JOHANNES KEPLER
UNIVERSITY LINZ

Author
David Alois Raab, BSc

Submission
Institute of
Networks and Security

Thesis Supervisor
Assoz.-Prof. Mag. DI Dr.
Michael Sonntag

Assistant Thesis
Supervisor
DI Dr. **Tobias Höller**, BSc

November 2025

Enhanced Privacy for DNS Resolution

Master's Thesis

to confer the academic degree of

Diplom-Ingenieur

in the Master's Program

Computer Science

JOHANNES KEPLER
UNIVERSITY LINZ
Altenberger Straße 69
4040 Linz, Austria
jku.at

Abstract

The Domain Name System (DNS) is one of the fundamental systems for Internet communication. Its underlying protocol was originally developed and standardized without consideration of privacy or security and has undergone only minor improvements since its introduction. This thesis examines DNS privacy threats in general, and specifically those that arise when DNS requests exit the Tor network.
An empirical evaluation of Tor exit relays was conducted to assess the extent of these issues on the Tor network, revealing that a substantial number of exit relays did not adhere to the Tor Project's recommendations. More than 32% of DNS requests were directed to only two organizations (Google: 20.8% and Cloudflare: 11.3%) and only about 65% of all DNS traffic supported DNSSEC validation.
To address the shortcomings of DNS, this thesis discusses and proposes privacy- and security-enhancing measures to DNS resolution, including the encryption and distribution of DNS requests and the use of the Tor network to anonymize the original requester.
To validate these improvements, publicly available encrypted DNS resolvers were assessed for mutual independence, applied security measures, censorship behavior, and performance. A proof-of-concept implementation demonstrates the feasibility of integrating the proposed measures.
This thesis aims to raise awareness of the privacy challenges inherent to DNS resolution and presents measures that can be applied to the existing DNS infrastructure without requiring modifications to the DNS protocol itself.

Kurzfassung

Das Domain Name System (DNS) ist eines der grundlegenden Systeme für die Kommunikation im Internet. Sein zugrunde liegendes Protokoll wurde ursprünglich ohne Berücksichtigung von Datenschutz und Sicherheit entwickelt und standardisiert und hat seit seiner Einführung nur geringfügige Verbesserungen erfahren. Diese Arbeit untersucht Datenschutzbedrohungen des DNS im Allgemeinen und insbesondere solche, die auftreten, wenn DNS-Anfragen das Tor-Netzwerk verlassen.
Eine empirische Untersuchung der Tor-Exit-Relays wurde durchgeführt, um das Ausmaß dieser Probleme im Tor-Netzwerk zu bewerten. Dabei zeigte sich, dass eine erhebliche Anzahl von Exit-Relays den Empfehlungen des Tor-Projekts nicht folgt. Mehr als 32% aller DNS-Anfragen wurden an lediglich zwei Organisationen geleitet (Google: 20,8% und Cloudflare: 11,3%) und nur etwa 65% des gesamten DNS-Datenverkehrs unterstützten die Validierung mittels DNSSEC.
Zur Behebung der Schwachstellen des DNS diskutiert und schlägt diese Arbeit Maßnahmen zur Verbesserung der Privatsphäre und Sicherheit bei der DNS-Auflösung vor, darunter die Verschlüsselung und Verteilung von DNS-Anfragen sowie die Nutzung des Tor-Netzwerks zur Anonymisierung des ursprünglichen Anfragenden.
Zur Validierung dieser Verbesserungen wurden öffentlich verfügbare verschlüsselte DNS-Resolver hinsichtlich ihrer gegenseitigen Unabhängigkeit, der angewandten Sicherheitsmaßnahmen, ihres Zensurverhaltens und ihrer Leistung untersucht. Eine Proof-of-Concept-Implementierung demonstriert die Umsetzbarkeit der vorgeschlagenen Maßnahmen.
Diese Arbeit soll das Bewusstsein für die Datenschutzprobleme bei der DNS-Auflösung schärfen und zeigt Maßnahmen auf, die auf die bestehende DNS-Infrastruktur angewendet werden können, ohne Änderungen am DNS-Protokoll selbst zu erfordern.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Description and Motivation

The Domain Name System (DNS) is one of the most important systems for online activities. Users rely on it when accessing websites, as well as for enabling applications and servers to communicate with each other.

The DNS originated in the 1980s, when the concept of a distributed system mapping domain names to their corresponding IP addresses was proposed and later standardized. Like many protocols from that era, it was designed without consideration for security or privacy. While secure successors have been standardized for protocols such as HTTP or FTP, DNS has received only a few enhancements, which still do not provide the level of privacy and security achieved by HTTPS or FTPS. Furthermore, these enhancements are not mandatory, are often not under the user's control, and only protect specific parts of the DNS communication path rather than providing a holistic approach.

In recent years, encrypted DNS protocols have emerged, and applications such as web browsers have started to implement them. However, overall adoption remains low, and problems such as the centralization of DNS data among large DNS resolver operators persist or have worsened.

DNS data is considered public, yet there is often no distinction made between the DNS information itself and associated metadata. Metadata such as the client's source IP address, combined with other data in the DNS request, enables adversaries to create user profiles. [1]

Because DNS traffic is easy to monitor and inexpensive to analyze, it is an attractive target not only for cybercriminals with monetary motives but also for surveillance agencies such as the National Security Agency (NSA) [2, 3].

Being such a crucial protocol affecting users' privacy while receiving little attention from them provides the primary motivation for this thesis.

Users often notice DNS only when it stops working, at which point they may switch from their ISP's DNS resolver to a public DNS resolver, thereby further contributing to DNS centralization [4].

These observations lead to the central question of this thesis: whether implementable measures or protocols already exist to enhance DNS privacy without changing the existing DNS infrastructure.

1.2 Objectives and Approach

The overarching goal of this thesis is to make DNS traffic more private and secure and to determine whether this can be achieved by encrypting, distributing, and anonymizing DNS requests.

A threat model is developed to define DNS privacy threats in general, as well as specific threats related to DNS resolution performed by Tor exit relays. To assess the current state of DNS on the Tor network, a practical analysis is conducted to test the DNS behavior of Tor exit relays.

The proposed improvements aim to enhance privacy by preventing adversaries from creating user profiles based on DNS data and by reducing the potential for correlation attacks on Tor, such as the DefecTor attack [88]. Additionally, users should be less exposed to censorship that relies on domain name blocking.

The proposed measures are applicable to the existing DNS infrastructure, using standardized and established protocols, such as encrypted DNS, without requiring changes to the DNS itself.

To evaluate the potential for improving decentralization through distributing DNS requests across multiple DNS resolvers, publicly available encrypted DNS resolvers are analyzed with respect to mutual independence, censorship, performance, and security measures such as DNSSEC and QNAME minimization. Furthermore, the feasibility of routing DNS requests through the Tor network to these DNS resolvers is examined.

Finally, a theoretical architecture incorporating the proposed improvements is presented, along with a proof-of-concept implementation.

Chapter 2

Background and Definitions

2.1 Secure Communication

The CIA triad describes the fundamental elements of security controls in information systems. CIA refers to *Confidentiality*, *Integrity*, and *Availability*. These terms have been complemented over time by non-repudiation and trust, among others [5].

- Confidentiality

  > *Confidentiality means that information is protected by preventing the unauthorized disclosure of information. When confidentiality is compromised, this means that there has been an unauthorized disclosure of information.* [6]

- Integrity

  > *Integrity means that information is protected by keeping it intact. When the integrity of information is compromised, this means that the information has been modified without authorization from its original form.* [6]

- Availability

  > *Degree to which a system or component is operational and accessible when required for use.* [7]

2.2 Trust

The Cambridge dictionary defines the verb trust as

> *to believe that someone is good and honest and will not harm you, or that something is safe and reliable.* [8]

In this thesis, some entities need to be trusted as they can see and, in some cases, manipulate unencrypted information. To reduce the amount of trust needed, the focus is on encrypting information where and when possible, and providing as little unencrypted information as possible to a single entity.

2.3 Client and User

In this work, a client is seen as the technical representation of a user. While a user can have personal behavior, a client has technical properties, e.g., an IP address.

2.4 The Domain Name System

The Domain Name System (DNS) maps domain names to IP addresses. IP addresses are needed when a client wants to access a resource on the Internet. Domain names are more memorable than IP addresses for humans, which makes them indispensable for web browsing. Additionally, they can contain brand names or describe the content of the resource. Domain names can serve as persistent identifiers for resources that change their IP addresses.

Transport Layer Security (TLS) provides a secure communication channel by ensuring confidentiality through encryption of data, authentication by verifying the server's identity (and optionally the client's identity), and integrity through protection against data modification. Each TLS connection begins with a handshake between the client and the server. In this handshake domain names are included as part of the Server Name Indication (SNI) extension. The server uses the SNI value to determine the correct TLS certificate, which is necessary when multiple domain names are hosted on the same server, e.g. multiple websites using HTTPS on a single IP address. [9, 10]

Before the DNS was in place, hosts on the Internet were listed in a global table, maintained by the Network Information Center (NIC). The size of the table and the high frequency of updates became unmanageable and the idea of a distributed database for domain names was introduced in 1983 in RFC 882 [11]. The first implementation of the current DNS was specified in 1987 in RFC 1034 [12] and RFC 1035 [13]. Since then, the DNS has been complemented, updated and extended several times.

The Domain Name System can be divided into three major components [12]:

- The Domain Name Space and Resource Records
- Name Servers
- Resolvers

2.4.1 Domain Name Space and Resource Records

The domain name space is a tree-structured name space. Its nodes and leaves hold a set of data in the form of resource records. The data can be extracted by query operations. It supports both UDP and TCP on port 53 for communication. [12]

[14] describes a domain name as an ordered list of one or more labels. Labels are separated by a dot and are ordered by the distance from the root, which is also a dot. In the common display format, the dot representing the root is not shown. The first label after the root is called the Top-Level Domain (TLD). Each

domain containing an additional label is called a subdomain. A zone combines the information for the domain names it is responsible for.

Resource Records

A resource record is the set of information for a node, which is identified by a domain name.

The following fields are specified for a resource record: [12, 13]

- *Name*: The domain name of the node.
- *Type*: Defines the record type of this resource record.
- *Class*: Identifies the protocol family. In this work class is always IN, which stands for Internet System.
- *TTL*: The time to live defines the maximum time in seconds a record can be cached until it needs to be queried again.
- *RDLENGTH*: Specifies the length of the RDATA field in octets.
- *RDATA*: Actual information of the record, having a variable length defined in RDLENGTH.

Record types and their information are: [12, 13, 15, 16]

- *A*: IPv4 address
- *AAAA*: IPv6 address
- *NS*: Domain name of the authoritative name server.
- *CNAME*: An alias pointing to a domain name.
- *PTR*: Domain name pointer, used for reverse lookups.
- *TXT*: Text strings that contain any information.
- *DNSKEY*: Public DNSSEC key and information about the type and the algorithm of the key.
- *RRSIG*: Resource record signature, record set signature of an DNSSEC entry.
- *DS*: Delegation signer, refers to a DNSKEY.

