Privacy analysis of DNS resolver solutions

J.H.C. van Heugten

Master of System and Network Engineering
University of Amsterdam, The Netherlands

August 3, 2018

Abstract

This research focuses on privacy in the Domain Name System (DNS). Techniques to improve privacy during specific phases of DNS resolution exists. The goal of this research is to evaluate those techniques and define what techniques are available to the user. We will then combine techniques to achieve the best privacy protection. This research will show that the best protection a DNS user could achieve is combining a DNS forwarder with a public resolver. Encryption should be used in the two steps between stub and public resolver. Required is a public resolver that has implemented at least one of the three evaluated encryption techniques (DNS-over-TLS, DNS-over-HTTPS or DNSCrypt) and QNAME minimisation.

1 Introduction

Online privacy concerns are growing [39], hence regulation like the General Data Protection Regulation (GDPR) [7] has been introduced across Europe. With the help of certificate authorities like Let's Encrypt offering free TLS certificates, the adoption rate of websites using HTTPS is increasing [19]. This prevents eavesdropping on the communication between users and websites. However, the system that translates memorable domain names into IP addresses, the Domain Name System (DNS), still communicates unencrypted by default [12]. This enables eavesdroppers [18] to gather insight in DNS requests/responses (e.g., what websites are visited) by a particular user.

To resolve domain names using the DNS, a computer will contact a (often non-local) DNS resolver. This resolver will contact other DNS servers and return the answer to the computer. In the last couple of years public DNS resolvers (e.g., Google – 8.8.8.8) have become increasingly popular [25] [41]. Unlike the DNS resolvers provided by the ISP, which are often within the same autonomous system, public DNS servers can be located anywhere on the Internet. This adds more possibilities for eavesdroppers: not just on the connection between the user and ISP, but also on intermediate networks.

Some (public) DNS resolvers claim to provide privacy, but this is usually limited to statements about their logging policies [33] [32]. However, there is no way to verify those policies, and it does not protect against eavesdropping on the connection. Additional techniques to prevent eavesdropping are necessary. New techniques to improve privacy in DNS resolution have been developed recently [13]. This paper aims at evaluating privacy issues in DNS, comparing modern techniques and advising on the most complete solution to protect the privacy of DNS users.

2 Research questions

This research will analyze different modern techniques that improve privacy in DNS resolution. In order to perform the research we defined the following research question:

How can modern techniques improve the privacy of DNS users?

In order to answer this question we defined several sub questions, to help structure the answering of the main research question.

Subquestions:

- What privacy sensitive data can be gathered in each phase of DNS resolution?
- What modern techniques are available to improve DNS privacy and to what extent do they improve privacy of DNS resolution?
- What combination of techniques available to the user results in the highest DNS privacy improvement?

3 Related work

The IETF DNS PRIVate Exchange working group (DPRIVE) is chartered to develop techniques to solve privacy issues in DNS. [26].

RFC7626 *DNS Privacy Considerations* [2] summarizes issues regarding DNS privacy. It covers basics about what information DNS packets contain and possible attack concepts.

The Internet draft *Evaluation of Privacy for DNS Private Exchange* [31] briefly explains some mechanisms to enhance DNS privacy. The draft also provides templates to asses the effectiveness of different DNS privacy techniques.

In the paper *Analysis of Privacy Disclosure in DNS Query* [42] Zhao et al. propose Range Query as a solution to improve DNS privacy. In this proposition a set of random hostnames is used to disguise the actual requested hostname. The paper does provide a theoretical possibility of guessing the actual hostname, but does not provide practical verification of this. Two years later Zhao et al. suggest improvements to Range Query by using a two server Privacy Information Retrieval (PIR) instead of a one server to increase efficiency and privacy [43] .

In *Evaluation of Two Privacy-Preserving Protocols for the DNS* **[6]** Castillo-Perez et al. compare the benefits and limitations of the solutions from Zhao et al. and design additional changes to improve the privacy.

Privacy-Preserving DNS: Analysis of Broadcast, Range Queries and Mix-based Protection Methods [16] compares multiple DNS privacy protection methods and their security. The authors also practically implement each solution and benchmark its performance.

4 Background

4.1 Domain Name System

The DNS provides a standardized solution for named resources across various computer systems, techniques and organizations connected to the Internet or private networks [30]. The DNS translates memorable domain names into Internet Protocol (IP) addresses by requesting information from a hierarchical and decentralized system.

This section explains the different DNS server types and their location, the DNS resolving process and a security extension for DNS.

4.1.1 DNS server types

In this section the different DNS server types are explained.

Stub resolvers can only contact recursive resolvers. They act as an entry point between application and recursive resolver, have the ability to differentiate between local policy and Internet locations and sometimes offer local caching.

Authoritative servers do not respond to recursive queries, act only as a server and will not contact other servers to answer a query. They know the correct answer to specific queries and will respond to those.

Recursive resolvers are generally contacted by multiple clients and try to resolve the requested DNS query by either answering it from a local cache or by contacting authoritative servers around the Internet for an answer. They act as both a server (receiving recursive queries) and a client (sending iterative queries).

Forwarding DNS servers look like a recursive resolver from a stub's perspective. However, this type of DNS server forwards all queries to a recursive resolver or another forwarding DNS server rather than contacting authoritative servers themselves. The results received can be cached locally.

4.1.2 DNS resolving

To resolve the domain name *example.com.* into an IPv6-address, the DNS has to execute the steps below. For a graphical overview of the steps, see figure 1.

1. The stub requests the AAAA record of domain name *example.com.* at the recursive DNS resolver configured.
2. The recursive DNS resolver will look up whether the answer can be served from cache. If this is the case, the recursive DNS server will answer the request from the stub with the IPv6-address from cache. If the question cannot be answered from cache, the steps below apply.
3. The recursive DNS resolver will do a iterative lookup of the question. Assuming an empty cache, it will start at the root zone (.), and forward the request to one of the 13 predefined root servers.
4. The root server does not know the answer to the request, but does know where to find the authoritative servers for the *com.* Top-Level Domain (TLD) zone. It will return (refer to) a list of authoritative name servers for this zone.
5. The recursive DNS resolver will now send the request to one of the returned authoritative name servers for the TLD-zone.
6. The authoritative TLD name server does not know the answer to the request, but does know where to find the authoritative servers for the Second-Level Domain (SLD) *example.com.* zone. It will return (refer to) a list of authoritative name servers for this zone.
7. Finally the recursive DNS resolver will contact one of the returned authoritative name servers for the SLD-zone (*example.com.*)

8. The authoritative SLD name server does know the IPv6 address of the server *example.com.* and will return the IPv6-address(es) of the requested AAAA record to the recursive DNS server.

9. The recursive DNS server will now answer the request from the stub with the received IPv6-addresses and might store the data in cache for future use.

