

DNS and DNSSEC

Geoff Huston

How can you tell whether a DNS
response is *true* or not?

How can you tell whether a DNS
response is *true* or not?

```
$ dig +short www.commbank.com.au  
prd.akamai.cba.commbank.edgekey.net.  
e6109.x.akamaiedge.net.  
104.116.115.138
```

How can you tell whether a DNS response is *true* or not?

```
$ dig +short www.commbank.com.au  
prd.akamai.cba.commbank.edgekey.net.  
e6109.x.akamaiedge.net.  
104.116.115.138
```

Where did this answer come from?

- It references a name associated with an Akamai cloud product, not the Commonwealth Bank
- Akamai is used by many folk – not all of them are honest
- So why should I enter my username and password into the web page that is found at this address?

How can you tell whether a DNS response is *true* or not?

```
$ dig +short www.commbank.com.au  
prd.akamai.cba.commbank.edgekey.net  
e6109.x.akamaiedge.net  
104.116.115.120
```

How can I tell if the

- Where did this answer come from?
- It references a name that is not the Commonwealth Bank
 - Akamai is used by many folk – not all of them are honest
 - So why should I enter my username and password into the web page that is found at this address?

DNS is telling me lies?

The Origins of the DNS

- The DNS was created as a replacement for a static list of hosts and addresses - /etc/hosts.txt
 - Which was a list of host names and their IP addresses

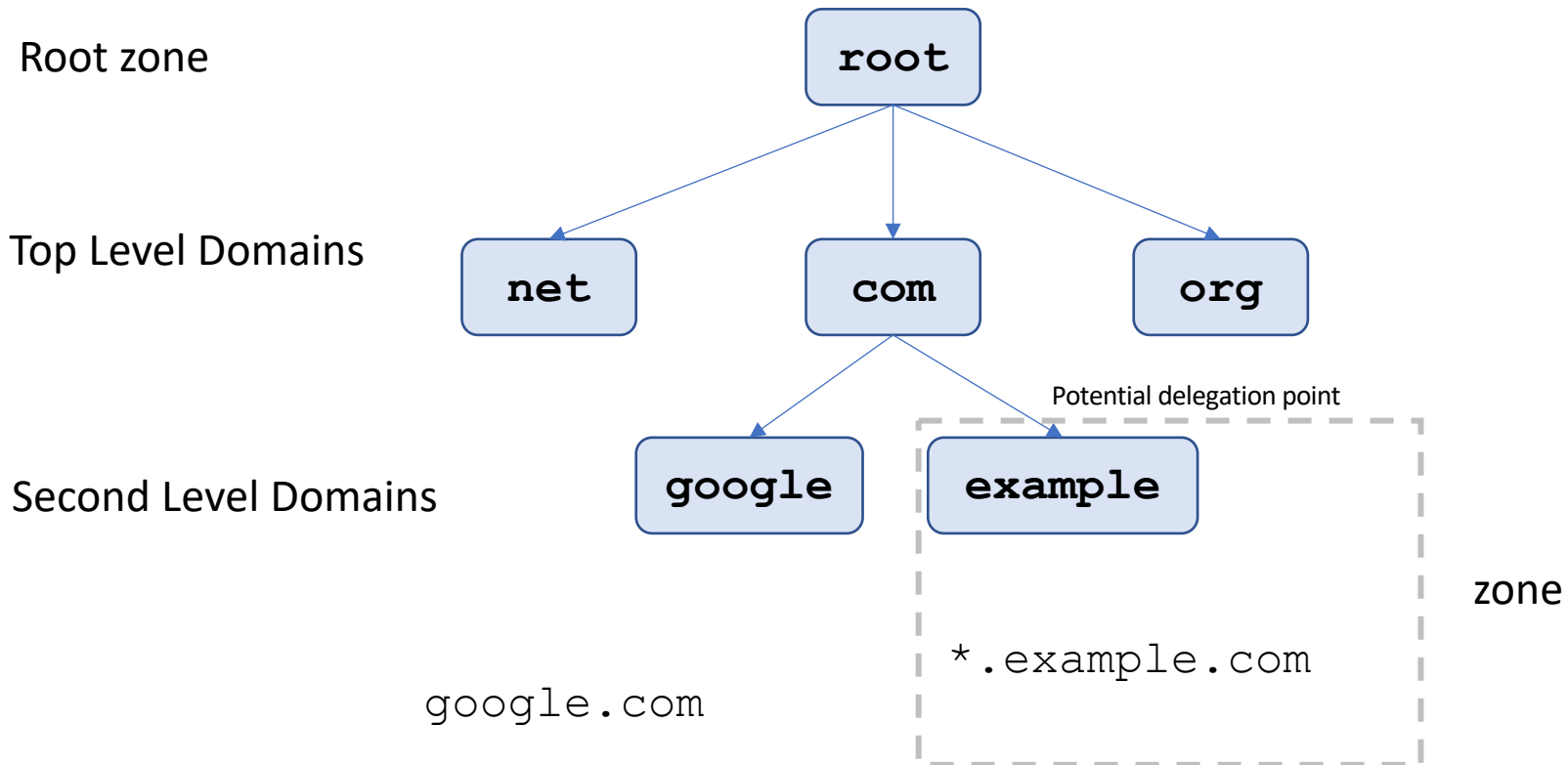
```
localhost      127.0.0.1
example.com    192.0.2.1
```