2.4.2 Name Servers

Name servers are servers holding information about one or more zones for which they are authoritative and respond to requests for their zones without querying other name servers. They are logically organized by a reverse tree structure and respond with referrals to the corresponding name servers for their child zones. [12, 13]

The IP addresses of 13 root name servers are defined and known by recursive resolvers. More than 1500 servers globally use DNS anycast for the 13 root name server identifiers[1].

[1]https://www.icann.org/en/rssac/faq#number-of-root-servers

2.4.3 Resolvers

Resolvers are programs that either resolve DNS requests directly or forward them to another resolver, returning the responses to their clients. They can implement a cache where already resolved domain names are stored for the maximum TTL. A cache therefore can increase the performance of the DNS resolution and reduce the number of queries. [12, 13]

Resolvers can be categorized according to their functionality: [17, 18]

- *Stub Resolver*: A stub resolver is implemented at the client and does not perform recursive resolution by itself. It answers requests from its cache or forwards them to a recursive or forwarding resolver.
- *Forwarding Resolver*: A forwarding resolver receives requests and answers them from the cache or passes them on to another resolver. They often stand between stub and recursive resolvers.
- *Recursive Resolver*: A recursive resolver does the actual resolving by sending requests to the corresponding name servers.

2.4.4 Process of Resolving DNS Requests

The following example describes the process for resolving the domain name www.tordns.ovh.

The client sends a DNS request for the domain name www.tordns.ovh from the stub resolver of its operating system to a forwarding resolver (1) or directly to a recursive resolver (2). There might be no forwarding resolver, one forwarding resolver or multiple forwarding resolvers. The recursive resolver resolves the domain name by its labels starting at the root. If QNAME minimization is applied, the recursive resolver only sends a request for .ovh, otherwise for www.tordns.ovh (3). The root name server responds with a referral to the name server of .ovh (4). The recursive resolver continues with a request for tordns.ovh (or www.tordns.ovh if QNAME minimization is not applied) to the name server responsible for the zone .ovh (5). The name server responds with a referral to the name server of tordns.ovh (6). This continues until the last label is reached. Finally, the recursive resolver sends a request for www.tordns.ovh to the name server responsible for the zone tordns.ovh (7) and gets the IP address for www.tordns.ovh, or an error if the domain does not exist (8). The recursive resolver sends a response back to the client (9, 10). Figure 2.1 illustrates this process.

Each resolver can cache responses and check the cache before forwarding requests. If a domain name is cached, the resolver can answer the request immediately without further lookups.

2.4.5 Extension Mechanisms for DNS (EDNS(0))

The extension mechanisms for DNS (EDNS(0), EDNS0, or EDNS) are used to extend the original limit of 512 byte for DNS using UDP. If a response does not fit into the UDP limit, the UDP response states that the answer is truncated

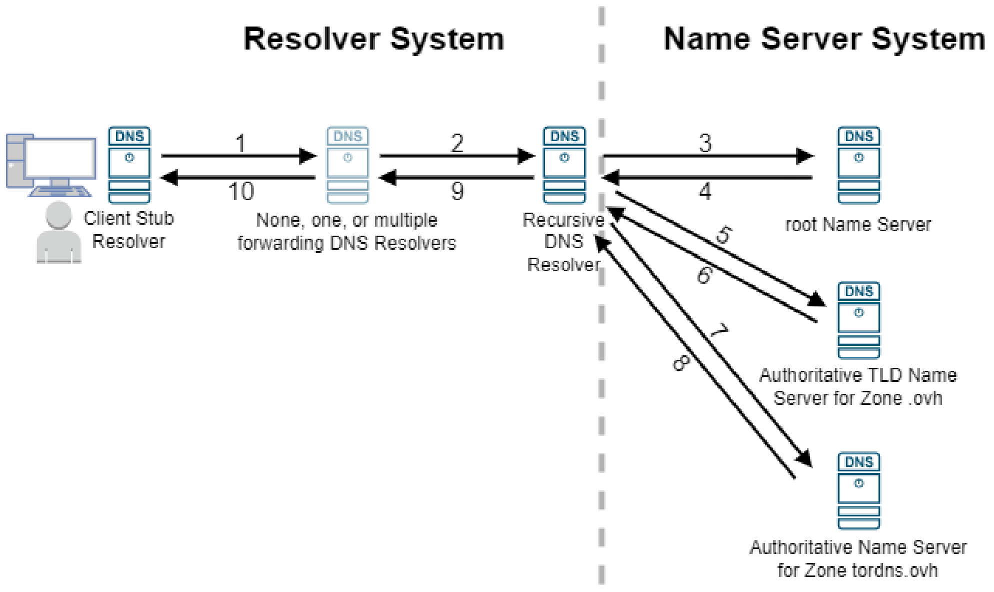
Resolver System
Name Server System
DNS
1
10
2
9
3
4
5
6
7
8
Client Stub Resolver
None, one, or multiple forwarding DNS Resolvers
Recursive DNS Resolver
root Name Server
Authoritative TLD Name Server for Zone .ovh
Authoritative Name Server for Zone tordns.ovh

Figure 2.1: Process of DNS resolution

which causes a retry over TCP. E.g. IPv6 and DNSSEC may require larger response sizes. EDNS allows the negotiation of larger message response sizes and recommends not exceeding a maximum payload size of 4096 bytes although networks may be capable of larger sizes. Apart from the extended message size, EDNS adapts to a more diverse use of DNS and makes it more scalable through adding control information by using the EDNS Option Codes (OPT). [19]

2.4.6 EDNS Client Subnet (ECS)

ECS is an EDNS option which allows resolvers to include information about the client's network, i.e. the IP address of the network and the subnet, in the DNS request. By identifying the client's network, an authoritative name server can respond with the closest IP address of the requested resource if there is more than one available. Although ECS information is meant to be used by intermediate resolvers, it can already be added by the client and forwarded by resolvers. However, the location of clients sending DNS requests might be different from the location of the recursive resolver. For privacy protection it is highly encouraged to use a subnet bit length not more specific than 24 for IPv4 and a subnet bit length not more specific than 56 for IPv6. ECS is optional and should be disabled by default. [17]

2.4.7 DNS Security Improvements

When the DNS was originally designed, aspects such as confidentiality, integrity, and authentication were not considered. Over time, the DNS received some non-mandatory security improvements.

Domain Name System Security Extensions (DNSSEC)

DNSSEC was designed to provide origin authentication and integrity of data in responses from name servers. Additionally, it allows the authentication of non-existence of a query name through next secure records (NSEC). However, it was never the goal of DNSSEC to provide confidentiality, or authentication of the requester. It requires EDNS as responses contain DNSSEC information larger than 512 bytes, and new resource types are introduced. [16, 20]

To validate a DNSSEC signed domain name each name server must provide a delegation signer (DS) record for the child zone, beginning at the root. A resolver queries the DS record of the parent zone to prove the validity of the DNSKEY of the child zone, and uses the DNSKEY and the RRSIG of the child zone to validate if the domain name is signed correctly. DNSSEC validation is done by recursive resolvers as each label of the domain name needs to be validated. Therefore, DNSSEC provides protection only to the path between a recursive resolver and an authoritative name server. [16, 20]

Next Secure Record 3 (NSEC3)

[21] introduced hashed authenticated denial of existence using NSEC3 instead of NSEC. This prevents zone walking, where all domain names of a zone are revealed. However, NSEC3 is vulnerable to offline enumeration of zone contents. To prevent this, a draft of its successor NSEC5 defines the use of verifiable random functions for authenticated denial of existence [22, 23].

DNS Query Name Minimization (QNAME Minimization)

QNAME Minimization reduces the information contained in queries sent by recursive DNS resolvers to authoritative name servers. It enhances the privacy of DNS by applying the principle of Data Minimization. It reduces the exposed information but does not provide confidentiality. Each name server receives the minimum information needed to answer the query. E.g. the root name server receives a query only for the TLD instead of the complete domain. The TLD name server receives only one more label of the domain name, and so on. Additionally, the query type can be obfuscated by using any type except for the final domain name, e.g. using type A in the query to the root name server when the final domain name asks for the MX type. Non-cached domain names containing a high number of labels would cause a high number of queries and lead to a performance decrease when applying QNAME minimization. Also, multiple labels can be in the same zone and be answered by the same authoritative name server. Mechanisms must be implemented to limit the number of queries per resolution. Therefore, the standard suggests limiting the number of queries where only one additional label is added to the domain name, and limiting the overall number of queries for resolving one request. [24]

Use of Bit 0x20 in DNS Labels

Bit 0x20 is determined by the case of the letters a-z and A-Z in the query name, i.e. 1 for lowercase letters and 0 for uppercase letters. Randomizing the case of the query name in the request and verifying the exact same case in the response gives additional protection against cache poisoning as randomizing the 16-bit transaction ID is not enough to prevent attacks on cache poisoning. The number of bits is the number of the letters in the query name. [13] defines that the sent information must be identical to the data returned in the response's question section. In practice the question section in the request is copied to the response, also retaining the case of the domain name. The draft [25] suggests making it mandatory that the domain name in the request is copied exactly to the response. [26]

In this thesis, the term *Case Randomization* is used to refer to this security measure.

Encrypted DNS

The use of encryption protocols can add confidentiality, integrity, and authentication to the DNS for the communication between a client and a recursive DNS resolver. The implementation of the DNS encryption protocols is performed at the application level and the authority of choosing the DNS resolver also shifts from the operating system to the application. DNS traffic between recursive DNS resolvers and authoritative name servers remains unencrypted.

- *DNS over HTTPS (DoH)*: DoH hides DNS traffic by tunneling it within HTTPS traffic where one query requires one HTTPS request. HTTPS uses Transport Layer Security (TLS), which provides integrity and confidentiality. A client connects to the URI of a recursive DNS resolver via port 443. A TLS certificate must be provided by the server and is used to authenticate the DoH server. Both GET and POST requests must be supported by a DoH server. Servers can provide both DoH and conventional HTTPS data making it harder to detect DoH. To resolve the IP address of the DNS resolver's domain name an initial bootstrap DNS request to another DNS resolver is needed, or IP-based URIs and certificates are used. [27]
- *DNS over TLS (DoT)*: The client establishes a TLS session to connect to a recursive DNS resolver. By default, port 853 is used, but a different port can be agreed on between client and server. A client should not wait for a response to send another request to minimize latency. The order of responses might differ from the order of requests sent to the server. The fields QNAME, QCLASS, and QTYPE in a question section of a response must be checked by the client to match the data in the request. There are two privacy profiles. In the opportunistic privacy profile, no authentication of the server is required. In the out-of-band key-pinned privacy profile the client authenticates the server by validating a set of subject public key info (SPKI) fingerprints. The SPKI pins are provided by the server to the client through an out-of-band channel in advance. In the latter privacy profile strong privacy can be guaranteed. [28]
- *DNSCrypt*: A DNS resolver offering DNSCrypt publishes an identifier called stamp. DNSCrypt clients use these stamps which contain all parameters for the connection. The client receives a public set of signed certificates from the DNS resolver and must validate them. UDP and TCP can be used by default on port 443. Various encryption algorithms are supported. The DNS

resolver sends a set of its supported algorithms, and the client can choose the algorithm for the communication. [29]

2.5 The Tor Project

The Tor Project is a non-profit organization which develops, maintains, and organizes Tor. The Tor network is a low-latency, circuit based, anonymization network, which relies on nodes that are run by volunteers providing resources, especially bandwidth. Software and guidance to run nodes and allow clients to connect to the network is provided and maintained by the Tor Project. The goal of Tor is to provide an open network that defends privacy and protects against tracking, surveillance, and censorship. [30]

The source code of the Tor Project is managed through the Tor GitLab repository [31].