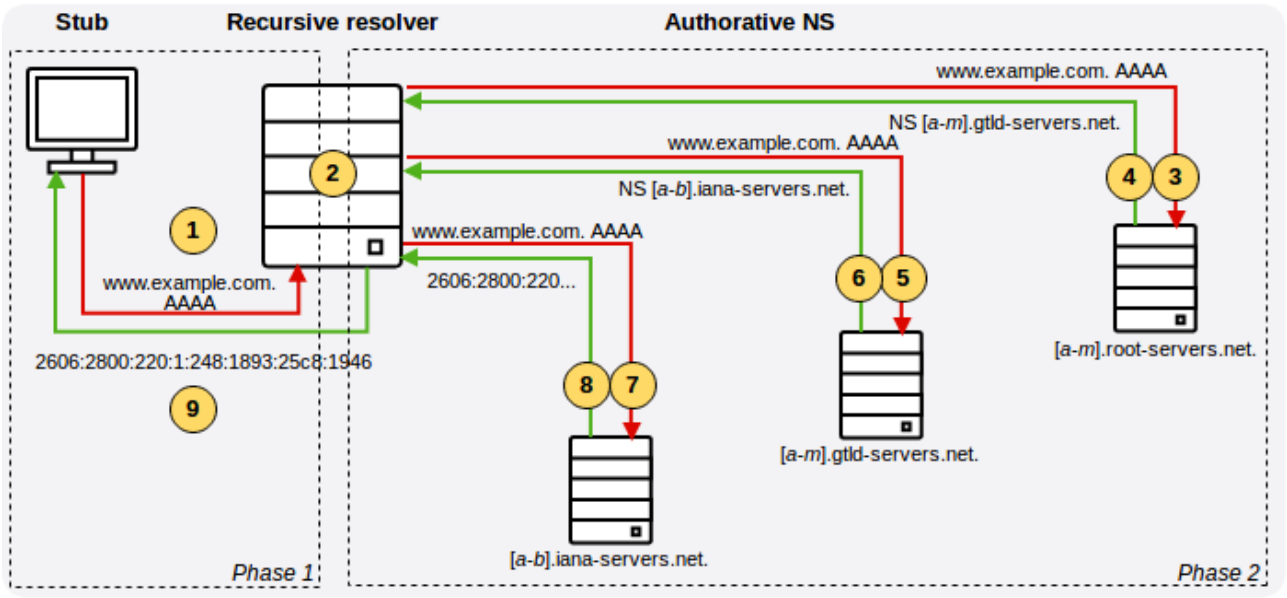

Figure 1: Domain name resolving using the DNS

4.1.3 Recursive resolver location

An important consideration from a user's point of view is the location of the recursive resolver. The recursive resolver has access to the full DNS communication and is therefore an important component from a privacy perspective. Recursive resolvers have the ability to cache (both positive and negative) answers. If answers can be served from cache, no upstream communication to/from authoritative servers is needed, which limits privacy risks to the first phase of DNS resolution.

There are four possible locations for a recursive resolver:

Local recursive resolvers are located within the user's network and share the same outgoing IP-address or network with the stub resolver. As the server is under a user's control, no third parties can use logging/monitoring on the recursive resolver. The disadvantage is that there is no obfuscation of the stub/user's IP-address in the recursive resolver to authoritative server phase of DNS resolution.

Private recursive resolvers are under control from the user, but use a different outgoing IP address or network than the stub resolver. This resolver location shares the advantages to logging/monitoring with the local resolver. There is obfuscation of the stub/user's IP-address, but only if the used outgoing IP-address cannot be linked to the particular user.

ISP recursive resolvers are under control of a particular ISP and are often located in their network. For Internet at home and mobile Internet this resolver is often pushed to users by the ISP as the default resolver to use. In many cases this ISP resolver does not offer any of the mentioned privacy techniques and neither offers a no-logging service.

Public recursive resolvers can be found in many flavors: Some offer (parental) filtering, some focus on performance or privacy [32] [33]. Popular examples of public recursive resolvers are Google's 8.8.8.8 and Cloudflare's 1.1.1.1.

4.1.4 DNSSEC

When the DNS was designed, confidentiality was not one of the design requirements [1]. Therefore the DNS does not encrypt communications. Nowadays, cyberattacks are common and methods to alter DNS data during transit exist [40]. To solve this, an

extension to DNS, called DNSSEC, was designed. In 2005 the ideas for DNSSEC were refined in the current standard (RFC 4033-4035) [1]. In 2010 the root zone became DNSSEC enabled.

DNSSEC adds signatures to DNS resource record sets and provides information necessary to verify those signatures. Verification of those signatures prevents third parties from using attacks such as DNS cache poisoning. While DNSSEC (if enabled) does improve security by validating the integrity of DNS query responses, it does not improve privacy in DNS as there is still no encrypted communication.

5 DNS privacy sensitive data

Determining privacy sensitive data in DNS resolution can be done by analyzing the different phases. In this section we define two phases of DNS resolution (see figure 1):

- Phase 1: Stub - Recursive resolver
- Phase 2: Recursive resolver - Authoritative servers

The first phase is the most interesting phase if one looks at privacy concerns. As described in section 4.1.2, users generally do not interact with authoritative DNS servers directly. Instead, they interact with a recursive resolver, which will in turn (unless the requested data can be served from cache) contact the authoritative DNS servers.

This DNS resolver can be located at the provider delivering Internet connectivity to the client, at some public DNS provider (e.g., Google) or in the network of an organization/user. Eavesdropping on those connections in public areas (e.g., WiFi in public transport) is quite easy and happens often [5]. Finally, there is the DNS resolver itself: there are no guarantees about the security and logging policies of this server.

In this first phase of DNS resolution, the IP address of the stub (or gateway in some scenarios), the DNS query and possibly some relevant meta-data is sent to the DNS resolver. This enables eavesdroppers listening in on this phase of DNS resolution to gather insight in the personal data of the user. We define the following privacy sensitive data in phase 1 of DNS communication:

1. stub's IP address,
2. requested domain name and record type,
3. response to requested query,
4. other user identifiable information[3],
5. relevant metadata

[3]When EDNS Client ID (section 5.1.2) is used

The IP address can be used to determine who is the user. The requested domain name can be used to gather insight into the browsing activities of the user, installed software (e.g., requests to license servers), e-mail communication (e.g., IMAP/POP/SMTP domain names, lookups for HTML content in e-mails). Other user identifiable information may include MAC-addresses or other data that can be used to fingerprint users. This is used in particular in parental control / filtering services that use EDNS Client ID (section 5.1.2). Relevant metadata, such as the TTL and timestamp, could be used to fingerprint users and/or analyze their online activity.

The second phase of DNS resolution, between the resolving DNS server and several authoritative servers, does usually not contain information about the stub IP address, unless EDNS Client Subnet (ECS) is used (section 5.1.1) or when the user uses a local DNS resolver. Data retrieved in this phase contains the requested domain name, the (intermediate) DNS response and might contain ECS information. We define the following privacy sensitive data in phase 2 of DNS communication:

1. DNS resolver IP address,
2. stub's network address[4],
3. requested domain name and record type,
4. (intermediate) response to requested query,
5. relevant metadata

5.1 EDNS

Standard DNS packets support a maximum packet size of 512 bytes. This is enough to fit basic DNS information, but in some cases more space for additional information is required. RFC6891 [9] describes a extension mechanism for DNS (EDNS). In this section we discuss two EDNS options that impact user privacy.

5.1.1 EDNS Client Subnet

EDNS Client Subnet (ECS) is an application of EDNS [8] that allows recursive DNS resolvers to disclose the stub's IP address upstream, usually in a /24 (IPv4) or /56 (IPv6) truncated form. This allows authoritative DNS servers to respond differently based on the network address of the stub (e.g., its geographical location).

Although not all the bits of the IP address are exposed, ECS does decrease the users privacy: some well-known DNS providers including Google use ECS to disclose the stub's truncated IP address to upstream servers. Excluding ECS when using those providers is only possible by custom configuration of the client software (i.e., by setting a source prefix-length of 0).