- To resolve a name you look up the name in /etc/hosts.txt and use the result

Question: What are the problems with this approach?

The DNS Name Space

is a structured hierarchy:



RFC 1034
RFC 1035

The DNS as a Distributed Database

- Each Zone is served by one or more *authoritative servers*
- Each server responds to *queries*
- Responses may be one of:
 - The queried data
 - A *referral* to a delegated authoritative server set
 - No such data
 - No such name

DNS "Resolvers"

- To query this DNS database we use specialised database query engines that are termed DNS *Recursive Resolvers*
- A recursive resolver performs a sequence of *discovery queries* to establish the authoritative server that can answer the query for this domain name
- It also uses a local *cache* to re-use responses for subsequent queries (these caches are super-important – without local caching the DNS would melt under the load!)

Resolving a DNS name is far more complex than it might seem

I want to resolve **www.commbank.com.au** in the DNS

Which DNS server is *authoritative* for **www.commbank.com.au**?

↳ I don't know.

But the servers for **commbank.com.au** should know

So lets ask one of them

Which DNS server is *authoritative* for **commbank.com.au**?

↳ I don't know.

But the servers for **com.au** should know

So lets ask one of them

Which server is authoritative for **.com.au**?

↳ I don't know

But **.au** servers should know

Which server is authoritative for **.au**?

↳ I don't know

But ROOT servers should know

So now lets start the name resolution process...

Resolving a DNS name is far more complex than it might seem

```
$ dig www.commbank.com.AU @a.root-servers.net
```

```
; <<>> DiG 9.18.11 <<>> www.commbank.com.AU @a.root-servers.net  
;; global options: +cmd  
;; Got answer:  
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15415  
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 13  
;; WARNING: recursion requested but not available
```

```
;; OPT PSEUDOSECTION:  
; EDNS: version: 0, flags:; udp: 4096  
;; QUESTION SECTION:  
;www.commbank.com.AU.          IN      A
```

```
;; AUTHORITY SECTION:
```

```
AU.          172800 IN      NS      d.AU.  
AU.          172800 IN      NS      q.AU.  
AU.          172800 IN      NS      t.AU.  
AU.          172800 IN      NS      s.AU.  
AU.          172800 IN      NS      r.AU.  
AU.          172800 IN      NS      c.AU.
```




Resolving a DNS name is far more complex than it might seem

```
$ dig www.commbank.com.AU @r.au
```

```
; <<>> DiG 9.18.11 <<>> www.commbank.com.AU @r.au
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 57200
;; flags: qr rd; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;www.commbank.com.AU.          IN      A

;; AUTHORITY SECTION:
commbank.com.au.    900     IN      NS      ns-2013.awsdns-59.co.uk.
commbank.com.au.    900     IN      NS      ns-302.awsdns-37.com.
commbank.com.au.    900     IN      NS      ns-748.awsdns-29.net.
commbank.com.au.    900     IN      NS      ns-1037.awsdns-01.org.
```



Resolving a DNS name is far more complex than it might seem

```
$ dig www.commbank.com.AU @ns-302.awsdns-37.com.
```

```
; <<>> DiG 9.18.11 <<>> www.commbank.com.AU @ns-302.awsdns-37.com.
```

```
;; global options: +cmd
```

```
;; Got answer:
```

