DNS and its Security

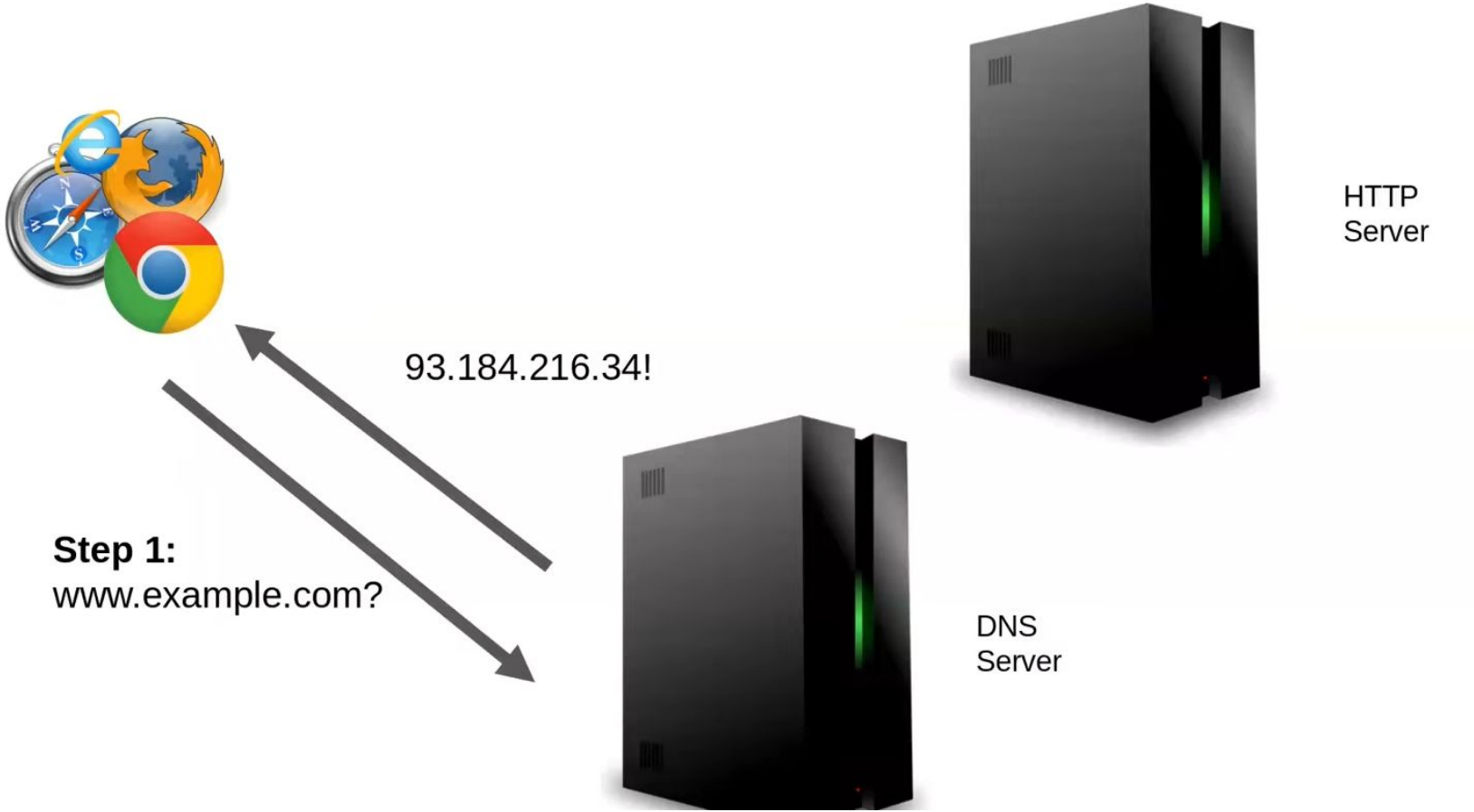
CSE 468: Computer Network Security
(November 14, 2023)
Sana Habib

Outline

- DNS Basics
- DNS Security
- Object Security
- Path Security
- Some exercises (if time permits)

What the novice Internet user sees?

What a Techy Internet user sees?



Step 2:
HTTP to 93.184.216.34

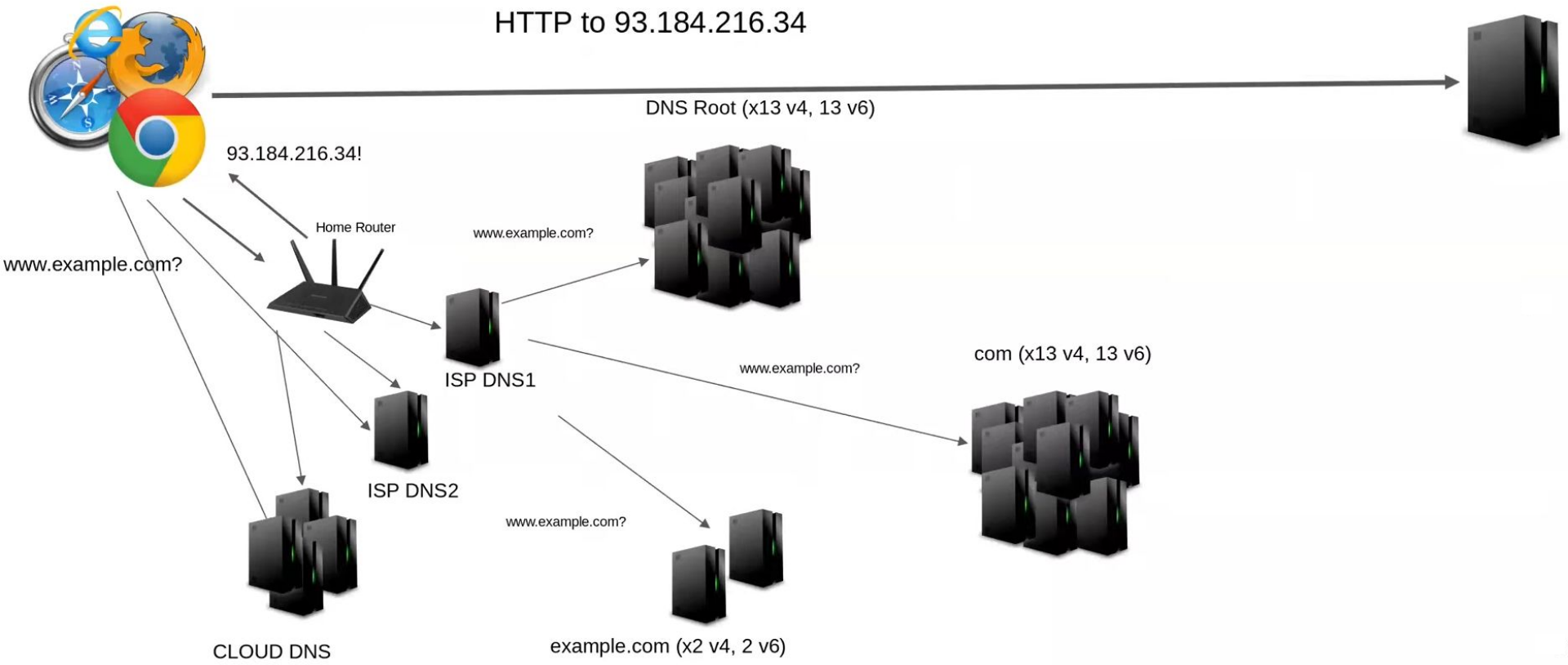
HTTP
Server

93.184.216.34!

Step 1:
www.example.com?

DNS
Server

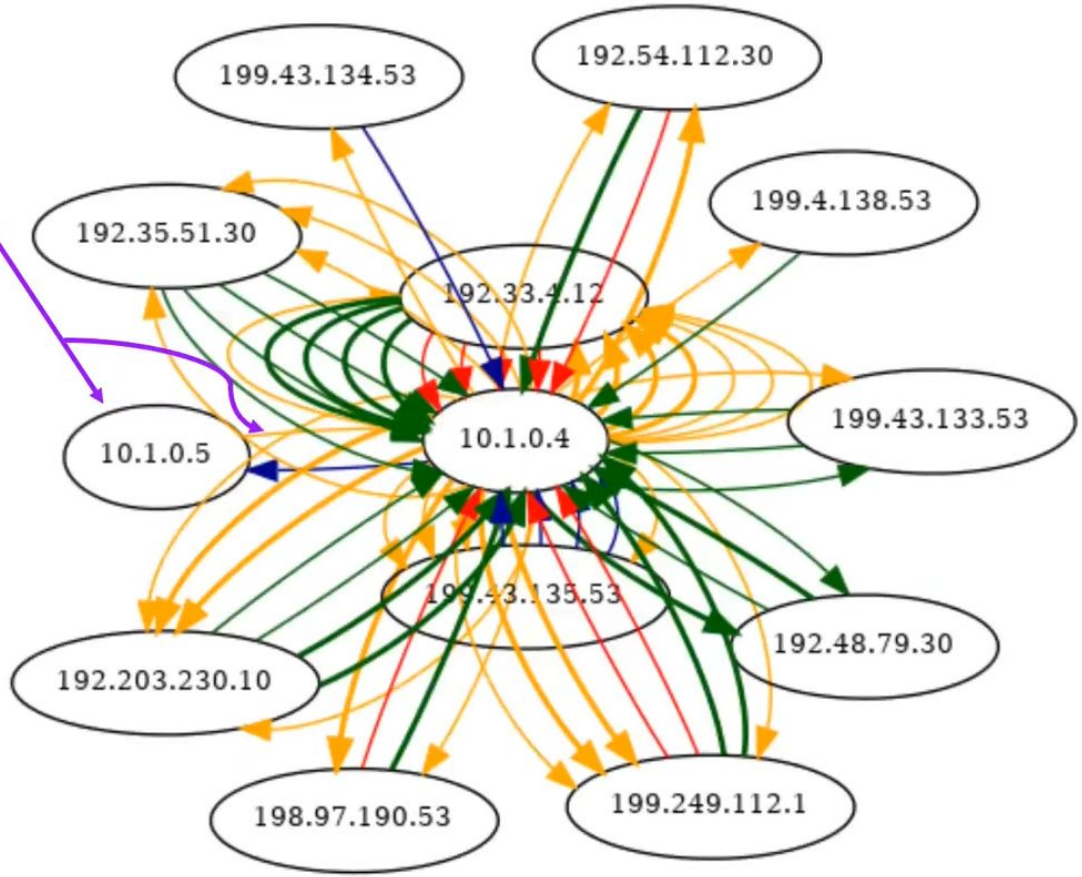
DNS is much much more complex!



The example.com web page

You make a single request

- Each line is a DNS request or response
- The center node is an ISP resolver

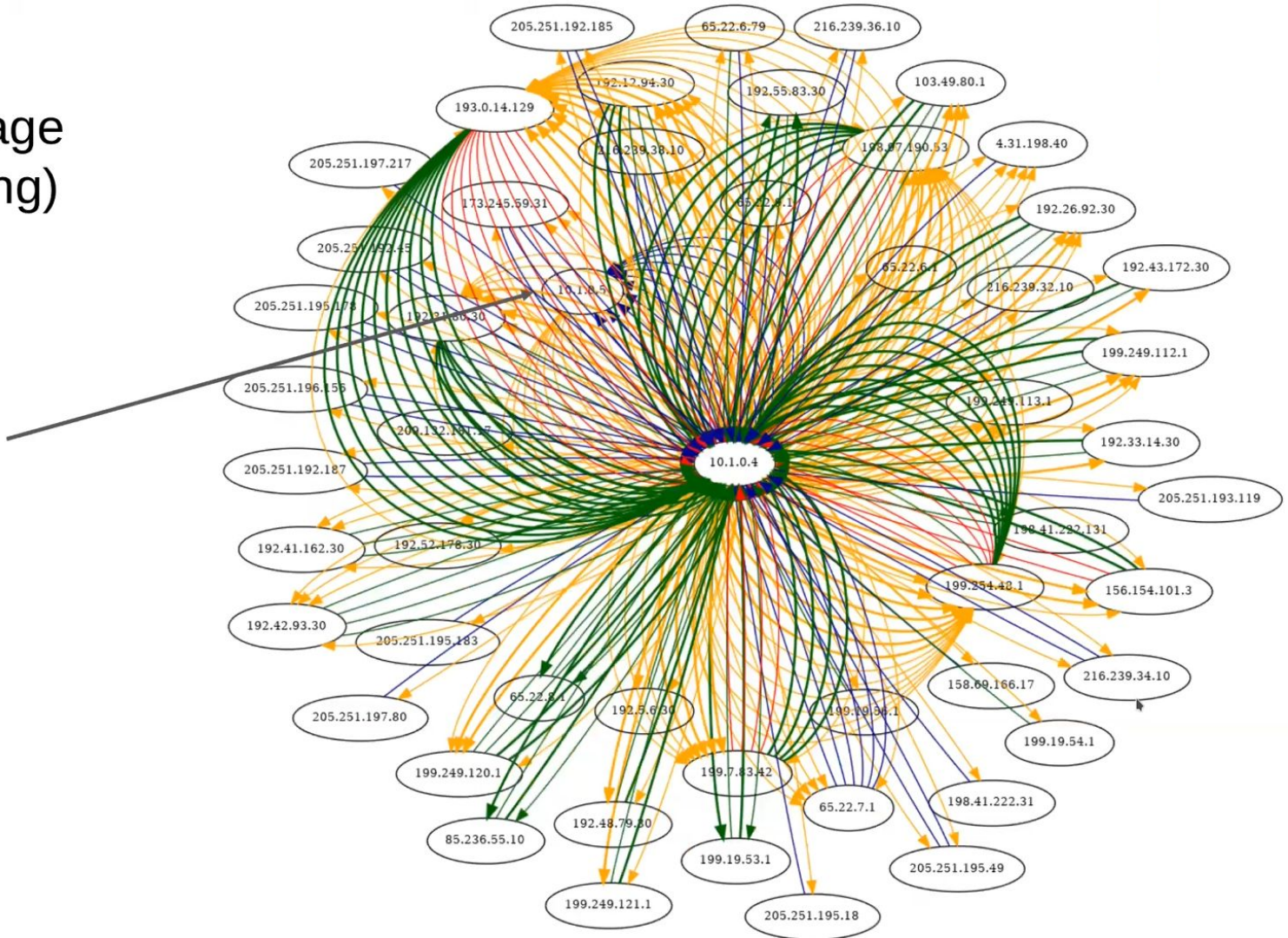


Query
Authoratative/DNSSEC

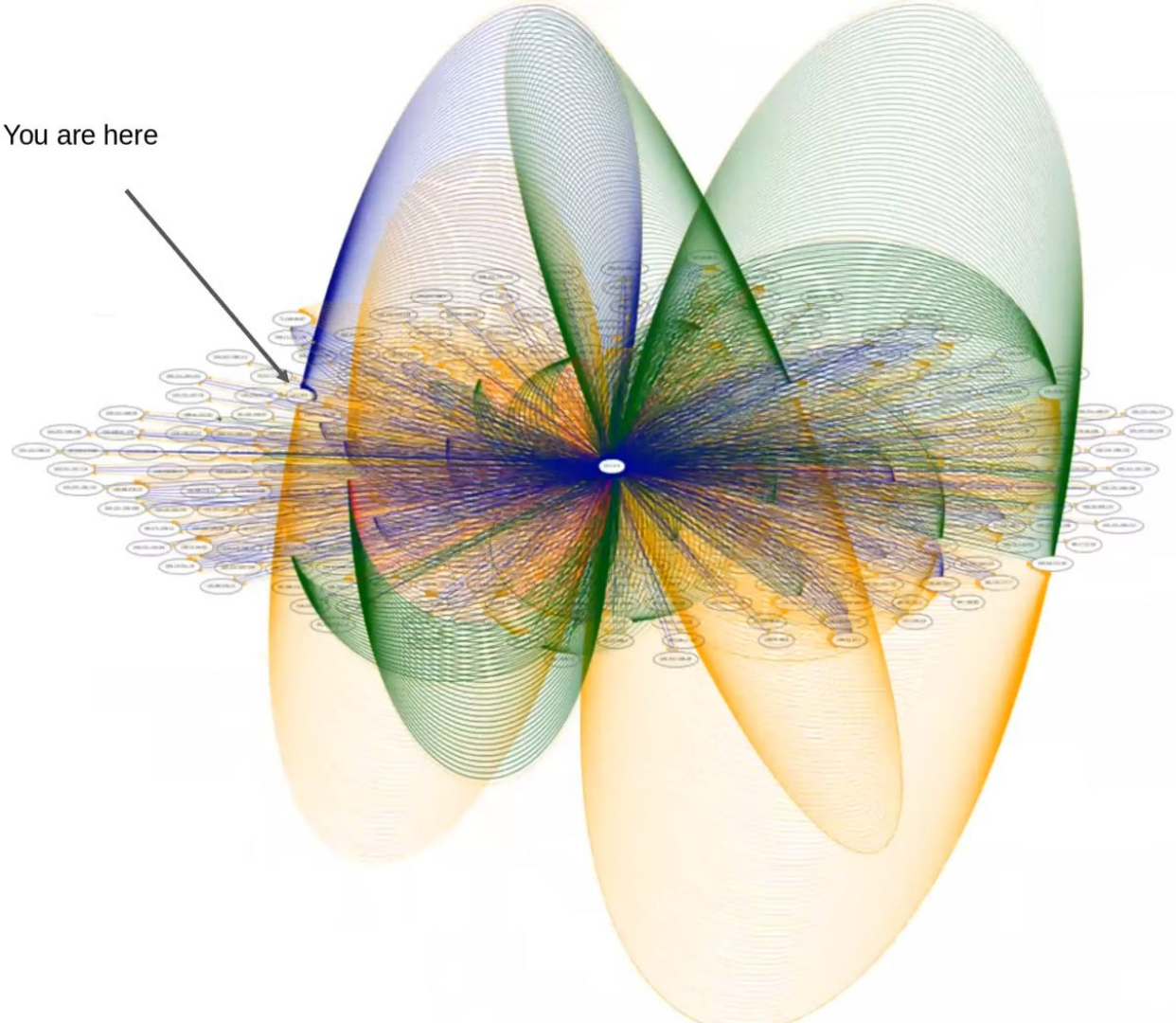
Truncated
Response

ietf.org web page (without caching)

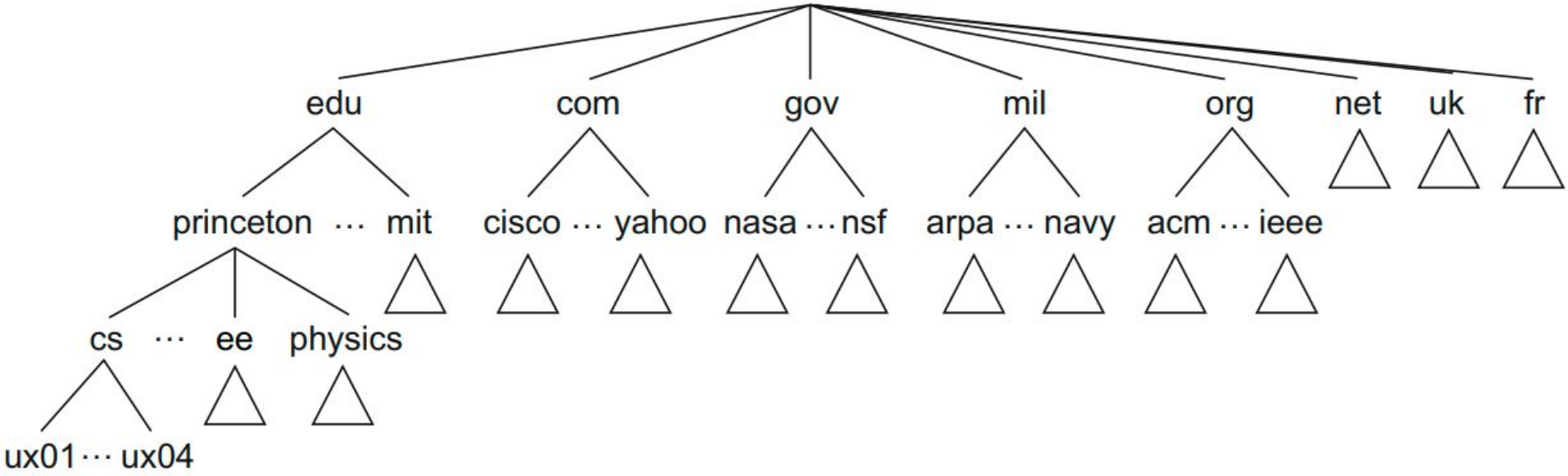
You are here



webmd.com



Example of Domain Hierarchy



https://cseweb.ucsd.edu/classes/wi19/cse124-a/courseoverview/compnetworks.pdf

What are parts of domain?

https://www.google.com,

Protocol, Subdomain, Domain Name, Top-level Domain

Root Domain (includes domain name and top level domain)

How about https://www.amazon.com/ and https://leetcode.com/?

Who maintains domains?

The Top Level Domain (TLD) or Top Level Zone is maintained by the Internet Corporation for Assigned Names and Numbers (ICANN) https://www.icann.org/

https://cseweb.ucsd.edu/classes/wi19/cse124-a/courseoverview/compnetworks.pdf

How DNS records are kept? Resource Records (RR)

Each name server implements the zone information as a collection of Resource Records (RR).

RR is a 5-tuple: <Name, Value, Type, Class, TTL>

Type → A, NS, CNAME, MX.

A → indicates value is an IP address.

NS → The value field gives the domain name for a host that is running a name server that knows how to resolve names within the specified domain.

CNAME—The value field gives the canonical name for a particular host; it is used to define aliases.

MX—The value field gives domain name for a host (running a mail server).

TTL → time to live shows how long a resource record is valid.

Linux Commands:

dig (domain information groper) → used for retrieving information about domain name servers.

nslookup (name server lookup) → allows you to query DNS service.

host → look up a variety of information available through the domain name system.

dog → a command-line DNS client for looking up DNS records for domain names.

whois → is a protocol that is used to look up information on domain names and IP addresses.

```
sana@pop-os:~$ dig google.com

; <<>> DiG 9.18.12-0ubuntu0.22.04.2-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 51819
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             221     IN      A       142.250.138.100
google.com.             221     IN      A       142.250.138.138
google.com.             221     IN      A       142.250.138.101
google.com.             221     IN      A       142.250.138.113
google.com.             221     IN      A       142.250.138.102
google.com.             221     IN      A       142.250.138.139

;; Query time: 3 msec
;; SERVER: 127.0.0.53#53(127.0.0.53) (UDP)
;; WHEN: Wed Sep 20 01:57:20 PDT 2023
;; MSG SIZE  rcvd: 135
```

The first field is the NAME: The domain name being returned.

The second field (221) is the TTL in seconds.

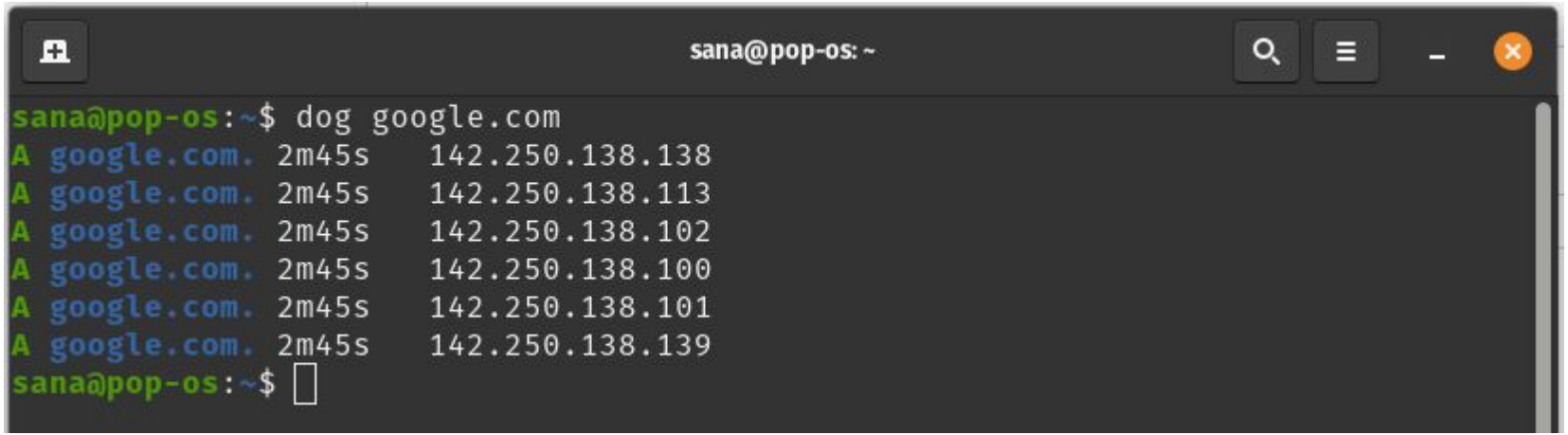
IN is the CLASS. Here, IN stands for Internet.

A is the TYPE. Here, A stands for mapping a domain name to an IPv4 address.