[4]When EDNS Client Subnet (section 5.1.1) is used

However, there is no guarantee that the recursive DNS resolvers of those providers honor this request.

To verify which popular DNS providers support ECS, we have queried the top 10 public DNS resolvers [17] as well as the DNS resolvers of the two biggest Dutch ISPs.

In figure 2 the experimental set-up is shown: Requests from our stub (using an outgoing IP address in subnet 195.114.238.0/24) are sent to each public resolver, which does a lookup on (amongst others) an authoritative name server under our control (ns1.transmixt.com). Each query is prefixed with the resolver IP address to be able to match it with the query log on the authoritative name server. The marked subnet (195.114.238.0/24) shows that ECS data is present and that the stub's IP address is truncated to a /24 network. The results show that Google, OpenDNS and Dyn support ECS (the full results can be found in appendix A).

The IP address 173.194.170.67 is related to the Google server that does the iterative lookup to our name server. Because our recursive lookup was sent to 8.8.8.8, this shows that Google is using different IP address(es) for the iterative phase.

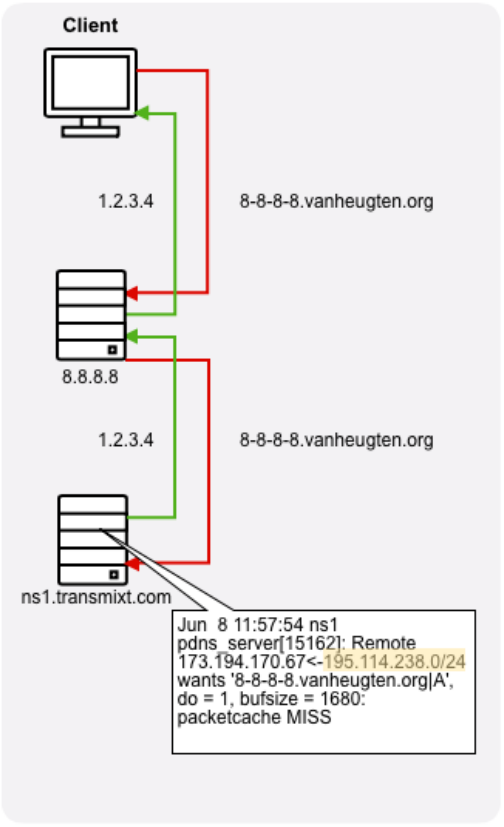

Figure 2: Google DNS (8.8.8.8) forwarding ECS

5.1.2 EDNS Client ID

Internet draft "Client ID in Forwarded DNS Queries" [28] defines the use of EDNS to forward data to identify a specific client in DNS. This (now expired) draft refers to two implementations (Cisco's Umbrella and Vantio_CacheServe) that use EDNS to forward client specific data (MAC address and IP address) to remote services. While those implementations are opt-it, addition of such specific data in the DNS can be used by third parties (including eavesdroppers) to fingerprint a specific user.

6 DNS privacy techniques

This section analyzes modern techniques to improve privacy in DNS resolution. We start by discussing encryption techniques for the first phase of DNS resolution in section 6.1 and 6.2. In section 6.3 we discuss techniques for both phases of DNS resolution. Section 6.4 covers a technique for the second phase of DNS resolution. We also study a new technique that has quite a different design and overlaps phases in section 6.5.

6.1 DNS over TLS

In 2016 DNS over TLS (DoT) was described in RFC7858 [23] as a solution to add privacy to the DNS. DoT encrypts the connection between the stub and the recursive DNS resolver with Transport Layer Security (TLS). To use DoT both implementation on the stub resolver and on recursive resolver is required. DoT is implemented by (amongst others) the stub resolver "Stubby" and three public resolvers from the list in appendix A: Google, Quad9 and Cloudflare. The DPRIVE working group is investigating the possibilities to use DoT for the recursive resolver to authoritative phase as well [4].

Concept. When using DoT, the stub resolver will connect to the recursive DNS resolver over Transmission Control Protocol (TCP) port 853. This non-standard DNS port is chosen to avoid complexity in differentiating between DoT and non-DoT connections. When the connection is made, a TLS handshake is initiated. When the TLS session is established, further communication will be encrypted (see figure 3). Authentication is provided by one of the six authentication mechanisms listed in RFC8310 [11].

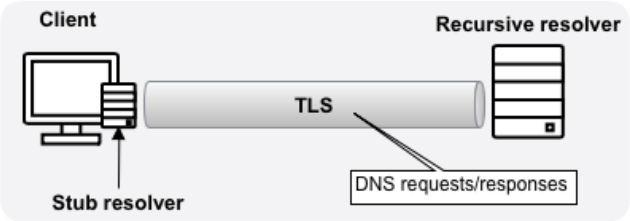

Figure 3: DNS over TLS

Performance. Unlike regular DNS resolution, DoT uses the TCP instead of the User Datagram Protocol (UDP). Both TCP and TLS will introduce extra latency [22] compared to UDP and will therefore have a negative impact on the performance of DoT compared to regular DNS resolution. To minimize this performance hit, the RFC states that stub resolvers should reuse the connection to prevent TCP/TLS overhead happening at each DNS request. It is also recommended to pipeline multiple DNS requests after each other, without waiting for an answer after each request. The "Message ID" is then used to match requests with incoming responses.

The overhead makes DoT not really viable for the second phase (recursive resolver to authoritative servers) of DNS resolution, because a one-to-many architecture is used there (one recursive resolver to many authoritative servers), where keeping the communication channel open is often not desired. There is an experimental proposal (RFC8094 [34] – DNS over DTLS) that suggests TLS over UDP instead. However, to our knowledge there are no practical implementations of this yet.

Privacy. By encrypting the communication between stub and recursive resolver, DoT protects against eavesdropping in the first phase of DNS resolution (excluding the recursive resolver). However, third parties may be able to gather insight in DNS communication based on traffic patterns. RFC7858 [23] suggests that clients/servers "may consider" using EDNS0 padding (RFC7830 [29] to solve this, but this is merely a recommendation.

6.2 DNS over HTTPS

DNS over HTTPS (DoH) is currently being standardized in Internet draft "DNS Queries over HTTPS" [20] (at the time of writing version 10). DoH is designed as a technique for confidential DNS resolution for both DNS clients and (native) web applications [20]. The web browser Firefox added support for DoH in version 60 (nightly build). Other client implementations are DNS proxies and a module for the PHP programming language. On the server side (based on the list in appendix A): Google and Cloudflare support DoH.

Concept. DoH uses HTTPS connections to encrypt DNS communication with TLS. Generally, the connection will be initiated in a comparable way to when a user would access any HTTPS (TLS) secured website. When the encrypted channel is created, DNS data will be encapsulated in HTTP POST/GET frames. Depending on the implementation the payload of the frame will contain wire-format (Internet draft) or JSON (Google implementation) DNS data. The used format is decided based on the header used.

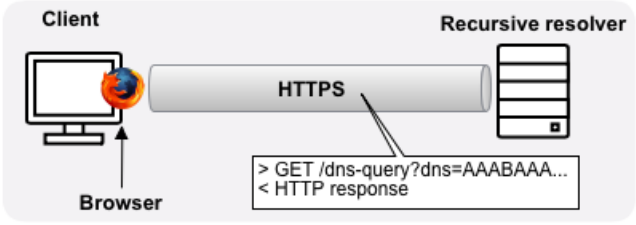

Figure 4: DNS over HTTPS

Performance. DoH suffers from the same performance overhead as DoT caused by using TCP and TLS. A similar solution to reduce the impact can be found in reuse of the connection (in HTTP: keep-alive). However, there is one advantage that DoH has over DoT: The possibility to use HTTP/2 Server Push. This mechanism allows a server to send additional DNS answers to the client/stub, even if it did not ask for them. Websites generally include content from other URLs (e.g., a CDN for static content). The server knows that the user is probably going to request those domain names anyway, so it can send their location already. However, the feasibility of this mechanism is unknown.

