DNS Oblivion

Geoff Huston AM
Chief Scientist
APNIC Labs

Why pick on the DNS?

The DNS is very **easy** to **tap** and **tamper**

- DNS queries are open and unencrypted
- DNS payloads are not secured and tampering cannot be detected
- DNS responses are predictable and false answers can be readily inserted

Why pick on the DNS?

The DNS is **hard for users to trace**

- Noone knows exactly where their queries go
- Noone can know precisely where their answers come from

Why pick on the DNS?

The DNS is **you!**

- Because pretty much everything you do on the net starts with a call to the DNS
- If we could see your stream of DNS queries in real time we could easily assemble a detailed profile of you and your interests and activities as it happens!
- And in the Internet Surveillance Economy this profile information is highly valuable to all kinds of actors

Countering DNS Surveillance

- Can we stop DNS surveillance completely?
  - Probably not!

- Can we make it harder to collect individual profiles of activity?
  - Well, yes
  - And that’s what I want to talk about today

What's the problem here?

- Is the **DNS label** being queried a secret?
  - Well, not normally *
    - Although there are DNS versions of steganography that can conceal data in the query string

What's the problem here?

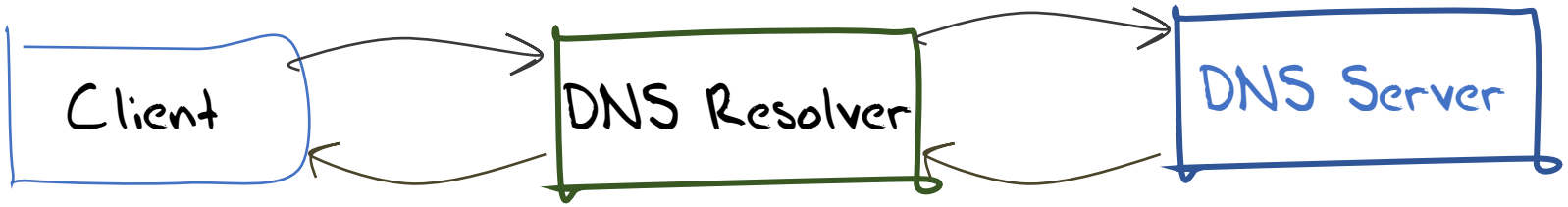
- Is the **DNS label** being queried a secret?
  - Well, not normally
- Is the **DNS response** to a query a secret?
  - Again, not normally *
    - Although there are DNS versions of steganography that can conceal data in the response value

What's the problem here?