The last field is the IP address.

```
sana@pop-os:~$ host google.com
google.com has address 142.250.113.139
google.com has address 142.250.113.100
google.com has address 142.250.113.101
google.com has address 142.250.113.102
google.com has address 142.250.113.113
google.com has address 142.250.113.138
google.com has IPv6 address 2607:f8b0:4023:1000::64
google.com has IPv6 address 2607:f8b0:4023:1000::8b
google.com has IPv6 address 2607:f8b0:4023:1000::66
google.com has IPv6 address 2607:f8b0:4023:1000::65
google.com mail is handled by 10 smtp.google.com.
sana@pop-os:~$
```

```
sana@pop-os:~$ nslookup google.com
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.113.139
Name:   google.com
Address: 142.250.113.113
Name:   google.com
Address: 142.250.113.101
Name:   google.com
Address: 142.250.113.100
Name:   google.com
Address: 142.250.113.102
Name:   google.com
Address: 142.250.113.138
Name:   google.com
Address: 2607:f8b0:4023:1000::65
Name:   google.com
Address: 2607:f8b0:4023:1000::64
Name:   google.com
Address: 2607:f8b0:4023:1000::8b
Name:   google.com
Address: 2607:f8b0:4023:1000::66
```

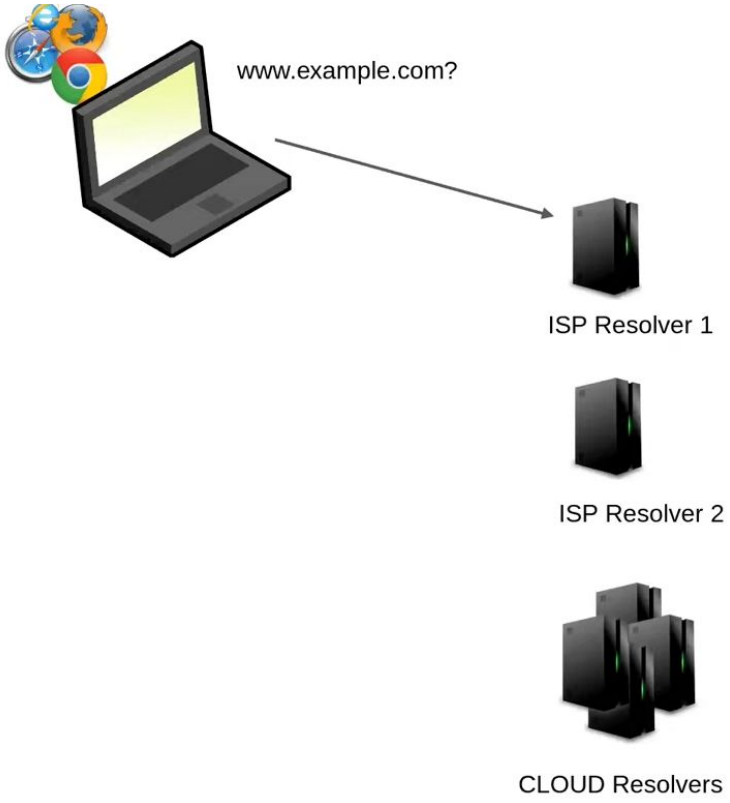

sana@pop-os: ~
sana@pop-os:~$ dog google.com
A google.com. 2m45s 142.250.138.138
A google.com. 2m45s 142.250.138.113
A google.com. 2m45s 142.250.138.102
A google.com. 2m45s 142.250.138.100
A google.com. 2m45s 142.250.138.101
A google.com. 2m45s 142.250.138.139
sana@pop-os:~$

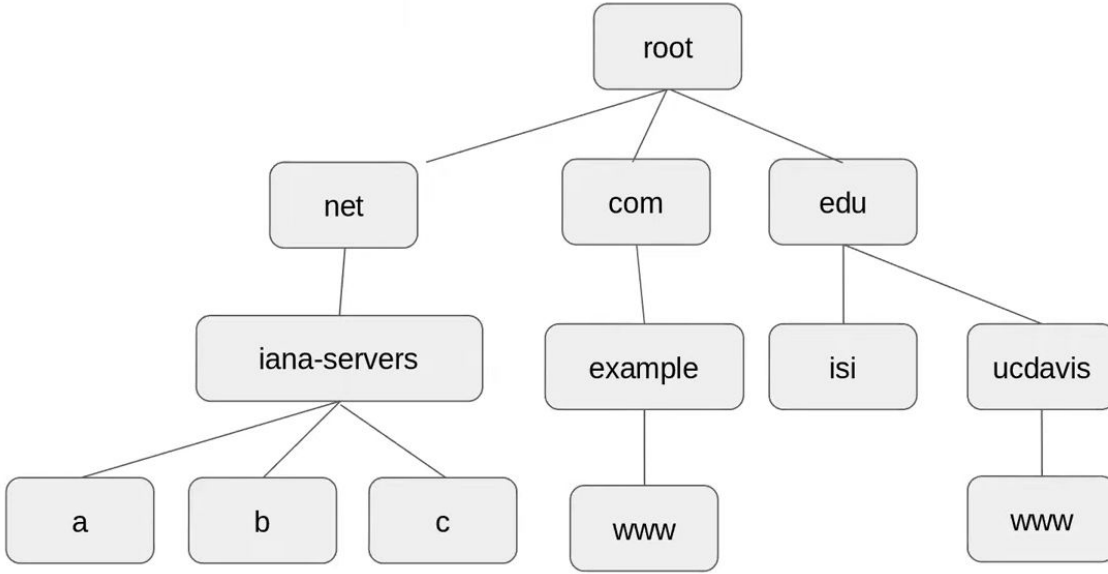
Subject Alternate Names (SAN)

Alternate domain names for the same website.

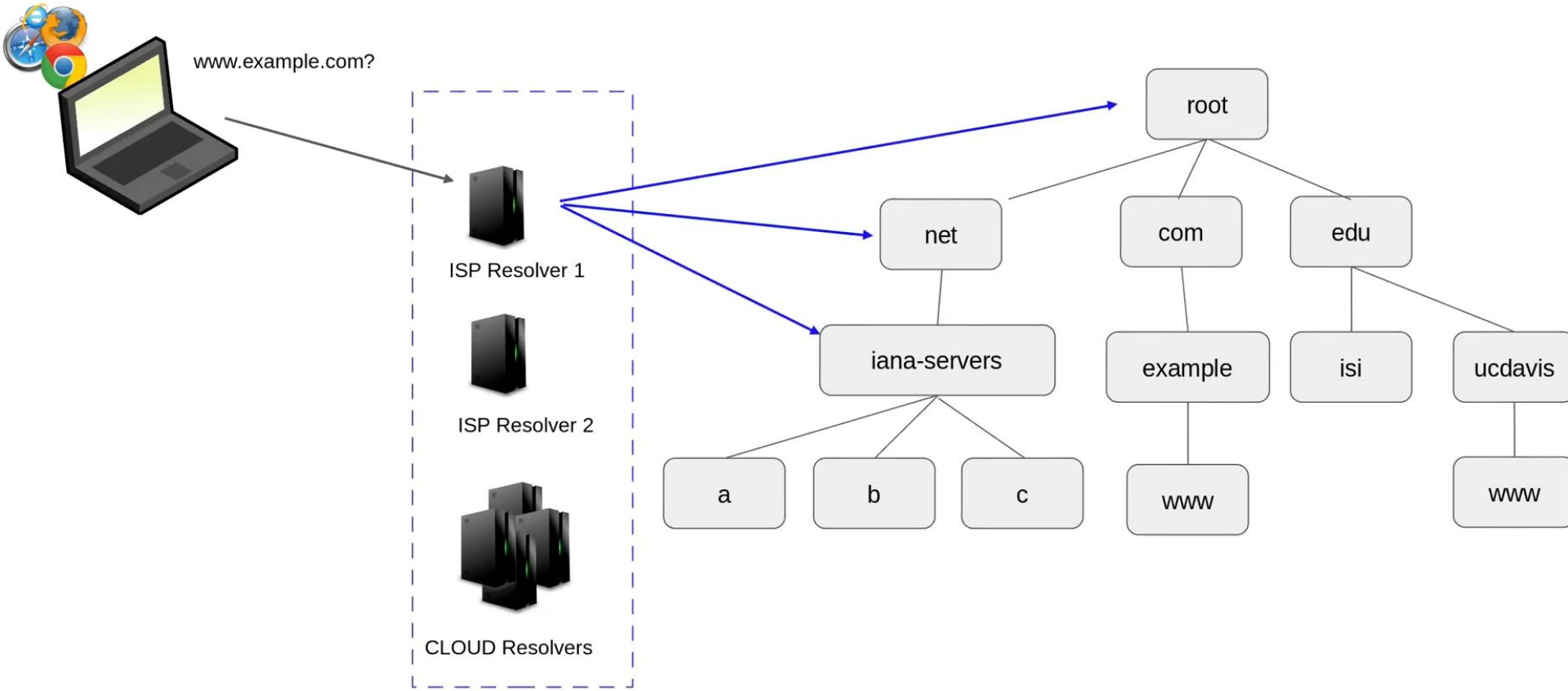
Exercise: What are the SANs for some of your favorite websites?

Resolvers do all the hard work!



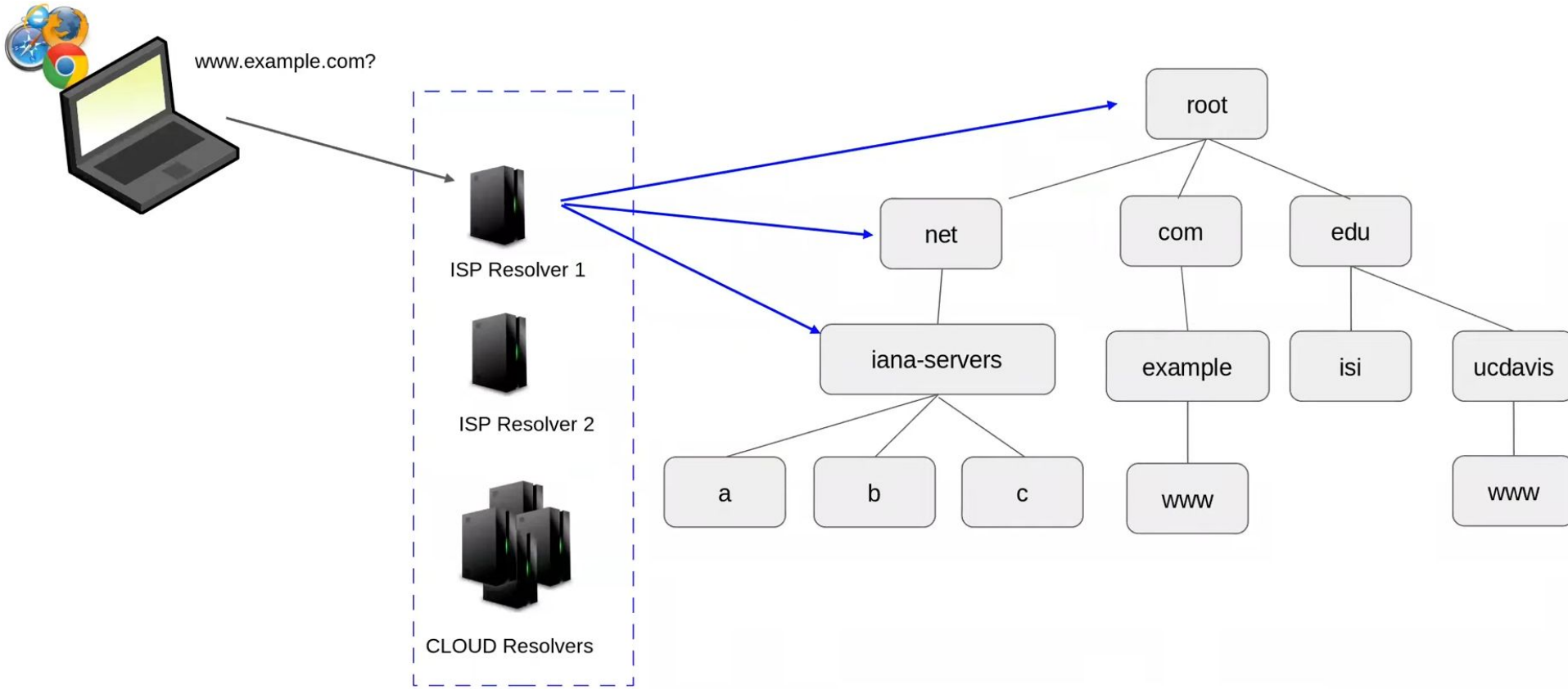


Resolvers do all the hard work!



Resolvers query the tree to find your answer

Resolvers do all the hard work!



Resolvers query the tree to find your answer

AND REMEMBER THIS ANSWER!

Resolvers do all the hard work!

www.example.com?

DNS resolver types:

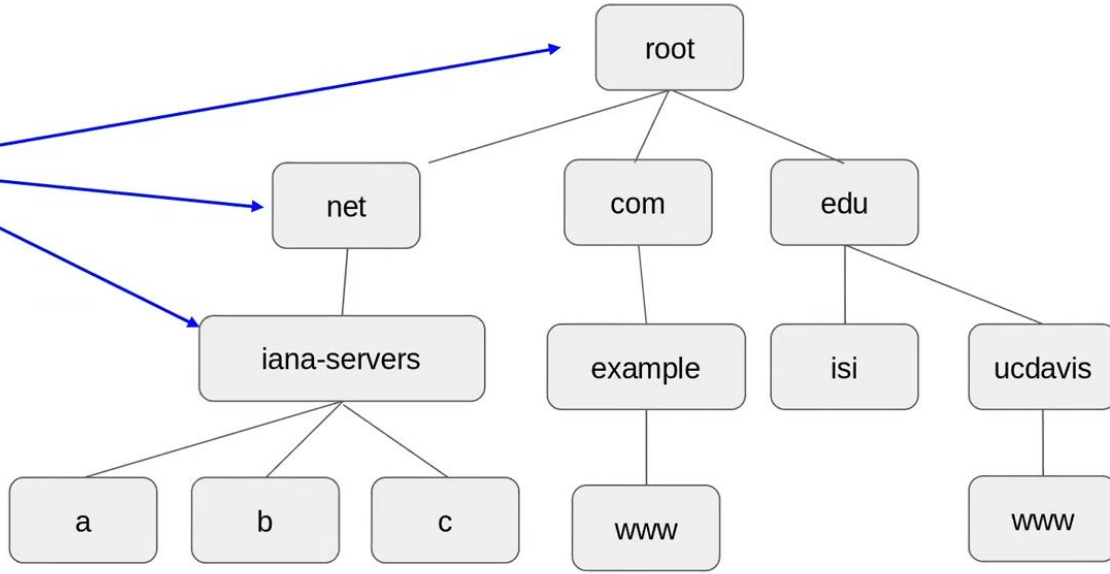
- Stub
- Recursive
- Forwarders
- Validating
- Pay Wall

(to be described later)

ISP Resolver 1

ISP Resolver 2

CLOUD Resolvers



Resolvers query the tree to find your answer

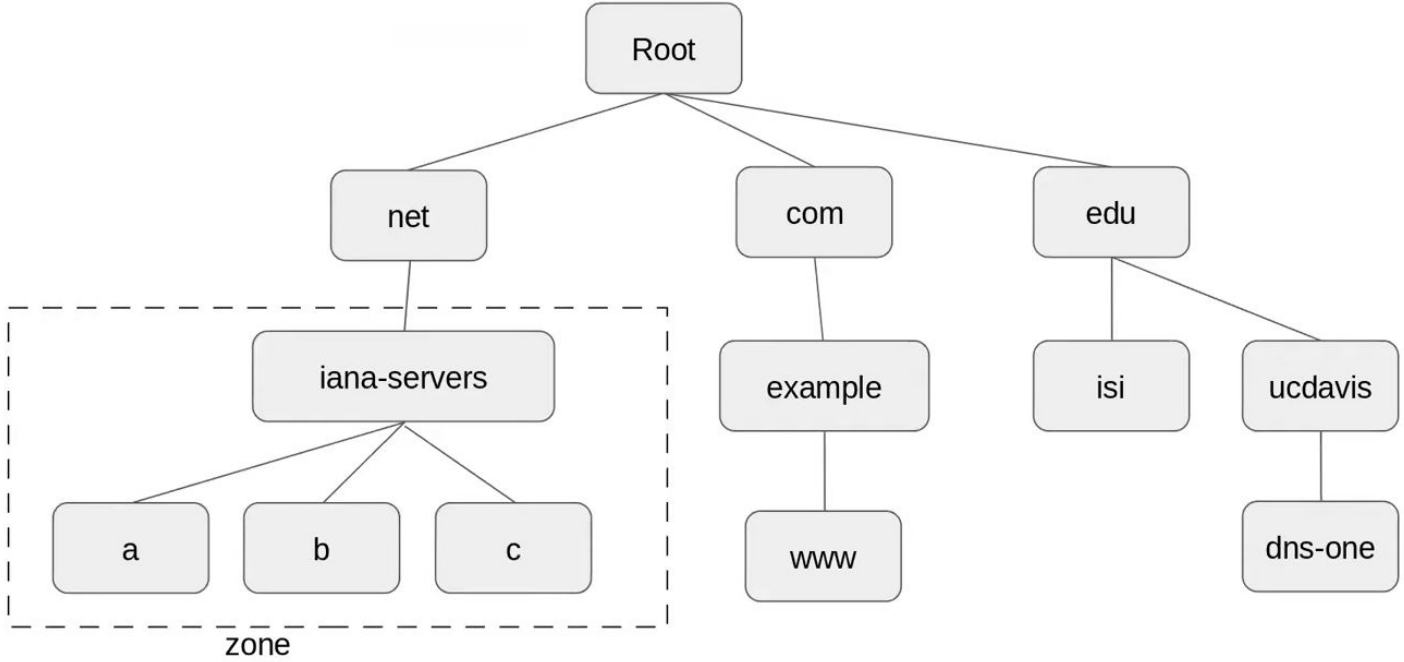
AND REMEMBER THIS ANSWER!

Structure of DNS…… The DNS ‘tree’

The Root (aka “.”)

Top Level Domains (TLDs)

Second Level Domains (SLDs)



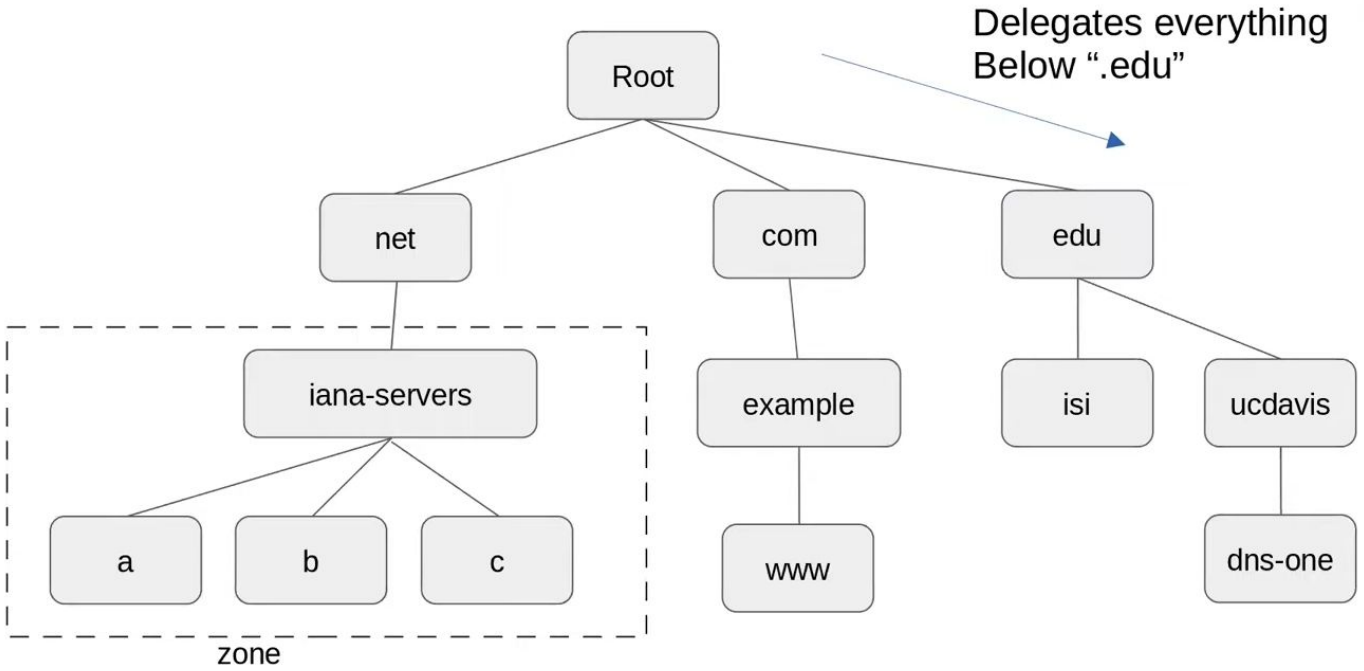
IMPORTANT: name server records in .net (13), .com (13), and .org (6) are not shown in these slides

Structure of DNS…… The DNS ‘tree’

The Root (aka “.”)

Top Level Domains (TLDs)

Second Level Domains (SLDs)



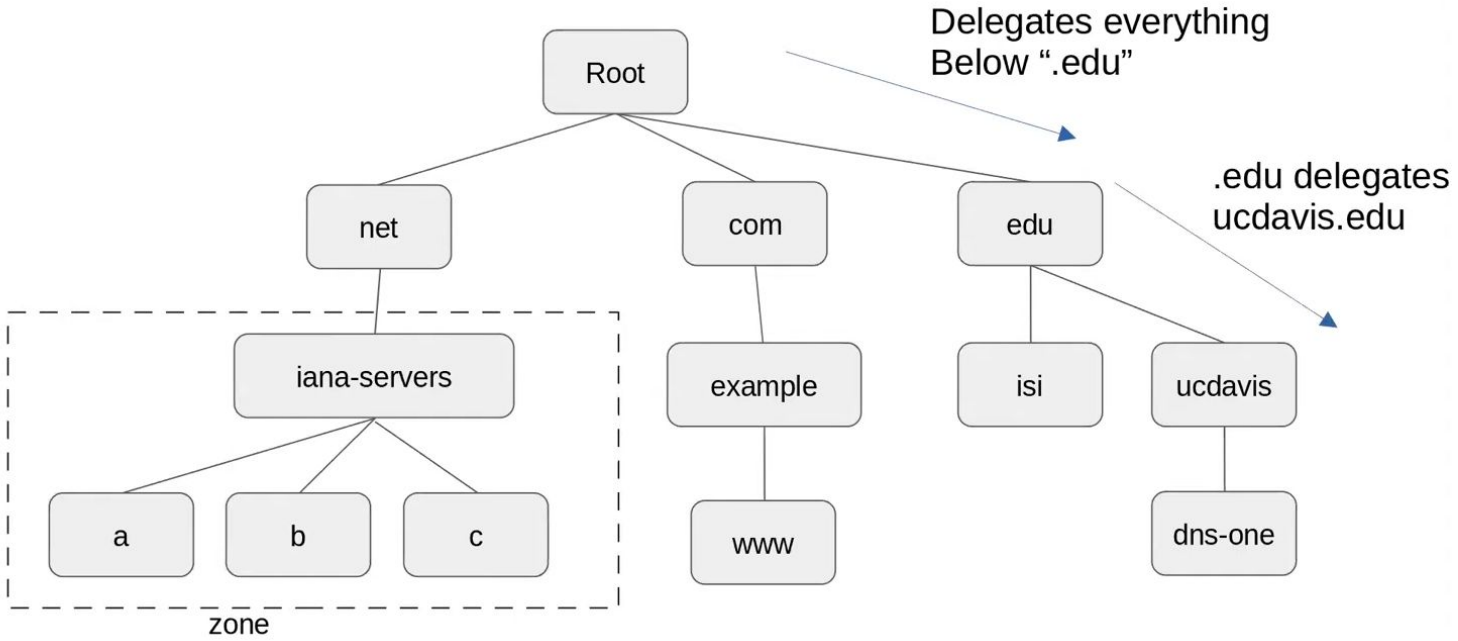
IMPORTANT: name server records in .net (13), .com (13), and .org (6) are not shown in these slides

Structure of DNS…… The DNS ‘tree’

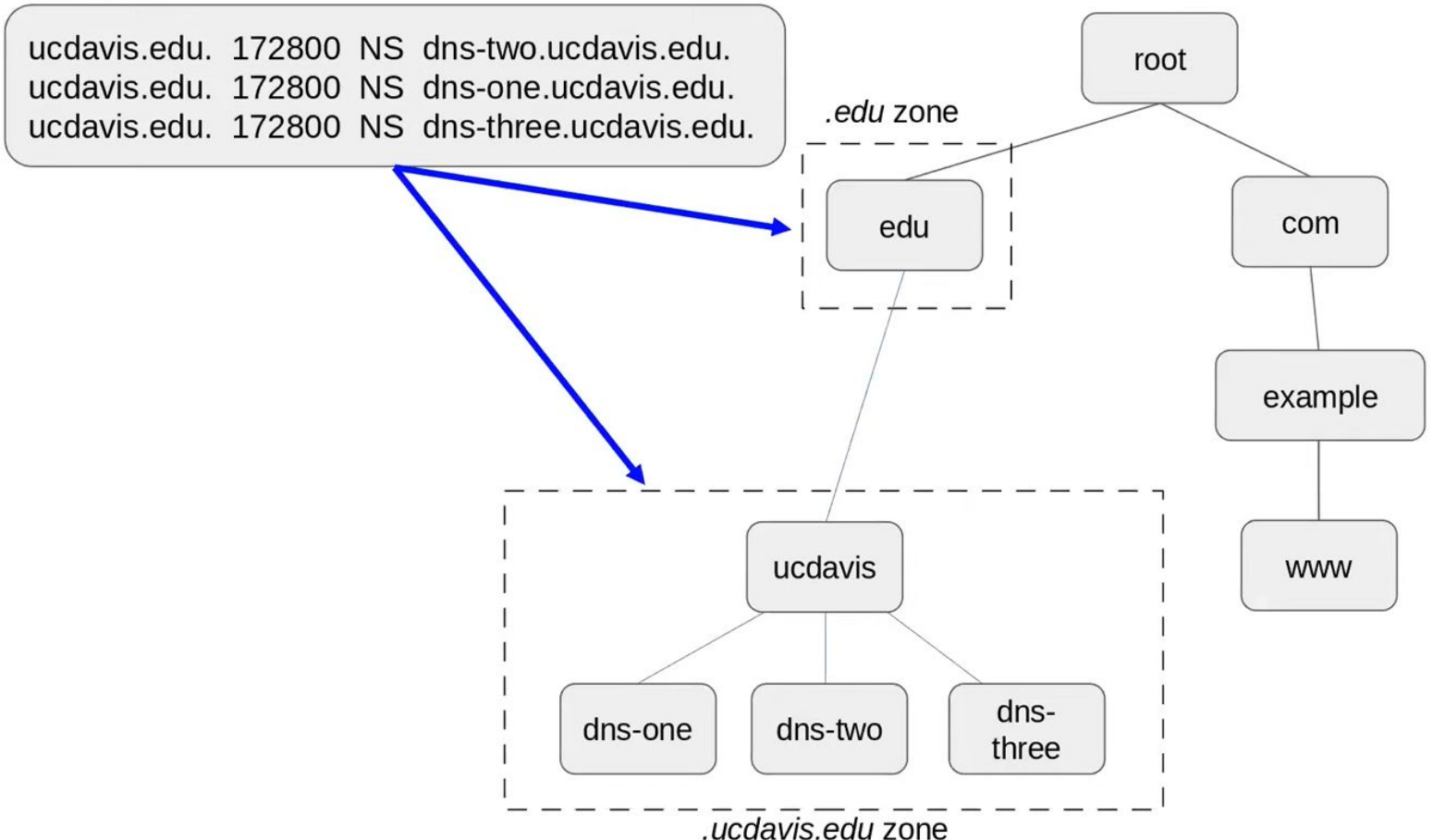
The Root (aka “.”)