Privacy. While the communication channel in phase one of DNS resolution in DoH is encrypted in a similar way compared to DoT, there are some advantages to privacy protection in DoH. The first is the use of common TCP port 443 could prevent censorship (if multiplexed with genuine web traffic): while the use of other DNS mechanisms could be prevented by blocking DNS ports, in this case only access to the full website can be blocked. The second advantage relates to the HTTP/2 Server Push mechanism described in the performance section above. The remote server can predict what URLs a user might request in the near future and can supply those already. Because of this, there is no need to request those domain names from other DNS servers. This limits the information spread. Similar to DoT, the use of EDNS0 padding is optional.

6.3 DNSCrypt & DNSCurve

DNSCrypt and DNSCurve are techniques which encrypt DNS communication using elliptic-curve cryptography on a link-level. DNSCrypt is based on DNSCurve, which was developed by Daniel J. Bernstein in 2008 [36]. DNSCrypt is not standardized by the IETF, but there exists a draft [10] written by OpenDNS in 2010. DNSCrypt.info published a (non-standardized) specification on their website [14]. The main difference between DNSCrypt and DNSCurve is the phase in DNS resolution where they operate. DNSCrypt encrypts DNS communication between the stub resolver and the recursive resolver (phase 1). DNSCurve does this for the communication between the recursive resolver and authoritative servers (phase 2).

Documentation and implementations of both techniques are scarce. On the stub/client side applications (proxies) like dnscrypt-proxy and pcap_dnsproxy support DNSCrypt. Looking at recursive resolvers, the only recursive resolver on our list in appendix A with support for DNSCrypt is OpenDNS. On the authoritative side, DNSCurve.io [1] recommends to use CurveDNS as a forwarder to the actual authoritative server (e.g., NSD, Bind, PowerDNS).

Concept. The techniques encrypt the DNS communication on link-level with the XSalsa20Poly1305 or XChaCha20-Poly1305 algorithm. The Curve25519 algorithm is used for the key exchange. Both TCP and UDP are supported and generally port 443 is used (but not the HTTPS protocol). Both parties in DNS resolving have to support the technique in order to use it. Support is negotiated automatically.

Performance. The use of elliptic curve cryptography allows much smaller keys and needs less computational power compared to algorithms like RSA. However, elliptic curve cryptography is also available for TLS. DNSCrypt recommends DNS resolvers to rotate the keys at most every 24 hours and clients not to reuse keys [14]. The latter recommendation increases the amount of necessary calculations drastically, as one cryptographical operation per query is executed. In an presentation for IOActive [27], Dan Kaminsky gives some insight in performance of DNSCurve compared to regular DNS resolution. In this presentation he estimates a 33% performance drop in the amount of queries per second when using DNSCurve. This might be relevant for high traffic recursive resolvers.

Privacy. DNSCrypt protects against eavesdropping between the stub and recursive resolver (phase 1). DNSCurve protects against eavesdropping on phase 2 of DNS resolution. Besides confidentiality, both offer authentication as well. However, from the client/stub position there is no way to enforce encryption for the second phase. Both techniques do not protect against logging on the recursive resolver.

6.4 QNAME minimisation

Query Name (QNAME) minimisation is a DNS privacy technique described in RFC7816 [3]. It is designed for the second phase of DNS resolution: between the recursive resolver and authoritative servers. The goal of QNAME minimisation is to supply each upstream authoritative server only with the minimal required information to resolve a query. Only an implementation at the DNS resolver is needed for QNAME minimisation. Newer versions of popular DNS resolvers including Bind, Unbound and Knot resolver support QNAME minimisation.

[1] https://dnscurve.io/documentation/install-curvedns.html

Concept. In figure 5 we show the DNS resolving of the domain name *www.example.com.* with QNAME minimisation implemented on the recursive resolver. As shown, the authoritative server for the root domain and for the com. TLD only receive the (for them) relevant part of the QNAME. Another important detail is that the requested QTYPE=AAAA has been substituted by a different QTYPE. The reason for this is both privacy (QTYPE is also privacy sensitive information) and as delegation is expected, the recursive DNS server expects a QTYPE=NS response. Therefor the most intuitive solution would be to use QTYPE=NS in the request. However, some authoritative servers do not respond properly to QTYPE=NS queries (which is actually a violation of the protocol). Unbound (as shown in figure 5) solved this by substituting the QTYPE with the most-common QTYPE (A).

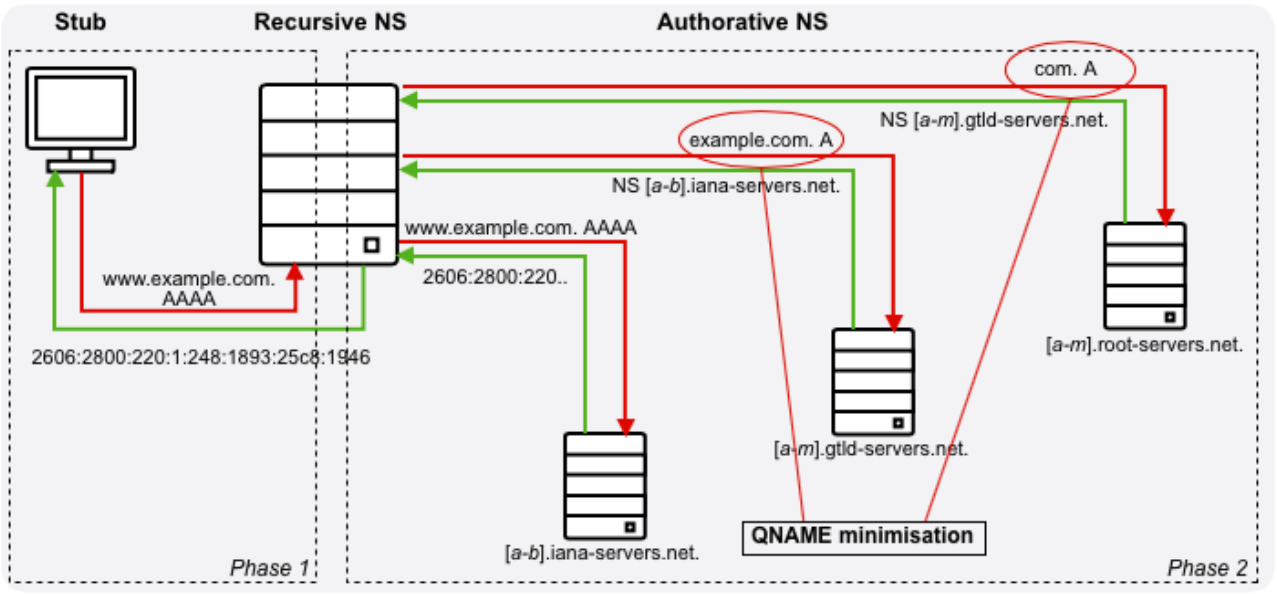

Figure 5: Domain name resolving with QNAME minimisation

Performance. Using QNAME minimisation can have both a positive and a negative effect on the performance [3]. The main positive effect comes from more optimal use of negative caching, due to the fact that less-unique domain names are sent to the higher level authoritative DNS servers. The negative effect is related to the QTYPE substitution. What is not shown in figure 5 is that there are actually two requests sent to iana-servers.net, the first with substituted QTYPE (A), the second one with the requested QTYPE (AAAA), because the recursive resolver does not know whether there is another level of delegation. This extra request has some impact on the performance.