2.5.1 Concept and Architecture of Tor

The concept is based on layered encryption, hence the name The Onion Router. Each layer employs a distinct encryption key, adding an additional encryption to the transmitted data. The Tor network consists of numerous Tor nodes, also referred to as relays, each serving specific functions within the network. In general, relays are publicly listed nodes that forward traffic and form the backbone of Tor's anonymizing structure.

- *Entry Guard*: An entry guard, also referred to as a guard relay or entry node, serves as the entry point for Tor clients connecting to the Tor network and represents the first hop in a Tor circuit. Traffic between clients and entry guards is encrypted. An entry guard knows the IP address of the client and the middle node of an established circuit. [30, 32]
- *Bridge Relay*: A bridge relay or bridge is an alternative to an entry guard and enables a client to connect to the Tor network. In contrast to publicly known entry guards, bridges are not listed in the public Tor directory and therefore are harder to identify and block for ISPs and governments. [32]
- *Middle Node*: A middle node, or middle relay, is a type of node that forwards traffic between relays and is neither used for entering nor exiting the Tor network. [32]
- *Exit Relay*: The exit relay is the representative for the Tor client. Traffic leaves the Tor network at the exit relay, and the destination, e.g. a website, sees the IP address of that exit relay. An exit relay sees the client's traffic but does not know its identity, i.e., its IP address. [32]

A typical Tor circuit for connecting a client to the Internet and routing its traffic over the Tor network includes an entry guard, one middle node, and an exit relay. Figure 2.2 [33] illustrates a typical three hop circuit.

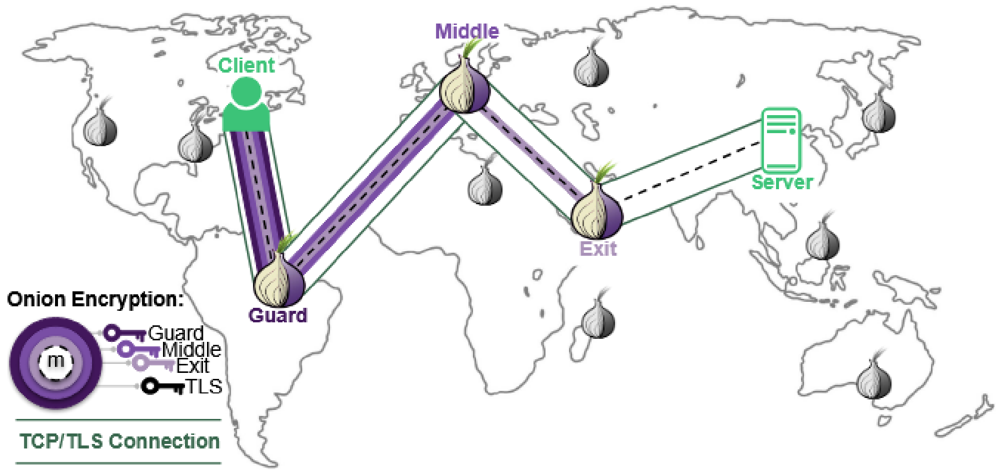

Figure 2.2: Three hop Tor circuit

2.5.2 DNS Resolution in Tor

Exit relays connect to resources on the Internet as representatives on behalf of clients using the Tor network. Consequently, they are also responsible for the DNS resolution of the requested resources. When using applications like Tor Browser or systems like TAILS and WHONIX, DNS requests are sent to Tor by their default configuration. Custom applications can use Tor's DNS resolution by sending the requests to a port, configured to listen to UDP DNS requests in the Tor software running on the client. It is deactivated by default and can be enabled with the option DNSPort in the configuration file. DNS requests sent to the exit relay use the Tor circuit and its encryption. Only DNS record types A, AAAA, and PTR are resolved[2]. Other records such as SRV or MX cannot be resolved by sending requests for these records to the Tor exit relay. However, the requests can be sent as TCP DNS requests to a specific DNS resolver over the Tor network. [34, 35]

Tor does not use DNS caching on the client side by default since Tor version 0.2.4.7-alpha[3] and DNS client-side caching was deprecated in Tor version 0.2.9.2-alpha[4]. An exit relay uses a common cache for all DNS requests received across its active circuits. It adjusts the TTL value in DNS responses by truncating it to five minutes if the original TTL is less than 5 minutes, or to 60 minutes if the original TTL is greater than 5 minutes. Additionally, an exit relay randomizes the truncated TTL to a value between (TTL − 4 minutes) and (TTL + 4 minutes) to mitigate the risk of attacks on the cache[5]. If a domain name of a request is not cached on the exit relay, the exit relay needs to resolve it on the Internet.

Exit relay operators can operate their own DNS resolver or select any DNS resolver available to the exit relay. However, there are recommendations by Tor to choose an appropriate DNS resolver. This is further discussed in section 4.2.

[2]https://gitlab.torproject.org/tpo/core/tor/-/blob/release-0.4.8/doc/man/tor.1.txt line 1209-1216

[3]https://gitlab.torproject.org/tpo/core/tor/-/blob/tor-0.2.4.7-alpha/ChangeLog line 51-68

[4]https://gitlab.torproject.org/tpo/core/tor/-/blob/tor-0.2.9.2-alpha/ChangeLog line 74-79

[5]Confidential: TROVE-2021-009: Improved DNS cache oracle, 08.09.2022, https://gitlab.torproject.org/tpo/core/tor/-/issues/40674

Exit relays apply case randomization for DNS requests sent to DNS resolvers by default[6].

2.5.3 Software and Applications

Two main implementations enable clients and servers to connect to the Tor network.

The first implementation, simply called Tor, also referred to as Tor network, Core Tor, or little-t tor, is written in the C programming language. It is free and open source, licensed under the 3-clause BSD license[7] [36]. Depending on the configuration defined in the torrc file, it can operate as any type of Tor node or function as a local proxy to the Tor network [36].

The second implementation, named Arti, is a complete rewrite of the C Tor codebase in the Rust programming language. Arti is designed as an embeddable library for integration into other applications and emphasizes a modular and reusable architecture. At the time of writing, Arti was ready for testing but not yet recommended for production use. It is intended to replace the C-based implementation in the future. [37]

The following applications and systems integrate Tor software to enable other software to connect to the Tor network.

- *Tor Browser*: The Tor Browser is a modified browser that allows easy access to the Tor network for browsing the web, also for novice users. It uses the Tor software for connecting to the Tor network and is based on Firefox ESR (Extended Support Release) [36], preconfigured with a secure browser configuration that should not be altered by users. Website visits are isolated from each other to prevent third-party trackers and ads from following a user. On closing the application, it deletes the browser history as well as cookies. It protects users from network surveillance, as observers can only detect a connection to the Tor network, but not specific websites visited. From the perspective of web servers, all Tor Browser users appear identical, preventing browser fingerprinting and device identification. [38, 39]
- *Mobile Applications* [40]:
  - *Orbot*: Orbot is a VPN application that integrates the Tor software to route the traffic of other apps, such as email clients or messaging apps, through the Tor network on Android and iOS. Due to memory restrictions on Apple devices, the iOS version may be less reliable. Orbot is maintained by The Guardian Project. [41]
  - *Tor Browser for Android*: Tor Browser for Android is the mobile version of the Tor Browser and provides privacy protections for Android users. It uses an app-integrated version of Orbot to connect to the Tor network. [42]
  - *Onion Browser for iOS*: For iOS, the Tor Project recommends the Onion Browser app, which is open source. Due to Apple's platform restrictions,

[6] https://gitlab.torproject.org/tpo/core/tor/-/blob/release-0.4.8/doc/man/tor.1.txt line 2655-2661

[7] https://gitlab.torproject.org/tpo/core/tor/-/raw/HEAD/LICENSE

all browsers on iOS are required to use the WebKit browser engine. Consequently, the Onion Browser cannot offer the same level of privacy protection as the Tor Browser for Android. [42, 43, 44]

- *Systems*
  - *TAILS*: TAILS is a free and open-source live operating system that can be started by booting from USB stick. It has already installed a selection of applications and uses the Tor network for all connections to the Internet. Every start of the operating system results in a clean state, every shutdown clears all traces from the operating system, e.g. visited websites, opened files history, or connected Wi-Fi networks. Data can be explicitly stored encrypted on a persistent storage to stay on the USB stick between system reboots. [45]
  - *Whonix*: Whonix is an operating system consisting of two virtual machines, the Whonix-Workstation and the Whonix-Gateway. The Whonix-Gateway is a gateway to the Tor network and allows only connections to the Tor network. The Whonix-Workstation runs user applications in an isolated network using the Whonix-Gateway as the only network connection to the outside. The operating system is free and open source, with focus on hardened security and privacy. [46]

Chapter 3

Threat Model

This threat model defines the attack surface, identifies potential adversaries, and discusses the resulting threats, whose motivations may include financial gain, surveillance, or censorship. The DNS represents an attractive target due to its lack of inherent security mechanisms and its critical role in accessing essential Internet services, especially because services are moving toward the cloud [2]. A survey [47] conducted among 1,000 organizations around the world in early 2023 found that 90% of them had already experienced one or more DNS-based attacks. The report lists as the most immediate consequences business downtime, loss of productivity, inability to access important data or applications, and regulatory fines. Long-term impacts include damaged brand reputation, loss of customers, decreased market share, and theft of sensitive data.

3.1 Scope of Threat Model

The scope of this threat model encompasses the use of DNS starting from the client's device, which issues conventional, unencrypted DNS requests, to the recursive DNS resolver, which processes these DNS requests by querying the appropriate authoritative name servers.