Top Level Domains (TLDs)

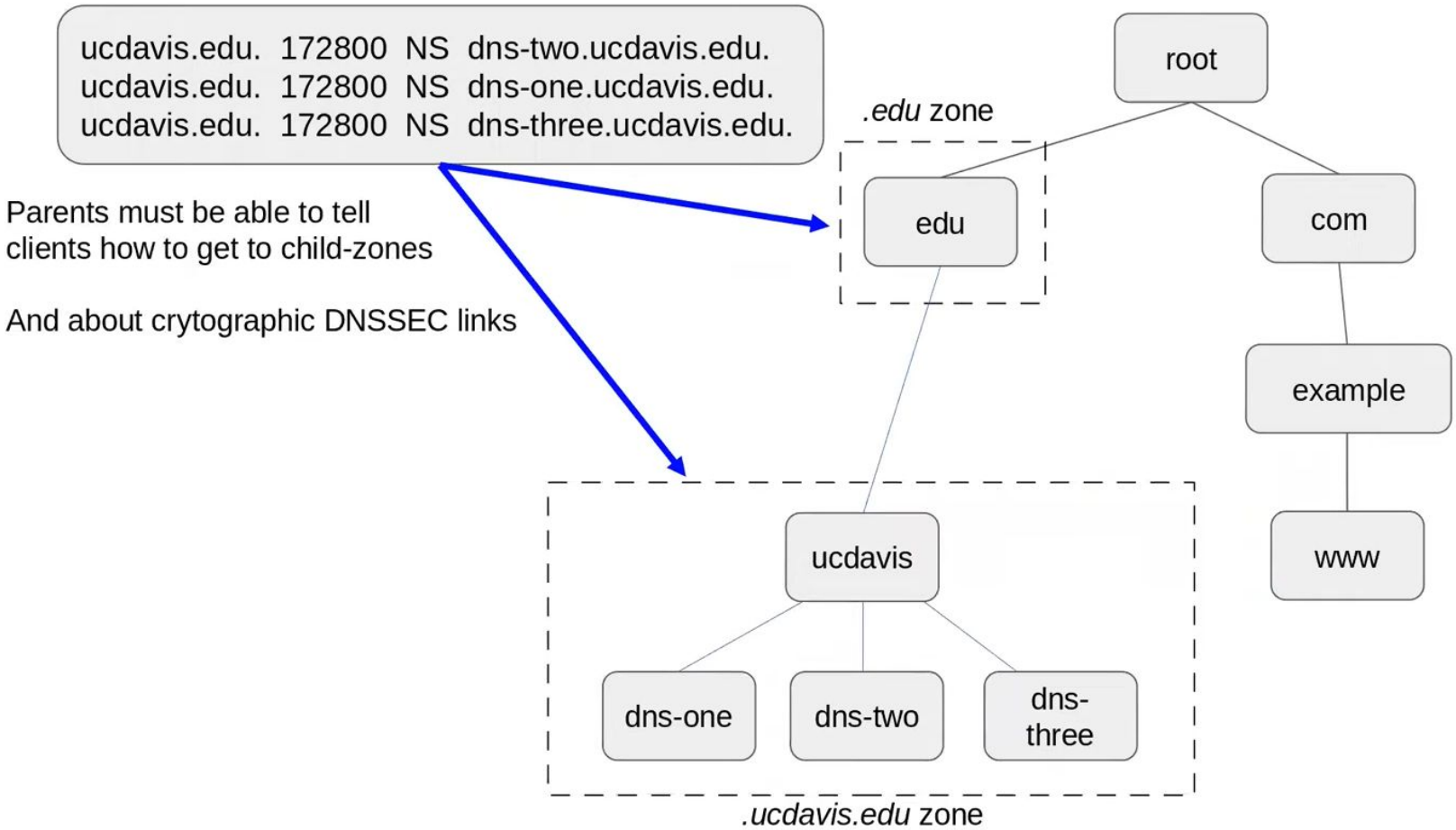
Second Level Domains (SLDs)

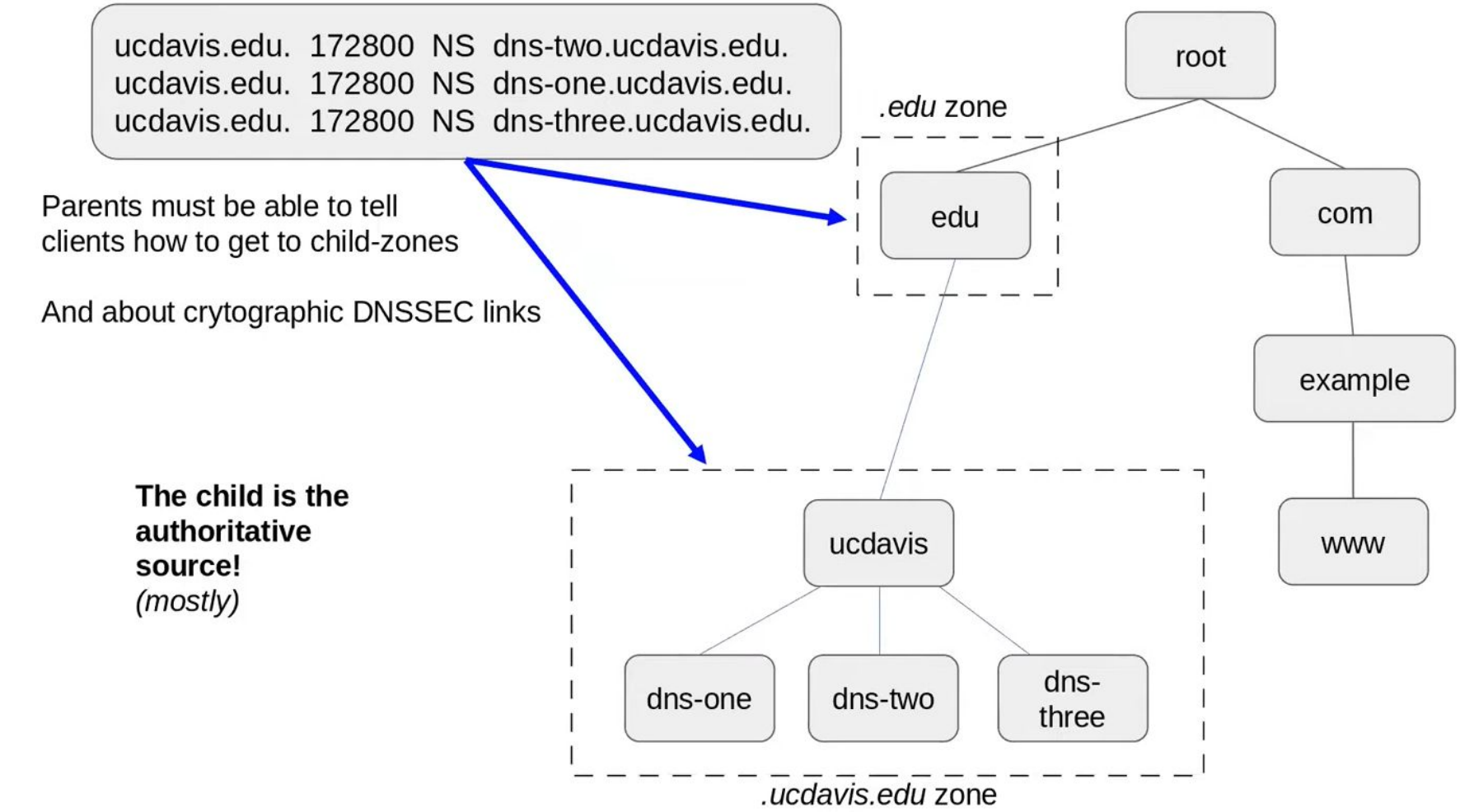


Some information needed in both parent and child zones

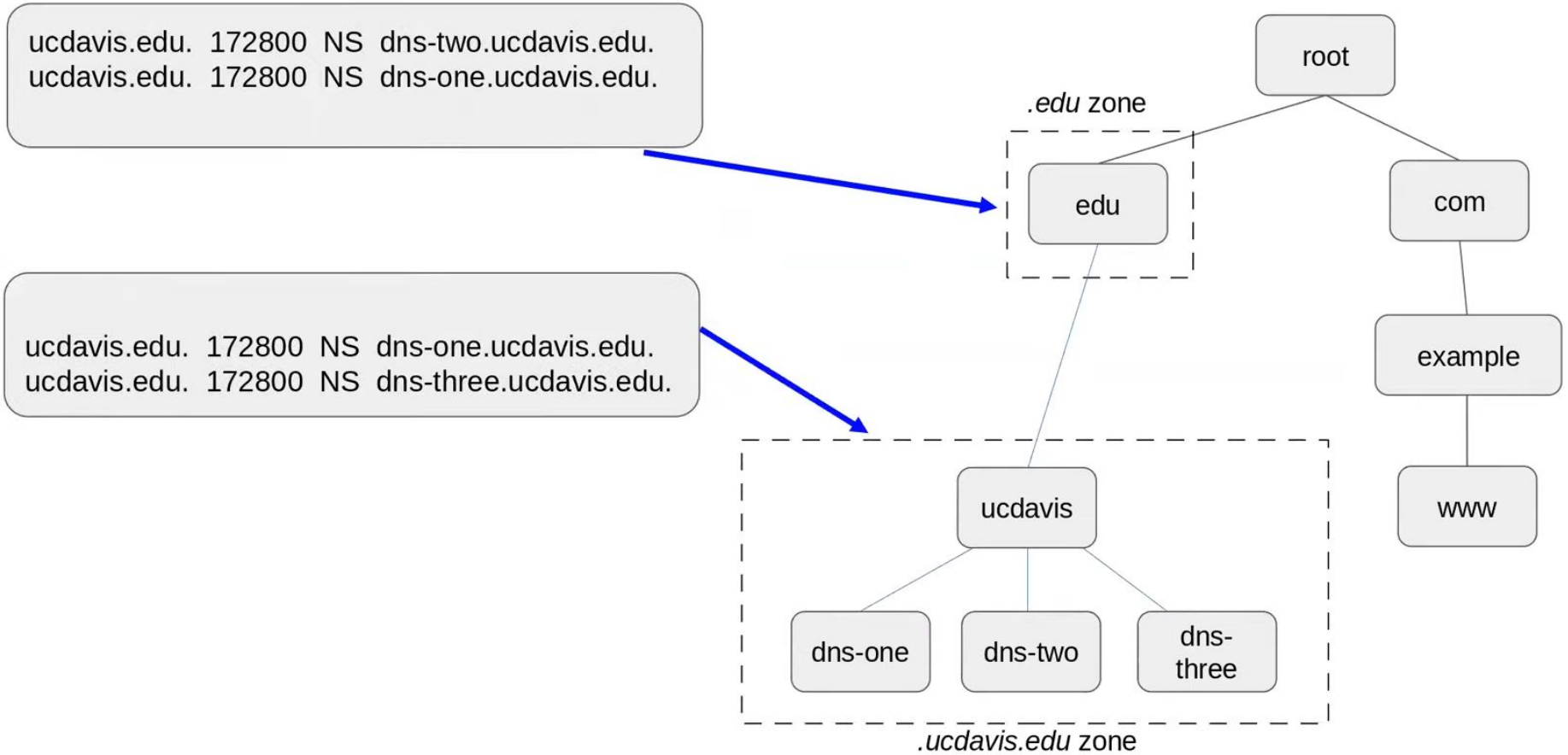


Some information needed in both parent and child zones




ucdavis.edu. 172800 NS dns-two.ucdavis.edu.
ucdavis.edu. 172800 NS dns-one.ucdavis.edu.
ucdavis.edu. 172800 NS dns-three.ucdavis.edu.
Parents must be able to tell clients how to get to child-zones
And about crytographic DNSSEC links
The child is the authoritative source!
(mostly)
root
.edu zone
edu
com
example
www
ucdavis
dns-one
dns-two
dns-three
.ucdavis.edu zone

What happens when things don't match…

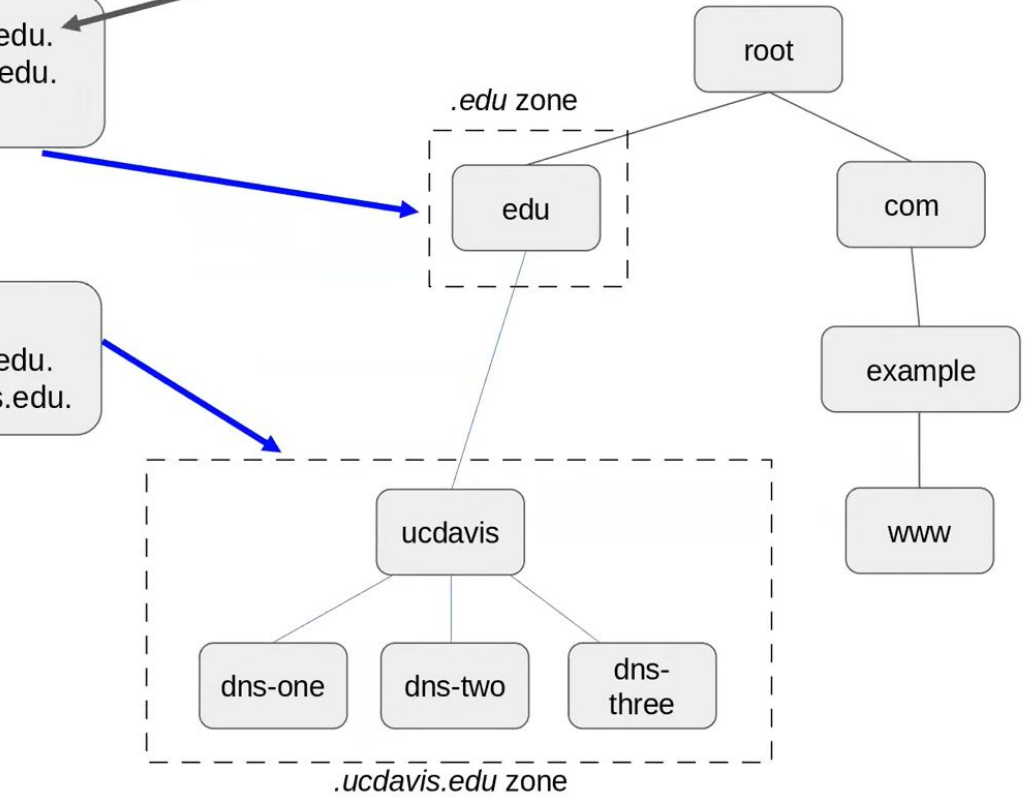


What happens when things don't match?

If *dns-two.ucdavis.edu* can't answer, this is a "lame delegation" (it's not authoritative, but .edu thinks it is)

```
ucdavis.edu.  172800  NS  dns-two.ucdavis.edu.
ucdavis.edu.  172800  NS  dns-one.ucdavis.edu.
```

```
ucdavis.edu.  172800  NS  dns-one.ucdavis.edu.
ucdavis.edu.  172800  NS  dns-three.ucdavis.edu.
```

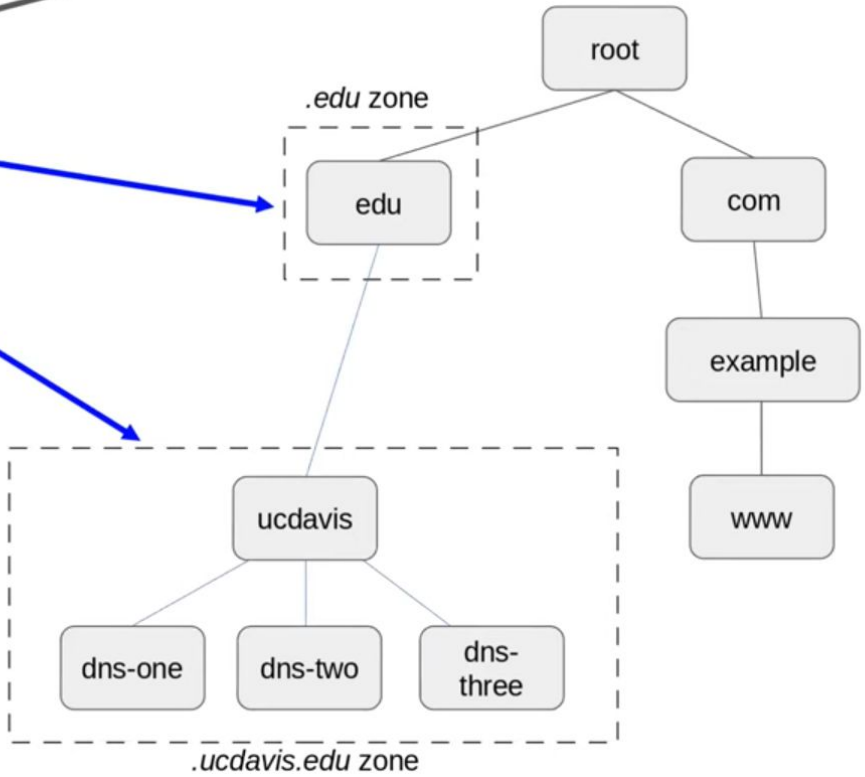


What happens when things don't match?

If *dns-two.ucdavis.edu* can't answer,
this is a "lame delegation"
(it's not authoritative, but .edu thinks it is)

```
ucdavis.edu.  172800  NS  dns-two.ucdavis.edu.
ucdavis.edu.  172800  NS  dns-one.ucdavis.edu.
```

```
ucdavis.edu.  172800  NS  dns-one.ucdavis.edu.
ucdavis.edu.  172800  NS  dns-three.ucdavis.edu.
```



Unfortunately **many**
zones exist with exactly
this problem

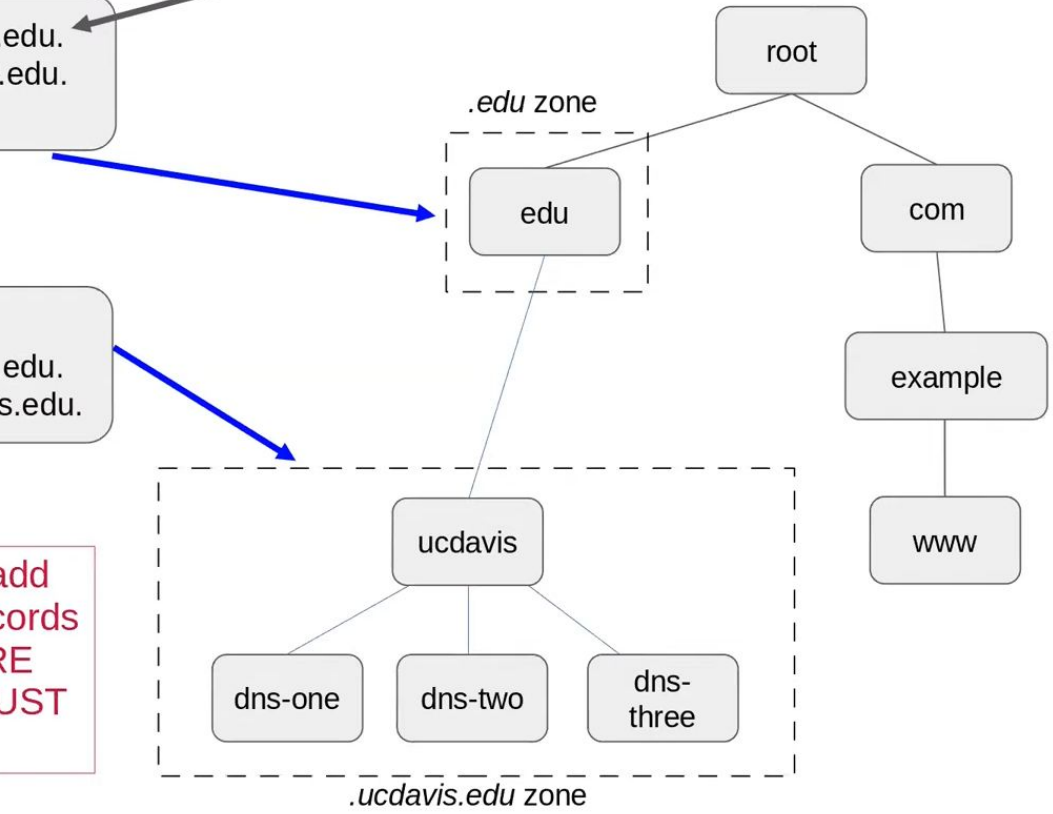
The result is timeouts
and delays for clients

What happens when things don't match?

If *dns-two.ucdavis.edu* can't answer, this is a "lame delegation" (it's not authoritative, but .edu thinks it is)

```
ucdavis.edu.  172800  NS  dns-two.ucdavis.edu.
ucdavis.edu.  172800  NS  dns-one.ucdavis.edu.
```

```
ucdavis.edu.  172800  NS  dns-one.ucdavis.edu.
ucdavis.edu.  172800  NS  dns-three.ucdavis.edu.
```



Unfortunately **many** zones exist with exactly this problem

The result is timeouts and delays for clients

When you add Security records EVEN MORE THINGS MUST MATCH

Always Remember….

- Probably 90% of the time errors are….
  - From mismatches
  - Often cached mismatches

- Generic rule for changing data:
  1. Add your new records first, then **WAIT**
  2. Publish to parent, then **WAIT**

     (NS, glue, DNSSEC)
  3. Remove old records from parent, then **WAIT**
  4. Remove your old records

Image source: Vivek Gite

WAIT := 2x max(your TTL, parent's TTL) – for resolver caches to refill

DNS Security Option Comparisons

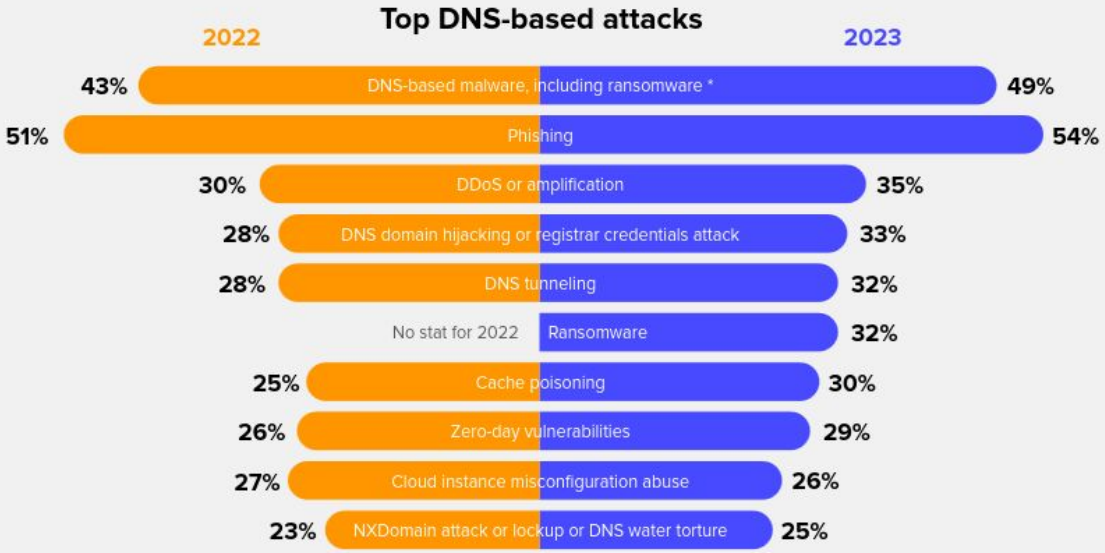
What is the role of DNS in cybersecurity landscape?

According to the International Data Corporation's (IDC), a European Security Service, Global DNS Threat Reports, the DNS attacks are continuously on the rise and impact major organizations and businesses.

[1] IDC 2021 Global DNS Threat Report. https://efficientip.com/resources/idc-dns-threat-report-2021/
[2] IDC 2022 Global DNS Threat Report. https://efficientip.com/resources/idc-dns-threat-report-2022/
[3] IDC 2023 Global DNS Threat Report.
https://efficientip.com/wp-content/uploads/2023/09/IDC-2023-DNS-Threat-Report.pdf
[4] https://www.idc.com/getdoc.jsp?containerId=IDC_P44612

Threat Landscape

Considering the essential role of DNS in the functioning of the Internet and the enterprise network, as well as its potential as a vector for the most dangerous cyberattacks, organizations need to prioritize DNS security as part of their overall cybersecurity strategy.



90%
experienced an attack

7.5
attacks on average

Attack sizes remain high

51%
over 5Gb/s

Ransomware:

A trend in ransomware is the use of multi-extortion: Attackers not only encrypt the data but also threaten to leak sensitive information if the ransom is not paid. This tactic is particularly effective against organizations that store large amounts of sensitive data, as the potential damage from a data breach can be much greater than the cost of paying the ransom.

The expansion of the threat landscape and the increasing sophistication of cyberattacks mean organizations need greater visibility and control over network activity. A purpose-built DNS security solution will enhance the security posture and help prevent data breaches and other cyberthreats, while actionable DNS data can be leveraged for threat intelligence.

Other Attacks…

Rogue DHCP Server. DNS Cache Poisoning. DNS Hijacking/Redirection Attacks. DNS Tunneling. DNS Hijacking - Framing/Phishing. Subdomain Hijacking.

Domain Squatting. C2 Domains. Botnet Domains. Grayware.

Wildcard DNS. Dangling DNS. Command and Control.

Data Theft. CNAME Cloaking. Strategically Aged Domains. DNS Infiltration.

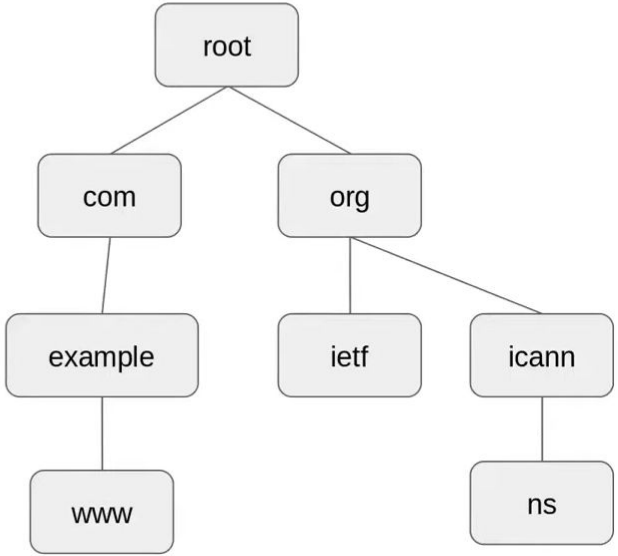
Direct DNS DoS Attack. DNS Amplification. DNS Reflection Attack. Bogus domain Attack.

Sloth Domain Attack. Phantom Domain Attack. Pseudo Random Subdomain Attack.

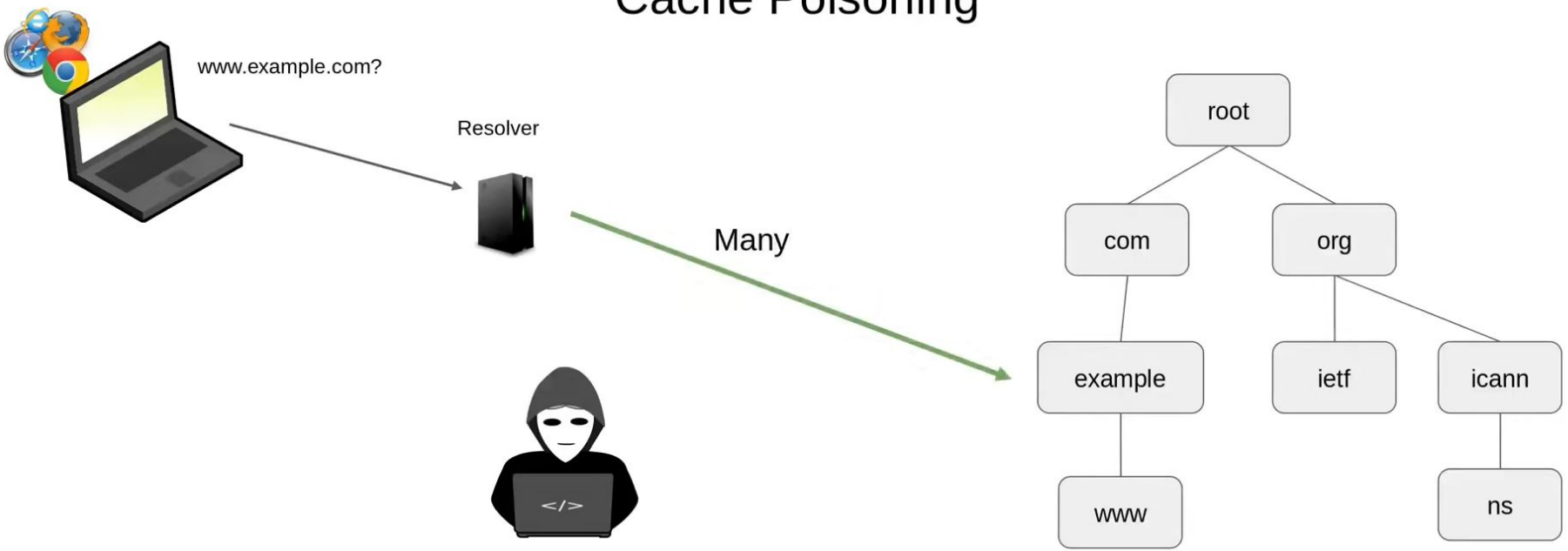
Cache Poisoning

www.example.com?