The same thing happens when there are multiple levels of prefixes with the same authoritative server. Each level will be queried, instead of the authoritative server responding with the answer at the first one (because it does not get the full query).

Privacy. QNAME minimisation protects privacy by supplying each authoritative server in chain with the minimum required information about the QNAME and substituting the QTYPE. Only the last authoritative server in chain receives the full QNAME and actual QTYPE. There is no encryption in communication, so QNAME minimisation reduces impact only from eavesdroppers close to the higher level authoritative servers.

6.5 Oblivious DNS

In June 2018 Schmitt et al. published a paper with the title *"Oblivious DNS: Practical Privacy for DNS Queries"* [35]. This paper describes Oblivious DNS (ODNS) as a technique to protect privacy in the DNS. This technique is different compared to the previously mentioned techniques because besides the protection against eavesdropping on the communication, ODNS protects privacy on the recursive resolver as well (e.g., against logging).

Because of the recentness of the paper, there are no public implementations of ODNS yet, so using it is not possible. According to the paper, the authors plan to write an IETF specification for ODNS.

Concept. The idea behind ODNS is that each step in DNS resolution may only contain either one of two important pieces of information: The stub IP address OR the domain name/location. ODNS decouples those two pieces of information: The recursive resolver does know the IP address of the stub, but not the domain name. The (ODNS) authoritative server does know the domain name, but not the stub IP address. To achieve this decoupling, ODNS encrypts the domain name and its location. To prevent attacks where databases with encrypted domain names and their corresponding plain text are used, each ODNS session uses its own unique session key for encryption. Recommended is to use a recursive resolver without ECS support to fully decouple the pieces of data.

In ODNS operation, the stub resolver will first request the public key of the closest ODNS server by querying for *special.odns.*. The recursive resolver will forward this request to one of the authoritative ODNS servers (which could be considered recursive resolvers - more about that later). By using an anycast address, the nearest server will be used. The ODNS authoritative server will now answer the query and adds its own public key as an additional resource record (EDNS).

The client now possess the public key of the nearest authoritative ODNS server. Subsequent DNS queries can now use encryption. The stub will generate a symmetric session key and encrypt the requested domain name with this session key (k). It will then append .odns. as a suffix, to route the query to the ODNS authoritative server. The stub will also encrypt the session key with the public key (PK) and append this to the DNS request. Figure 6 illustrates this query.

$$\{www.example.com\}_k.odns.$$
$$\{k\}_{PK}$$

Figure 6: Encrypted data in ODNS request

The ODNS authoritative server decrypts the session key with its local private key, and uses this session key to decrypt the domain name. The authoritative server will now act as a recursive resolver and will resolve the DNS request the conventional way. When the final answer is received, the ODNS authoritative server encrypts the requested domain name and a part of the answer with the session key (figure 7) and return it to the recursive resolver.

$$\{www.example.com\}_k$$
$$\{93.184.216.34\}_k$$

Figure 7: Encrypted data in ODNS response

Performance. Because the ODNS authoritative server acts as a recursive resolver, an extra step in the DNS resolution process is introduced. This extra step will cause additional latency and therefore decrease the performance of DNS resolution.

Privacy. ODNS offers broad privacy coverage: Both phases of DNS resolution are covered, and the (public) recursive resolver is not able to gather much insight in DNS communication. Because only the domain name and answer are encrypted, other data could still be used to fingerprint users (e.g., QTYPE, TTL). For ODNS it would be important to use a public resolver without ECS support.

7 Combining techniques

All of the techniques evaluated in section 6 improve privacy in a certain phase in DNS resolution. In this section we will analyze combinations of techniques and determine the level of DNS privacy protection for each combination from a user's perspective.

In figure 8 we show the coverage of each technique in DNS resolution. Techniques in group one and two operate in the first phase of DNS resolution: Between stub and recursive resolver. Techniques in group three and four operate in the second phase of DNS resolution: Between recursive resolver and authoritative servers.

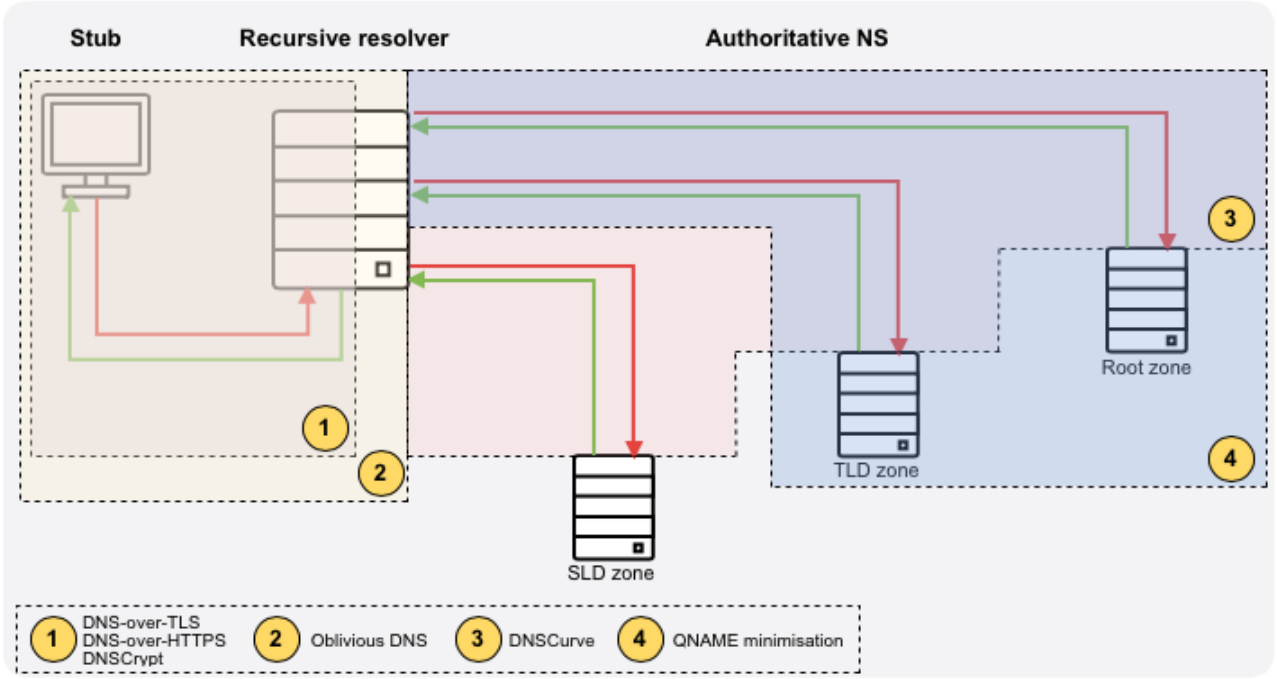

Figure 8: Coverage of DNS privacy techniques

We will omit the techniques *Oblivious DNS* (2) and *DNSCurve* (3) from our combinations. The reasoning behind this is that there is no publicly available technical implementation for Oblivious DNS, making the technique at this time unusable from a user's point of view. DNSCurve could be used by selecting a compatible public recursive resolver or running a private (local) recursive solver. However, DNSCurve requires support on the authoritative side as well. Unfortunately we were not able to find any DNSCurve compatible authoritative servers, which makes the technique useless for the user.