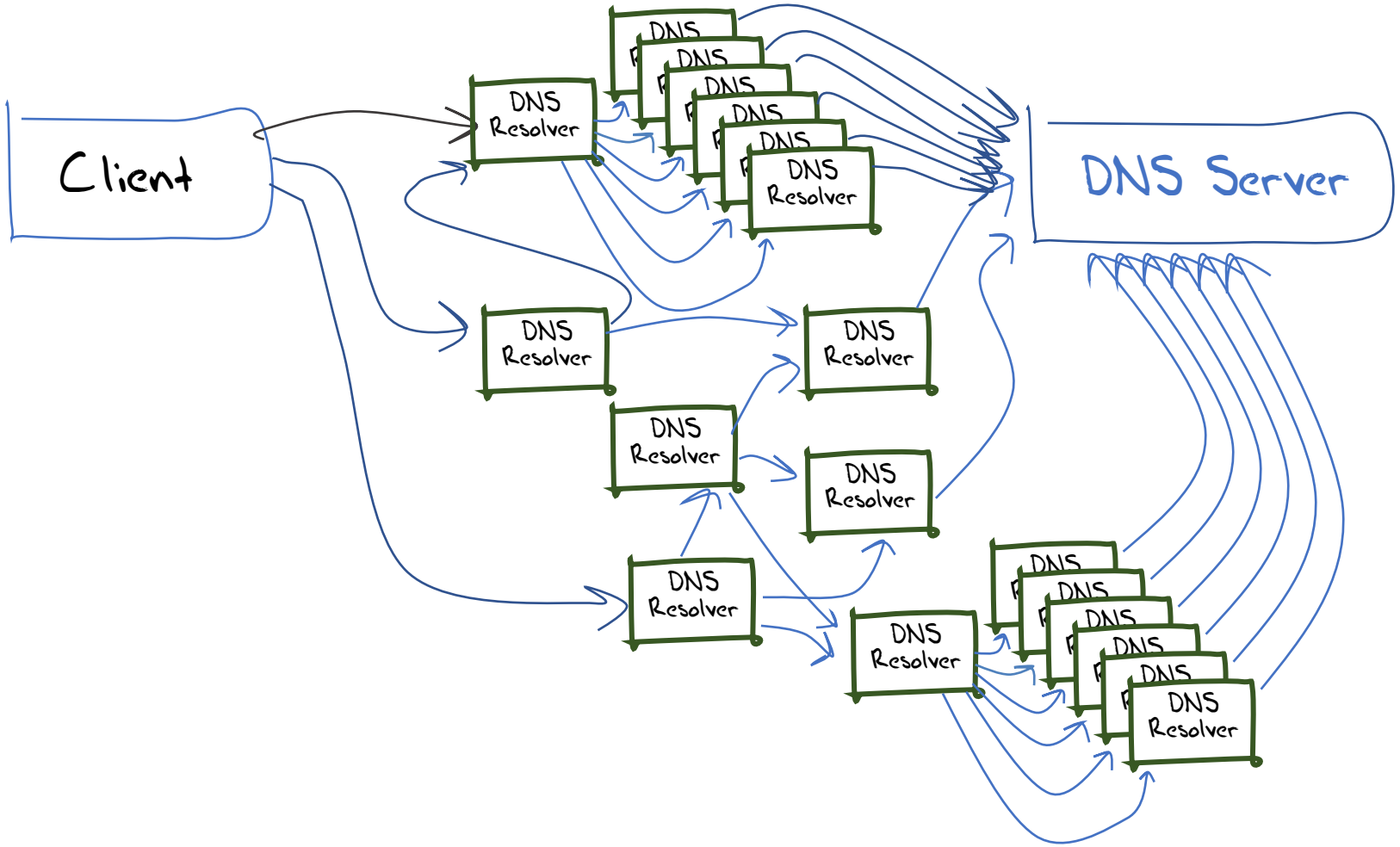
- Is the **DNS label** being queried a secret?
  - Well, not normally
- Is the **DNS response** to a query a secret?
  - Again, not normally
- So what is the issue here?
  - It’s the combination of query and the meta-data around a query that creates a problem:
    - The end user identity, from the IP packet header
    - The DNS label (or sequence of labels) being queried, from the payload
    - The date and time

Let's take a step back...