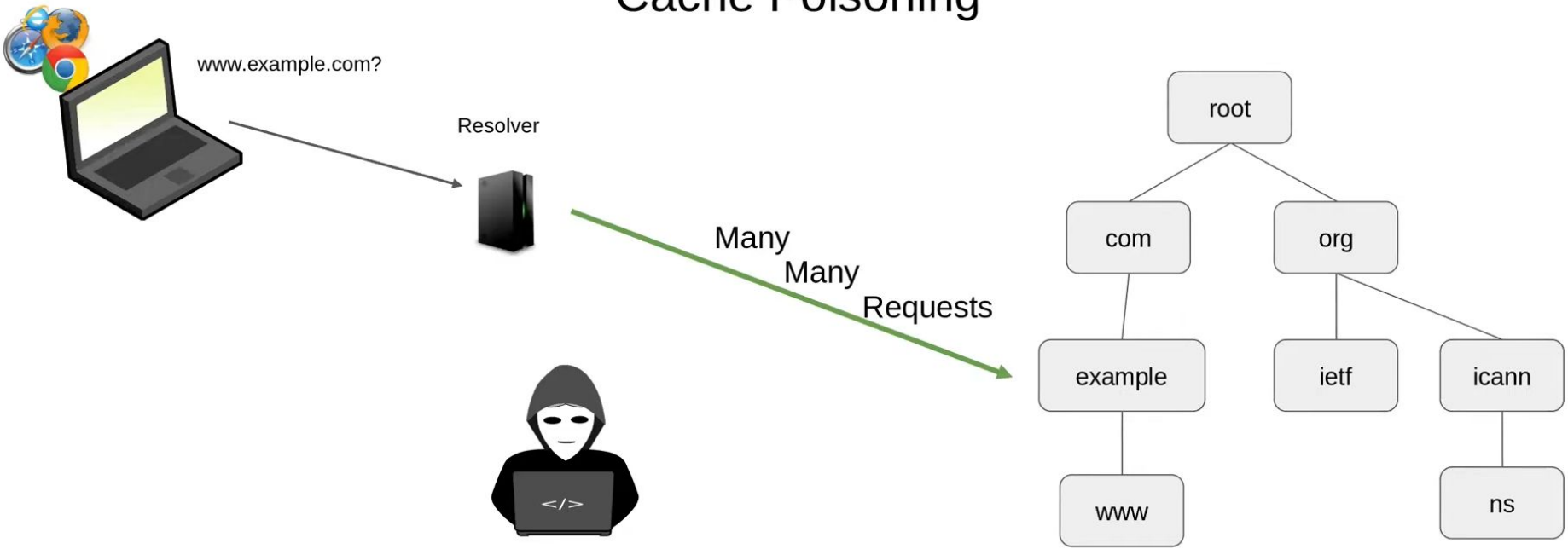
Resolver



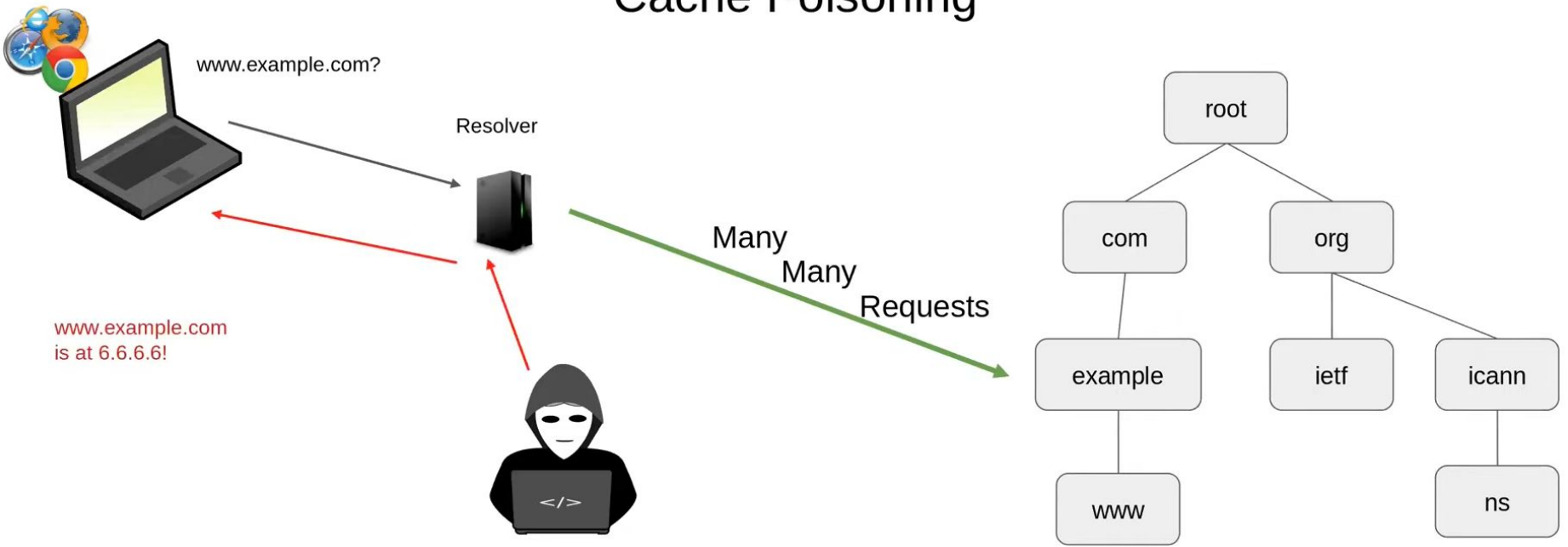
Cache Poisoning



Cache Poisoning



Cache Poisoning



Cache Poisoning

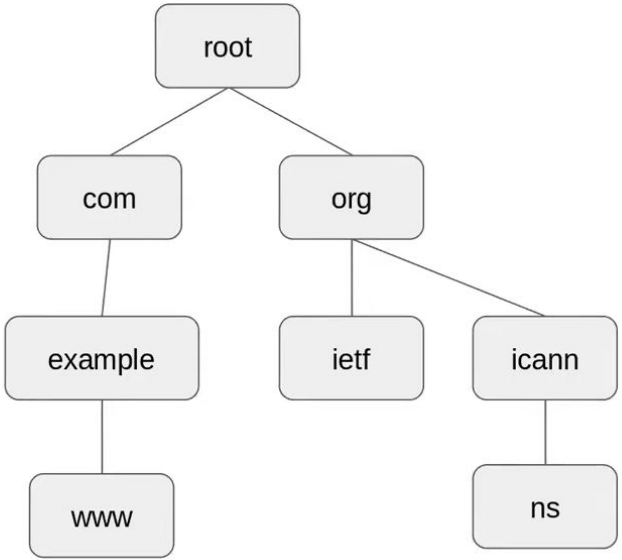
www.example.com?

Resolver

Many
Many
Requests

www.example.com
is at 6.6.6.6!

Without security,
First answer wins



Cache Poisoning

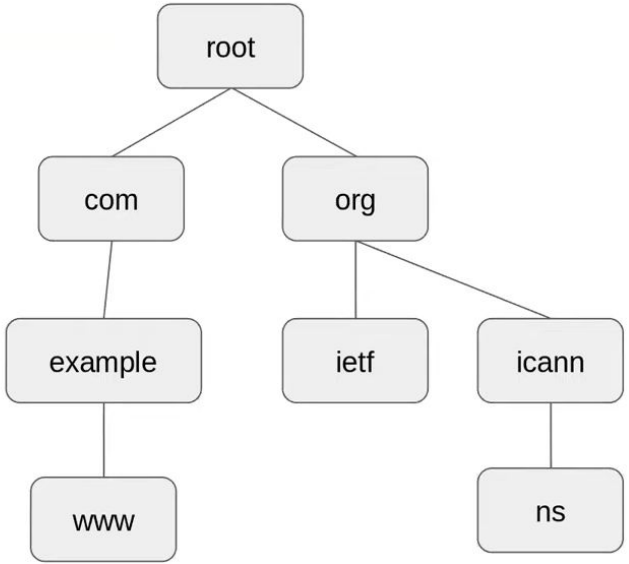
www.example.com?

Resolver

Many
Many
Requests

www.example.com
is at 6.6.6.6!

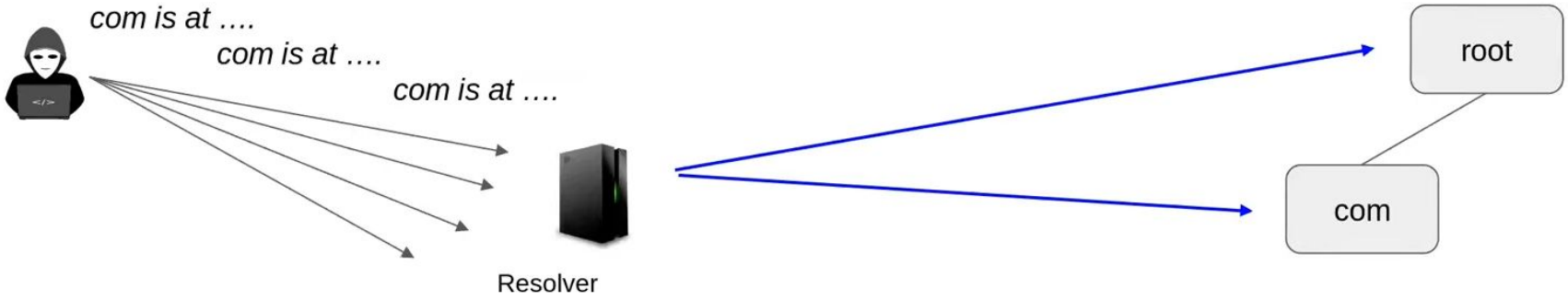
Without security,
First answer wins



And the resolver remembers the answer for a long time!

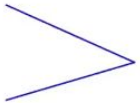
Early and Easiest Cache Poisoning **Attacks**

- Extra data in the additional section was considered authentic
  - I know you asked for www.example.com,
  - but did you know www.yourbank.com is at 6.6.6.6?
    - It is! And you should trust me!
- Protocol attacks
  - Just **flood the DNS resolver** with response packets and hope they accept them
  - Possibly **guessing IP/protocol values** (name, source address, DNS ID field)
  - Answer is cached for **TTL chosen by the attacker**



Early and easiest methods to **combat cache poisoning**

- Ignore non-authoratative answers
  - Only accept answers to questions you asked
  - Exception: parents can supply “glue” address records for in-zone child NS servers
- Resolvers must check that:
  - The IP source and UDP port is correct (handled by the UDP stack)
  - The DNS ID field is correct
- Senders must make it harder for attackers to guess these
  - Randomize the source port number
  - Randomize the ID field
- Note that this
  - Isn’t cryptographically strong
  - Doesn’t work at all for on-path attackers that can see and copy the requests

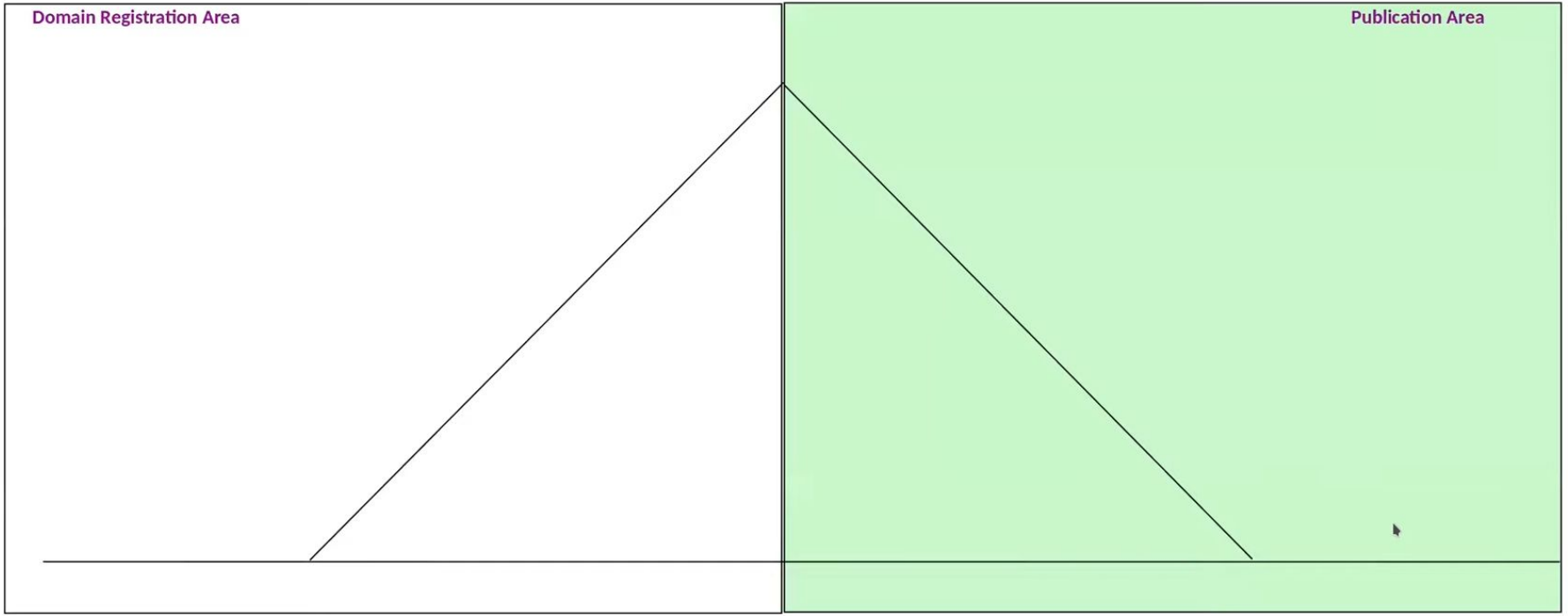


32 bits of randomness

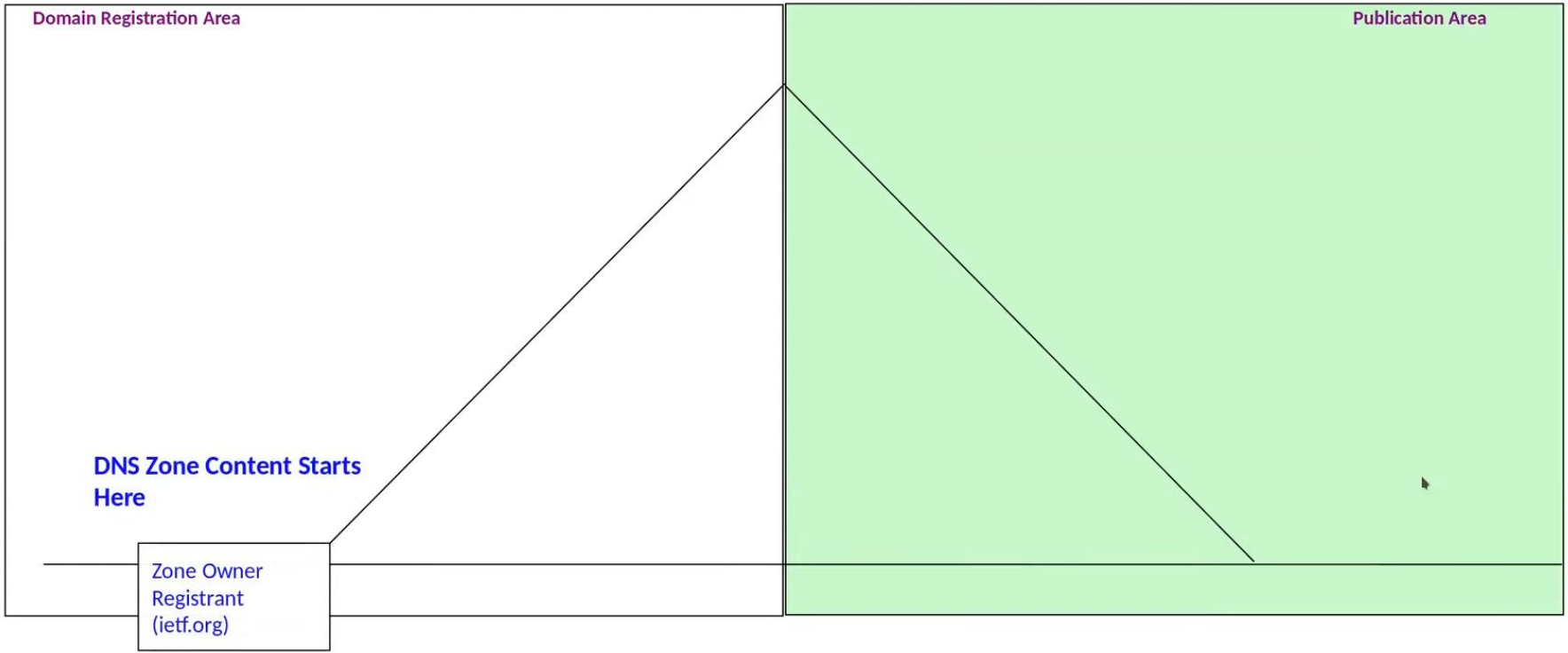
Can we fix it?

Yes, we can!