The model does not include malware present on the client's device. It is assumed that the local network is secured such that only the network operators can observe client traffic, while clients cannot access each other's traffic. Furthermore, it is assumed that there is no malware present on servers of the DNS resolver or the name server system. Nonetheless, operators may act maliciously and use DNS resolvers or name servers to their advantage, e.g., extracting information or manipulating DNS traffic. DNS resolvers may or may not implement existing security enhancements, including DNSSEC, QNAME minimization, or case randomization. Each DNS request issued by a client is considered valid. Therefore, attacks such as phishing, e.g., when a user mistakenly requests a rogue domain resembling a legitimate one, as well as the authenticity of domain ownership or the content of the retrieved resource, are outside the scope of this model.

The focus is on ensuring that a client receives the correct IP address for the requested domain name. Web traffic itself, which may follow network paths different from those of DNS traffic, is not covered in this threat model.

3.2 Attack Surface Scenarios

The attack surface of the DNS is defined by the path of DNS traffic between the client and the authoritative name servers. It can be examined from two main perspectives. One is the exposure of the client's IP address. This enables an observer to correlate queried domain names with a specific client. The other perspective is the application of DNS security enhancements, which is primarily controlled by the DNS resolver.

The communication path between the client and the recursive DNS resolver is not protected by security enhancements, and as there is typically no cache before the recursive DNS resolver, every DNS request of the client passes this section of the path [48]. Consequently, the location of the recursive DNS resolver and the trustworthiness of its operator are critical factors. In transit, DNS requests may pass through multiple Autonomous Systems (ASes) and Internet Exchange Points (IXPs), increasing their exposure.

Three scenarios are distinguished based on the DNS resolver's location:

- A local recursive DNS resolver
- A DNS resolver provided by the Internet Service Provider
- A public DNS resolver located in a different autonomous system than the client

3.2.1 Local Recursive DNS Resolver

In this scenario, the client's stub resolver sends DNS requests to a recursive DNS resolver within the local network, typically operated by the network provider. Multiple clients on this network may share this DNS resolver. The IP address of the local DNS resolver is visible along the entire path to the authoritative name servers, and all DNS requests can be correlated with clients using that DNS resolver. The segment between the client and the DNS resolver, as well as the DNS resolver itself, are under the control of the local network operator, who may choose to implement security enhancements such as QNAME minimization, DNSSEC validation, and case randomization. The operator may also employ caching or apply DNS-based filtering for connected clients. In this scenario, the local network's recursive DNS resolver effectively represents the client from the outside perspective. This scenario is illustrated in Figure 3.1.

3.2.2 DNS Resolver of Internet Service Provider (ISP)

In this scenario, the client's stub resolver sends DNS requests to a recursive DNS resolver operated by the ISP. This is the most common setup for residential users, as the DNS resolver is typically configured automatically via DHCP [4, 48, 49]. ISPs often operate DNS resolvers within the same network as they assign their clients to, minimizing the path length between clients and the DNS resolver [48]. The implementation of a DNS cache or other DNS security enhancements depends on the ISP. DNS requests processed by the ISP's recursive DNS resolver are aggregated with those of potentially many other clients and

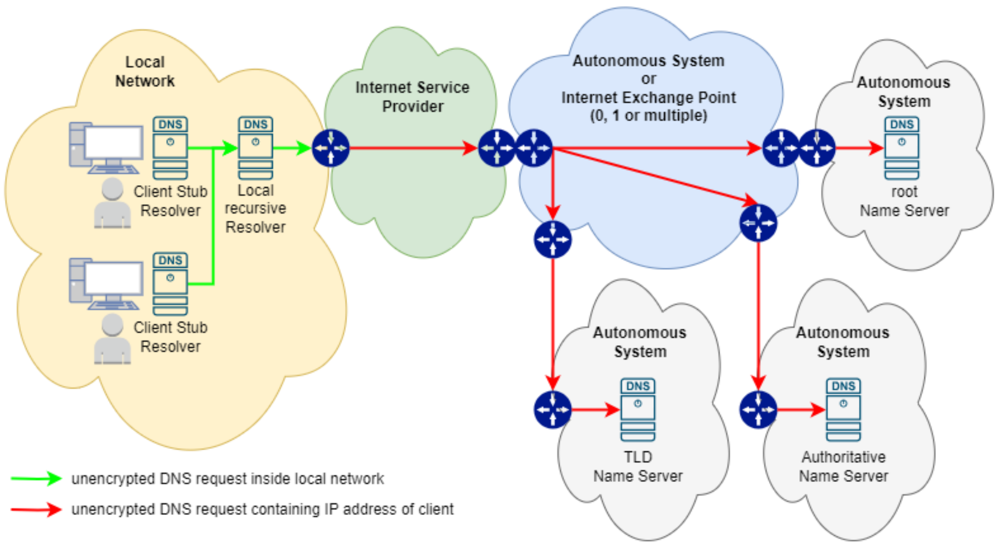
Local Network
Client Stub Resolver
Local recursive Resolver
Client Stub Resolver
Internet Service Provider
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Autonomous System
root Name Server
Autonomous System
TLD Name Server
Autonomous System
Authoritative Name Server
DNS
unencrypted DNS request inside local network
unencrypted DNS request containing IP address of client

Figure 3.1: Scenario of local recursive DNS resolver

no longer contain individual client IP addresses in the DNS request on the path beyond the recursive DNS resolver. Hence these requests can no longer be correlated to a particular client. The degree of indistinguishability depends on how many clients share the ISP's DNS resolver. This scenario is illustrated in Figure 3.2.

3.2.3 Public DNS Resolver

In this scenario, the client's stub resolver sends DNS requests to a public recursive DNS resolver located in a different AS from the client. Such DNS resolvers are often provided by large organizations, such as Google[1] or Cloudflare[2], offering DNS resolution as a service. The client's IP address is exposed across a potentially long path between the client and the public DNS resolver. Large public DNS resolvers process DNS requests from vast numbers of clients, leading to significant centralization of DNS traffic. Within these systems, DNS requests from numerous clients are aggregated, and the DNS resolver operator typically employs a cache and determines which security enhancements are applied.

Public DNS resolvers may also perform DNS-based filtering, either required by law[3] or provided as a service, such as blocking ads. Some providers, such as Control D[4], offer multiple DNS resolvers with different filtering policies. Although residential clients typically use the ISP's DNS resolver, they may switch to public DNS resolvers if the DNS resolver of the ISP experiences an outage and retain the public DNS resolver configuration afterward [4]. ISPs themselves may also distribute public DNS resolvers to clients via DHCP configuration. This scenario is illustrated in Figure 3.3.

[1]https://developers.google.com/speed/public-dns
[2]https://developers.cloudflare.com/1.1.1.1/
[3]https://developers.google.com/speed/public-dns/blocking
[4]https://controld.com/free-dns

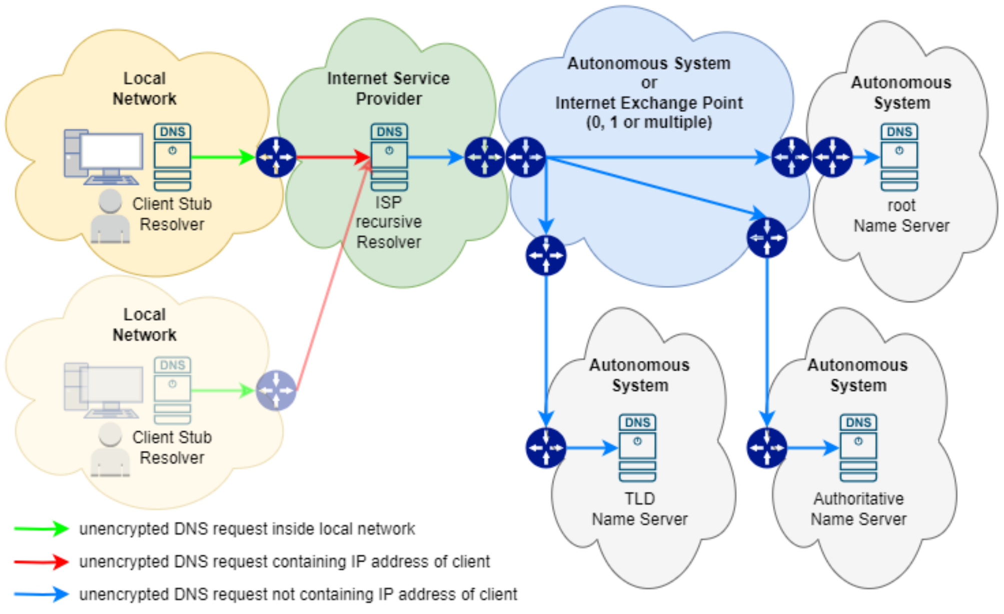
Local Network
DNS
Client Stub Resolver
Internet Service Provider
DNS
ISP recursive Resolver
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Autonomous System
DNS
root Name Server
Local Network
DNS
Client Stub Resolver
Autonomous System
DNS
TLD Name Server
Autonomous System
DNS
Authoritative Name Server
unencrypted DNS request inside local network
unencrypted DNS request containing IP address of client
unencrypted DNS request not containing IP address of client

Figure 3.2: Scenario of ISP's DNS resolver

3.3 Potential Adversaries

Potential adversaries are entities capable of monitoring or manipulating DNS traffic, regardless of whether they actually do so, and irrespective of their intentions, legal status, or obligations. In some cases, such entities may be legally required to monitor DNS traffic and to share collected data with third parties, such as governmental or judicial authorities.

3.3.1 Local Network Operator

The local network operator has visibility into network traffic of connected clients, including connection information such as protocols and destination IP addresses. It knows the internal IP addresses of all clients and may monitor, restrict, or manipulate network traffic, particularly unencrypted DNS traffic, for security purposes. The operator may also apply DNS-based filtering and inform users about the applied filtering policies.