We will not combine an *ISP recursive resolver* with any techniques either, as those ISPs generally do not support any of the techniques. We have verified this for two major Dutch ISPs: Ziggo and KPN. A second reason is that in some countries ISPs are obliged by regulation to implement mechanisms for censorship [21].

As shown in figure 9 there are three privacy improving combinations a user could choose from. Below each combination is evaluated based on the level of privacy improvement. In the tables in appendix B we show the protection for the most important pieces of DNS data for each technique. Those tables do not take caching into account, which can make a huge difference. Therefore the combinations below use those tables as a guideline rather than a definitive source.

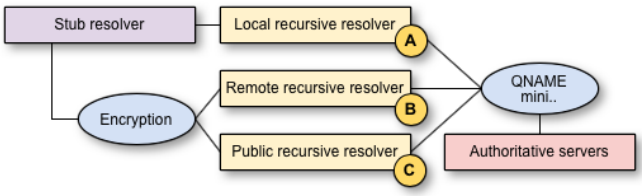

Figure 9: Resolver and technique combinations

Encryption. The techniques DoT, DoH and DNSCrypt (Group one techniques from figure 8) all implement encryption between the stub resolver and the recursive resolver. While the implementation and algorithm used differs, they all encrypt the same data. This results in the same level of privacy improvement for all three techniques. This research does not evaluate security of encryption techniques and therefore those techniques will be considered equal in the combinations below. DNS-over-HTTPS "Server Push" might offer advantages for privacy, but there are no known implementations of this yet.

Cache sharing. In section 4.3 of the research "Analysis of DNS Cache Effects on Query Distribution" [37] Zheng Wang describes the effects of cache sharing amongst multiple users. Wang explains the aggregation effect of the caching DNS server. Advantages of cache sharing are the higher cache rate and the flattening effect on the domain name distribution. The flattening effect is relevant for this research, as this effect results in less communication between the (caching) recursive resolver and authoritative servers.

7.1 Combination A

The first combination that is evaluated uses a local recursive resolver combined with QNAME minimisation.

Privacy phase 1. As the local network is usually a trusted network, using encryption between the stub and recursive resolver might not be required (but is possible).

Privacy recursive resolver. The local recursive resolver is under user's control. Therefore third parties cannot eavesdrop on communications and/or use query logging on this resolver.

Privacy phase 2. When using a local recursive resolver, the outbound IP address is shared between the recursive resolver and the stub resolver. This results in no obfuscation of the stub's IP address upstream. QNAME minimisation truncates the QNAME of DNS queries to the minimum required for each authoritative server. However, eavesdroppers close to the recursive resolver or close to the last authoritative server in the chain will still be able to intercept the full DNS packet and the stub's IP address.

Caching limits upstream communication to authoritative servers for repeated DNS requests. The relatively small user base of this recursive resolver reduces the effect of caching compared to a public resolver.

7.2 Combination B

The second combination we will evaluate is a combination of a group 1 technique with a private remote recursive resolver and QNAME minimisation.

Privacy phase 1. Encryption (a group 1 technique) between the stub and recursive resolver ensures data confidentiality in this phase of DNS communication.

Privacy recursive resolver. The remote recursive resolver is under users' control. Therefore third parties cannot eavesdrop on communications and/or use query logging on this resolver.

Privacy phase 2. When using a remote recursive resolver, the stub's IP address is obfuscated by the recursive resolver. Without configuring a remote recursive resolver, the stub will use the (ISPs) recursive resolver set at each Internet location (e.g., WiFi hot spot). The remote recursive resolver usually has a static (= permanent) IP address. Therefore, when the remote recursive resolvers cache is not shared amongst multiple users, the permanent IP address would decrease privacy compared to a changing ISPs resolver address at every Internet location.

QNAME minimisation truncates the QNAME of DNS queries to the minimum required for each authoritative server. However, eavesdroppers close to the recursive resolver or close to the last authoritative server in the chain will still be able to intercept the full DNS packet but without the stub's IP address.

Caching limits upstream communication to authoritative servers for repeated DNS requests. The relatively small user base of this recursive resolver reduces the effect of caching compared to a public resolver.

7.3 Combination C

Combination C involves one of the group 1 techniques (DoT, DoH or DNSCrypt) together with a public recursive resolver and QNAME minimisation.

Privacy phase 1. Encryption between the stub and public recursive resolver ensures data confidentiality in this phase of DNS communication.

Privacy recursive resolver. The public recursive resolver is not under users' control. Third parties could therefore eavesdrop and/or log DNS communication. Some public recursive resolvers (e.g., Cloudflare and Quad9) offer a privacy service (without logging), but there is no way to verify this.

Privacy phase 2. When using a public recursive resolver, the stub's IP address is obfuscated by the recursive resolver. Some public resolvers use QNAME minimisation which limits the contents of requests to the minimum required. However, eavesdroppers close to the recursive resolver or close to the last authoritative server in the chain will still be able to intercept the full DNS frame but without the stub's IP address.

Caching limits upstream communication to authoritative servers for repeated DNS requests. Popular public resolvers have a bigger user base, which increases the efficiency of caching. More users relates to a bigger chance in them requesting the same DNS queries.

7.4 Comparing A,B and C

When comparing the three previous combinations, we conclude that combination A offers the least privacy improvement for a user: The recursive resolver does not obfuscate the user's IP address, which links DNS frames in the iterative phase (phase 2) of DNS resolution to the user. Combination B offers more privacy, because the user's IP address is obfuscated. However, both combinations A & B lack efficient caching as the amount of users using the recursive resolver is generally much smaller compared to a public resolver

(e.g., Google).

Combination C features efficient caching and IP obfuscation to upstream authoritative servers. However, unlike the previous two combinations third parties could log DNS communication on the recursive resolver. Fortunately, we can reduce the impact of logging by making another combination, which is discussed below.

7.5 Adding a forwarding resolver

The combinations evaluated so far assume a common DNS setup is used like shown in figure 1. However, adding extra types/locations of DNS resolvers in sequence might have a positive effect on privacy in DNS. In this paragraph we will evaluate such a setup. The only DNS server type we can add is a forwarding resolver. Stub (clients) and authoritative servers can't be added for obvious reasons, and only one server in the chain can do iterative lookups to authoritative servers.

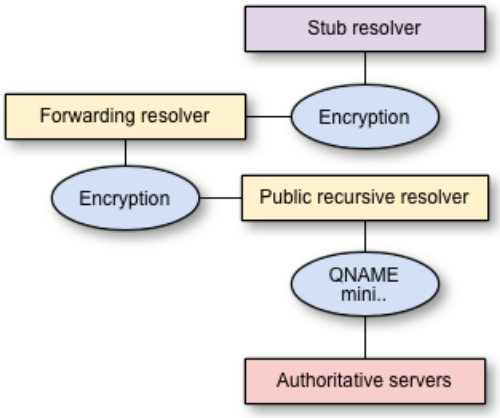

Figure 10: Combination C with a forwarding resolver

The disadvantage of using a public resolver is that a third party could log DNS communication on the recursive resolver. This log will then include both the stub's IP address and the requested domain name. If we add a remote forwarding resolver before the public resolver (as shown in figure 10) we decouple this data and still may enjoy optimal caching. The stub's IP address is now obfuscated by the forwarding server. The latter still has a full view of DNS communication, but this is not an issue as this server is under users' control.