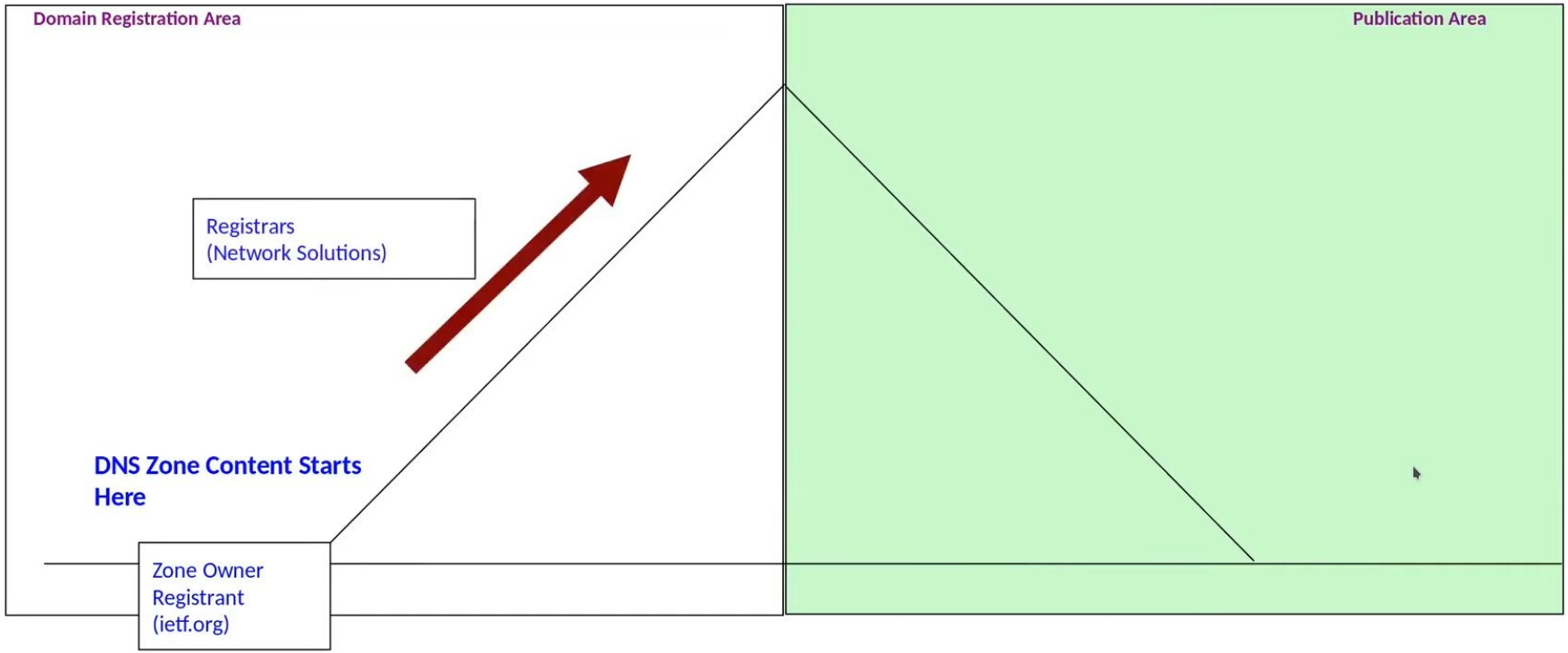
DNS Publication Architecture



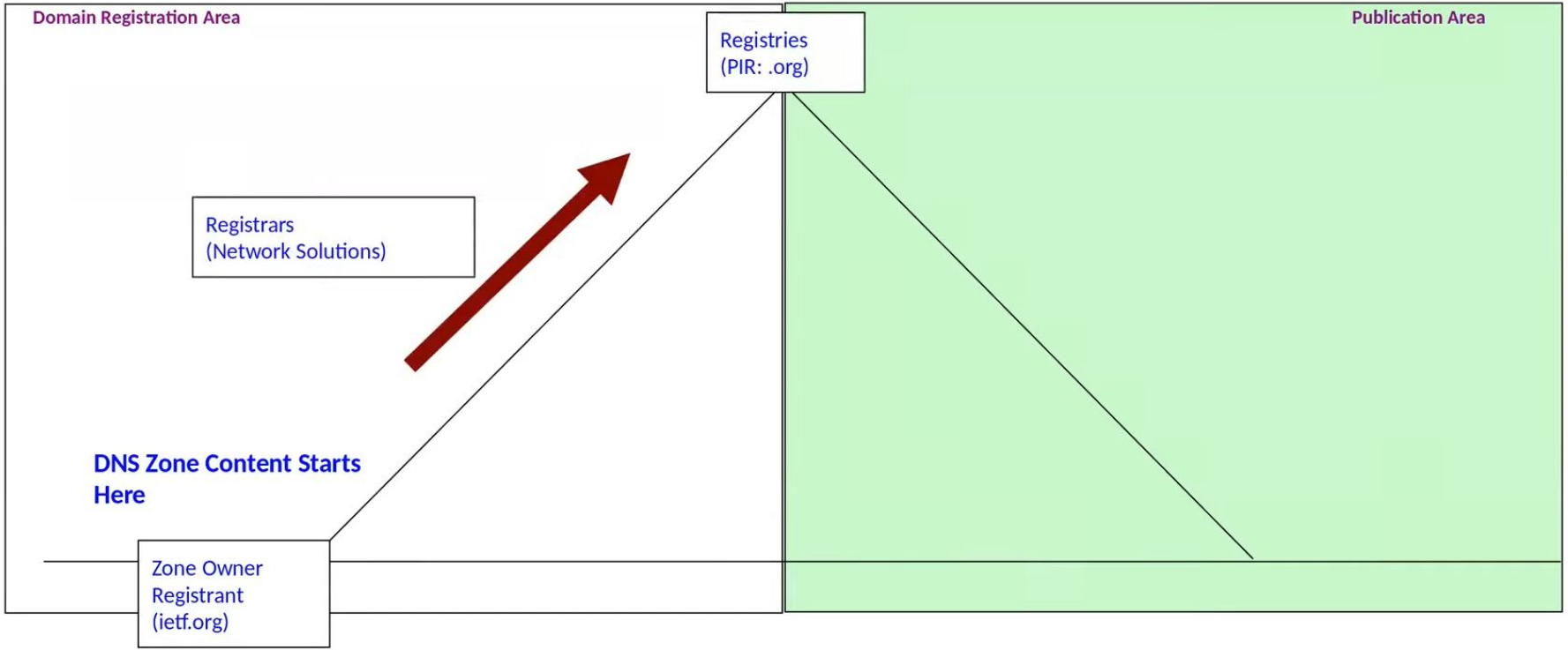
DNS Publication Architecture



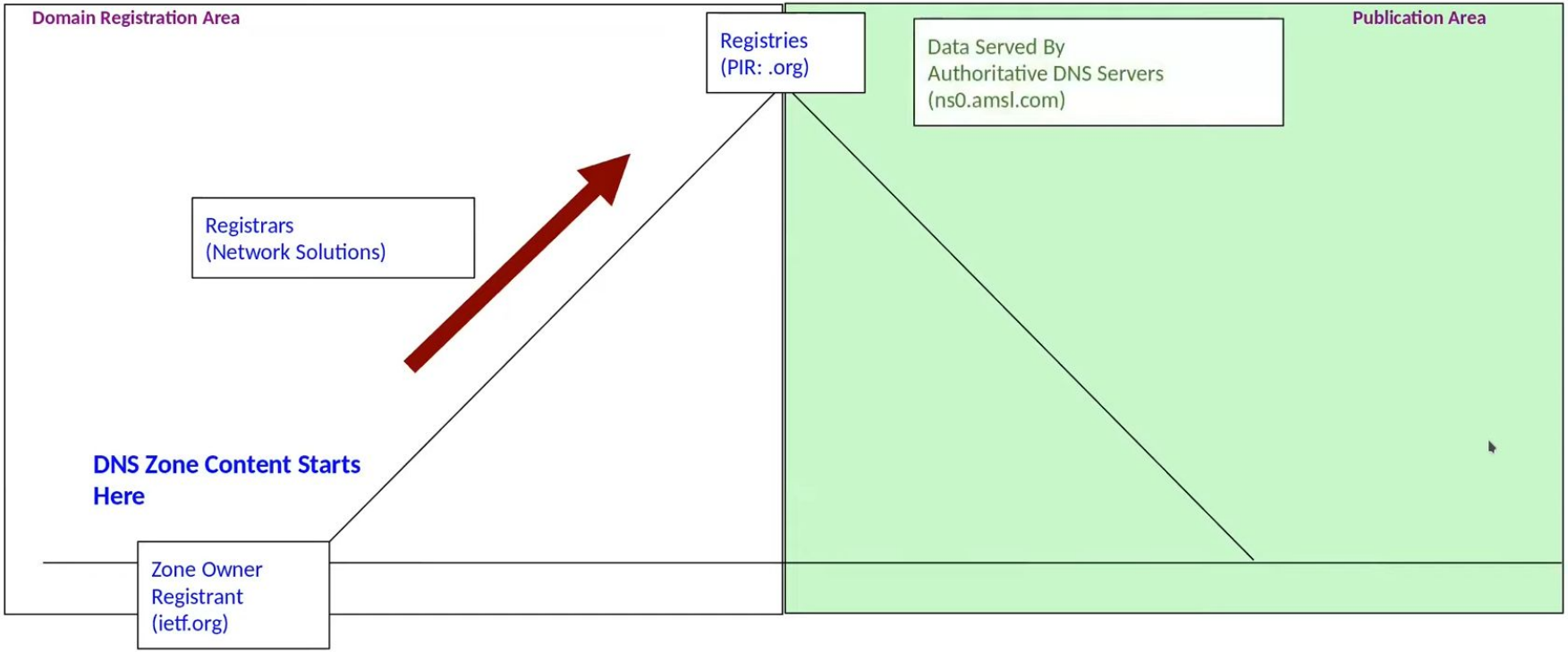
DNS Publication Architecture



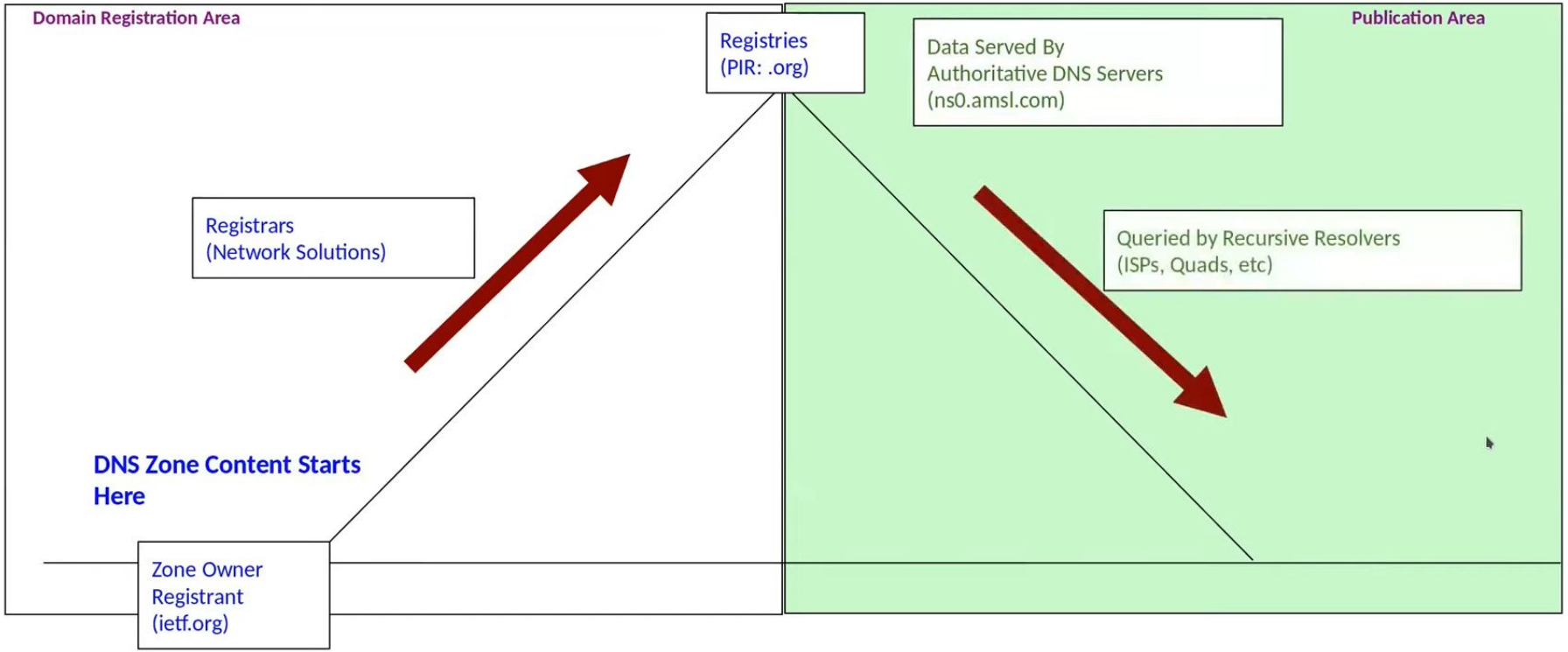
DNS Publication Architecture



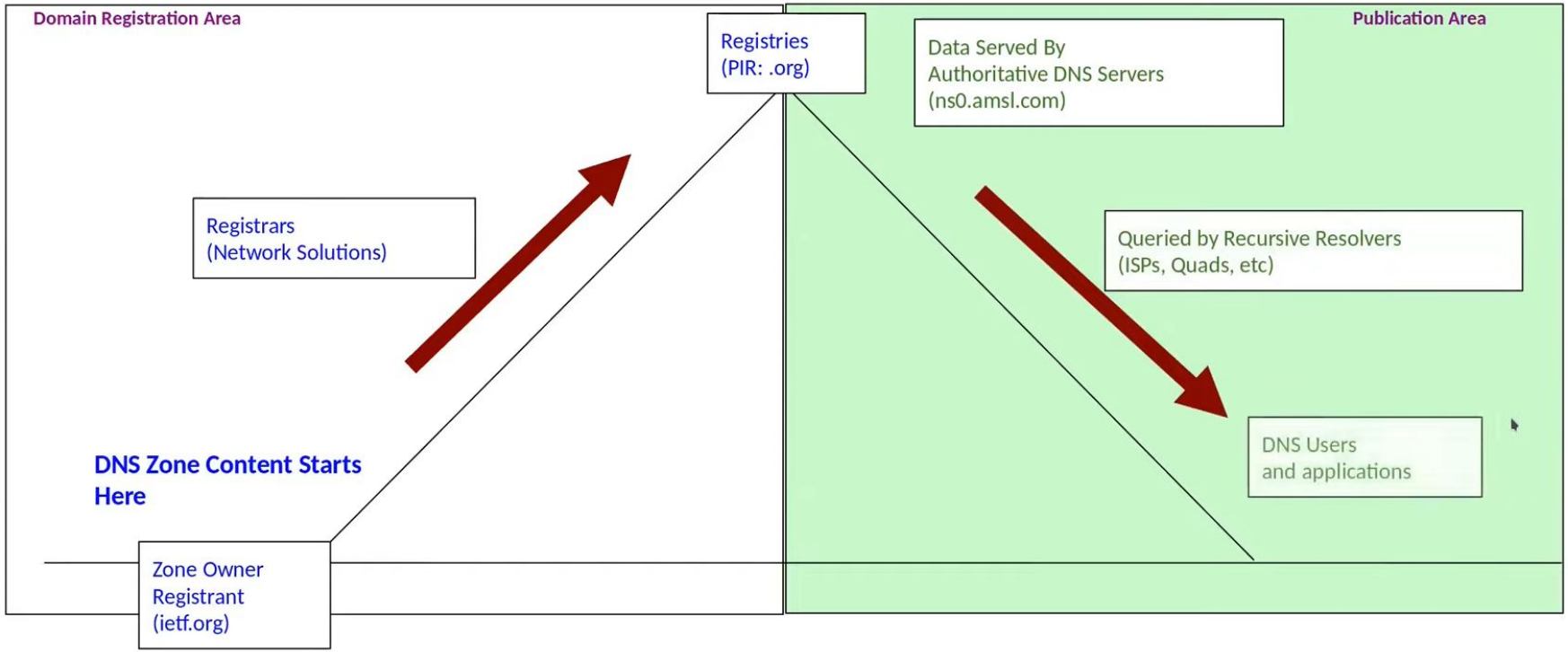
DNS Publication Architecture



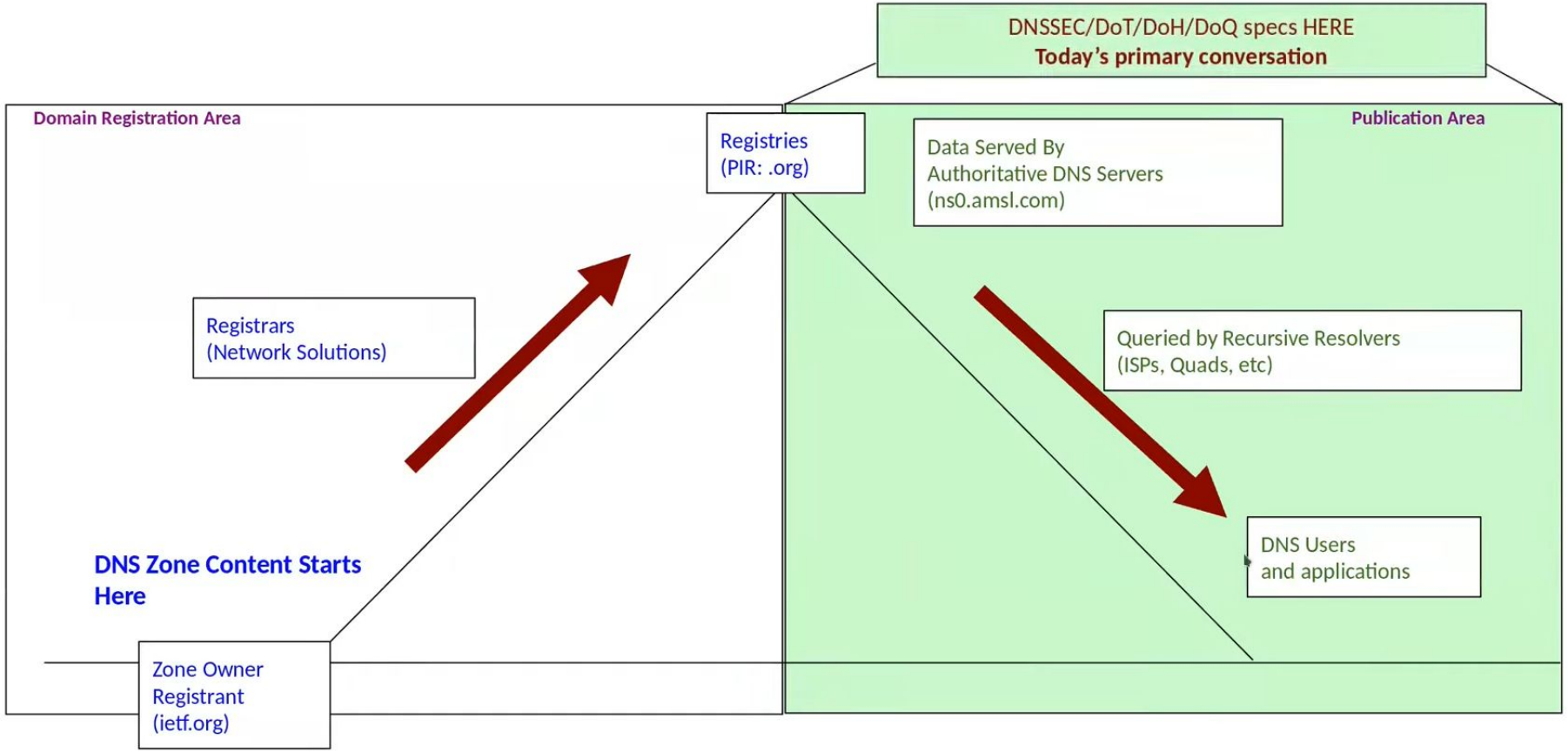
DNS Publication Architecture



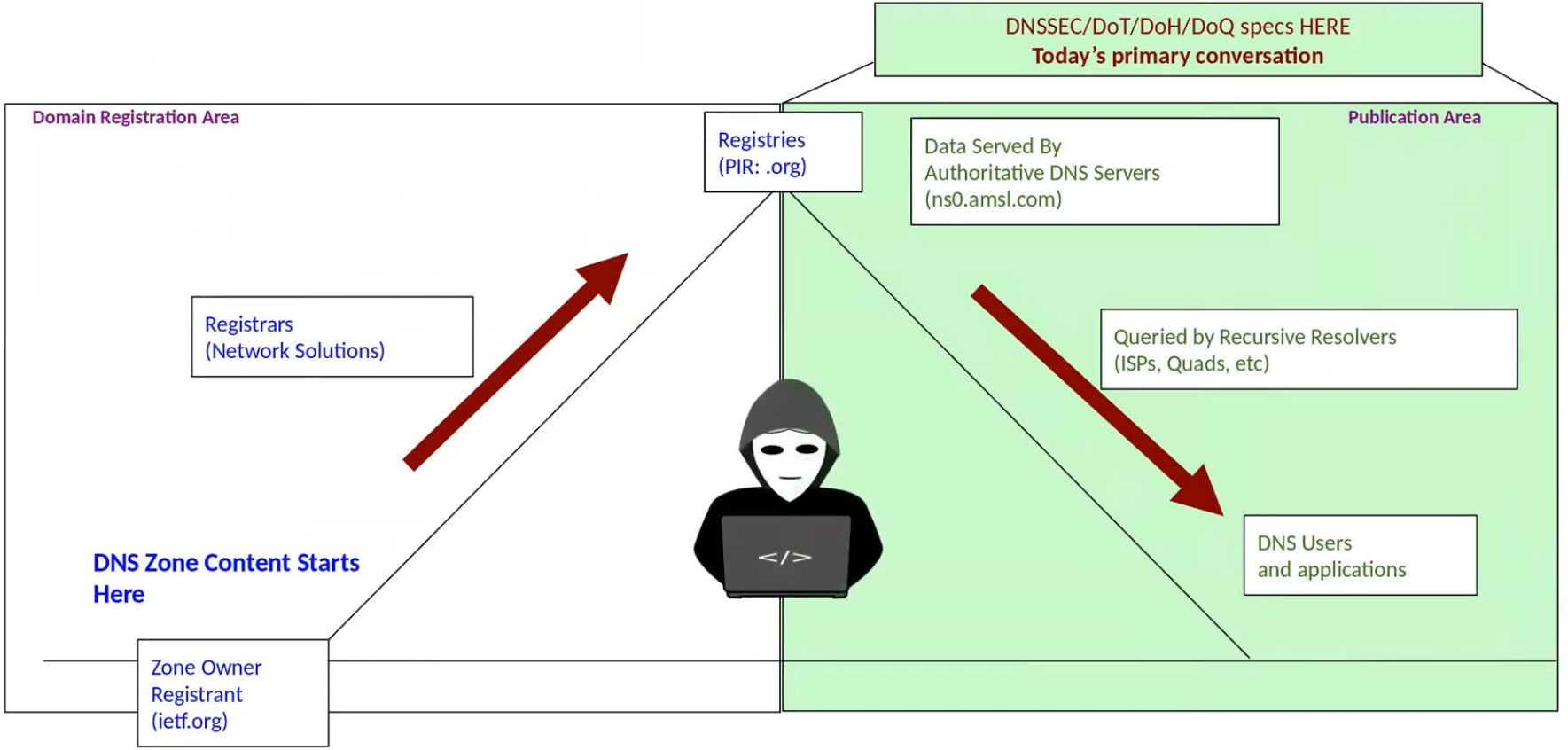
DNS Publication Architecture



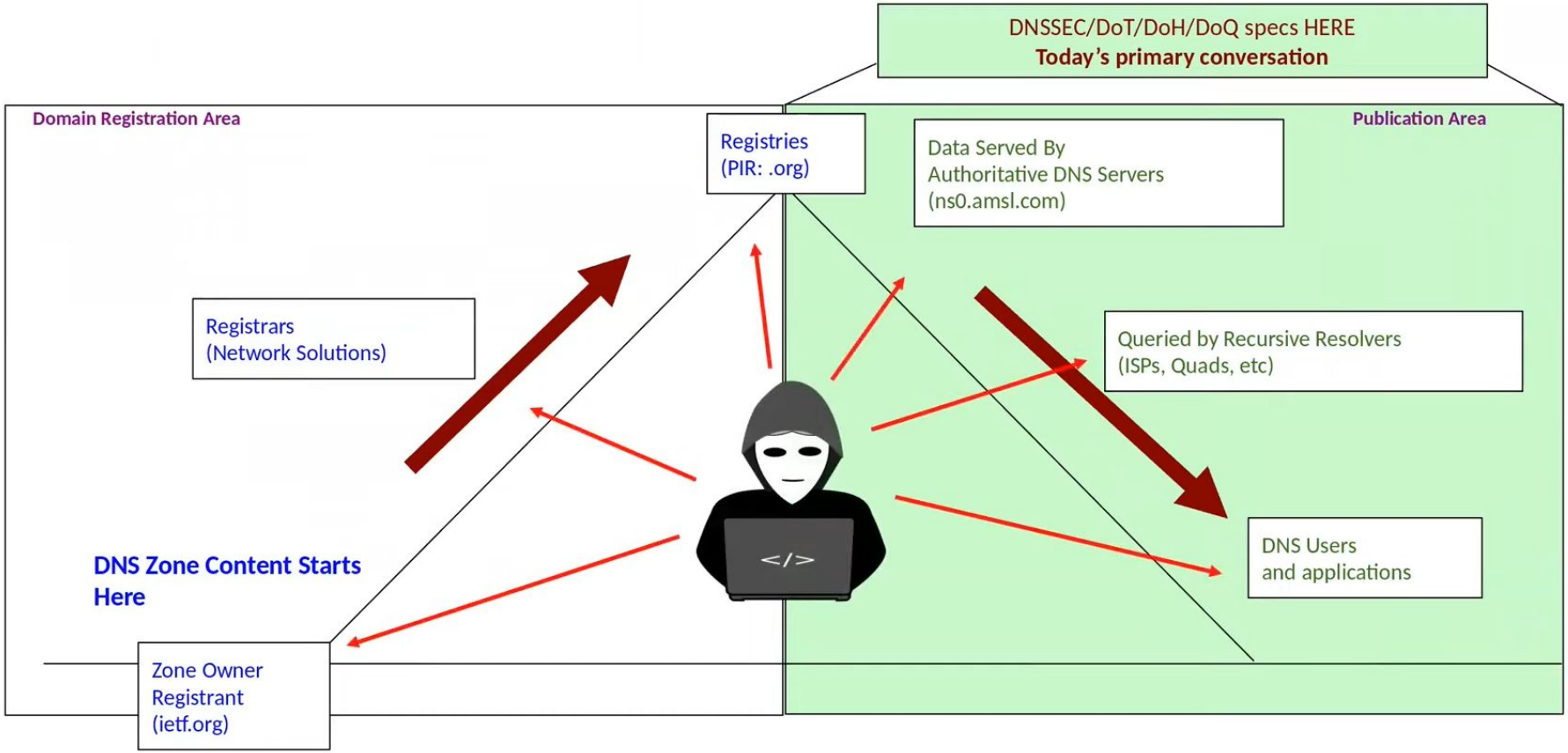
DNS Publication Architecture



DNS Publication Architecture



DNS Publication Architecture



DNS Data Protection Mechanisms

Object security

DNS Data Protection Mechanisms

Object security

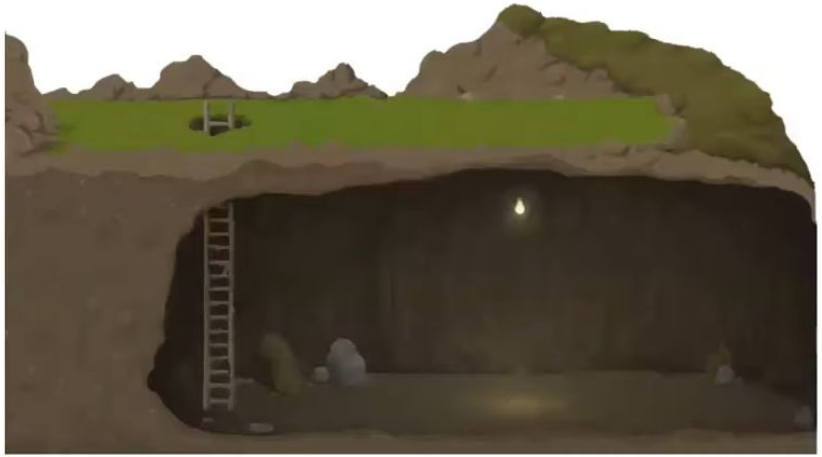
Path security

Object Security -- DNSSEC

- DNSSEC adds cryptographic **signatures on record data**
- Public/Private keys
- Signed at or near the data's origin
- Verifiable in the middle
- Verifiable at the end
- Only provides integrity protection -- **no privacy protection**

Path Security -- DOT, DOH, DOQ, ...

- Tunnel's answers securely between two points
- Provides integrity protection and encryption
  - But, offers **point-to-point protection only**
  - Verifies **who you got it from**, but **not what it is**



Path vs Object Differences

Object security *(DNSSEC)*

- Pro: Ensures end-to-end integrity is available everywhere
- Con: **Doesn't provide privacy protection**
- Pro: Distributed trust model with minimal configuration (typically 1 trust anchor)
- Con: Typically not deployed all the way to the user ("the last mile")

Path security *(DoT, DOH, …)*

- Pro: Provides point-to-point integrity **and privacy** protection
- Con: Doesn't verify data actually came from the origin *(trust everyone in the path?)*
- Con: For true security, **requires that every link be protected & trusted**
- Pro: Solves the last mile problem

Object vs Path Security

They have very different complementary properties

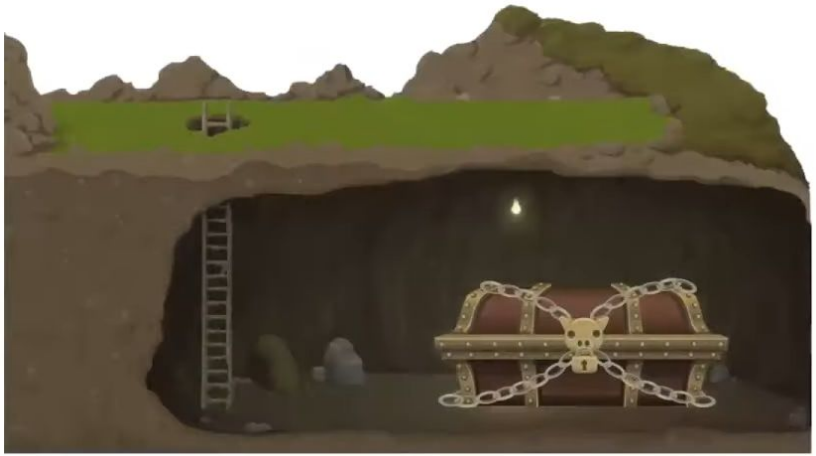
Object vs Path Security

They have very different complementary properties

and

Object vs Path Security Compatibility

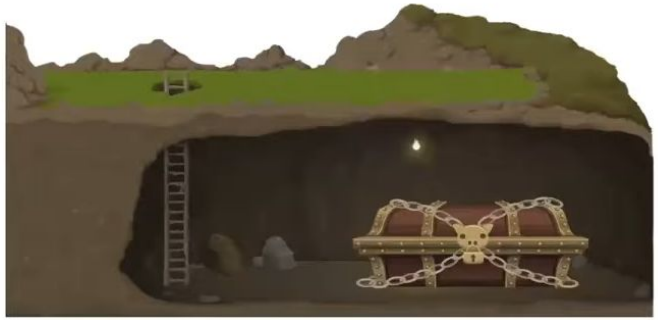
They have very different complementary properties



Path and Object security

Stronger together:

- Object security protects the data
- Object security ties the date to its source
- Path security fixes the last mile problem
- Path security provides encryption
  - Protects clients from on-path eavesdropping



DNS Security Technologies

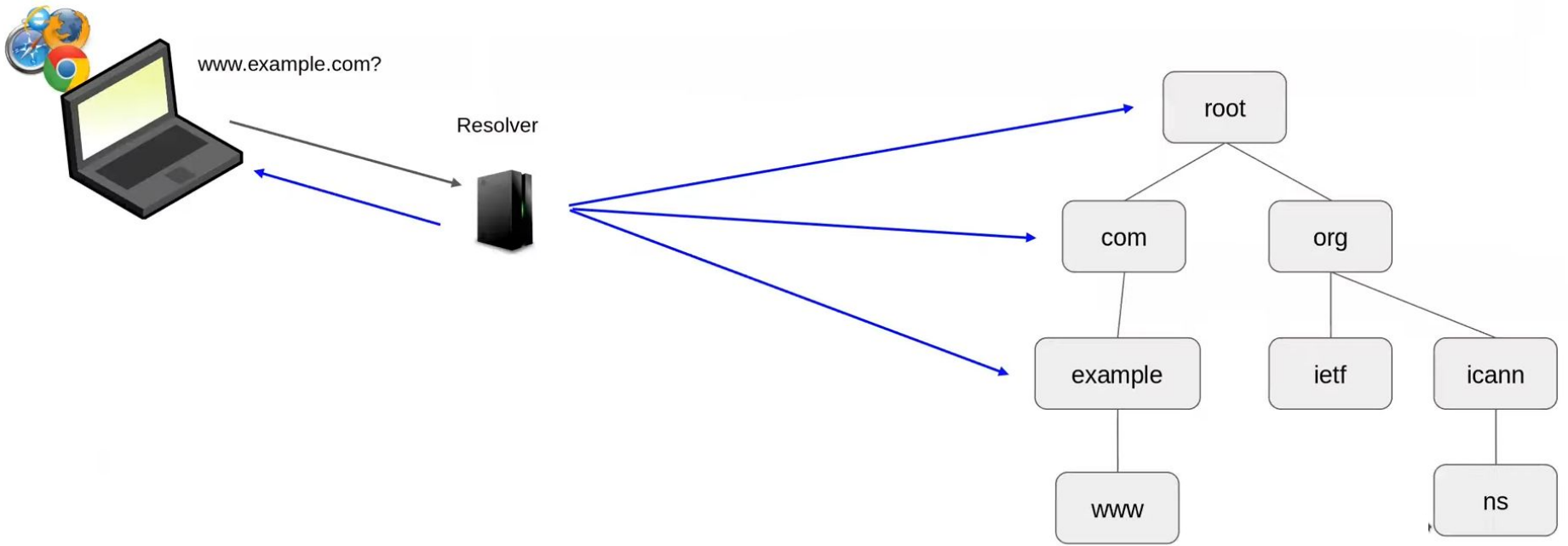
Object security (integrity, but no encryption)

- DNSSEC — DNS records signed by their authoritative source — RFC4033+

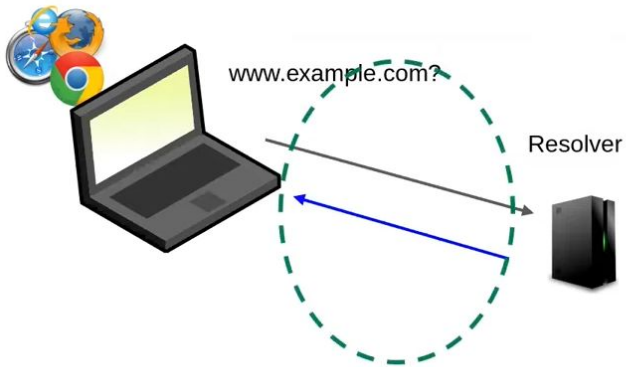
Path security (point-to-point only integrity, and provides encryption)

- DoT — DNS over TLS — RFC7858
- DoDT — DNS over DTLS — RFC8094
- DoH — DNS over HTTPS — RFC8484
- TSIG — DNS over DNS with shared keys — RFC8945
- DNS Curve: point-to-point encryption and authentication with elliptic curve