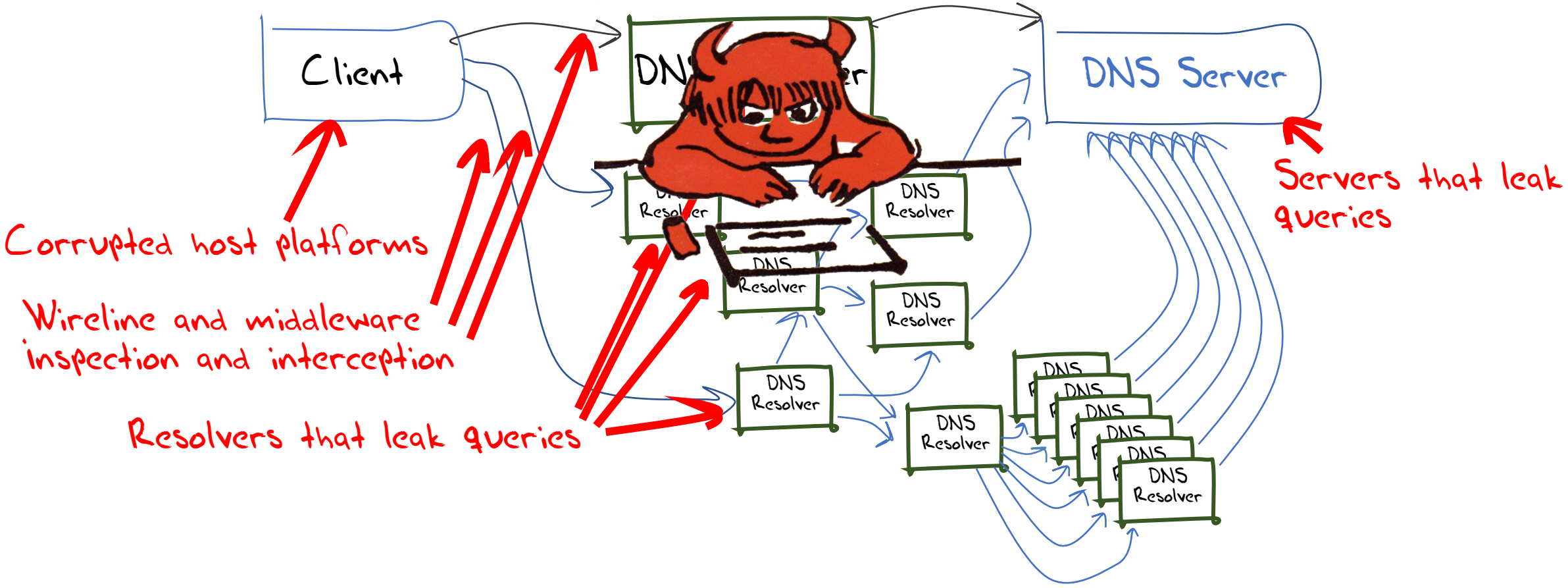
How the idealised model of the DNS works



What we suspect the DNS is like



What we suspect the DNS is like



Second-hand DNS queries are a business opportunity these days

How can we improve DNS Privacy?

Let’s look at a few behaviours of the DNS and see what we are doing to try and improve its privacy properties

QNAME Minimisation

- A resolver technique intended to improve DNS privacy where a DNS resolver no longer sends the entire original query name to the upstream name server
- Described in RFC 7816

> Instead of sending the full QNAME and the original QTYPE upstream, a resolver that implements QNAME minimisation and does not already have the answer in its cache sends a request to the name server authoritative for the closest known ancestor of the original QNAME. The request is done with:
>
> o the QTYPE NS
>
> o the QNAME that is the original QNAME, stripped to just one label more than the zone for which the server is authoritative

Yes, but…

It’s a technique to minimise the information leak between a recursive resolver and authoritative servers, as stub resolvers pass the full query label to the recursive resolver

So while it sounds good, its actually not a major improvement

DNSSEC pixie dust

- A DNSSEC-signed zone can be used to allow clients to verify that the DNS answer they that receive about an entry in that zone is authentic

  - A DNS response that has been modified will fail to validate under DNSSEC when:
    - a client asks a security-aware resolver to resolve a name, and
    - sets the EDNS(0) DNSSEC OK bit, and
    - the zone is DNSSEC-signed

  - A DNSSEC-validating recursive resolver will only return a RRset for the query if it can validate the response using the associated digital signature, and It will set the AD bit in the resolver response to indicate validation success

    Otherwise it will return SERVFAIL

Yes, but…

- The zone (and all its parent zones) must be DNSSEC-signed
- If the recursive resolver performs DNSSEC validation (using the recursive resolver to validate is the most prevalent deployment model) then the all-important stub-to-recursive link is still vulnerable to interception and re-writing
- And if your recursive resolver is performing the re-writing of the response then the stub is none the wiser if the stub does not perform DNSSEC validation
- Stub resolvers don’t generally perform DNSSEC validation
  - It’s too slow!

