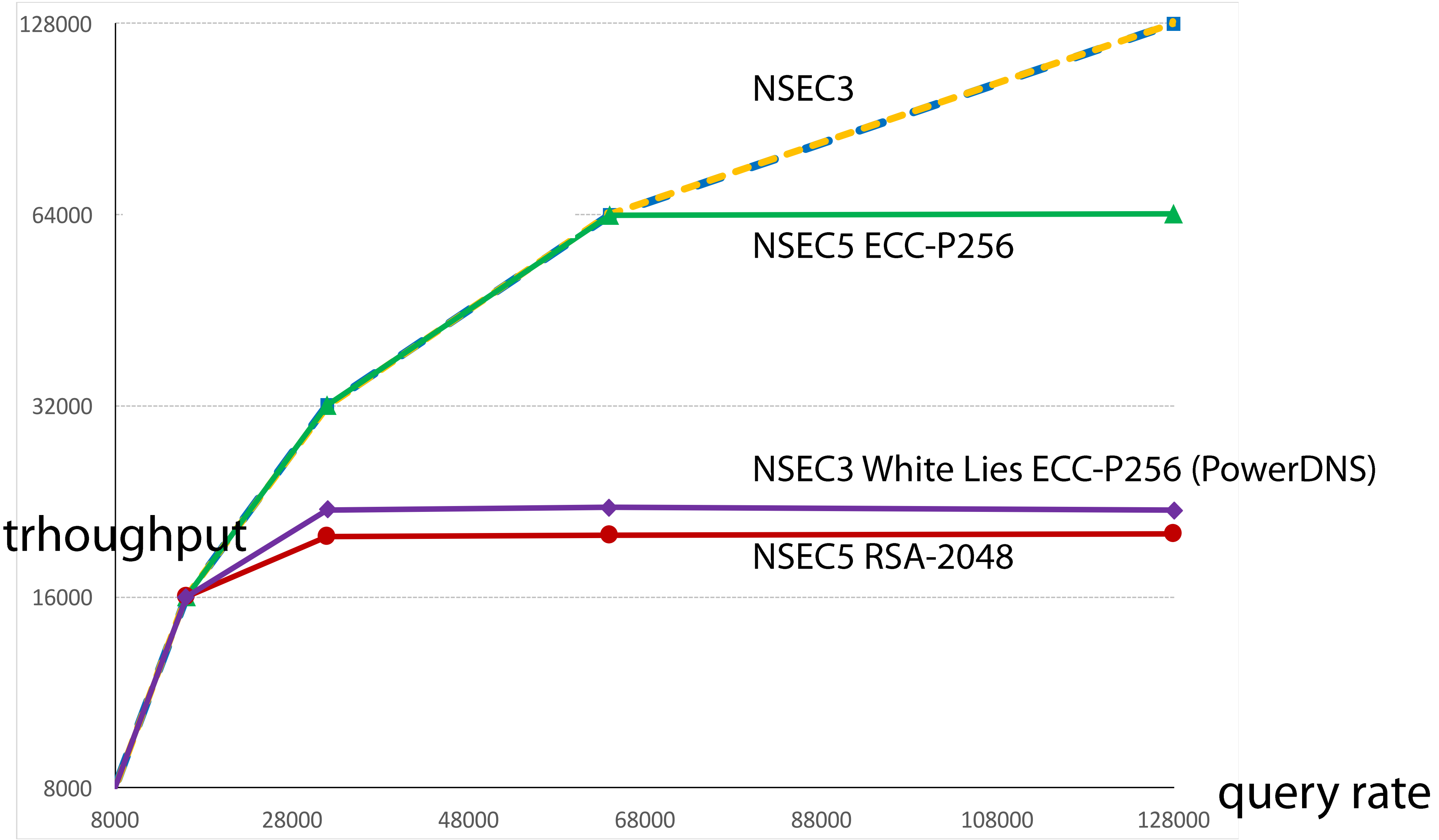
9K Lines of Code, no new libraries (openSSL) or system optimizations

*** Work done while on internship at Verisign Labs**

empirical measurement of NXDOMAIN response sizes



nameserver query throughput (pure NXDOMAIN traffic)



Machine specs: 20X 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

NSEC5 project resources

Full results in our new tech report (Feb 2017)
https://ia.cr/2017/099

Project page: https://www.cs.bu.edu/~goldbe/papers/nsec5.html
Internet Draft: https://datatracker.ietf.org/doc/draft-vcelak-nsec5/

Implementation coming soon.