In this setup the stub contacts a remote forwarding resolver securely using an encryption technique. This forwarding resolver does contact a public recursive resolvers, again using encryption. The public recursive resolver will then contact the specific authoritative servers using QNAME minimisation.

To make this set-up even more effective, the remote forwarding DNS server could be shared amongst multiple trusted users. Multiple users could be family members, multiple devices (computers / mobile) and such.

8 Conclusion & Discussion

In this section we discuss the results of our research, its limitations and identify possibilities for future research.

8.1 Conclusion

This paper has presented a solution to improve privacy in DNS resolution available to the user. We analyzed the privacy issues in DNS resolution. An experiment to investigate the implementation of the ECS extension on various public recursive resolvers was conducted. Modern techniques to improve DNS privacy were evaluated, and we discussed a technique that was recently published. Finally, we combined techniques available to the user to achieve the highest privacy improvement in DNS resolution.

We conclude that there are several techniques and combinations which all improve privacy. However, decoupling data by introducing a forwarding resolver offers the highest privacy improvement available to the user. Encryption ensures confidentially between the stub, forwarding and recursive resolver. Caching on multiple levels and QNAME minimisation limits the (unencrypted) DNS communication between the recursive and authoritative DNS servers. For better privacy a public recursive resolver that does not implement ECS and offers a no-logging server (although there is no guarantee on that) should be chosen.

8.2 Discussion

We have identified some limitations in our research.

The effectiveness of the proposed solution is influenced by unknown factors. For instance, the caching policies and amount of users on the forwarding and recursive resolver can make a lot of difference. Other factors are the details of the Internet connection of a user and different regulation in countries.

The combinations of evaluated techniques do improve DNS privacy, but also impose new challenges. Even when using encryption for both DNS and HTTP, there are still ways to gather insight in this communication by eavesdroppers. On web servers it is possible to have multiple TLS certificates on a single IP address. For this use case a mechanism called SNI (Server Name Indication) is used to determine which TLS certificate should be used for a specific connection. Because the certificate is needed for the TLS handshake, the server name (often equal to the domain name) is unencrypted. There is work in progress [24] to mitigate this issue.

Another concern is fingerprinting based on metadata like timestamps and packet size. Without encryption in DNS, there is no harm in such metadata, because the DNS data is accessible anyway. Especially the packet size is interesting, because it could be used to derive the question that was asked. To solve this, a specification [29] is written that proposes padding packets to a generic size.

8.3 Future work

When Oblivious DNS (ODNS) publishes a specification or implementation, research could analyze the advantages of using Oblivious DNS in combination with other techniques. Also interesting would be to see what the implications are of ODNS not encrypting the full communication, and how metadata like TTL and QTYPE could be used to fingerprint users.

A different approach to combining techniques that cover a certain phase of DNS resolution is to use a network designed for anonymous communication, like The Onion Router (Tor). Research could evaluate whether using such networks is a feasible way for DNS resolution.

In section 5.1.1 we note that there is no guarantee that DNS providers will honor the /0 source prefix length to prevent Client Subnet forwarding in DNS. Future research could investigate whether popular public DNS resolvers do honor this.

Evaluation of the differences between the phase one techniques DoT and DoH is necessary. There are currently discussions in the Internet community about which one of the two protects the user the most for fingerprinting based on metadata.

Finally, research could look into the parallel use of public DNS recursive resolvers to improve privacy. Randomly splitting requests over multiple recursive resolvers might impact DNS privacy. Research could look into positive/negative effects of such an implementation.

A Appendix: Public DNS resolvers

Public resolver	IP addresses	ECS used
Google	8.8.8.8 8.8.4.4 2001:4860:4860::8888 2001:4860:4860::8844	**Yes**
Cloudflare	1.1.1.1 1.0.0.1 2606:4700:4700::1111 2606:4700:4700::1001	No
Norton ConnectSafe	199.85.126.10 199.85.127.10	No
OpenDNS	208.67.222.222 208.67.220.220 2620:0:ccc::2 2620:0:ccd::2	**Yes** [1]
Open NIC DNS	(closest two resolvers) 193.183.98.66 51.254.25.115 2a00:dcc0:dead:b242::42 2001:41d0:2:73d4::125	No
DNS.Watch	84.200.69.80 84.200.70.40 2001:1608:10:25::1c04:b12f 2001:1608:10:25::9249:d69b	No
Level3 DNS	209.244.0.3 209.244.0.4 4.2.2.[1-6]	No
Comodo Secure DNS	8.26.56.26 8.20.247.20	No
UltraDNS	156.154.70.1 156.154.71.1	No
Dyn	216.146.35.35 216.146.36.36 2607:f590:f2::2	**Yes**
Ziggo	62.179.104.196 213.46.228.196 212.54.44.54 212.54.40.25 2001:b88:1002::10 2001:b88:1202::10	No
KPN	194.151.228.18 194.151.228.34	No

1. Registration required to receive ECS data [15] [38].

B Appendix: DNS data protection per technique

area of communication	data	DNS-over-TLS	DNS-over-HTTPS	DNSCrypt	QNAME mini...	Local	Remote	ISP	Public
Stub-Recursive	IP-address of stub resolver	-	-	-	-	-	-	-	-
Stub-Recursive	IP-address of recursive resolver	-	-	-	-	-	-	-	-
Stub-Recursive	Request contents	✓	✓	✓	-	-	-	-	-
Stub-Recursive	Request record type(s)	✓	✓	✓	-	-	-	-	-
Stub-Recursive	Response contents	✓	✓	✓	-	-	-	-	-
Stub-Recursive	Response record type(s)	✓	✓	✓	-	-	-	-	-
Recursive-Authoritative	IP-address of recursive resolver	-	-	-	-	-	-	-	-
Recursive-Authoritative	IP-address of authoritative resolver	-	-	-	-	-	-	-	-
Recursive-Authoritative	Network address of stub resolver	-	-	-	-	-	✓[5]	✓[5]	✓[5]
Recursive-Authoritative	Request contents	-	-	-	✓[6]	-	-	-	-
Recursive-Authoritative	Request record type	-	-	-	✓[6]	-	-	-	-
Recursive-Authoritative	(intermediate) Response contents	-	-	-	✓[6]	-	-	-	-
Recursive-Authoritative	(intermediate) Response record type(s)	-	-	-	✓[6]	-	-	-	-

Table 1: Eavesdropping: Protection of each technique against a certain data type

location	data	DNS-over-TLS	DNS-over-HTTPS	DNSCrypt	QNAME mini...	Local	Remote	ISP	Public
Recursive resolver	IP-address of stub resolver	-	-	-	-	✓	✓	-	-
Recursive resolver	IP-address of recursive resolver	-	-	-	-	✓	✓	-	-
Recursive resolver	Request contents	-	-	-	-	✓	✓	-	-
Recursive resolver	Request record type(s)	-	-	-	-	✓	✓	-	-
Recursive resolver	Response contents	-	-	-	-	✓	✓	-	-
Recursive resolver	Response record type(s)	-	-	-	-	✓	✓	-	-
Authoritative server	IP-address of recursive resolver	-	-	-	-	-	-	-	-
Authoritative server	IP-address of authoritative resolver	-	-	-	-	-	-	-	-
Authoritative server	Network address of stub resolver	-	-	-	-	-	✓[5]	✓[5]	✓[5]
Authoritative server	Request contents	-	-	-	✓[6]	-	-	-	-
Authoritative server	Request record type	-	-	-	✓[6]	-	-	-	-
Authoritative server	(intermediate) Response contents	-	-	-	✓[6]	-	-	-	-
Authoritative server	(intermediate) Response record type(s)	-	-	-	✓[6]	-	-	-	-

Table 2: Logging: Protection of each technique against a certain data type

5. Only if ECS is not used.
6. Depends on the position of this server in the chain.

Acknowledgements

We are grateful to Ralph Dolmans and Martin Hoffmann from NLnet Labs for facilitating this research, their help with various subjects and reviewing this paper.