Middleware and WireTapping

- If we want to make DNS surveillance harder we should look at encrypting the transport used by DNS queries and responses between stub and recursive resolvers

- Today's standard tool is **Transport Layer Security**, which uses a dynamically generated session key to encrypt all traffic between two parties

DoT - DNS over TLS

- TLS is a TCP ‘overlay’ that adds server authentication and session encryption to TCP
- DoT uses a persistent stub-to-recursive relationship to amortize the setup costs of TCP and TLS over many subsequent queries
  - Which works efficiently in a stub-to-recursive scenario, but its not even a little bit efficient for recursive-to-authoritatives!
- If the initial server name certificate is validated by the client then
  - The client can be assured (to some extent) of who it is talking to by name
  - No third parties can observe or intrude in the DNS queries and responses in this DoT session

Yes, but…

- The TCP session state is on port 853
  - DNS over TLS can be readily blocked by CPE and middleware
- DoT will generate a higher recursive resolver memory load as each client may have a held state with one or more recursive resolvers
- The privacy is relative, as the recursive resolver still knows all about you and your DNS queries
- And until ECH* in TLS 1.3 is widely supported, the identity of the TLS server is still in the clear, which also facilitates blocking even if the DoT session jumps over to use TCP port 443

* Encrypting the Server Name in the Client Hello message of TLS setup

DoH - DNS over HTTPS

- DNS over HTTPS
- Uses an HTTPS session for the stub-to-recursive link
- Similar to DNS over TLS, but with HTTP object semantics interposed between the DNS and TLS
- Uses TCP port 443, so can be masked within other HTTPS traffic
- Uses DNS wire format

Why prefer DoH over DoT?

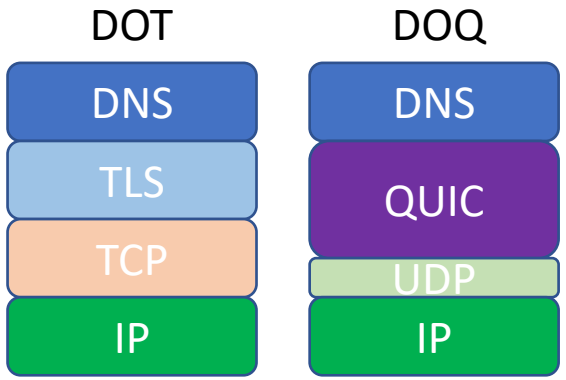
- To bypass middleware blocking of TCP port 853 (DoT)
- DoH allows the stub resolver function to be merged into the application at the client end and DNS resolver to be multiplexed at the server side (browsers and web servers)
  - HTTP object semantics allow for HTTP object caching in the client
  - This enables server-side HTTP push of DNS responses
  - Resolver-less DNS!
  - Can speed up transactions through pre-provisioning of DNS responses
- Applications can switch to use DoH without waiting for the OS platform or the ISP service infrastructure to also support DoH

Yes, but…

- Aside from changing the TCP port to 443 there is little difference between DoH and DoT from a conventional DNS perspective
- Most of the DNS issues with DoH are about the use of resolver-less DNS and content-based DoH-server switching using the HTTP framing shim, which are still largely speculative matters these days
- Application-level DoH can be readily hidden from the platform and from the local network – this can be seen as a good or bad thing!

DNS over QUIC

- QUIC is a transport protocol originally developed by Google and passed over to the IETF for standardised profile development
- QUIC uses a thin UDP shim and an encrypted payload
  - The payload is divided into a TCP-like transport header and a payload
- QUIC allows for multiple DNS queries without TCP Head Of Line blocking



Yes, but…

- QUIC on UDP port 443 has issues with port blocking in middleware
- There is little difference between DoQ, DoH and DoT from a conventional DNS perspective
  - The remote end recursive resolver still is privy to all your DNS queries and your identity

Hiding in the Crowd

- What if you use a DoT or DoH session to a very busy open resolver?
  - No third party can see you queries to the open resolver
  - Noone else can see the responses from the open resolver
  - The open resolver asks the authoritative servers which makes it challenging to map the query back to the end user

  - So if you you are prepared to trust Google, Open DNS, Cloudflare, Quad9, etc with your DNS, and you use DoH or DoT on the stub-to-recursive hop then its far harder for any third party to associate your identity with your queries

I'm still uncomfortable

- It the intention of all of these measures is to prevent a third party from known who I am and what DNS queries I make, then none of these approaches really work!
- Is sharing my identity and all my DNS with Google* really a step forward in protecting my privacy?
  - I don’t pay Google
  - I don’t have a contract with Google
  - It’s not even clear if Google obey the laws of the country where I reside
  - So how am I “protected” here?