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9009
```

```
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 1
```

```
;; WARNING: recursion requested but not available
```

```
;; OPT PSEUDOSECTION:
```

```
; EDNS: version: 0, flags:; udp: 4096
```

```
;; QUESTION SECTION:
```

```
;www.commbank.com.AU.          IN      A
```

```
;; ANSWER SECTION:
```

```
www.commbank.com.AU.3600      IN      CNAME  prd.akamai.cba.commbank.edgekey.net.
```



Resolving a DNS name is far more complex than it might seem

- This is an alias, so we now have to resolve the name:
`prd.akamai.cba.commbank.edgekey.net.`
- But this is also an alias, so we have to resolve the name:
`e6109.x.akamaiedge.net`
- Finally:

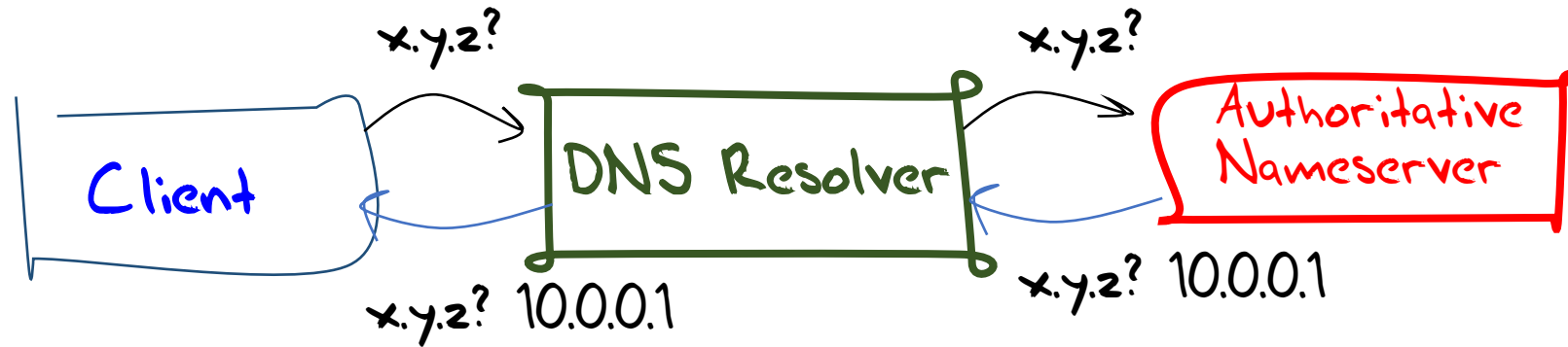
```
$ dig +short e6109.x.akamaiedge.net.  
104.116.115.138
```

The DNS is far more complex than it might seem

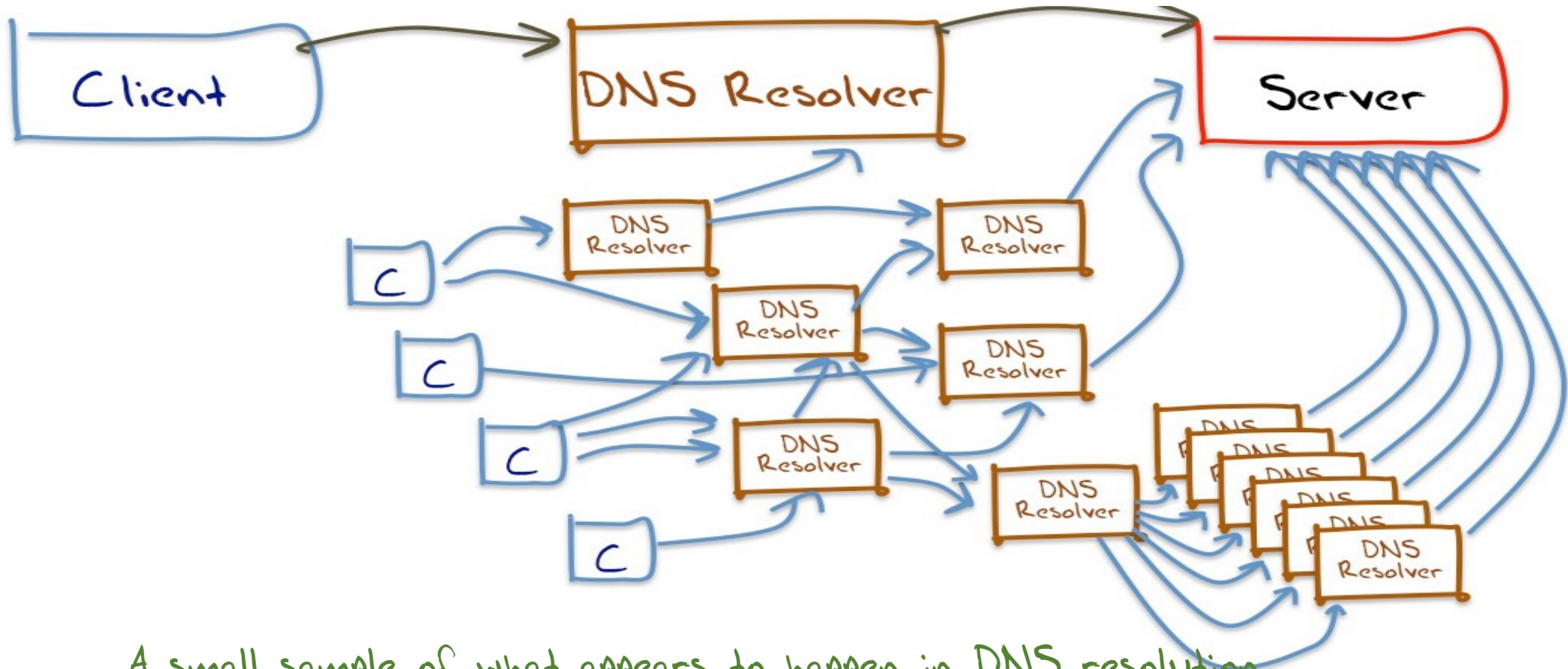
- Phew!
- That simple name resolution query actually took 13 queries to discover who to ask and what query name to use, and a final query for the data.
- What's going on here?

Understanding DNS Resolvers is "tricky"

What we would like to think happens in DNS resolution!



Understanding DNS Resolvers is "tricky"



A small sample of what appears to happen in DNS resolution

The DNS and Trust

- When you pass a question to a recursive resolver you naturally assume that the answer is *authentic*
 - i.e. the answer is an accurate representation of the current contents of the zone
- But how can you *confirm* that this is the case?
 - You don't have a genuine copy of the zone file to compare it to the response
 - You don't know if the response was generated from a cache or directly from a query to one of the zone's authoritative servers
 - You can't even tell *who* provided the response!

The DNS is vulnerable

- It's in the clear
 - So anyone on the wire can tap the DNS transactions
- It's UDP
 - So its at risk from injection, substitution and fragmentation attacks
- Its unprotected
 - The DNS works on the principle that if you send a query to the "right" IP address then you can trust the answer you get back that contains this same address as a source address