Security Protection Areas

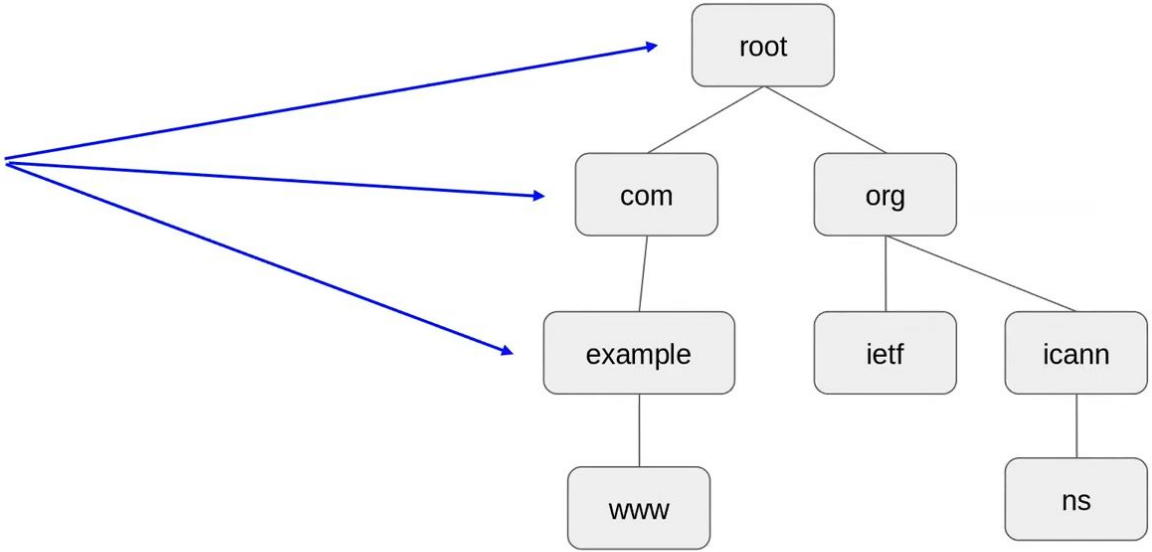


Security Protection Areas

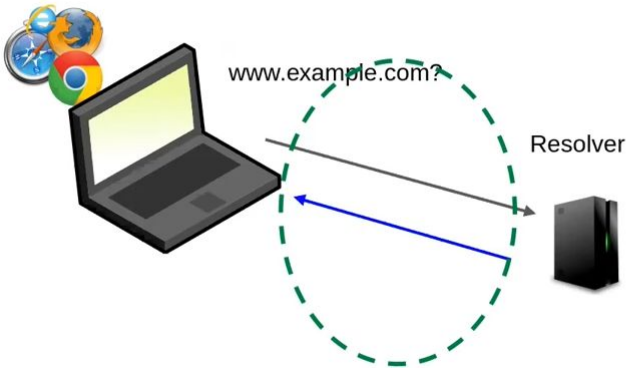


DoT / DoDT / DoH

Authenticates and Encrypts the user to the resolver's protocol traffic

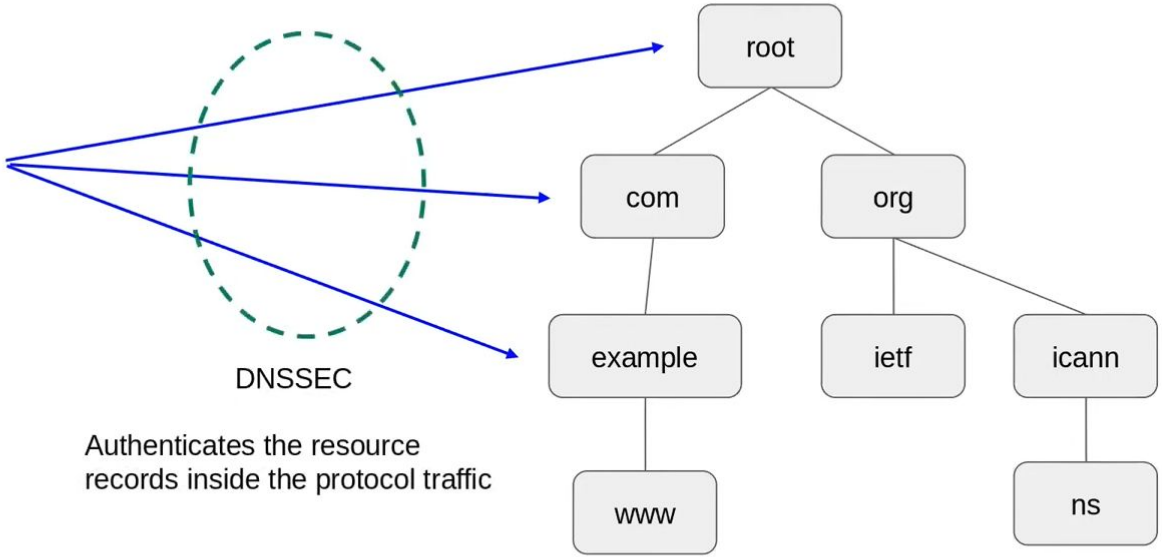


Security Protection Areas



DoT / DoDT / DoH

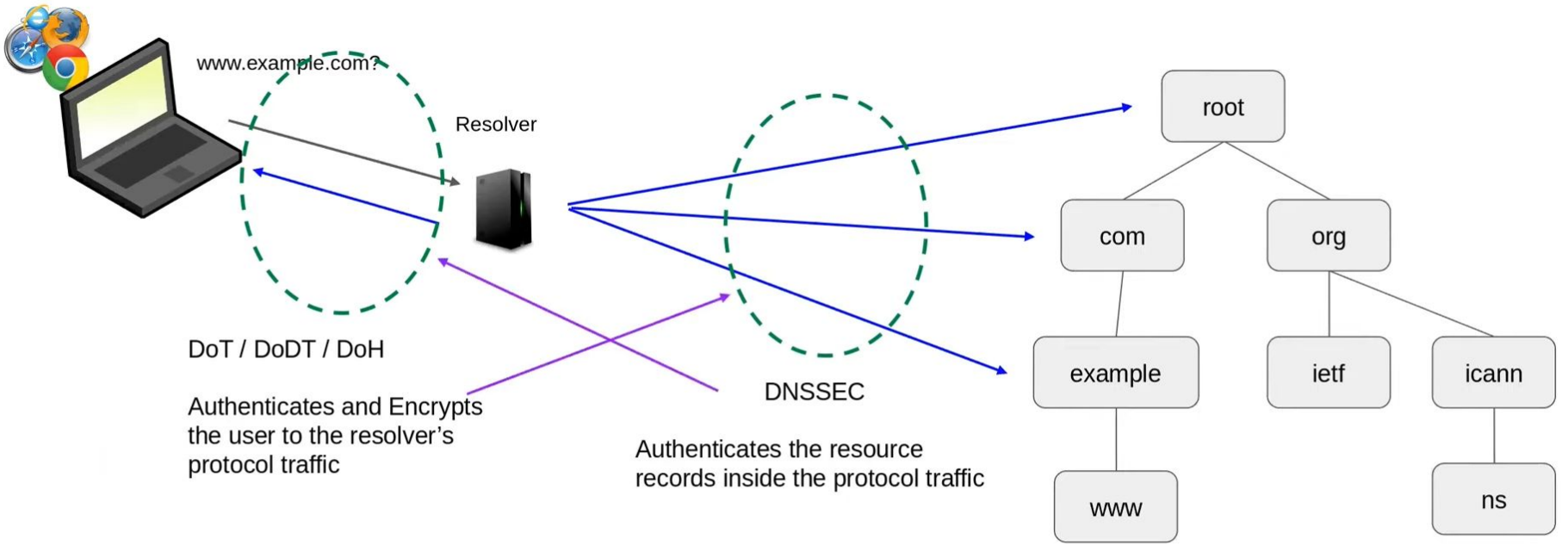
Authenticates and Encrypts the user to the resolver's protocol traffic

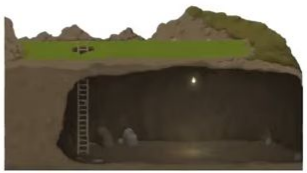


DNSSEC

Authenticates the resource records inside the protocol traffic

Security Protection Areas

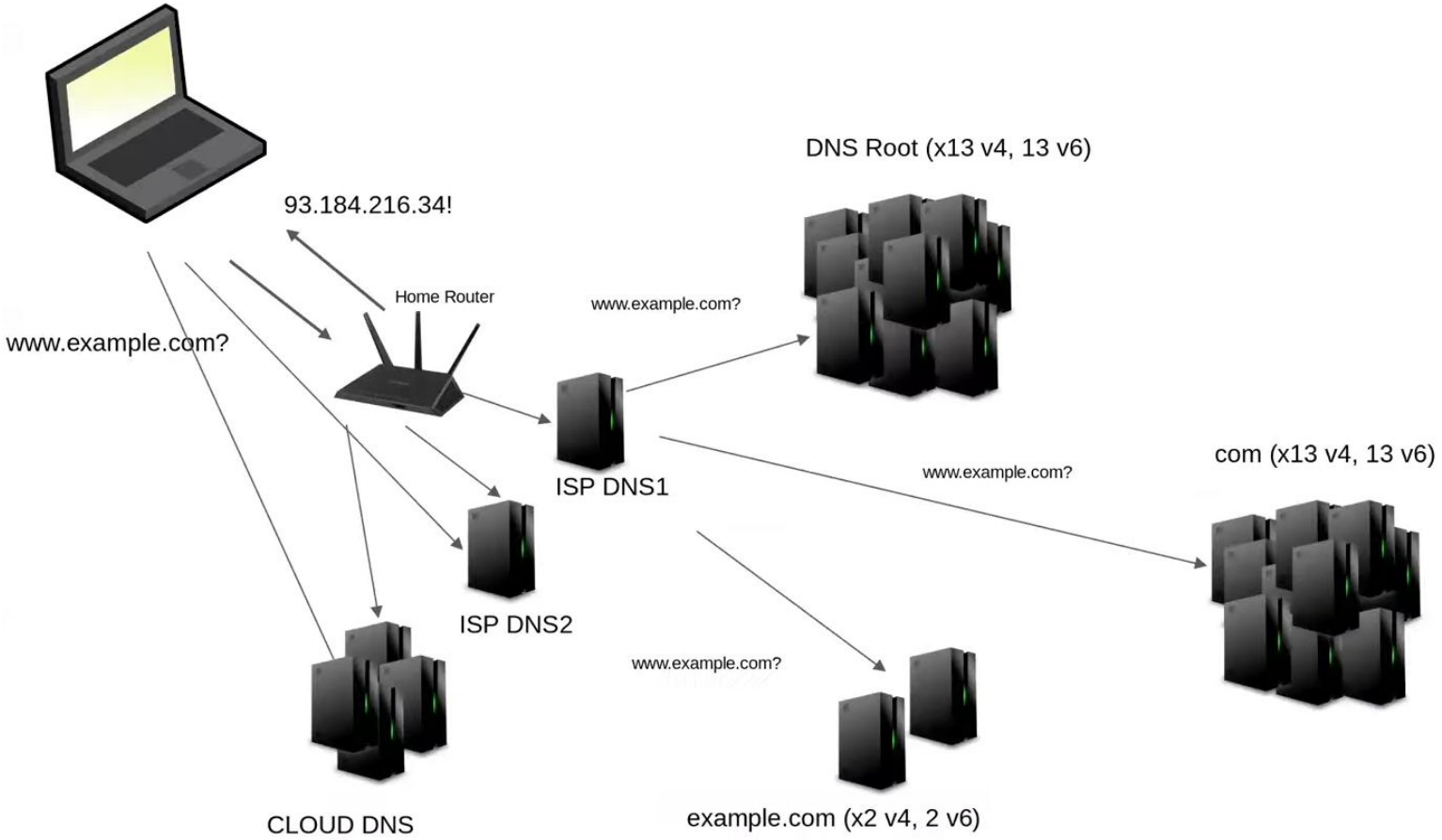




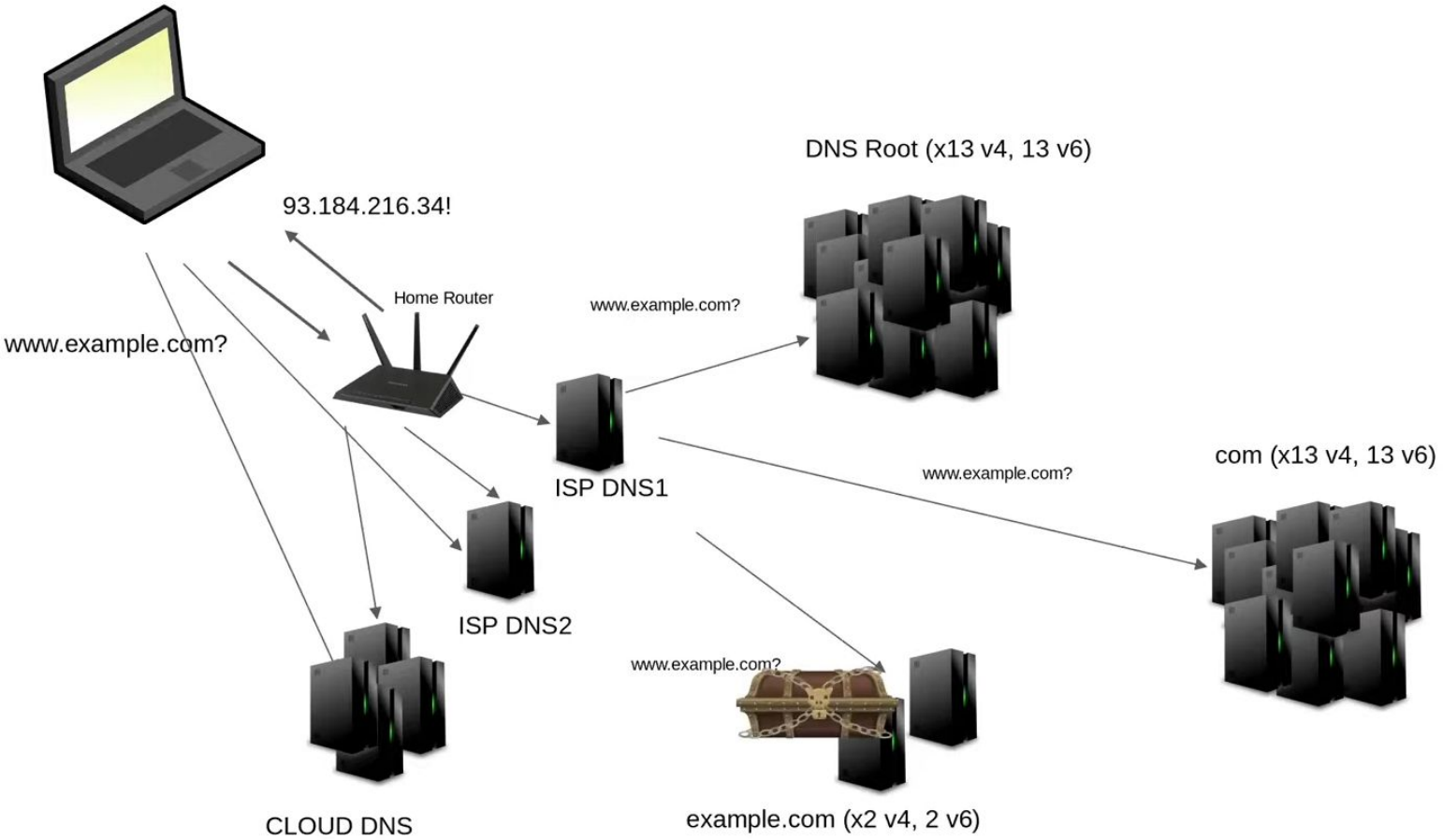
Can be used in both places, but typically isn't

DNSSEC – Object Security at Work

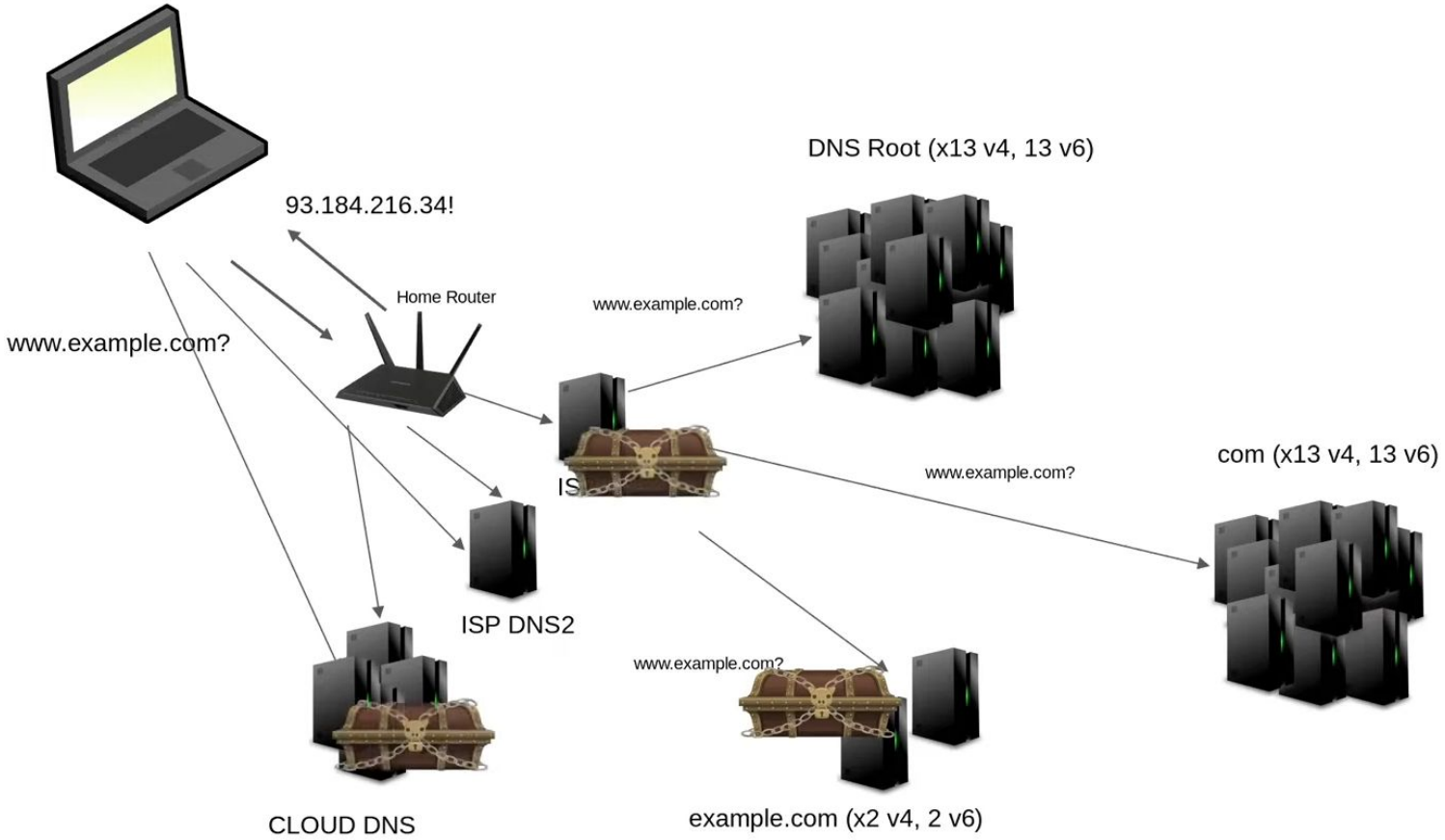
DNSSEC secures records no matter where they are



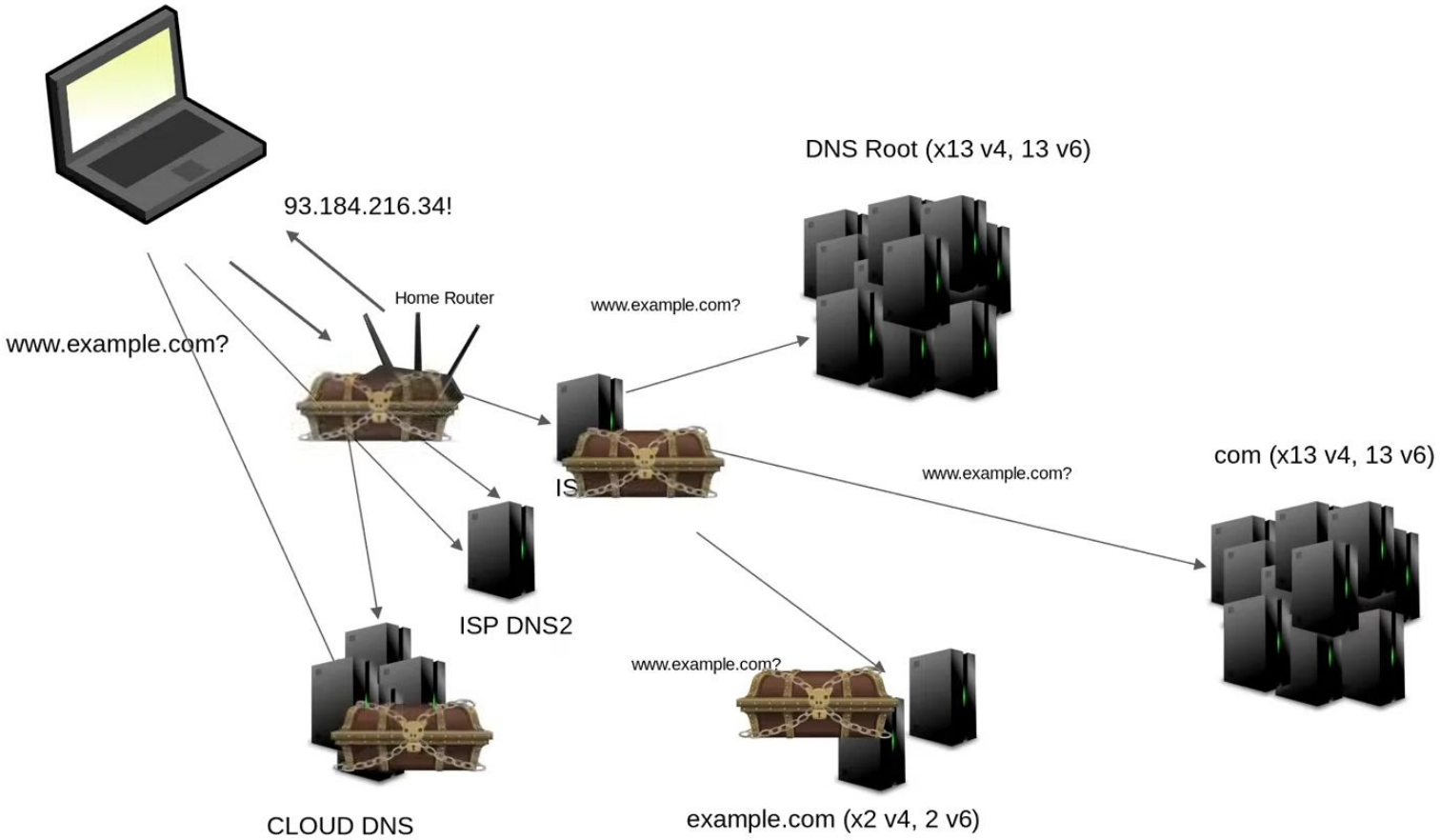
DNSSEC secures records no matter where they are



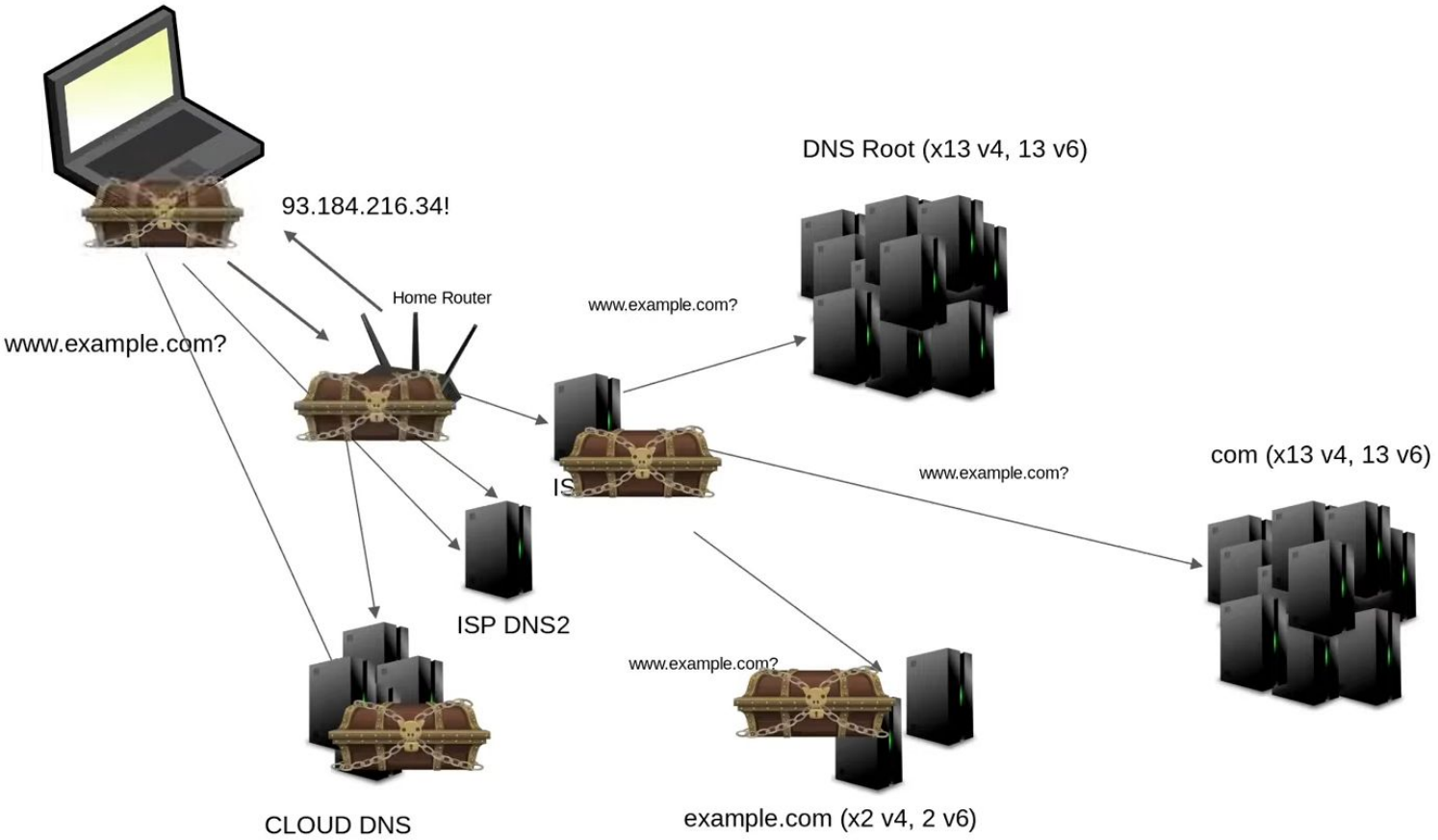
DNSSEC secures records no matter where they are



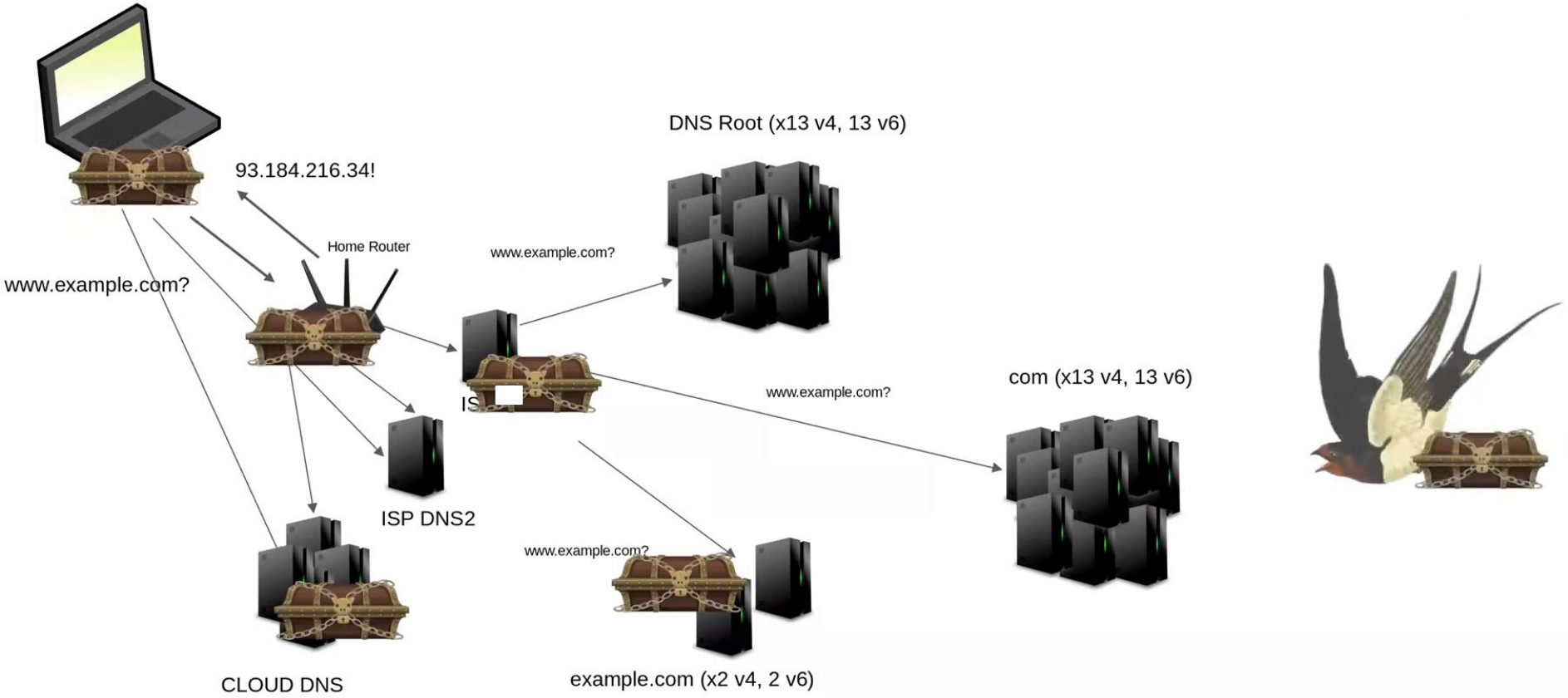
DNSSEC secures records no matter where they are