* Or any other third party DNS resolution service provider

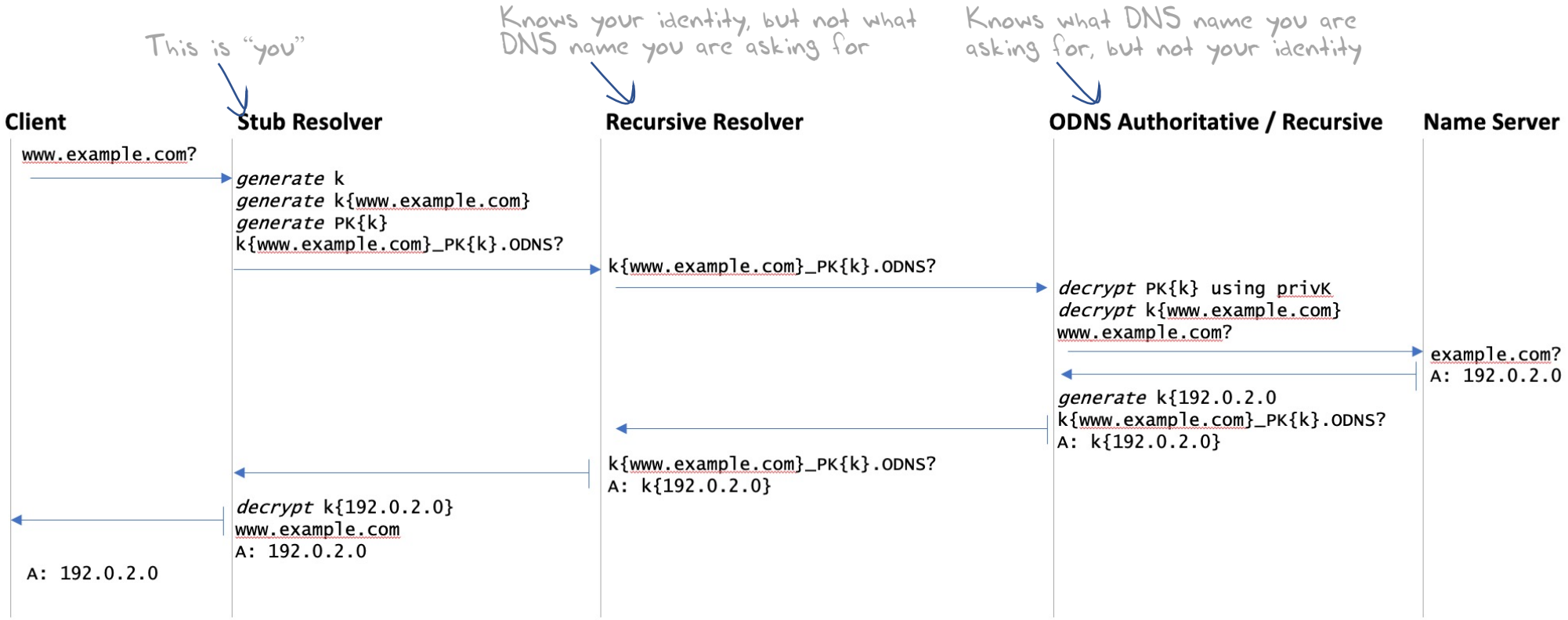
Can we improve this?

- A big part of the problem is being able to associate my identity with DNS queries
- So can we break this association?