Why attack the DNS?

Because perverting the infrastructure does not require a successful attack on the host

For example, If you can control the resolution of a DNS name you can:

- Coerce an automated CA to issue a domain name certificate using the attacker's public key by using a faked DNS response

- Then alter the DNS resolution of the target name to the attacker's site

- And use a faked service that passes a TLS test

How can you detect DNS attacks that alter DNS responses?

How can you detect DNS attacks that alter DNS responses?

- Use digital signatures in the DNS:
 - Associate a public/private key pair with a DNS zone, and use the private key to generate a digital signature for each zone entry
 - Deliver the digital signature along with the DNS response
- A DNS response is *authentic* if:
 - the client can be satisfied that the zone's private key has been used to sign the digital signature associated with the DNS response record, and
 - the DNS client can be assured that the zone's public/private key pair is authentically associated with the zone
- We use a digital signature framework for DNS called "DNSSEC"

How does DNSSEC do "Trust"?

- If we are talking "trust" when should we be talking X.509 public key certificates as well?
 - No, no X.509 certificate is needed or used in DNSSEC
- This entire process is based on the keys themselves
- Its strength lies in the transitive trust model of interlocking keys...

DNSSEC Design Basics

- DNSSEC does not alter the DNS in any way, nor does it alter the basic query/response protocol
- DNSSEC adds 5 new Resource Record Types:
 - RRSIG – the digital signature of a zone resource record
 - DNSKEY – the public key(s) used to "sign" the zone
 - DS – the hash of the zones entry key, placed in the parent zone
 - NSEC – a spanning record used to sign across the "gaps" in a zone
 - NSEC3 – a variant of the NSEC spanning record used to sign across the "gaps" in a zone
- If a query sets the DNSSEC OK flag then the signature (RRSIG record) is added to the response (if one exists in the zone)

What's DNSSEC?

It's the ability to add digital signatures to DNS responses.