3.3.2 Internet Service Provider (ISP)

> *Internet Service Providers (ISPs) connect end users and businesses to the public Internet. They compete with each other on price, performance, and reliability, but they also must cooperate with each other to provide global connectivity to all other attachments on the Internet.* [50]

ISPs typically provide DNS resolvers to their customers, automatically configured via DHCP [4, 51]. The applied DNS security enhancements and the implementation of a cache are under their control. ISPs might be required by national

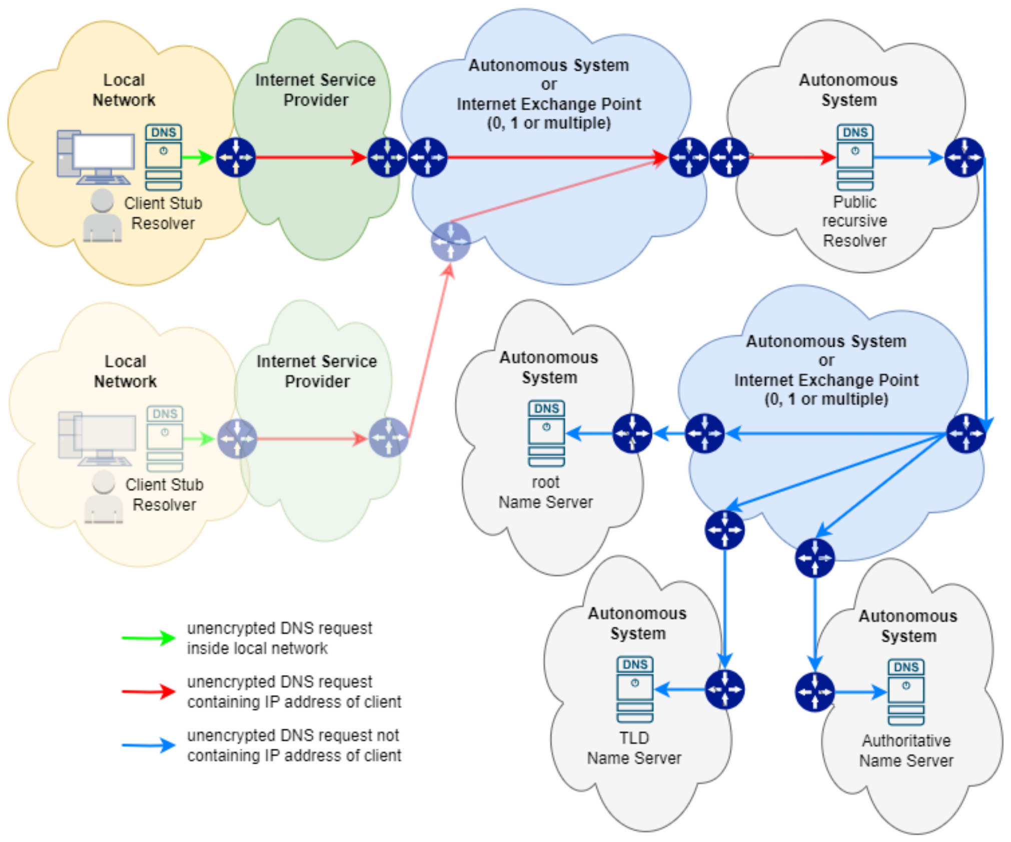
Local Network
Internet Service Provider
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Autonomous System
DNS
Client Stub Resolver
Public recursive Resolver
Local Network
Internet Service Provider
Autonomous System
Autonomous System or Internet Exchange Point (0, 1 or multiple)
root Name Server
Autonomous System
TLD Name Server
Autonomous System
Authoritative Name Server
unencrypted DNS request inside local network
unencrypted DNS request containing IP address of client
unencrypted DNS request not containing IP address of client

Figure 3.3: Scenario of public DNS resolver

laws to block access to specific domain names, particularly in cases involving copyright infringements in Europe [52].

As an entry point for a client's Internet access, the ISP can observe unencrypted DNS traffic and associate DNS requests with individual client IP addresses. Consequently, the ISP must be trusted to a certain degree, as it can monitor, filter, or manipulate DNS requests, regardless of the configured DNS resolver.

3.3.3 Autonomous System (AS) Operator

An Autonomous System specifies a network, mostly an organization that can own or announce network addresses to the Internet. [53]

An AS connects directly to one or more other ASes via Internet Exchange Points [54]. Traffic can originate, terminate, or transit within an AS, depending on the network location of services and clients. The AS operator can potentially monitor or manipulate any traffic within its infrastructure.

3.3.4 Internet Exchange Point (IXP) Operator

An Internet Exchange Point provides a shared interconnection infrastructure that enables multiple ASes to exchange traffic. Without IXPs, interconnections would require dedicated, pairwise links between ASes [54]. They typically operate on network layer 2 and may offer additional services such as traffic statistics and looking glasses [55]. Consequently, IXP operators are capable of monitoring or manipulating all traffic passing through their infrastructure.

3.3.5 Recursive DNS Resolver Operator

Operators of recursive DNS resolvers can view both the client IP address and the contents of DNS requests. They receive all DNS requests as there is typically no cache before them [49].

Recursive DNS resolvers can be operated by ISPs, public service providers, local network operators, or even individual users on their own devices. DNS resolver operators can therefore gain deep insights into client DNS behavior and may monitor, block, or manipulate DNS traffic.

3.3.6 Authoritative Name Server Operator

An authoritative name server is authoritative (i.e. responsible) for one or more DNS zones and manages the corresponding resource records. The amount of information exposed to an authoritative name server depends on whether the recursive DNS resolver applies QNAME minimization. Without minimization, each queried authoritative name server can see full domain names, which can be reduced to the relevant domain name portion when QNAME minimization is applied.

Authoritative name server operators can observe which recursive DNS resolvers query their zones and may keep log files containing these DNS requests. When EDNS client subnet is used, DNS requests may also contain partial client IP address information.

3.4 Threats of the conventional Domain Name System

3.4.1 DNS Traffic Interception

Unencrypted DNS traffic can be identified by filtering for UDP port 53 and TCP port 53. Such traffic exposes sensitive information, including the requested domain names and potentially the IP address of the original sender. As defined in section 3.3, multiple adversaries are capable of intercepting and inspecting DNS traffic.

Intercepting and inspecting DNS traffic is often the basis for attacks, but it can also serve benign purposes and may be necessary to provide security in networks, especially for network operators. According to [56], pervasive monitoring is defined as a

> *widespread (and often covert) surveillance through intrusive gathering of protocol artefacts, including application content, or protocol metadata such as headers.*

The authors classify pervasive monitoring as an attack, but also acknowledge certain beneficial actions that would fit into their definition such as network management functions and anti-spam mechanisms. Similarly, [57] proposes payload analysis to detect DNS tunneling based on features of the payload, such as the size of requests and responses, the entropy of hostnames, and the use of uncommon record types. This approach requires monitoring of DNS requests and responses, as well as other associated indicators. [58] further presents a machine learning algorithm applied on live traffic for detecting DNS tunneling.

However, ASes and especially IXPs present single points for traffic analysis [59]. [60] inspected 148,478 residential and cellular IP addresses across 3,047 ASes and discovered that 259 ASes exhibited DNS interception behavior, particularly for DNS traffic from and to public DNS resolvers. [61] lists countries known to collect metadata or mandate ISP participation in surveillance activities. The study also observes countries taking efforts to prevent Internet traffic from being routed through specific surveillance states, particularly when both the source and destination of a DNS request are located within the same country.

3.4.2 Privacy and Information Exposure

In general, DNS data is considered public, as the idea was originally to distribute DNS data and make it available to everyone. However, DNS transaction data, which includes the client IP address, the queried domain name, and the timestamps of the DNS request and DNS response, may reveal sensitive information. This particularly applies when the transaction data contains the IP address of the original client, i.e. DNS data captured before any external DNS resolver. When combined with time information, sequences of DNS transaction data can show DNS patterns of individual users. Notably, confidentiality was never a design goal of the DNS [49].

Monitoring DNS traffic can disclose which websites or services a client accesses. Aggregating DNS data over time enables the reconstruction of a comprehensive user profile of web activities and habits. When ECS is used, clients can be associated with a subnet of IP addresses, even if the IP address of the original sender is no longer the IP address of the client from the perspective of an observer beyond a recursive or forwarding DNS resolver [62].

Results of tests conducted in section 5.4.4 show that among the two major public DNS resolvers, Google employs ECS while Cloudflare does not. Depending on the subnet size and the number of simultaneous clients within that subnet, it may still be possible to reidentify clients or estimate their geographical location.

[49] differentiates between primary and secondary requests. Primary requests are DNS requests for the domain name a user intentionally wants to visit. Secondary requests are automatically triggered by primary requests without direct involvement of the user to retrieve additional resources, such as embedded content on a website. The combination of a primary request and its secondary requests can be unique to a particular website.

Even if users cannot be personally identified by name, DNS data can be used to construct behavioral profiles that enable reidentification. Such profiles can reveal commercial habits, sexual or political orientations, health conditions, and the geographic location at a given time, and can provide insights into businesses, strategic relationships, and future plans of organizations. Commercial organizations and advertising agencies exploiting this information create a market for buying and selling data of ordinary citizens [2]. ISPs can include the costs of operating a DNS resolver in the costs of their overall service and may not explicitly disclose them to their customers.

Reasons for providing public recursive DNS resolvers might be to promote other services, or altruistic by using other paid services to cover the costs because of beliefs in preserving privacy. Major operators, however, may get their revenue from selling DNS data to third parties [63].

Even passive observation of DNS data can disclose OS information and enhance finding vulnerable systems, services, or devices for subsequent exploitation [3, 64]. The use of specific software can be identified when application names appear within the requested domain names [49].

[65] analyzed DNS data from a large campus network (Xi'an Jiaotong University, China) using a system named DNSMiner, which identifies behavioral fingerprints based on the domain name, the inter-domain relationship, and temporal behavior. Regular activities such as booting a system, starting applications, and visiting specific websites generate sequences of DNS requests that are characteristic for a user. From two weeks of data collected from clients with static IP addresses, DNSMiner successfully reidentified 69.63% of users by their DNS activities in a new DNS data stream with an accuracy of 98.74% and a false positive rate of 1.26%.

As [65] created behavioral fingerprints of clients with static IP addresses, [66] demonstrated that users can be identified even when their IP addresses change. Using unsupervised learning techniques over a 56-day period, during which users' IP addresses changed daily, the authors successfully linked all sessions of the 19% highly active users and nearly all sessions of 73% of users.

The original client's IP address is among the most revealing pieces of DNS information. Due to the structure of DNS, it remains visible up to the first external DNS resolver. DNS data can be correlated with other information sources, such as web traffic, where the same IP address is very likely used.

3.4.3 Centralization

Centralization consolidates the privacy exposure of numerous clients under the control of a single potential adversary. Large public recursive DNS resolvers, their ISPs, and the ASes in which they are hosted represent key points of centralization.

For most customers, configuring an alternative recursive DNS resolver is too complex, leading them to rely on the default DNS resolver provided by their ISP. While many ISPs operate their own recursive DNS resolvers, some forward all DNS queries to one or more public recursive DNS resolvers, or even configure such resolvers directly for their customers. A study found that this practice is employed by 36.2% of small consumer ISPs and 4.0% of large consumer ISPs. [67]

Blocking of DNS requests for specific domain names can also prompt users to switch to public recursive DNS resolver [51].

DNS data can be combined with information from other sources. For example, Alphabet Inc., which operates the Google public DNS resolver in addition to numerous other services, such as a search engine, mail platform, web browser, and cloud infrastructure, has access to multiple, potentially interrelated data streams. Although these services appear to operate independently, the aggregation and correlation of data across them can provide competitive advantages in the market [51, 68].

Trust in single instances is a key component as they gain insights into data and pose technical, economic, and political risks. Centralization on public DNS resolvers introduces a single point of failure where all clients across several ISPs can be affected [51].

A large public recursive DNS resolver experienced an outage on July 14, 2025. Cloudflare's DNS service, also known as 1.1.1.1., experienced a downtime of 62 minutes due to an internal configuration error, impacting the majority of its users. The DNS over HTTPS service of Cloudflare remained operational, as the domain cloudflare-dns.com, used for connecting to the DoH service, resided on a separate set of IP addresses [69].