Oblivious DNS (oDNS)

- Uses the **QUERY name** to disassociate stub identity from query
  - Stub resolver encrypts the DNS query name into a new query name
    - Encryption uses the public key of a known oDNS server, and appends the name of the oDNS server to the encrypted query name
  - Stub resolver queries a ‘normal’ recursive resolver with this encrypted query name
  - Recursive resolver queries an oDNS server with this encrypted query name
  - oDNS server strips out its own name and decrypts the query name, and resolves this name
  - oDNS server encrypts the DNS response to send back to the stub via the recursive resolver

Oblivious DNS

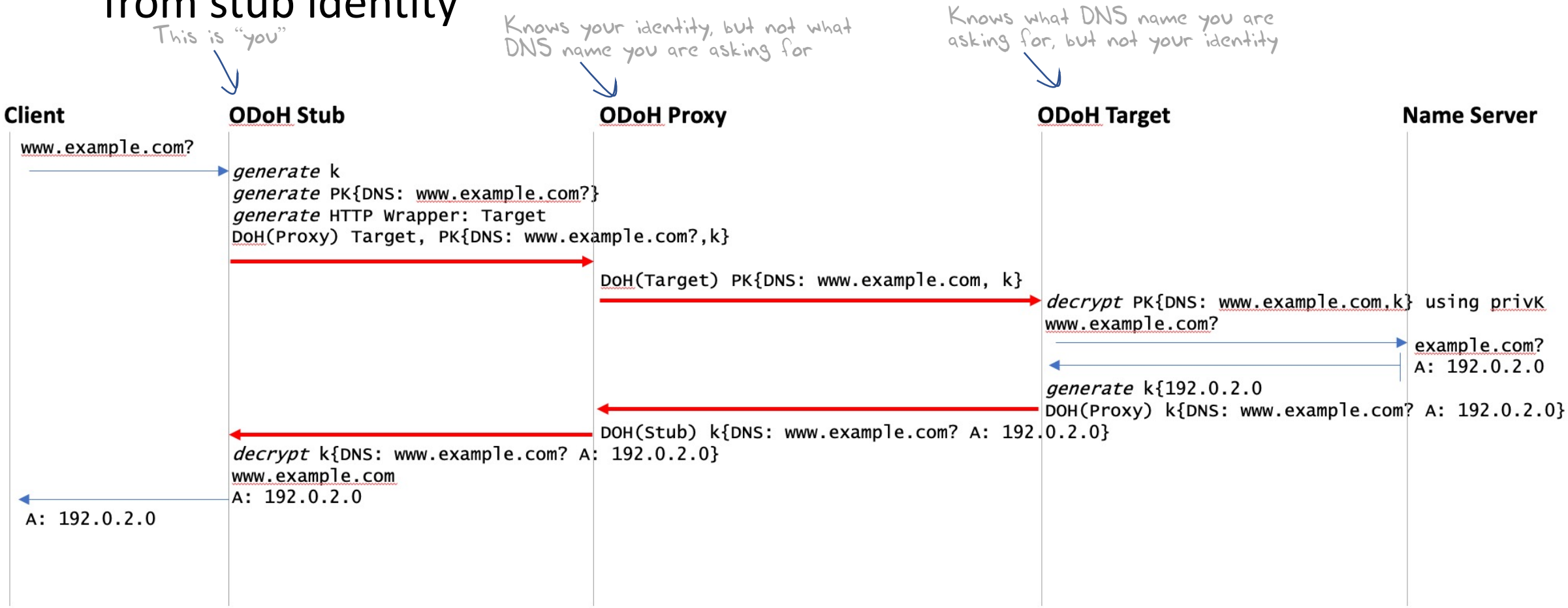


Yes, but…

- The DNS is still DNS over UDP port 53
  - But nothing prevents a oDNS stub using Do[THQ] to a recursive resolver. The recursive resolver has no knowledge of oDNS and process the DNS query like any other
- The encryption is limited due to limited size and alphabet of the query name field