References

[1] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Dns security introduction and requirements. RFC 4033, RFC Editor, March 2005.

[2] S. Bortzmeyer. Dns privacy considerations. RFC 7626, RFC Editor, August 2015.

[3] S. Bortzmeyer. Dns query name minimisation to improve privacy. RFC 7816, RFC Editor, March 2016.

[4] S. Bortzmeyer. Next step for dprive: resolver-to-auth link. Internet-draft, IETF.org, december 2016.

[5] Danny Bradbury. Hacking wifi the easy way. Technical report, February 2011.

[6] S. Castillo-Perez and J. Garcia-Alfaro. Evaluation of two privacy-preserving protocols for the dns. 2009.

[7] European Commission. 2018 reform of eu data protection rules. https://ec.europa.eu/commission/priorities/justice-and-fundamental-rights/data-protection/2018-reform-eu-data-protection-rules_en, 2018. [Online; accessed June-1-2018].

[8] C. Contavalli, W. van der Gaast, D. Lawrence, and W. Kumari. Client subnet in dns queries. RFC 7871, RFC Editor, May 2016.

[9] J. Damas, M. Graff, and P. Vixie. Extension mechanisms for dns (edns(0)). RFC 6891, RFC Editor, April 2013.

[10] M. Dempsky. Dnscurve: Link-level security for the domain name system. Internet-draft, IETF.org, February 2010.

[11] S. Dickinson, D. Gillmor, and T. Reddy. Usage profiles for dns over tls and dns over dtls. RFC 8310, RFC Editor, March 2018.

[12] Sara Dickinson. Dns privacy - the problem. https://dnsprivacy.org/wiki/display/DP/DNS+Privacy+-+The+Problem, 2018. [Online; accessed June-1-2018].

[13] Sara Dickinson. Dns privacy - the solutions. https://dnsprivacy.org/wiki/display/DP/DNS+Privacy+-+The+Solutions, 2018. [Online; accessed June-1-2018].

[14] DNSCrypt.info. Dnscrypt version 2 protocol specification. https://dnscrypt.info/protocol/. [Online; accessed June-25-2018].

[15] doileak.com. The privacy risk of edns-subnet-client (ecs) or why using a public dns server might not improve your privacy. https://www.doileak.com/blog-Public-DNS-might-not-%20improve-privacy.html. [Online; accessed June-12-2018].

[16] H. Federrath, K. Fuchs, D. Hermmann, and C. Piosecny. Privacy-preserving dns: Analysis of broadcast, range queries and mix-based protection methods. https://svs.informatik.uni-hamburg.de/publications/2011/2011-09-14_FFHP_PrivacyPreservingDNS_ESORICS2011.pdf, 2011. [Online; accessed June-4-2018].

[17] Dharmesh Goyal. 10 best free and public dns servers you must try. https://technofizi.net/best-free-and-public-dns-servers/, 2018. [Online; accessed June-12-2018].

[18] Christian Grothoff, Matthias Wachs, Monika Ermert, and Jacob Appelbaum. Nsas morecowbell: Knell for dns. Technical report, 2017.

[19] Lawrence Hecht. Ssl adoption continues to rise. https://thenewstack.io/ssl-adoption-continues-to-rise/, 2018. [Online; accessed June-1-2018].

[20] P. Hoffman and P. McManus. Dns queries over https (doh). Internet-draft, IETF.org, June 2018.

[21] Henry L. Hu. The political economy of governing isps in china: Perspectives of net neutrality and vertical integration. Technical report, September 2011.

[22] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. Dns over tls: Initiation and performance considerations. Internet-draft, IETF.org, January 2016.

[23] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman. Specification for dns over transport layer security (tls). RFC 7858, RFC Editor, May 2016.

[24] C. Huitema and E. Rescorla. Issues and requirements for sni encryption in tls. Internet-draft, IETF.org, May 2018.

[25] Geoff Huston. The resolvers we use. https://labs.ripe.net/Members/gih/the-resolvers-we-use, 2014. [Online; accessed June-12-2018].

[26] IETF. Dns private exchange (dprive). https://datatracker.ietf.org/wg/dprive/about/. [Online; accessed June-12-2018].

[27] Dan Kaminsky. Dns 2008 and the new (old) nature of critical infrastructure. http://slideplayer.com/slide/6582796/. [Online; accessed June-25-2018].

[28] R. Licht and D. Lawrence. Client id in forwarded dns queries. Internet-draft, IETF.org, march 2017.

[29] A. Mayrhofer. The edns(0) padding option. RFC 7830, RFC Editor, May 2016.

[30] P. Mockapetris. Domain names - implementation and specification. RFC 1035, RFC Editor, November 1987.

[31] A. Mohaisen and A. Mankin. Evaluation of privacy for dns private exchange. Internet-draft, IETF.org, october 2015.

[32] Matthew Prince. Announcing 1.1.1.1: the fastest, privacy-first consumer dns service. https://blog.cloudflare.com/announcing-1111/, 2018. [Online; accessed July-4-2018].

[33] Quad9. Quad9 frequently asked questions. https://www.quad9.net/faq/#What_does_Quad9_log/store_about_the_DNS_queries, 2018. [Online; accessed July-4-2018].

[34] T. Reddy, D.Wing, and P. Patil. Dns over datagram transport layer security (dtls). RFC 8094, RFC Editor, February 2017.

[35] P. Schmitt, A. Edmundson, and N. Feamster. Oblivious dns: Practical privacy for dns queries. eprint arxiv:1806.00276, Princeton University, June 2018.

[36] M. Timmers. Dnscurve analysis. Research project 1, University of Amsterdam, February 2009.

[37] Zheng Wang. Analysis of dns cache effects on query distribution. Research article, Computer Network Information Center, July 2013.

[38] Kevin Wetter. Akamai to enable ecs for opendns & googledns on ipa/sxl network. https://community.akamai.com/customers/s/article/Akamai-to-Enable-ECS-for-OpenDNS-GoogleDNS-on-IPA-SXL-Network?language=en_US, 2015. [Online; accessed June-12-2018].

[39] Therese L. Williams. A longitudinal study of privacy awareness in the digital age and the influence of knowledge. Technical report, University of Arkansas, May 2017.

[40] Cory Wright. Understanding kaminsky's dns bug. https://www.linuxjournal.com/content/understanding-kaminskys-dns-bug. [Online; accessed June-12-2018].

[41] Nykolas Zal. Dns market share analysisâĂŁ-identifying the most popular dns providers. https://medium.com/@nykolas.z/dns-market-share-analysis-identifying-the-most-popular-dns-providers-80fefb2cfd05, 2018. [Online; accessed June-12-2018].

[42] F. Zhao, Y. Hori, and K. Sakurai. Analysis of privacy disclosure in dns query. May 2007.

[43] F. Zhao, Y. Hori, and K. Sakurai. Two-servers pir based dns query scheme with privacy-preserving. 2009.