```
$ dig +dnssec www.potaroo.net AAAA @127.0.0.1
```

```
; <<>> DiG 9.18.2 <<>> +dnssec www.potaroo.net AAAA @127.0.0.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 36348
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; COOKIE: 037e7ff2970bd29801000000628b47b76d96ecc2d227fae5 (good)
;; QUESTION SECTION:
;www.potaroo.net.                IN                AAAA

;; ANSWER SECTION:
www.potaroo.net.                6394              IN                AAAA              2401:2000:6660::108
www.potaroo.net.                6394              IN                RRSIG            AAAA 13 3 6400 20320331235230 20220324225230 41284 potaroo.net.
W9CDfQ3nCl35ZuFCIxzg+RI4f+L8O/RRpJLwpPVq6wMgP5CPpP8sSiQc ySCB5sclFBN5aeqG1/jOBeywVYfp0g==

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)
;; WHEN: Mon May 23 18:37:11 AEST 2022
;; MSG SIZE rcvd: 207
```



response

digital signature

So what?

- If the client can *validate* this digital signature, then it can be assured that:
 - The response the client received is **authentic** and **complete**
 - The response is **current**
- It doesn't matter how the DNS client learned this response: if the response correctly validates then the data is a genuine and complete extract from the authoritative zone file
- If it wasn't signed and validated then you really can't tell if the data has been altered by some intermediary

What about non-existence?

\$ dig +dnssec _80_tcp.www.potaroo.net A @127.0.0.1 *Query for a non-existent name*

```
; <<>> DiG 9.18.2 <<>> +dnssec _80_tcp.www.potaroo.net A @127.0.0.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 47442
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 1
```

```
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
; COOKIE: b6c04852ce47b40901000000628b4b20b3104cdb1bfb9e90 (good)
;; QUESTION SECTION:
;_80_tcp.www.potaroo.net.      IN      A
```

```
;; AUTHORITY SECTION:
potaroo.net.      6018      IN      SOA      ns1.potaroo.net. gih.potaroo.net. 2022032501 10800 3600 3600000 6400
potaroo.net.      6018      IN      RRSIG    SOA 13 2 6400 20320331235230 20220324225230 41284 potaroo.net.
oQZTmjoMBb8r8FUjHbp+62ZjSV1aXU9GI6K28ngh6RXHFPWmzTJIIEA dCkf7fzA3d9ANqm5I5UiMikBRPceFw==
www.potaroo.net. 6018      IN      NSEC     _443._tcp.www.potaroo.net. A AAAA RRSIG NSEC
www.potaroo.net. 6018      IN      RRSIG    NSEC 13 3 6400 20320331235230 20220324225230 41284 potaroo.net.
lhP13N+YR6m3dBYLUxfgv8fGsuiF4f14UcpznpqlevlJyEumLgHtzUV Y6k6MXpiygGql70KzZidqzAhglVCcQ==
```

There are no names between these two labels

```
;; Query time: 6 msec
;; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)
;; WHEN: Mon May 23 18:51:44 AEST 2022
;; MSG SIZE rcvd: 396
```

NXDOMAIN digital signature

What about non-existence?

```
$ dig +dnssec www.potaroo.net TXT @127.0.0.1
```

Query for a non-existent record type

```
; <<>> DiG 9.18.2 <<>> +dnssec www.potaroo.net TXT @127.0.0.1
```

```
;; global options: +cmd
```

```
;; Got answer:
```

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32884
```

```
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 0, AUTHORITY: 4, ADDITIONAL: 1
```

NODATA flags

```
;; OPT PSEUDOSECTION:
```

```
; EDNS: version: 0, flags: do; udp: 4096
```

```
; COOKIE: 660df460a7f4ed0101000000628b49a2d9c0f21dc394dd81 (good)
```

```
;; QUESTION SECTION:
```

```
www.potaroo.net.                IN                TXT
```

```
;; AUTHORITY SECTION:
```

```
www.potaroo.net.                6400              IN                NSEC              _443._tcp.www.potaroo.net. A AAAA RRSIG NSEC  
www.potaroo.net.                6400              IN                RRSIG             NSEC 13 3 6400 20320331235230 20220324225230 41284 potaroo.net.  
lhP13N+YR6m3dBYLUxfgv8fGsuiF4f14UcpznpqlevlJyEumLgHtzUV Y6k6MXpiygGqI70KzZidqzAhglVCcQ==  
potaroo.net.                    6400              IN                SOA               ns1.potaroo.net. gih.potaroo.net. 2022032501 10800 3600 3600000 6400  
potaroo.net.                    6400              IN                RRSIG             SOA 13 2 6400 20320331235230 20220324225230 41284 potaroo.net.  
oQZTmjoMBb8r8FUihbp+62ZjSV1aXU9GI6K28ngh6RXHFPWmzTJlilEA dCkf7fzA3d9ANqm5I5UiMikBRPceFw==
```