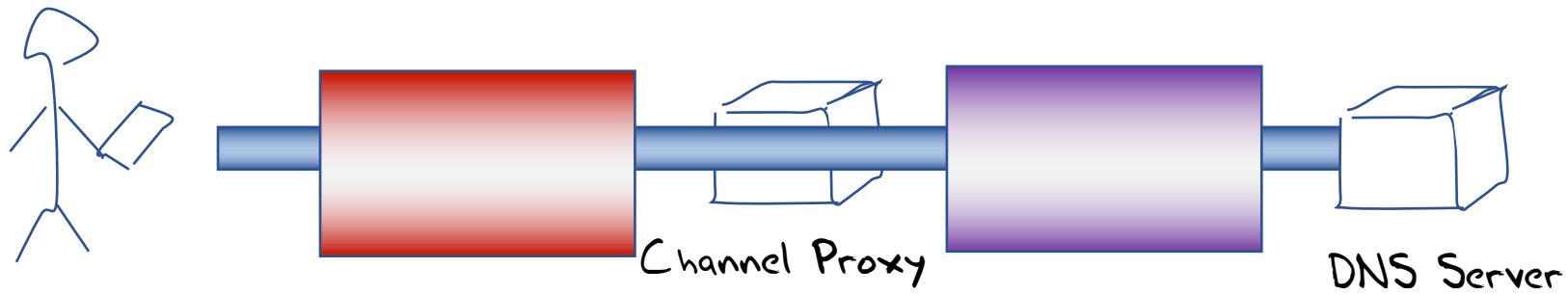
Oblivious DoH

Use double TLS wrapper on a DoH transport to dissociate query name from stub identity



Oblivious DoH

- An outer TLS wrapper is used for the stub-to-oDoH Proxy hop and a different TLS wrapper is used for the oDoH Proxy-to-oDoH Target hop
- The inner TLS wrapper is used to encrypt the DNS query, encrypted using the public key of the target
- The response is encrypted using a session key generated by the client



Yes, but…

- This requires a modified DNS stub resolver that can send and receive oDoH messages, an ODoH Proxy and an ODoH Target
- Oh, and the ODoH Proxy and the ODoH Target must not collude!
  - But we can’t ensure that no collusion happens!

Where is this heading?

- Will any of these DNS Privacy approaches becomes mainstream in the public Internet?

The DNS Economy

- In the public Internet, end clients don't normally pay directly for DNS recursive resolution services
- Which implies that outside of the domain of the local ISP, DNS resolvers are essentially unfunded by the resolver's clients
- And efforts to monetise the DNS with various forms of funded misdirection (such as NXDOMAIN substitution) are generally viewed with extreme disfavour
- Open Resolver efforts run the risk of success-disaster
  - They more they are used, the greater the funding problem
  - The greater the funding problem the greater the temptation to monetise the DNS resolver function in more subtle ways

The DNS Economy

- The default option is that the ISP funds and operate the recursive DNS service, funded by the ISP's client base
  - 70% of all end clients use same-AS recursive resolvers *
- However the fact that is works today does not mean that you can double the input costs and expect it to just keep on working tomorrow
- For ISPs the DNS is a cost department, not a revenue source
  - We should expect strong resistance from ISPs to increase their costs in DNS service provision
- The DNS is also highly resistant to changes in the edge infrastructure

* https://stats.labs.apnic.net/rvrs

My Opinion

- ISP-based provisioning of DNS servers without channel encryption will continue to be the mainstream of the public DNS infrastructure
- Most users don’t change their platform settings from the defaults and CPE based service provisioning in the wired networks and direct provisioning in mobile networks will persist
- But that’s not the full story...

"Fragmented" DNS

- Is appears more likely that applications who want to tailor their DNS use to adopt a more private profile will hive off to use DoH to an application-selected DNS service, while the platform itself will continue to use libraries that will default to DNS over UDP to the ISP-provided recursive DNS resolver
- That way the application ecosystem can fund its own DNS privacy infrastructure and avoid waiting for everyone else to make the necessary infrastructure and service investments before they can adopt DNS privacy themselves
- The prospect of application-specific naming services is a very real prospect in such a scenario

It's life Jim, but not as we know it!*

- The progression here is an evolution from network-centric services to platform-centric services to today's world of application-centric services
- It's clear that the DNS is being swept up in this shift, and the DNS is changing in almost every respect
- The future prospects of a single unified coherent name space as embodied in the DNS, as we currently know it, for the entire Internet service domain are looking pretty poor right now!

* Apologies to Trekkies!

Thanks!