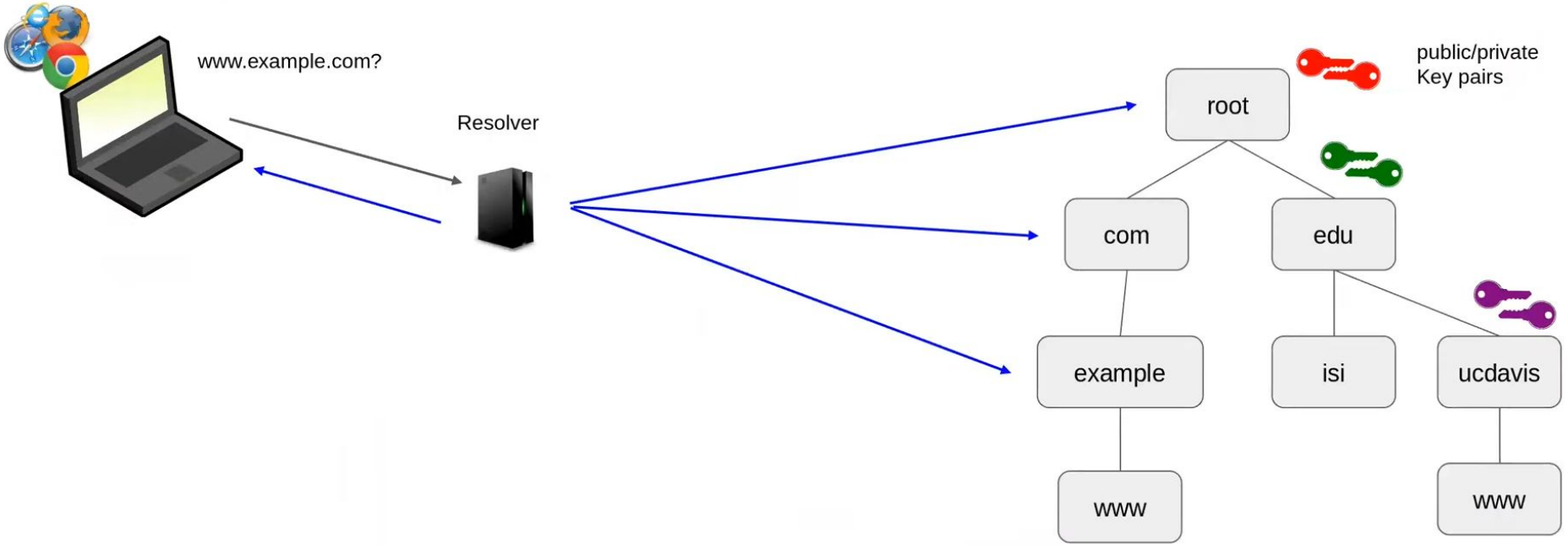
DNSSEC secures records no matter where they are



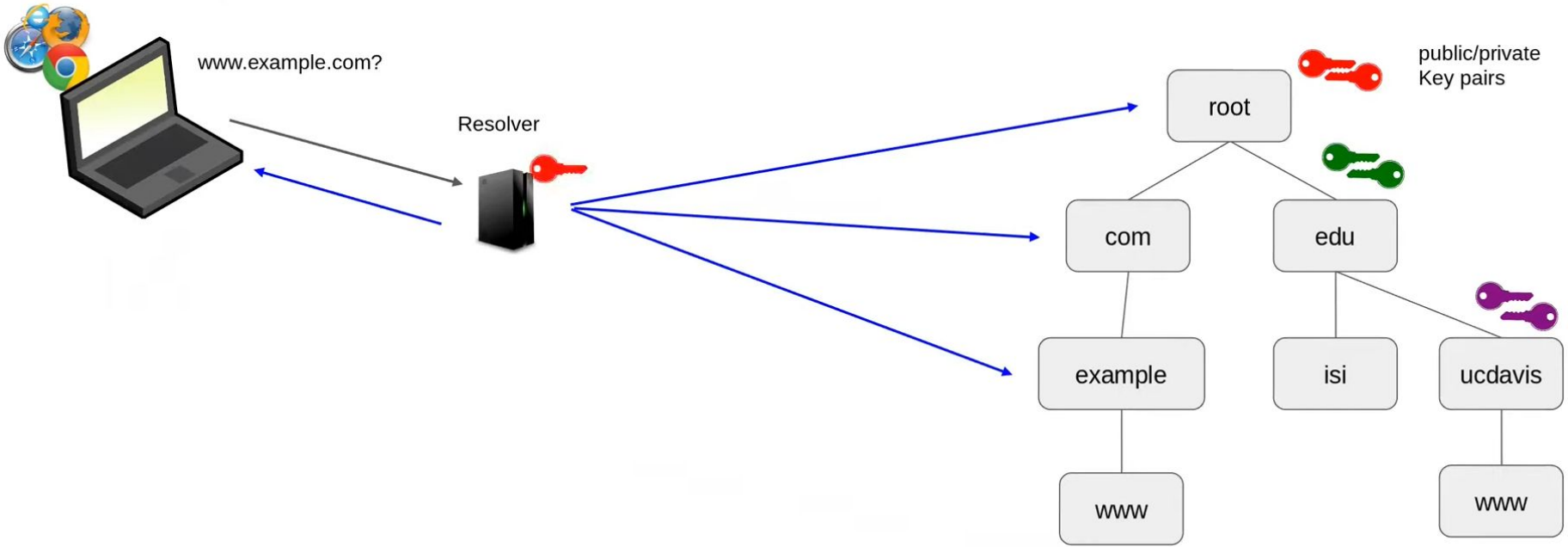
DNSSEC secures records no matter where they are



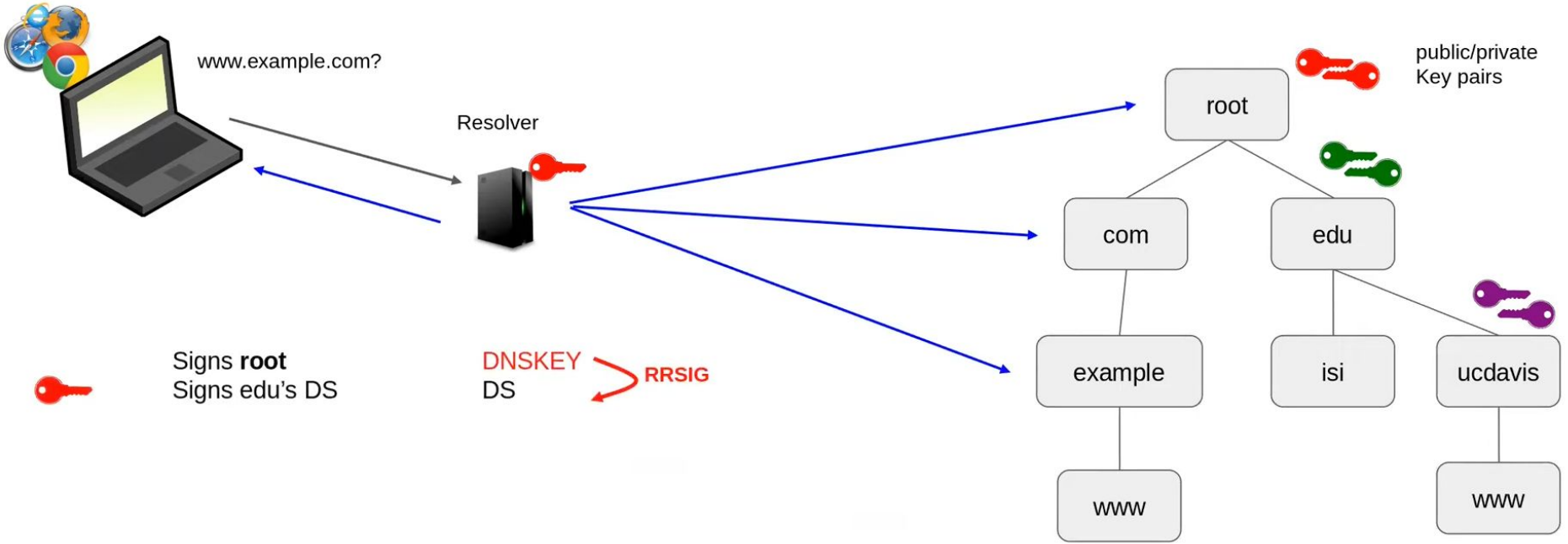
DNSSEC – hierarchical object signing security – “chain of trust”



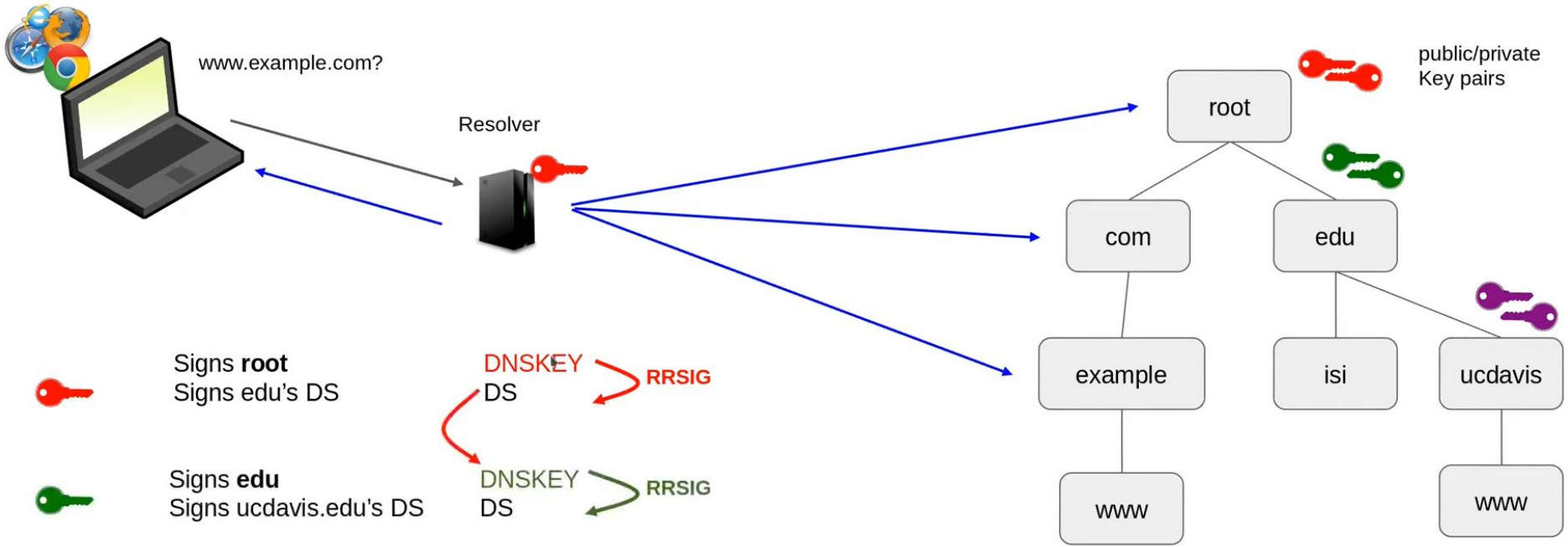
DNSSEC – hierarchical object signing security – “chain of trust”



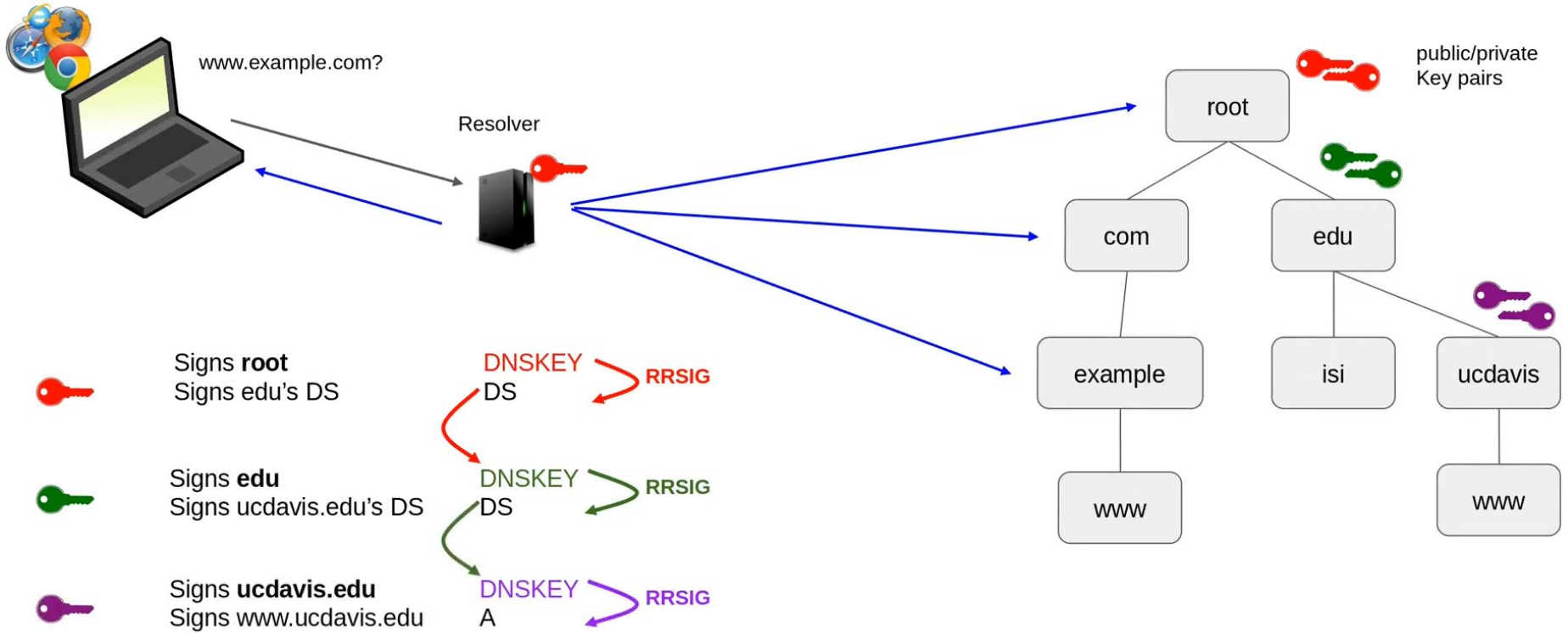
DNSSEC – hierarchical object signing security – “chain of trust”



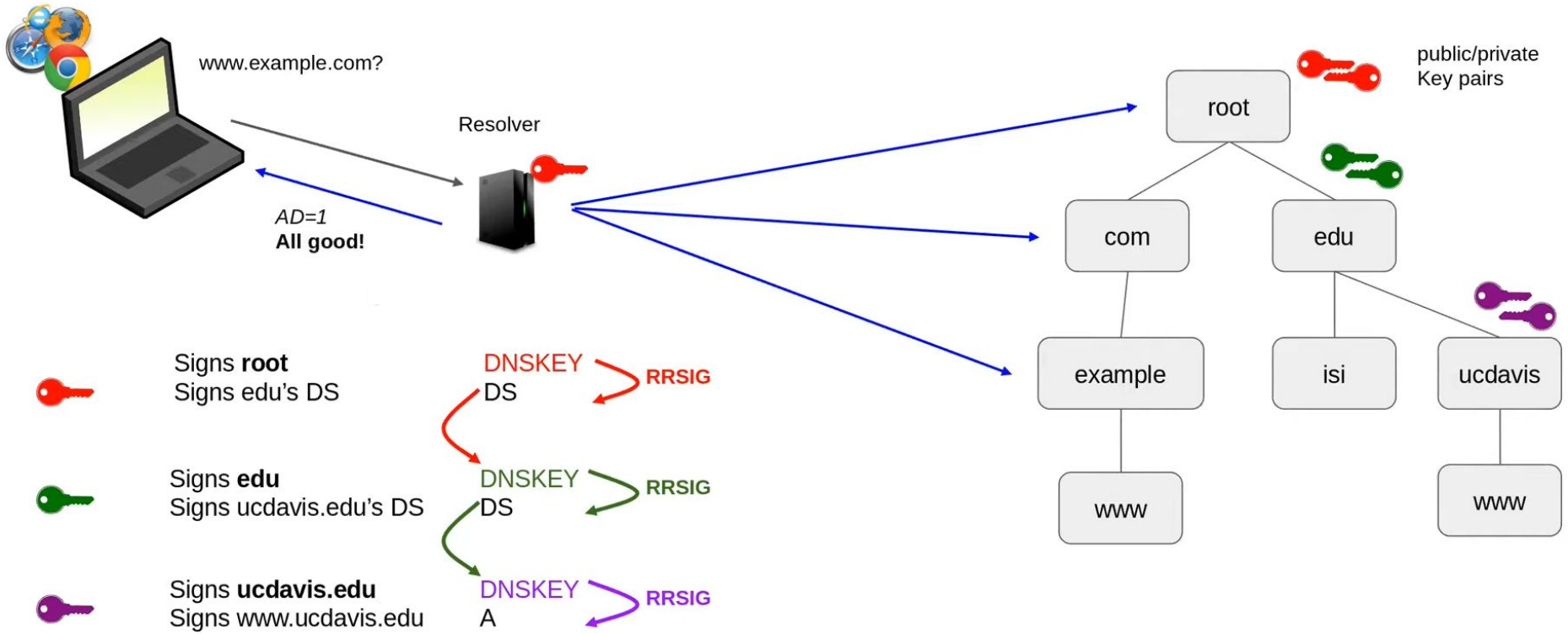
DNSSEC – hierarchical object signing security – “chain of trust”



DNSSEC – hierarchical object signing security – “chain of trust”



DNSSEC – hierarchical object signing security – “chain of trust”



DNSSEC Component Records

(we use the DNS to secure the DNS)

- **DNSKEY:** The public key used to verify a zone's signatures
- **DS:** A cryptographic hash of a child's *DNSKEY* for the parent to publish
- **CDS:** Key rolling record for parents to query in children
- **RRSIG:** A signature record that signs <CLASS, QTYPE, QNAME, CLASS>
  - Example: One RRSIG exists for all three *ucdavis.edu* NS records: *<IN, NS, ucdavis.edu>* but another exists for the single *<IN, A, ucdavis.edu>* IPv4 record
- **NSEC:** Proves other names don't exist between two names
  - NSEC apple orange: banana can't exist
- **NSEC3:** A hashing version of NSEC – use only if you need it (hint: you don't)

Real World Views of DNS~~SEC~~

```
# dig ucdavis.edu NS

;; flags: qr rd ra ad; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 6

;; ANSWER SECTION:
ucdavis.edu.            9985   IN NS dns-three.ucdavis.edu.
ucdavis.edu.            9985   IN NS dns-one.ucdavis.edu.
ucdavis.edu.            9985   IN NS dns-two.ucdavis.edu.

;; ADDITIONAL SECTION:
dns-two.ucdavis.edu.    9986   IN A      128.120.252.10
dns-one.ucdavis.edu.    9985   IN A      128.120.252.9
dns-three.ucdavis.edu.  9985   IN A      169.237.243.171
dns-two.ucdavis.edu.    9986   IN AAAA   2607:f810:3f0:2::2
dns-three.ucdavis.edu.  9987   IN AAAA   2607:f810:ce0:10::2
```

Real World Views of DNS~~SEC~~

Your resolver is DNSSEC validating (even if you didn't ask)

```
# dig ucdavis.edu NS

;; flags: qr rd ra ad; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 6

;; ANSWER SECTION:
ucdavis.edu.            9985   IN NS dns-three.ucdavis.edu.
ucdavis.edu.            9985   IN NS dns-one.ucdavis.edu.
ucdavis.edu.            9985   IN NS dns-two.ucdavis.edu.

;; ADDITIONAL SECTION:
dns-two.ucdavis.edu.    9986   IN A      128.120.252.10
dns-one.ucdavis.edu.    9985   IN A      128.120.252.9
dns-three.ucdavis.edu.  9985   IN A      169.237.243.171
dns-two.ucdavis.edu.    9986   IN AAAA   2607:f810:3f0:2::2
dns-three.ucdavis.edu.  9987   IN AAAA   2607:f810:ce0:10::2
```

Real World Views of DNSSEC

Your resolver is DNSSEC validating (even if you didn't ask)

Seconds left in the resolver's cache before it asks again

```
# dig ucdavis.edu NS

;; flags: qr rd ra ad; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 6

;; ANSWER SECTION:
ucdavis.edu.            9985    IN NS dns-three.ucdavis.edu.
ucdavis.edu.            9985    IN NS dns-one.ucdavis.edu.
ucdavis.edu.            9985    IN NS dns-two.ucdavis.edu.

;; ADDITIONAL SECTION:
dns-two.ucdavis.edu.    9986    IN A      128.120.252.10
dns-one.ucdavis.edu.    9985    IN A      128.120.252.9
dns-three.ucdavis.edu.  9985    IN A      169.237.243.171
dns-two.ucdavis.edu.    9986    IN AAAA   2607:f810:3f0:2::2
dns-three.ucdavis.edu.  9987    IN AAAA   2607:f810:ce0:10::2
```

Real World Views of DNSSEC

```
# dig +dnssec ucdavis.edu NS

;; ANSWER SECTION:
ucdavis.edu.    9936   IN NS dns-three.ucdavis.edu.
ucdavis.edu.    9936   IN NS dns-one.ucdavis.edu.
ucdavis.edu.    9936   IN NS dns-two.ucdavis.edu.
ucdavis.edu.    9936   IN RRSIG  NS 8 2 14400 20230106061807 20220106051807
                          40986 ucdavis.edu. KitFCqJ28ppQjZB……hK S+4=



;; ADDITIONAL SECTION:
dns-two.ucdavis.edu.  9986   IN A      128.120.252.10
dns-two.ucdavis.edu.  8409   IN RRSIG  A 8 3 14400 20230106061807 20220106051807
                             40986 ucdavis.edu. FOqxEMGl4h……+to lh8=

(other names/AAAA records truncated for brevity)
```

A signature on the NS record **SET**

Real World Views of DNSSEC

```
# dig +dnssec ucdavis.edu NS

;; ANSWER SECTION:
ucdavis.edu.    9936   IN NS dns-three.ucdavis.edu.
ucdavis.edu.    9936   IN NS dns-one.ucdavis.edu.
ucdavis.edu.    9936   IN NS dns-two.ucdavis.edu.
ucdavis.edu.    9936   IN RRSIG  NS 8 2 14400 20230106061807 20220106051807
                          40986 ucdavis.edu. KitFCqJ28ppQjZB……hK S+4=



;; ADDITIONAL SECTION:
dns-two.ucdavis.edu.  9986   IN A      128.120.252.10
dns-two.ucdavis.edu.  8409   IN RRSIG  A 8 3 14400 20230106061807 20220106051807
                             40986 ucdavis.edu. FOqxEMGl4h……+to lh8=

(other names/AAAA records truncated for brevity)
```

Real World Views of DNSSEC

A signature on the NS record **SET**

```
# dig +dnssec ucdavis.edu NS

;; ANSWER SECTION:
ucdavis.edu.    9936   IN NS dns-three.ucdavis.edu.
ucdavis.edu.    9936   IN NS dns-one.ucdavis.edu.
ucdavis.edu.    9936   IN NS dns-two.ucdavis.edu.
ucdavis.edu.    9936   IN RRSIG  NS 8 2 14400 20230106061807 20220106051807
                       40986 ucdavis.edu. KitFCqJ28ppQjZB……hK S+4=
```

Key lookup hint

```
;; ADDITIONAL SECTION:
dns-two.ucdavis.edu.   9986   IN A      128.120.252.10
dns-two.ucdavis.edu.   8409   IN RRSIG  A 8 3 14400 20230106061807 20220106051807
                              40986 ucdavis.edu. FOqxEMGl4h……+to lh8=

(other names/AAAA records truncated for brevity)
```

Real World Views of DNSSEC

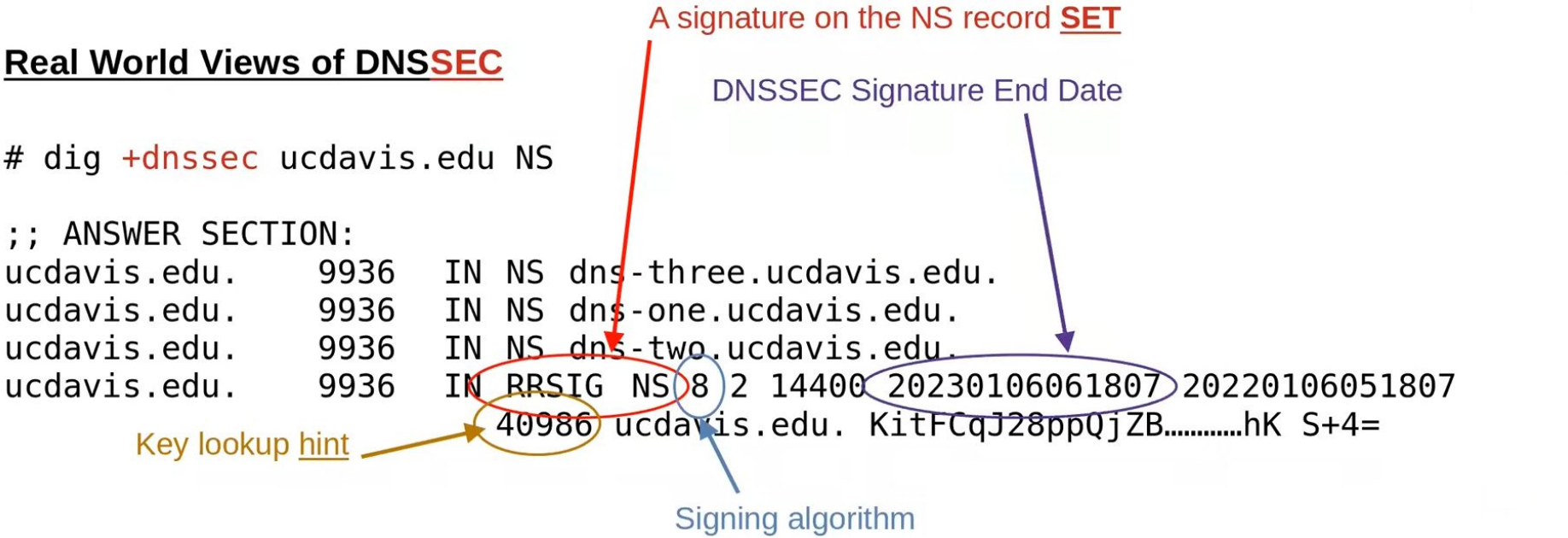
A signature on the NS record **SET**

```
# dig +dnssec ucdavis.edu NS

;; ANSWER SECTION:
ucdavis.edu.    9936   IN NS dns-three.ucdavis.edu.
ucdavis.edu.    9936   IN NS dns-one.ucdavis.edu.
ucdavis.edu.    9936   IN NS dns-two.ucdavis.edu.
ucdavis.edu.    9936   IN RRSIG  NS 8 2 14400 20230106061807 20220106051807
                       40986 ucdavis.edu. KitFCqJ28ppQjZB……hK S+4=

;; ADDITIONAL SECTION:
dns-two.ucdavis.edu.  9986   IN A      128.120.252.10
dns-two.ucdavis.edu.  8409   IN RRSIG  A 8 3 14400 20230106061807 20220106051807
                             40986 ucdavis.edu. FOqxEMGl4h……+to lh8=

(other names/AAAA records truncated for brevity)
```