Defined RRtypes for this label



NODATA response

NODATA signature

```
;; Query time: 143 msec
```

```
;; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)
```

```
;; WHEN: Mon May 23 18:45:22 AEST 2022
```

```
;; MSG SIZE rcvd: 377
```

"Signing" a zone

- Generate a key pair
- Add the public key to the zone as a DNSKEY record
- Use the private key to generate signatures for all records in the zone (RRSIG records)
- Publish the signed zone file
- Pass the hash of the public key to the zone's parent to publish as a DS record alongside the NS delegation records

DNSSEC is great! Right?

- We can now tell if a DNS response is authentic or not
- Any effort to intercept a DNS response and substitute something else will fail on validation of the digital signature
- So if we are worried about the DNS being a covert attack channel then DNSSEC will fix it!
- So everyone should be using DNSSEC to sign their DNS names – right?

Really?

There are a whole lot of zones the are **not** signed – like:

www.google.com

www.westpac.com.au

www.telstra.com.au

my.gov.au

anu.edu.au

Some 90% of all domain names are **not** signed with DNSSEC


Why don't we use DNSSEC everywhere?

- It depends on your “parent” zone being signed
 - If your parent zone is unsigned then nobody can validate the signed entries in your zone
- Signing an entire zone can be an operational nightmare for very large zones
 - Those NSEC records need to span all the gaps in the zone, which means that you can only sign discrete “snapshots” of the zone file

And validation is no fun

- To validate the key used to generate the digital signature, you need to retrieve the public key that is associated with this zone (DNSKEY), and validate that key
- The way DNSSEC associates a key pair with a zone is to place a record of the hash of the public key into the parent zone file, and have this DS record signed by the parent zone key
- To answer whether a parent zone key is authentic you need to ask for the parent's DS record for this delegated zone label
- And then you need to authenticate the parent zone key, and so on...