Conversely, public DNS resolvers can offer faster response times than DNS resolvers provided by the ISP, even though DNS traffic typically traverses more ASes [70]. Another benefit of centralized DNS resolvers is the faster adoption of security enhancements, such as QNAME minimization [71].

3.4.4 Manipulation of DNS Responses

Unencrypted DNS traffic can be easily intercepted and manipulated by adversaries along the communication path, as the original DNS design provides neither integrity protection nor authentication. The likelihood of DNS requests being manipulated depends on the domain name, the transport protocol, and the query type. QNAME minimization and DNSSEC validation provide countermeasures. However, DNSSEC validation is performed only by recursive resolvers, and protection is limited to domains that are properly DNSSEC signed. [72] reports, that only 1% of .com, .net, and .org domains were properly DNSSEC signed in 2017.

[73] notes that the detection of DNS manipulation might result in a high false-positive rate when relying on consistency-based heuristics. Their study identified 55 ASes across 26 countries where DNS manipulation was performed at the ISP-level.

3.4.5 DNS Blocking

DNS blocking is a form of DNS manipulation intended to prevent access to specific online information or services by interfering with the DNS resolution process. This can involve denying the existence of a valid domain name or forging falsified responses.

DNS blocking is typically implemented at the recursive DNS resolver, though any adversary positioned along the DNS path can perform it. While DNS blocking is simple to deploy, it can also be circumvented with minimal effort, for instance by switching to an alternative DNS resolver. [74]

DNS blocking can be detected in several ways. When DNSSEC validation is applied, the domain name resolution fails, indicating that the received DNS response is invalid. On the application layer, users may encounter a warning about an invalid TLS certificate while browsing the web. However, users often fail to associate these warnings with DNS manipulation and may be tempted to proceed or even become accustomed to disable security controls. The Motivation for blocking DNS can be censorship, protecting clients from accessing malicious domain names, controlling access to content within an organization, or legal or political reasons. DNS blocking is often controversial, particularly when implemented without user consent, leaving users unaware of the reason for a website being inaccessible. Moreover, blocking a domain name at the second level can result in result in overblocking when domain names of legitimate resources are subdomains of the blocked domain name. [74]

DNS blocking can also be based on the source IP of the client sending the DNS request. This approach is used when globally operating recursive DNS resolvers are legally required to block specific domain names only within certain jurisdictions or countries. Google, for instance, explicitly states the jurisdictions in which it must comply with such blocking requirements. In affected cases, Google's public DNS resolver responds with RCODE REFUSED, and optionally includes an extended DNS error code 16 (Censored)[5].

Some public DNS resolvers offer blocking domain names associated with malicious or fraudulent resources. [75] identified 17,601 DNS servers offering protective DNS (PDNS) services across 1,473 ASes in 117 countries, representing 9.1% of all probed DNS servers. Despite performing additional filtering steps, these services introduced negligible latency for users. PDNS offers an easy and fast way without requiring changes to the DNS protocol. Users can typically select blocking categories, such as adult, malicious, spam, or phishing, though they cannot modify the applied blocklists, which may be public or proprietary.

According to the ICANN DAAR monthly report published in September 2024 [76], more than 2.4 million domains were classified as security threats, consisting of 2.8% malicious domains, 24.1% phishing domains, 1.6% Botnet C&C domains, and 71.5% spam domains.

3.4.6 Cache Poisoning

Cache poisoning occurs when a DNS resolver's cache stores a manipulated DNS response. The information in the cache is used for all clients, independent of which client triggered the original DNS request, and remains in the cache for the time defined in the TTL value. For a successful attack the domain name must not already be present in the cache, and the attacker must be able to predict or guess certain parameters of the legitimate DNS response. Countermeasures are DNSSEC validation and case randomization. [2, 77]

3.4.7 Cache Snooping

For an attacker, it can be beneficial to know whether a domain name was recently requested from a given DNS resolver. Cache snooping reveals whether

[5]https://developers.google.com/speed/public-dns/blocking

and, in some cases, when a domain name was requested, which in turn can facilitate correlation attacks. To check whether a domain name is present in the cache, an attacker can send two requests. The first request is for the target domain name, followed immediately by the second request, which queries a domain name that is already cached. If the response for the second request is received before the first request, it indicates that the domain name was very likely not cached and authoritative name servers had to be queried [2, 78].

Chapter 4

Evaluation of DNS on the Tor Network

While chapter 3 examined DNS in general, this chapter evaluates and discusses the impact of DNS weaknesses and the resulting threats in the context of DNS resolution performed by Tor exit relays.

4.1 Considered Use Cases

The evaluation adopts the perspective of an exit relay handling DNS requests, i.e., receiving DNS requests from the Tor network, resolving them on the public Internet, and returning the corresponding DNS responses to the requesting Tor clients.

Exit relays are responsible for DNS resolution on their Tor circuits. Various applications and systems utilize the Tor network and depend on exit relays to translate domain names into corresponding IP addresses. All DNS requests received by exit relays are considered valid.

Applications that send TCP DNS requests over Tor directly to DNS resolvers chosen by the client are excluded from this evaluation, as in such cases the exit relay does not perform the DNS resolution. Likewise, non-DNS traffic, such as web traffic, is outside the scope of this analysis. Access to Onion services, which do not rely on conventional DNS, is also excluded.

4.2 Tor's Recommendations for Exit Relay Operators

Although anyone is welcome to operate a Tor exit relay, doing so requires technical and legal considerations. ISPs or hosting providers may prohibit the operation of Tor exit relays within their networks.

The Tor Project provides information for operating Tor nodes[1], as well as an explicit guide for exit relay operators[2].

The latter emphasizes the importance of DNS in ensuring performance and reliability and includes the following recommendations regarding DNS on exit relays:

- To avoid centralization, none of the big DNS resolvers, e.g., Google or Cloudflare, should be used, neither as primary nor as fallback DNS resolver.

[1]https://community.torproject.org/relay/setup/
[2]https://community.torproject.org/relay/setup/exit/

- Run a local, caching, DNSSEC-validating resolver as the primary resolver. Forwarding resolvers should not be used. If a secondary resolver is desired, a resolver provided by the service provider within the same Autonomous System as the exit relay can be chosen. The secondary resolver should only be used as a fallback resolver.
- To limit the exposure of DNS queries at the autonomous system level, no more than two resolvers should be configured.
- The resolver should not use outbound IP addresses used by any Tor nodes. These IP addresses are publicly known and may be blocked.
- Monitoring and optimizing the DNS resolution timeout rate is recommended for operators running exit relays with a bandwidth $\geqslant$ 100 Mbit/s.
- The chosen DNS resolver software should support DNSSEC validation and QNAME minimization.

In summary, Tor recommends operating a local, caching DNS resolver that supports DNSSEC validation and QNAME minimization and, ideally, monitors and optimizes the query timeout error rate.

4.3 Tor DNS Exposure

The exposure of DNS traffic between Tor exit relays and their corresponding DNS resolvers depends primarily on the network location of the DNS resolver, i.e. its IP address and the associated AS. For this evaluation, exit relays are categorized into three scenarios based on the DNS resolver's location:

- *Exit relay itself*: The exit relay operator runs a recursive DNS resolver on the same host as the exit relay, and the DNS resolver uses the exit relay's outbound IP address.
- *DNS resolver in same AS*: The DNS resolver is operated within the same AS as the exit relay but uses a different IP address than the exit relay's outbound IP address.
- *DNS resolver in external AS*: The exit relay sends DNS requests to a DNS resolver located in a different AS.

A key consideration in all three scenarios is whether an observer can classify the source IP address in a DNS request or the destination IP address in a DNS response as belonging to a Tor exit relay. Other relevant factors include the trustworthiness of the DNS resolver operator and the use of DNS security enhancements. Potential adversaries are the same as identified in section 3.3, with the addition of a rogue exit relay.

4.3.1 Exit Relay Itself

In this scenario, the exit relay operator runs a recursive DNS resolver on the same host, and the exit relay uses this resolver for DNS requests received on its connected circuits. Because exit relay IP addresses are publicly known, all DNS traffic sent or received can be classified as Tor DNS traffic. Anyone capable

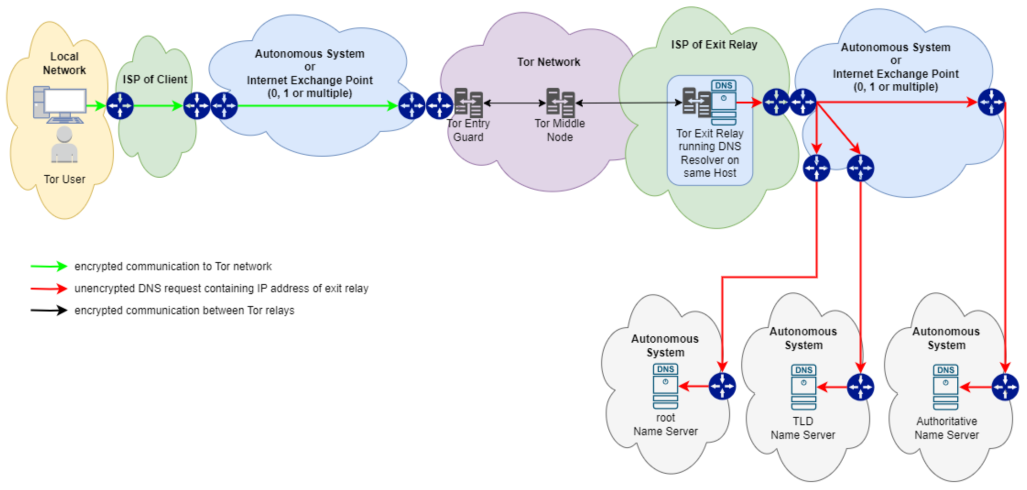
Local Network
Tor User
ISP of Client
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Tor Network
Tor Entry Guard
Tor Middle Node
ISP of Exit Relay
DNS
Tor Exit Relay running DNS Resolver on same Host
Autonomous System or Internet Exchange Point (0, 1 or multiple)
encrypted communication to Tor network
unencrypted DNS request containing IP address of exit relay
encrypted communication between Tor relays
Autonomous System
root Name Server
Autonomous System
TLD Name Server
Autonomous System
Authoritative Name Server

Figure 4.1: Exit relay doing DNS resolution itself

of monitoring the traffic path, from the exit relay to the authoritative name servers, can link this traffic to the specific exit relay.

Since authoritative name servers are likely located in different ASes, DNS traffic may traverse multiple ASes and IXPs. QNAME minimization can reduce the exposure of full domain names to some authoritative name servers but cannot prevent disclosure on the path to the final authoritative name server. The exit relay operator controls QNAME minimization, caching, and DNSSEC validation. Because the exit relay performs the recursive resolution, integrity can be provided for DNSSEC-signed domain names. Figure 4.1 illustrates this scenario.