Key lookup hint

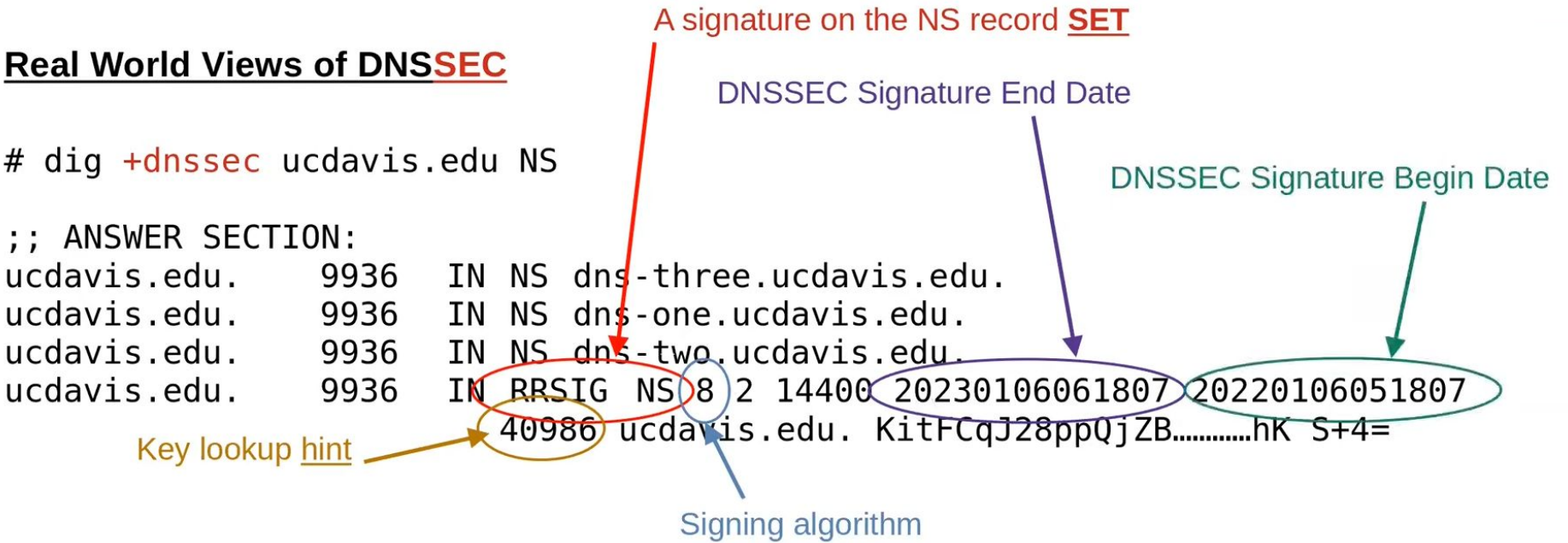
Signing algorithm

Real World Views of DNSSEC




```
# dig +dnssec ucdavis.edu NS

;; ANSWER SECTION:
ucdavis.edu.    9936   IN NS dns-three.ucdavis.edu.
ucdavis.edu.    9936   IN NS dns-one.ucdavis.edu.
ucdavis.edu.    9936   IN NS dns-two.ucdavis.edu.
ucdavis.edu.    9936   IN RRSIG  NS 8 2 14400 20230106061807 20220106051807
                                 40986 ucdavis.edu. KitFCqJ28ppQjZB……hK S+4=

;; ADDITIONAL SECTION:
dns-two.ucdavis.edu.  9986   IN A      128.120.252.10
dns-two.ucdavis.edu.  8409   IN RRSIG  A 8 3 14400 20230106061807 20220106051807
                             40986 ucdavis.edu. FOqxEMGl4h……+to lh8=

(other names/AAAA records truncated for brevity)
```

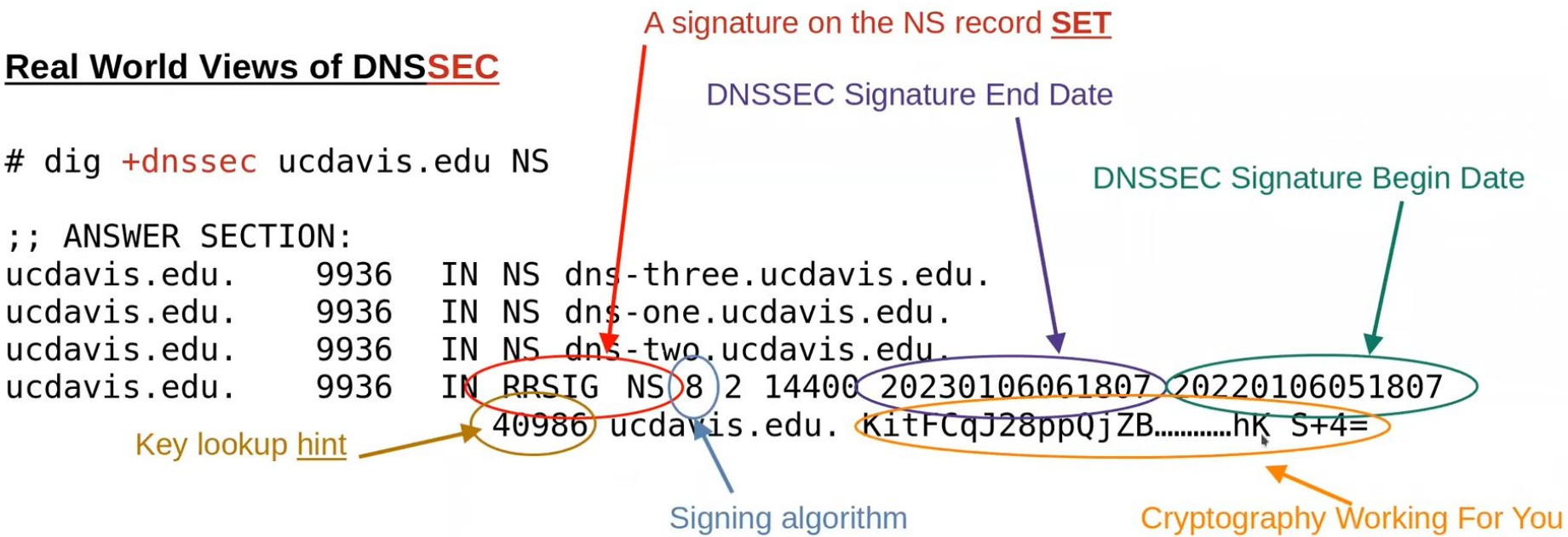
Real World Views of DNSSEC




```
# dig +dnssec ucdavis.edu NS

;; ANSWER SECTION:
ucdavis.edu.    9936   IN NS dns-three.ucdavis.edu.
ucdavis.edu.    9936   IN NS dns-one.ucdavis.edu.
ucdavis.edu.    9936   IN NS dns-two.ucdavis.edu.
ucdavis.edu.    9936   IN RRSIG  NS 8 2 14400 20230106061807 20220106051807
                       40986 ucdavis.edu. KitFCqJ28ppQjZB……hK S+4=

;; ADDITIONAL SECTION:
dns-two.ucdavis.edu.   9986   IN A      128.120.252.10
dns-two.ucdavis.edu.   8409   IN RRSIG  A 8 3 14400 20230106061807 20220106051807
                              40986 ucdavis.edu. FOqxEMGl4h……+to lh8=
```

(other names/AAAA records truncated for brevity)

Real World Views of DNSSEC




```
# dig +dnssec ucdavis.edu NS

;; ANSWER SECTION:
ucdavis.edu.    9936   IN NS dns-three.ucdavis.edu.
ucdavis.edu.    9936   IN NS dns-one.ucdavis.edu.
ucdavis.edu.    9936   IN NS dns-two.ucdavis.edu.
ucdavis.edu.    9936   IN RRSIG NS 8 2 14400 20230106061807 20220106051807
                       40986 ucdavis.edu. KitFCqJ28ppQjZB……hK S+4=

;; ADDITIONAL SECTION:
dns-two.ucdavis.edu.  9986   IN A     128.120.252.10
dns-two.ucdavis.edu.  8409   IN RRSIG A 8 3 14400 20230106061807 20220106051807
                             40986 ucdavis.edu. FOqxEMGl4h……+to lh8=
```

(other names/AAAA records truncated for brevity)

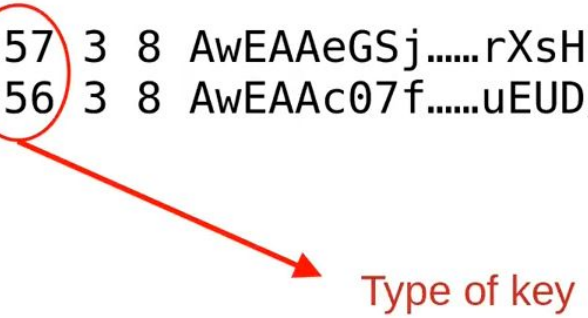
Real World Views of DNSSEC / DNSKEYs

```
# dig ucdavis.edu DNSKEY

;; ANSWER SECTION:
ucdavis.edu.    79818  IN DNSKEY 257 3 8 AwEAAeGSj……rXsHE=
ucdavis.edu.    79818  IN DNSKEY 256 3 8 AwEAAc07f……uEUD/7
```

ZSK (256): signs ALL records
"zone signing key"
KSK (257): signs only the keys themselves
"key signing key"
DS records are created from this one to give to your parent

Real World Views of DNSSEC / DNSKEYs

```
# dig ucdavis.edu DNSKEY

;; ANSWER SECTION:
ucdavis.edu.    79818  IN DNSKEY 257 3 8 AwEAAeGSj……rXsHE=
ucdavis.edu.    79818  IN DNSKEY 256 3 8 AwEAAc07f……uEUD/7
```



ZSK (256): signs ALL records
"zone signing key"
KSK (257): signs only the keys themselves
"key signing key"
DS records are created from this one to give to your parent

Why does DNSSEC have 2 key types????

ZSK (256): signs ALL records
"zone signing key"

Designed to be rotated frequently if desired
Requires no parent communication to roll them
Current software can **entirely automate this**
If rotated frequently, can generally be shorter in size

KSK (257): signs only the keys themselves
"key signing key"

DS records are created from KSKs to give to your parent
Harder to rotate
Often longer in size
Rolling only requires special considerations

Note: you can have just a ZSK – the DS in the parent should point to the ZSK

Real World Views of DNSSEC / DNSKEYs → DS at a parent

```
# dig ucdavis.edu DNSKEY

;; ANSWER SECTION:
ucdavis.edu.    79818  IN DNSKEY 257 3 8 AwEAAeGSj……rXsHE=
ucdavis.edu.    79818  IN DNSKEY 256 3 8 AwEAAc07f……uEUD/7



# dig @k.edu-servers.net. ucdavis.edu DS
ucdavis.edu.    86400  IN DS 31642 8 2 2F9EC3E5……48CB394
ucdavis.edu.    86400  IN DS 31642 8 1 71F9C0…6E8FCB



SHA256 (2):   the algorithm you really MUST have
SHA1   (1):   stop using this at this point – no longer needed and less secure
```

Real World Views of DNSSEC / DNSKEYs → DS at a parent

```
# dig ucdavis.edu DNSKEY

;; ANSWER SECTION:
ucdavis.edu.    79818  IN DNSKEY 257 3 8 AwEAAeGSj……rXsHE=
ucdavis.edu.    79818  IN DNSKEY 256 3 8 AwEAAc07f……uEUD/7
```

Key lookup hint

```
# dig @k.edu-servers.net. ucdavis.edu DS
ucdavis.edu.    86400  IN DS 31642 8 2 2F9EC3E5……48CB394
ucdavis.edu.    86400  IN DS 31642 8 1 71F9C0…6E8FCB
```

```
SHA256 (2):   the algorithm you really MUST have
SHA1   (1):   stop using this at this point – no longer needed and less secure
```

Real World Views of DNSSEC / DNSKEYs → DS at a parent

```
# dig ucdavis.edu DNSKEY

;; ANSWER SECTION:
ucdavis.edu.     79818  IN DNSKEY 257 3 8 AwEAAeGSj……rXsHE=
ucdavis.edu.     79818  IN DNSKEY 256 3 8 AwEAAc07f……uEUD/7
```

Key lookup hint

```
# dig @k.edu-servers.net. ucdavis.edu DS
ucdavis.edu.     86400  IN DS 31642 8 2 2F9EC3E5……48CB394
ucdavis.edu.     86400  IN DS 31642 8 1 71F9C0…6E8FCB
```

Hashing type

```
SHA256 (2):   the algorithm you really MUST have
SHA1   (1):   stop using this at this point – no longer needed and less secure
```

One DS → DNSKEY match is always REQUIRED

```
# dig ucdavis.edu DNSKEY

;; ANSWER SECTION:
ucdavis.edu.    79818  IN DNSKEY 257 3 8 AwEAAeGSj……rXsHE=
ucdavis.edu.    79818  IN DNSKEY 256 3 8 AwEAAc07f……uEUD/7


# dig @k.edu-servers.net. ucdavis.edu DS
ucdavis.edu.    86400  IN DS 31642 8 2 2F9EC3E5……48CB394
```

One DS → DNSKEY match is always REQUIRED

```
# dig ucdavis.edu DNSKEY

;; ANSWER SECTION:
ucdavis.edu.    79818  IN DNSKEY 257 3 8 AwEAAeGSj……rXsHE=
ucdavis.edu.    79818  IN DNSKEY 25[illegible] 3 8 AwEAAc07f……uEUD/7


# dig @k.edu-servers.net. ucdavis.edu DS
ucdavis.edu.    86400  IN DS 31642 8 2 2F9EC3E5……48CB394
```

CRITICAL for validation to succeed:

1. **One DS record in your parent must match one DNSKEY in your zone at all times**

One DS → DNSKEY match is always REQUIRED

```
# dig ucdavis.edu DNSKEY

;; ANSWER SECTION:
ucdavis.edu.     79818  IN DNSKEY 257 3 8 AwEAAeGSj……rXsHE=
ucdavis.edu.     79818  IN DNSKEY 256 3 8 AwEAAc07f……uEUD/7


# dig @k.edu-servers.net. ucdavis.edu DS
ucdavis.edu.     86400  IN DS 31642 8 2 2F9EC3E5……48CB394
```

CRITICAL for validation to succeed:

1. **One DS record in your parent must match one DNSKEY in your zone at all times**
2. **This includes all records potentially in resolver caches – TTLs really really matter**

OK, but how does one actually roll keys?

(and what is rolling???)

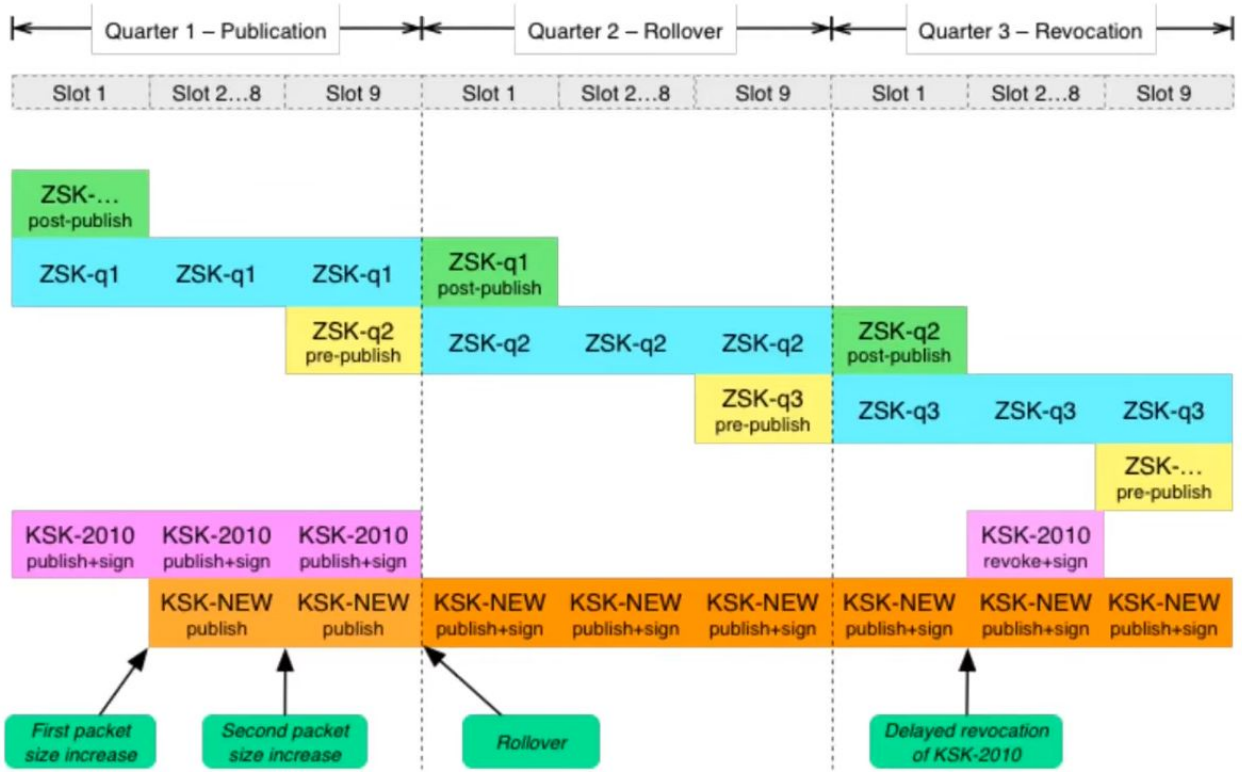


Figure 1. Rollover Scheduling

ICANN's multi-step process for rolling the root zone's KSK

Wes' rules to remember...

- Generic rule for changing **ANY** data:
  1. Add your new records first, then **WAIT**
  2. Publish to parent, then **WAIT**

     (NS, glue, DNSSEC DS)
  3. Remove old records at parent, then **WAIT**
  4. Remove your old records

Image source: Vivek Gite

WAIT := 2x max(your TTL, parent's TTL)
This is for resolver caches to refill

DNSSEC Deployment Recommendations

- Use modern tools
  - “back in my day” we had to self-manage keys, signing, rollovers, etc
  - These days excellent tools do all the hard work: infoblox, registratars (google), knot, bind, …
- Use modern algorithms
  - DS records: SHA256 only in your parent
  - Use elliptic curve algorithms (eg, algorithm 13)
  - Don’t use NSEC3
- Monitor monitor monitor
  - Do all your parent/child records match?
  - Are your signatures still good far enough out?

DNSSEC TTL Recommendations

The only real DNSSEC consideration:

- DNSKEY queries can cause fragmentation
- You might slightly larger TTLs on these

- Short TTLs:
  - Ability to change values quickly
  - Better for load balancing
- Long TTLs:
  - Better protection against outages
  - Faster responses for end-users *(resolvers don't need to send queries if they have the answer!)*
  - Lower traffic and lower scalability costs
- Personal opinion: 3600 (1 hour) <= YOUR TTL <= 86400 (1 day)
- Further reading:
  - "Cache Me If You Can: Effects of DNS Time-to-Live" – IMC-2018
    - *Giovane Moura, Richardo de O.Schmidt, John Heiemann, Wes Hardaker*

~~DNSSEC~~ TTL Recommendations

The only real DNSSEC consideration:

- DNSKEY queries can cause fragmentation
- You might slightly larger TTLs on these

- Short TTLs:
  - Ability to change values quickly
  - Better for load balancing
- Long TTLs:
  - Better protection against outages
  - Faster responses for end-users *(resolvers don't need to send queries if they have the answer!)*
  - Lower traffic and lower scalability costs
- Personal opinion: 3600 (1 hour) <= YOUR TTL <= 86400 (1 day)
- Further reading:
  - "Cache Me If You Can: Effects of DNS Time-to-Live" – IMC-2018
    - *Giovane Moura, Richardo de O.Schmidt, John Heiemann, Wes Hardaker*

DoT, DoH: Path Security At Work

Zeek (A useful tool)

Zeek — put a pcap in and get a bunch of log files out.

Installation: sudo apt install zeek

Running zeek on a pcap gives a list of files, e.g.,

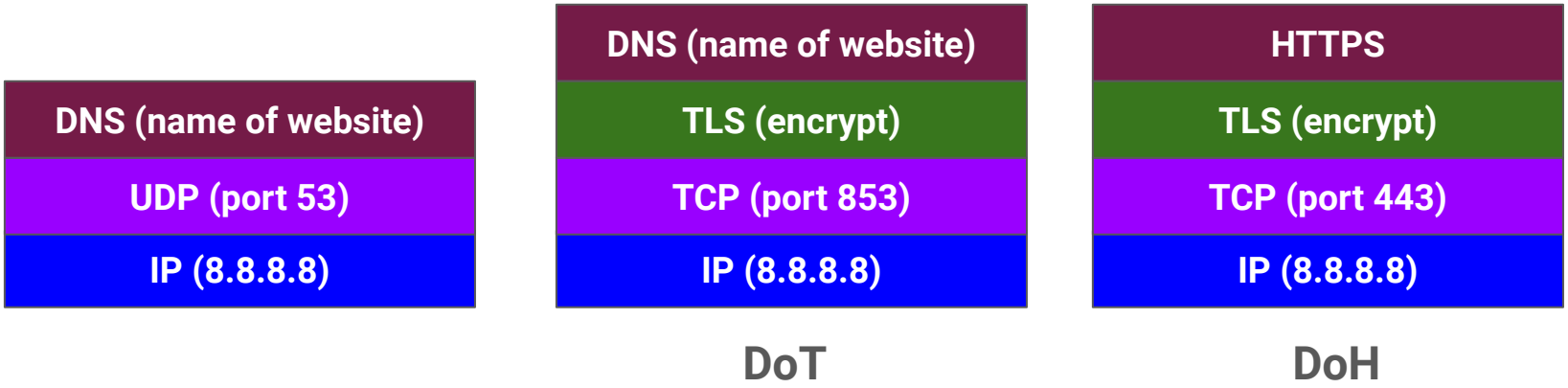
conn.log → contains src/dst ip, src/dst port,

dns.log → contains dns requests

less -S conn.log (displays results in an organized column format)

Exercise: Run zeek on a pcap and analyze the details.

DNS over TLS (DoT) and DNS over HTTP (DoH)



DNS over HTTP (DoH)

Exercise: Decrypt DNS traffic using DoH.

1. Generate DNS traffic and see that in wireshark.
2. Turn DoH on in Firefox and see how the traffic changes.
3. How to decrypt this traffic?

DoT can be observed in wireshark using the same process with some tweaks.

TLSv1.2 vs TLSv1.3

Difference?

Encrypted Server Name Indication (ESNI)

Issue: people can tell where you are going even if you’re using TLS.

Exercise: Find all the SNI values within a capture?

What is Server Name Indication?

Server Name Indication (SNI) is an extension to the Transport Layer Security (TLS) computer networking protocol by which a client indicates which hostname it is attempting to connect to at the start of the handshaking process.

Turn to wireshark to show SNI.

Continued…..

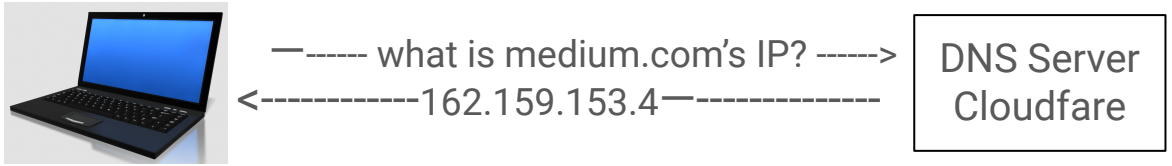
Encrypted Server Name Indication (ESNI) is an extension to TLS 1.3 which prevents eavesdroppers from knowing the domain name of the website network users are connecting to. When combined with encrypted DNS, it is not possible to know which websites a user is visiting.

DoH or DoT should be in place before ESNI.

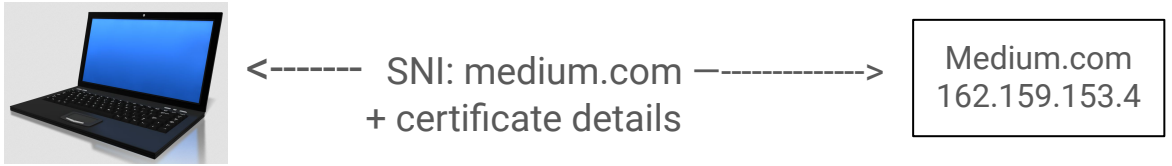
ESNI to Protect TLS

To visit a TLS website:

1. Get IP from DNS server



2. Connect to medium's IP using TLS



ESNI to Protect TLS

To visit a TLS website:

1. Get IP from DNS server

When DoH is enabled, we will only see that someone is connecting to the Cloudfare DNS server? But, we don't know what they are asking….

—------ what is ????????????? IP? ------>

<---------- ????????????? —----------

DNS Server
Cloudfare

2. Connect to medium's IP using TLS

<------- SNI: medium.com —---------->
+ certificate details

Medium.com
162.159.153.4

In TLSv1.2, we would see SNI and certificate details with DoH enabled.

ESNI to Protect TLS

To visit a TLS website:

1. Get IP from DNS server

When DoH is enabled, we will only see that someone is connecting to the Cloudfare DNS server? But, we don't know what they are asking….

—------ what is ????????????? IP? ------>

<---------- ????????????? —----------

DNS Server
Cloudfare

2. Connect to medium's IP using TLS

<------- SNI: medium.com —---------->

Medium.com
162.159.153.4

In TLSv1.3, the certificate details are encrypted.

ESNI to Protect TLS

To visit a TLS website:

1. Get IP from DNS server

When DoH is enabled, we will only see that someone is connecting to the Cloudfare DNS server? But, we don't know what they are asking….

—------ what is ?????????????? IP? ------>

<---------- ????????????? —----------

DNS Server
Cloudfare

2. Connect to medium's IP using TLS

<------- ???????????????? —---------->

Medium.com
162.159.153.4

In TLSv1.3, the ESNI encrypts the SNI.

The websites that support ESNI will have their public key in DNS records so that the clients can use that to encrypt their traffic.

ESNI to date is not heavily supported.

We are going towards intercepting the TLS connection traffic to get information about what someone is connecting to.

Shortcomings of ESNI

Incomplete protection.

For example, during session resumption, the Pre-Shared Key extension could, legally, contain a cleartext copy of exactly the same server name that is encrypted by ESNI.

Real-world use of ESNI has exposed interoperability and deployment challenges that prevented it from being enabled at a wider scale.

https://blog.mozilla.org/security/2021/01/07/encrypted-client-hello-the-future-of-esni-in-firefox/

Encrypted Client Hello (ECH)

Firefox decided to kill ESNI.

But there is a replacement for encrypting the destination of TLS traffic.

The standard that replaced ESNI is called ECH.

Useful Resources:

- 12 days of Defense series (https://www.youtube.com/watch?v=9H8wAXg2vrs)
- Zeek (https://zeek.org/get-zeek/)
- Malware Traffic Analysis.net (https://www.malware-traffic-analysis.net/index.html)
- Wes Hardaker lecture on DNS security (https://www.linkedin.com/posts/whardaker_dns-security-overview-activity-6901910613824565250-XGmL?trk=public_profile_share_view)