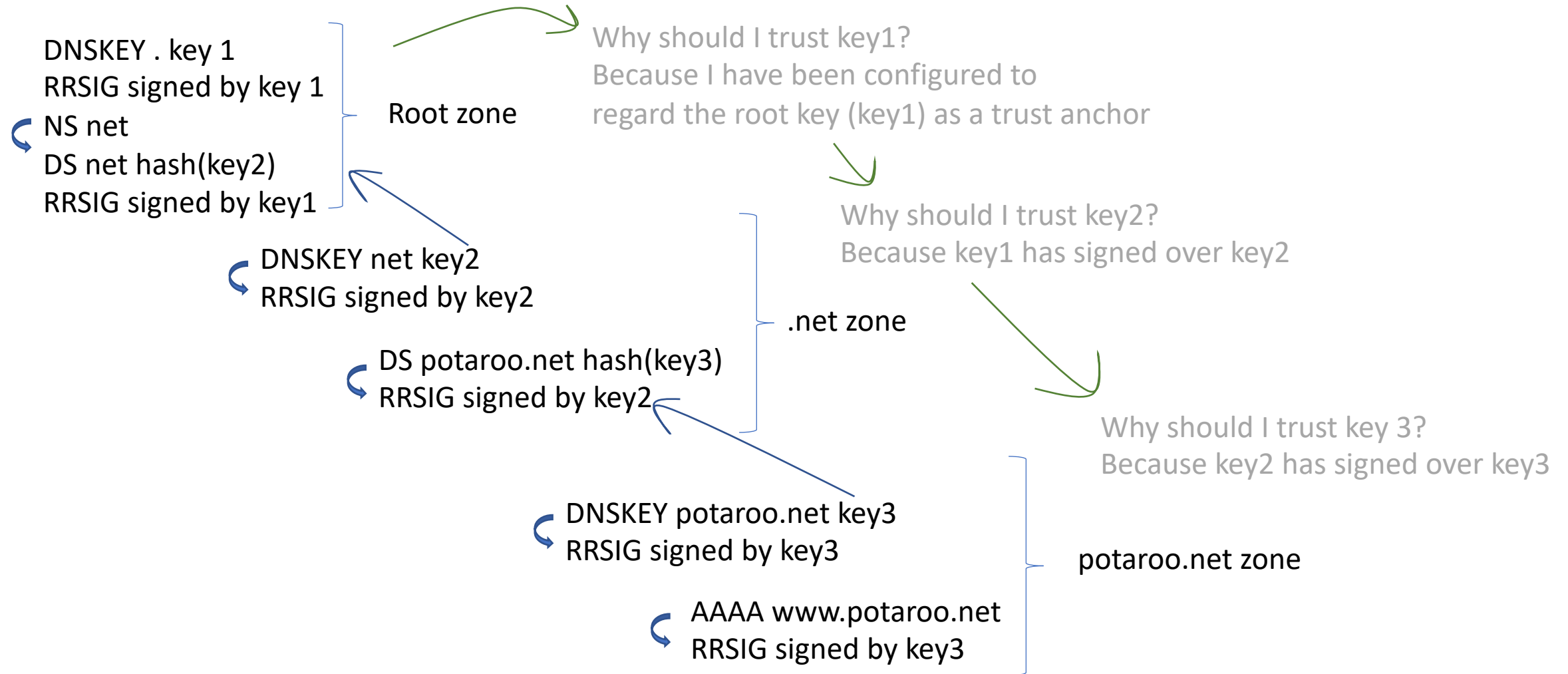
DNSSEC Validation

- 
- Retrieve the zone signing key(s) for this zone (DNSKEY)
 - Check that the signature matches the couplet of the RRdata and the zone key
 - Check that the signature of the DNSKEY record matches the couplet of the RRdata and the zone key
 - So if I trust the zone key, then I can trust this record
 - Why should I trust the zone key (DNSKEY)?
 - Query the zone parent for the Delegation Signer (DS) record
 - Validate the signature of the DS record in the parent zone
 - Repeat for the parent zone
-
- Once you get to the root zone check that the key you have retrieved from the DNS matches the root zone key that you have pre-loaded as your single trust point

DNSSEC Validation

- This is like Authoritative Server discovery in the DNS, but in reverse
- At each level the client retrieves the DS and DNSKEY resource records and then moves UP a level to the parent zone
- Until it reaches the root zone
- Then it performs the sequence of crypto operations to validate the chain of signatures

DNSSEC "key chains"



DNSSEC Validation can be slow

- For each level the validating client needs to retrieve the DNSSEC-signed DS and DNSKEY records
- For each record the validating client needs to perform a crypto validation operation
- E.g. for www.potaroo.net that's 5 additional DNS queries and 6 crypto operations:
 - DNSKEY potaroo.net @ns1.potaroo.net
 - DS potaroo.net @a.gtld-servers.net
 - DNSKEY net @a.gtld-servers.net
 - DS net @a.root-servers.net
 - DNSKEY . @a.root-servers.net

Validation is no fun

- But there's more!
- Not all crypto algorithms and crypto keys are the same
- Crypto systems that are more resistant to attack tend to use longer key sizes and more complex crypto algorithms

How "good" is DNSSEC?

- Like all crypto, the choice of crypto algorithms to use to generate keys and signatures is crucial
- RSA is fast to use, but it has a low crypto strength, so crypto strength is achieved by using longer RSA keys
- Elliptical Curves are "denser" – slower to use, but have a higher crypto strength for a given key size
- DNS over UDP prefers smaller keys!

DNSSEC and UDP

```
$ dig +dnssec +bufsize=1232 DNSKEY au @2a01:8840:bf::1
; Truncated, retrying in TCP mode.

; <<>> DiG 9.16.27 <<>> +dnssec +bufsize DNSKEY au @2a01:8840:bf::1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22246
;; flags: qr aa rd; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 1232
;; QUESTION SECTION:
;au.                                IN                DNSKEY

;; ANSWER SECTION:
au.                                43200             IN                DNSKEY
au.                                43200             IN                DNSKEY
au.                                43200             IN                RRSIG

;; Query time: 36 msec
;; SERVER: 2a01:8840:bf::1#53(2a01:8840:bf::1)
;; WHEN: Sun Jun 05 23:28:51 UTC 2022
;; MSG SIZE rcvd: 1385
```

Query using IPv6 with the UDP buffer size set to the current recommended value of 1232

The large response cannot fit in UDP and the query is retried using TCP, adding 2 additional round trip intervals to the time to complete the DNS response

Crypto Strength

Algorithm	Private Key size	Public Key Size	Signature Size	Strength Equivalence
RSA -1024	1,102	438	259	80
RSA-2048	1,776	620	403	112
RSA-4096	3,3112	967	744	140
ECDSA P-256	187	353	146	128
Ed25519	179	300	146	128

And if we look at post-quantum computing crypto algorithms, then the key sizes get a whole lot larger very quickly!