4.3.2 DNS Resolver in the Same AS

In this scenario, the DNS resolver is located in the same AS as the exit relay but uses a different IP address. It may be operated by the exit relay operator as recommended by Tor, by the ISP of the exit relay, or by a public DNS resolver within the same AS. In all cases, the operator controls the DNS security enhancements applied and knows the IP address of the exit relay.

DNS traffic between the exit relay and the DNS resolver remains within a single AS. The exposure of DNS traffic from the DNS resolver to authoritative name servers depends on whether DNS requests can be associated with the Tor network, e.g., when Tor node operators maintain their own AS, or if the DNS resolver is shared with non-Tor clients, e.g. using the ISP's DNS resolver. Figure 4.2 illustrates this scenario.

4.3.3 DNS Resolver in a Different AS

In this scenario, a DNS resolver located in a different AS than the exit relay is used. Consequently, DNS traffic between the exit relay and the resolver routes to another AS or even traverses one or more intermediary ASes. Along this path,

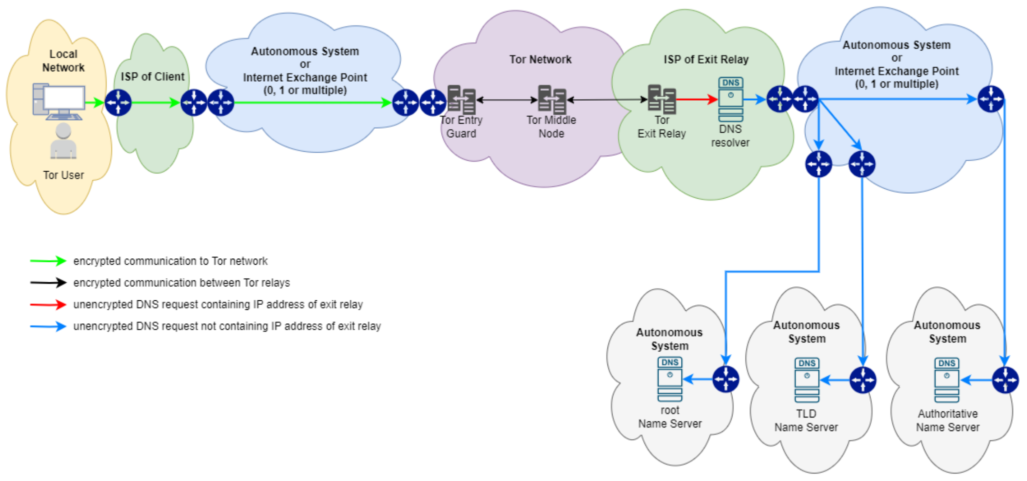
Local Network
Tor User
ISP of Client
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Tor Network
Tor Entry Guard
Tor Middle Node
ISP of Exit Relay
Tor Exit Relay
DNS resolver
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Autonomous System
root Name Server
Autonomous System
TLD Name Server
Autonomous System
Authoritative Name Server
encrypted communication to Tor network
encrypted communication between Tor relays
unencrypted DNS request containing IP address of exit relay
unencrypted DNS request not containing IP address of exit relay

Figure 4.2: Exit relay using a DNS resolver in the same AS

the domain names and the IP address of the Tor exit relay are exposed. Additionally, some degree of trust in the DNS resolver operator is required, as the application of DNS security enhancements is under its control.

At the DNS resolver side, DNS requests from exit relays may be mixed with those from non-Tor clients, and the exit relay's IP address is no longer contained in the DNS traffic between the external DNS resolver and authoritative name servers. Figure 4.3 illustrates this scenario.

4.3.4 Actual Exposure

The actual DNS exposure for a Tor user depends on the selected exit relay, which determines the DNS resolver used. Each exit relay is assigned a probability of being chosen by the circuit creation algorithm based on various factors such as relay flags and bandwidth weights[3].

Although not recommended by Tor, test results in section 4.5.2 show that some exit relays send DNS requests to multiple DNS resolvers to resolve a single DNS request.

4.4 Tor-specific DNS Threats

Based on DNS traffic that can be classified as Tor DNS traffic, or even attributed to a specific exit relay, the following threats arise specifically for the Tor network.

4.4.1 Privacy Exposure of the Tor Network

Tor's primary goal is to protect user privacy. By observing DNS traffic of one or many exit relays conclusions about the interests of Tor users can be drawn. DNS traffic may also be used to facilitate attacks, such as deanonymization.

[3]https://metrics.torproject.org/onionoo.html#details_relay_exit_probability

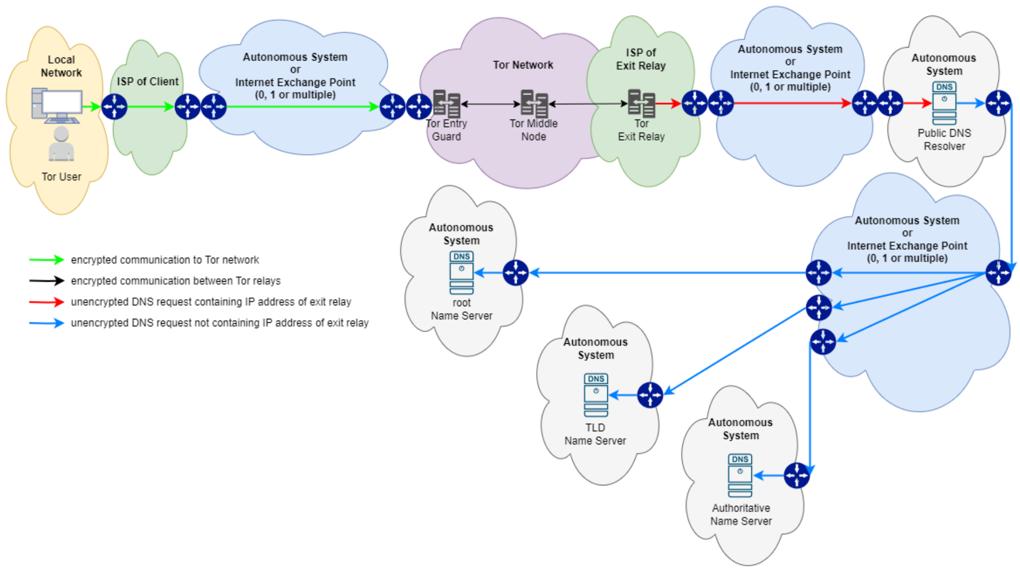
Local Network
Tor User
ISP of Client
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Tor Network
Tor Entry Guard
Tor Middle Node
ISP of Exit Relay
Tor Exit Relay
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Autonomous System
DNS
Public DNS Resolver
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Autonomous System
DNS
root Name Server
Autonomous System
DNS
TLD Name Server
Autonomous System
DNS
Authoritative Name Server
encrypted communication to Tor network
encrypted communication between Tor relays
unencrypted DNS request containing IP address of exit relay
unencrypted DNS request not containing IP address of exit relay

Figure 4.3: Exit relay using a DNS resolver in a different AS

Accessing resources over the Tor network without ensuring that the corresponding DNS requests are also routed through the Tor network exposes requested resources. This is particularly relevant for the .onion pseudo-top-level domain, which is accessible only inside the Tor network.

4.4.2 Centralization

Exit relays determine the DNS resolver used for all clients on the connected circuits. Therefore, exit relays that are more likely to be chosen by the circuit creation algorithm have a higher impact on centralization on DNS resolvers, assuming similar per-user DNS request rates. Centralization increases when multiple frequently chosen exit relays select the same DNS resolver. In addition to the DNS resolver operator, the DNS resolver's ISP and its AS operators can observe DNS traffic.

4.4.3 Tor-specific Blocking and Manipulation

DNS requests coming from the Tor network can be blocked or modified by any actor able to observe and intercept traffic between an exit relay and its DNS resolver, and by the DNS resolver operator itself. Blocking or manipulation may be applied to all Tor-related DNS requests for all domain names or only to specific domain names, resulting in censorship for Tor users.

4.4.4 Attacks on Exit Relay Cache

Exit relays typically operate a caching stub DNS resolver. [78] demonstrates a timeless timing attack[4] where sending a single TLS record can indicate whether

[4]Confidential: TROVE-2021-009: Improved DNS cache oracle, 08.09.2022, https://gitlab.torproject.org/tpo/core/tor/-/issues/40674

a domain name is currently cached on the exit relay. By repeating this attack, the time when the domain name was inserted into the cache can be determined. The timeless timing attack provides insights into the requested resources of an exit relay and facilitates perfectly reliable timing attacks. Their proposed short-term solution of implementing a clipping and randomization of the TTL value of a cached domain name to a random number between 60 and 540 seconds for an original TTL value lower than 300 seconds, and to a random number between 3,360 and 3,840 seconds for an original TTL value greater than or equal to 300 seconds was implemented in Tor version 0.4.5.15[5].

However, testing exit relays for this thesis revealed that the TTL value actually applied is disclosed to Tor clients. The TTL value returned in DNS responses for the corresponding domain name shows the randomized start value for the duration of the cached entry. When the entry expires and is stored again into the cache, the TTL value is clipped and randomized again. Therefore, the TTL value changes with probability of 0.9979, i.e., the probability of not randomly selecting the same value again. By monitoring TTL value changes for a specific domain name, a client can still infer cache presence and age. The attack is available to any Tor client connected to the same exit relay. The issue was reported to the Tor Project, approved, and permission to disclose it in this thesis was granted.

Exit relays also apply case randomization of domain names by their default configuration to reduce the probability of successful spoofing attacks, which would affect all Tor clients of a compromised exit relay.

4.4.5 Correlation Attacks

Tor's low latency exposes it to correlation attacks that link and correlate traffic entering and exiting the network. [79] finds that DNS traffic provides an additional attack surface because it often goes through a different path of ASes as the corresponding TCP connections. The authors describe DefecTor, an attack where observing DNS requests and responses of an exit relay enhances existing correlation and website fingerprinting attacks on Tor users. Relevant adversaries include the client's ISP, intelligence agencies, or malicious entry guards on the ingress side; and actors monitoring the path between the exit relay and the DNS resolver, and the DNS resolver itself on the egress side. Figure 4.4 illustrates the DefecTor attack.

4.5 Evaluation of the State of DNS on Tor

4.5.1 Test Setup

The test setup consists of three main components:

- *Evaluation Server*: This server receives information needed for testing, assigns specific test cases to the test servers, hosts a database for storing test results, and processes log files.