Anonymous posts (not from our team!) from
http://dnsreactions.tumblr.com/

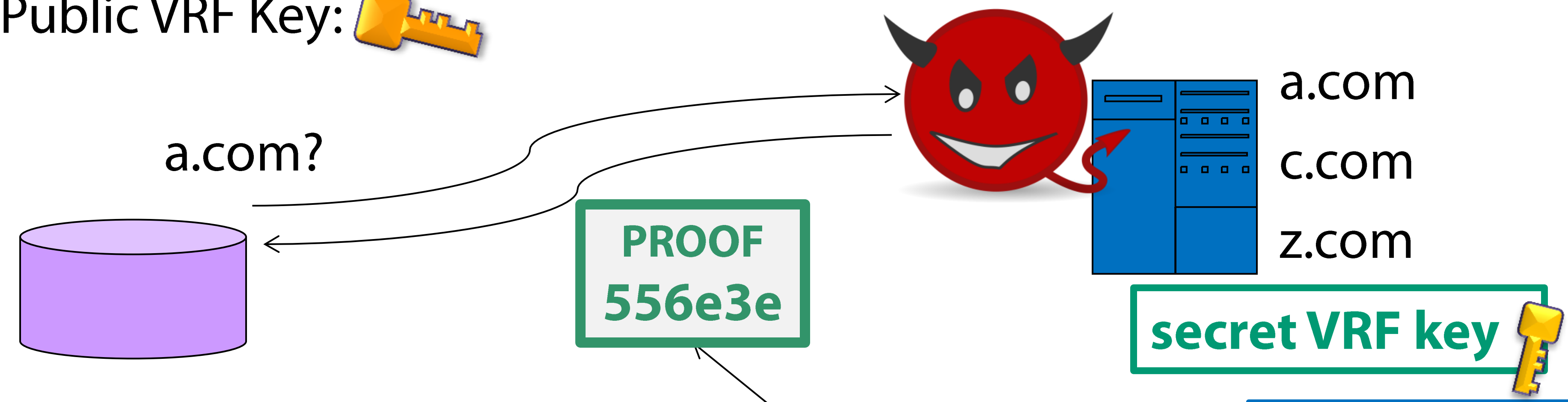
Hearing about NSEC5

When I finally grasp NSEC5

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

back to talk

Public parameters. Let q be a prime number, Z_q be the integers modulo q, $Z_q^* = Z_q - \{0\}$, and let G a cyclic group of prime order q with generator g. We assume that q, g and G are public parameters of our scheme. Let H_1 be a hash function (modeled as a random oracle) mapping arbitrary-length bitstrings onto the cyclic group G. (See Appendix A for a suggested instantiation of H_1.) Let H_3 be a hash function (modeled as a random oracle) mapping arbitrary-length bitstrings to fixed-length bitstrings. We can use any secure cryptographic function for H_3; in fact, we need only the first ℓ bits of its output for ℓ-bit security. Let H_2 be a function that takes the bit representation of an element of G and truncates it to the appropriate length; we need a 256 bit output for 128-bit security.

Keys. The secret VRF key $x \in Z_q$ is chosen uniformly at random. The public VRF key is g^x.

Hashing. Given the secret VRF key x and input α, compute the proof π as:

1. Obtain the group element $h = H_1(\alpha)$ and raise it to the power of the secret key to get $\gamma = h^x$.
2. Choose a nonce $k \in Z_q$.
3. Compute $c = H_3(g, h, g^x, h^x, g^k, h^k)$.
4. Let $s = k - cx \mod q$.

The proof π is the group element γ and the two exponent values c, s. (Note that c may be shorter than a full-length exponent, because its length is determined by the choice of H_3). The VRF output $\beta = F_{SK}(\alpha)$ is computed by truncating γ with H_2. Thus

$$\pi = (\gamma, c, s) \qquad\qquad \beta = H_2(\gamma)$$

Notice that anyone can compute β given π.

Verifying. Given public key g^x , verify that proof π corresponds to the input α and output β as follows:

1. Given public key g^x, and exponent values c and s from the proof π, compute $u = (g^x)^c \cdot g^s$. Note that if everything is correct then $u = g^k$.
2. Given input α, hash it to obtain $h = H_1(\alpha)$. Make sure that $\gamma \in G$. Use h and the values (γ, c, s) from the proof to compute $v = (\gamma)^c \cdot h^s$. Note that if everything is correct then $v = h^k$.
3. Check that hashing all these values together gives us c from the proof. That is, given the values u and v that we just computed, the group element γ from the proof, the input α, the public key g^x and the public generator g, check that:

$$c = H_3(g, H_1(\alpha), g^x, \gamma, u, v)$$

Finally, given γ from the proof π, check that $\beta = H_2(\gamma)$.

Figure 2: An EC-based VRF for NSEC5. We use a multiplicative group notation. This VRF adapts the Chaum-Pederson protocol [28] for proving that two cyclic group elements g^x and h^x have the same discrete logarithm x base g and h, respectively.