DNSSEC and UDP

- DNS usually operates over UDP
 - It's fast and efficient
 - But UDP is unreliable when carrying large payloads
 - Because IP fragmentation is unreliable
- So the DNS has TCP as a “fallback”
 - If the server does not want to send a large response over UDP it sets a “truncated” flag in a shorter response and the client is expected to open up a TCP session and resend the query over TCP
 - TCP is slower, and takes more resources at both the server and the client

Is DNSSEC worth the effort?

The case for “Yes”

- Too Much Blind Trust.
- We are trusting that the DNS mapping of the name to an IP address is genuine, trusting that the routing system is passing the IP packets to the ‘correct’ endpoint, trusting that the representation of the name on your screen is actually the name of the service you intended to go to, trusting that the TLS connection is genuine, and trusting that the WEB PKI is not corrupted, to name but a few critical points of trust
- We really have no alternatives – we have no other way of securing the DNS content
- The DNS is central – if an attacker can corrupt the DNS at will, then many other kinds of attacks are possible as a consequence

Is DNSSEC Worth the effort?

The case for “No!”

- Its One More Thing to go wrong
 - It adds the tasks of secure key management, regular key rotation, synchronisation with the parent zone
- DNS Responses are larger
 - All responses include a digital signature
 - DNSKEY responses include the entire key set plus the digital signature
 - DNS over UDP has reliability issues with large responses
 - UDP fragmentation for large responses is unreliable
 - TCP failover on truncated responses is unreliable
- Validation takes additional time
 - The validator must separately query for DS and DNSKEY records up the delegation chain and then perform a sequence of crypto operations

Is DNSSEC Worth the effort?

The case for “No!”

- Sub resolvers generally don't validate responses anyway!
 - They rely on the AD bit being set in the response from the recursive resolver
 - Which defeats the entire purpose of DNSSEC!! Its crazy!
- Signalling DNSSEC validation failure is extremely badly handled in the DNS
 - There was no defined DNSSEC validation error code, so the standard reused the SERVFAIL error code
 - SERVFAIL as a response code triggers an exhaustive search across **all** servers
- What's the realistic assessment of threat?
 - As a result, the only threat that DNSSEC protects the stub against is tampering with the response sent from the Authoritative server to the Recursive resolver, which is a pretty abstract threat model

If not DNSSEC, then what?

- Nobody “important” seems to be signing here in .au
 - No gov.au records
 - Not even AFP or ASD!
 - Not the federal shop front (my.gov.au), nor the ATO
 - No major retail banks
 - More generally, few folk DNSSEC-sign their DNS names in .au
- Instead, they are trusting that TLS is robust
 - TLS relies on the certificate infrastructure of the web PKI
 - So they are trusting that the web PKI is robust
- And this is a problem

PKIs have problems too!

<https://www.feistyduck.com/ssl-tls-and-pki-history/>

- The problem here is that with so many points of trust and no easy way of limiting the trust domain each client is forced to trust every single CA that all of its actions are absolutely correct all of the time
- Every CA in the PKI simply must never lie
- Which is an impossible objective
- And we have no robust *certificate revocation* mechanism to “unsay” dud certificates

Certificates are a Failure?

- We persist with long-lived certificates and non-functional revocation mechanisms, because it's the path of least resistance
- The problem with certificates that provide a trust window of a few hours, is that the existing CA infrastructure and the use models of locally stashed certificates just can't cope with such an increased intensity of certificate re-issuance.
- If certificates are incapable of informing a client that they are about to be drawn into misplaced trust then what exactly are they good for anyway?
- The entire objective here was to answer the simple question: **“Is the service that I am about to connect to the service that I intended to connect to?”** And the problem is that this entire certificate structure can only answer a question that relates to the past, not the present!

Where to from here?

- We've been trying to patch up the PKI system for some decades, and the result is a system that is not much more robust, but now has a greater level of external dependencies/vulnerabilities
- DNSSEC could be a more robust approach here but adoption resistance and operational immaturity count heavily against it
 - It's not a clear and useful "solution" to a current set of opsec issues
- But doing nothing seems to be irresponsible as well!

Where to from here?

I really don't know!

- I don't think we can “fix” the certificate system
 - Too many points of trust create vulnerabilities for the entire system
 - Revocation is broken so mis-issuance and key compromise create persistent vulnerabilities
- But there is a lot of resistance to DNSSEC
 - The single point of all trust is not at all reassuring
 - Validation is too slow and too fragile
 - Key management is fragile

Questions?