[5]Changelog of Tor version 0.4.5.15, 16.12.2022, https://gitlab.torproject.org/tpo/core/tor/-/blob/tor-0.4.5.15/ChangeLog lines 1, 26-28

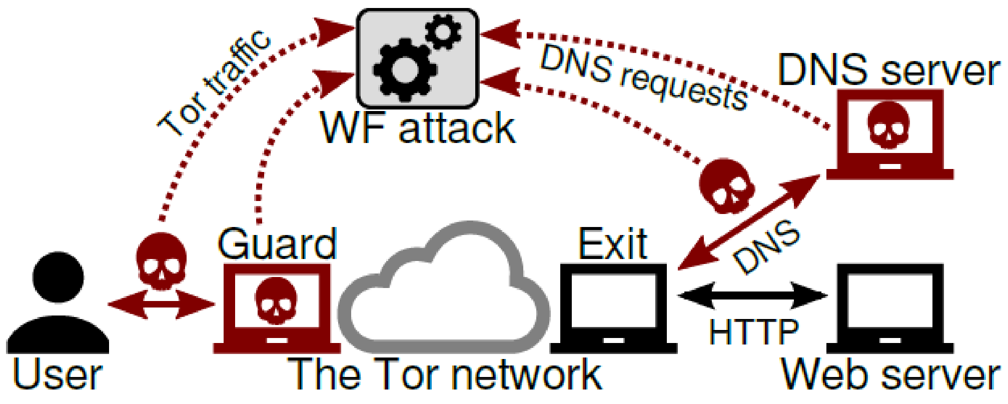

Figure 4.4: DefecTor attack [79]

- *Two Test Servers*: These servers are responsible for issuing DNS requests to exit relays.
- *Two Authoritative Name Servers*: These servers are authoritative for the domains tordns.ovh and dnssec-check.ovh, which are used throughout the testing procedures.

All servers are virtual servers rented from a hosting provider and configured with dedicated IPv4 addresses. They operate within the AS of Hetzner Online GmbH, AS number 24940, AS name *Hetzner Online GmbH*. Figure 4.5 illustrates an overview of the test setup.

Evaluation Server

The evaluation server retrieves the information required for testing exit relays on a daily basis from the Tor API[6], including each relay's fingerprint, IP address, and probability of being selected by the circuit creation algorithm. It stores this data for all testable exit relays into the database. An exit relay is considered testable if the running flag is true, the exit IP address is not empty, the fingerprint has a length of 40 characters, and the exit probability is greater than 0. This subset defines the exit relays to be tested for the day. The evaluation server then assigns the testable exit relays randomly to the two test servers.

Once per day, the evaluation server also downloads files containing AS information such as AS numbers and the associated IP address ranges and AS names using pyasn[7].

At the end of each day, the evaluation server receives the log files from the authoritative name servers, identifies all test-related DNS requests, and inserts the corresponding entries into the database.

Test Servers

Both test servers are identical in terms of operating system, hardware resources, and running software. They are hosted in the same data center and

[6]https://metrics.torproject.org/onionoo.html
[7]https://github.com/hadiasghari/pyasn

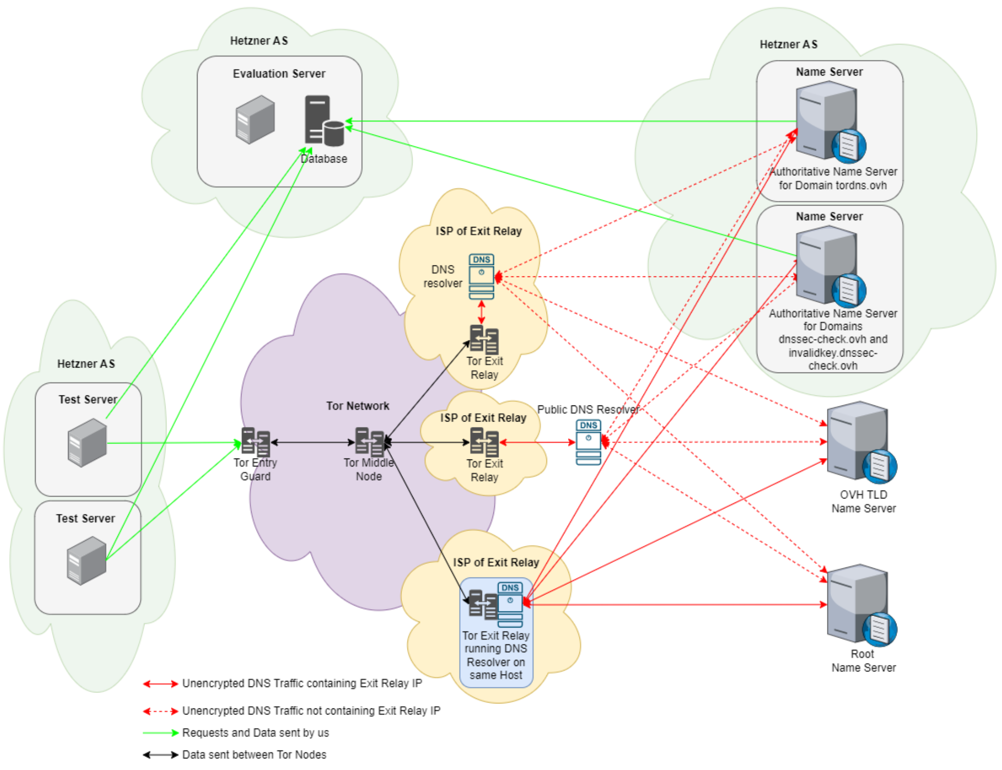
Hetzner AS
Evaluation Server
Database
Hetzner AS
Name Server
Authoritative Name Server for Domain tordns.ovh
Name Server
Authoritative Name Server for Domains dnssec-check.ovh and invalidkey.dnssec-check.ovh
ISP of Exit Relay
DNS resolver
Tor Exit Relay
Hetzner AS
Test Server
Test Server
Tor Network
Tor Entry Guard
Tor Middle Node
ISP of Exit Relay
Tor Exit Relay
Public DNS Resolver
OVH TLD Name Server
ISP of Exit Relay
Tor Exit Relay running DNS Resolver on same Host
Root Name Server
Unencrypted DNS Traffic containing Exit Relay IP
Unencrypted DNS Traffic not containing Exit Relay IP
Requests and Data sent by us
Data sent between Tor Nodes

Figure 4.5: Server and network overview of test setup

operate within the same AS. However, it cannot be guaranteed that network bandwidth and server performance remain identical at all times.

Each day, every test server schedules its tests according to the exit relays assigned to it. For each test run, the server starts a container and provides it with the necessary parameters, including information about the designated exit relay and test configuration. Within the container, a test script connects to the specified exit relay and verifies that the connection has been successfully established. The script then executes a series of tests by sending DNS requests to the exit relay for various test purposes. After issuing the DNS requests, the connection to the exit relay is checked again to ensure that it remained active. The test output generated within the container is stored on the test server outside the container. Once the container completes its execution, the test server processes the output and stores the results into the database. If a test fails and sufficient time remains in the daily schedule, it is rescheduled to run again at the end of the current test schedule. Figure 4.6 provides an overview of the test server architecture.

Authoritative Name Server

Both authoritative name servers are identical in terms of operating system, hardware resources, and running software. Logging is enabled and each DNS request received is recorded in a log file. Each authoritative name server is responsible for a domain.

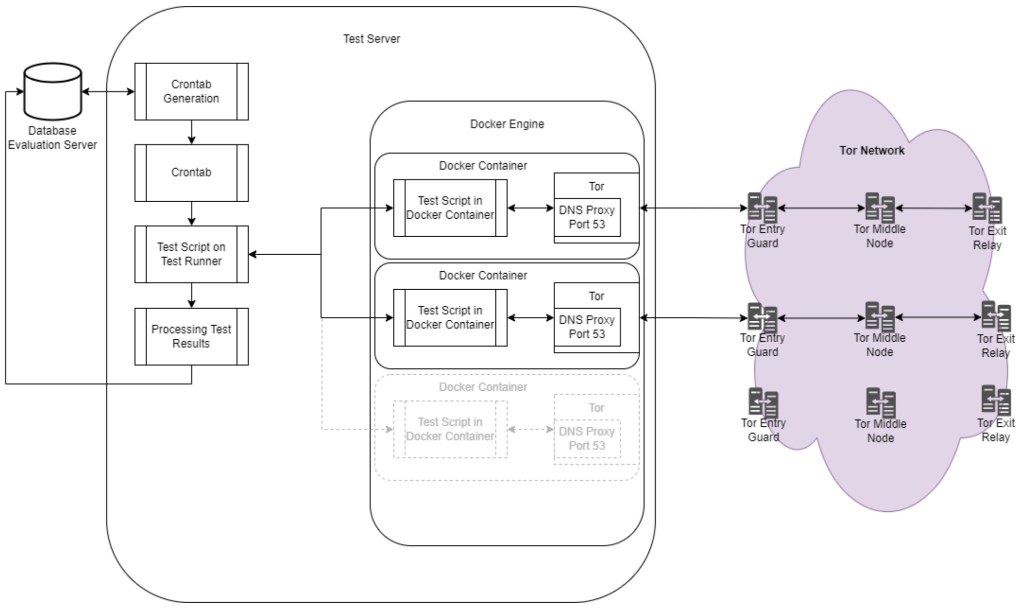

Figure 4.6: Test server structure

The first server is authoritative for the domain tordns.ovh. The domain is not DNSSEC-signed and includes the wildcard record *.tordns.ovh to respond to queries for all subdomains. The TTL value is set to the maximum allowed duration of 604,800 seconds, i.e. 7 days.

The second server is authoritative for the domain name dnssec-check.ovh. The domain zone includes wildcard records for *.dnssec-check.ovh and *.invalidkey.dnssec-check.ovh, where the former is correctly DNSSEC-signed and the latter is intentionally signed with an invalid key. The TTL value is set to the maximum allowed duration of 604,800 seconds, i.e. 7 days.

At the end of each day, the log files containing all received DNS requests are transmitted to the evaluation server for processing and analysis.

General Assumptions, Conditions, and Limitations of Tests

As shown in Figure 4.5, tests are conducted from outside the Tor network. No Tor nodes were operated as part of this evaluation.

Information about available exit relays is obtained by the Tor API, which is queried once a day. Consequently, any changes occurring during the day, such as the appearance of new exit relays, the removal of existing exit relays, or changes of exit relay IP address, are not reflected in the tests the same day.

Each DNS request sent to the authoritative name server includes both a test ID and an exit relay ID. The test ID uniquely identifies a single test run. Multiple test runs can occur per exit relay and day if a previous test attempt fails, resulting in distinct test IDs. The exit relay ID identifies a specific exit relay on a given day. Each tested exit relay is assigned such an internal ID daily, allowing both successful and failed tests to be associated with the corresponding exit relay for that day. Exit relays can also be identified across multiple days by their unique Tor exit relay fingerprint, which is not included in the domain name but can be linked through the database.

The default Tor circuit creation algorithm was used for establishing connections to exit relays, with only the target exit relay explicitly defined. Performance therefore depends not only on the exit relay itself but also on the entry guard and the middle node chosen for the circuit. Since no data about prior Tor connections is retained, entry guards do not persist for new connections.

A test was considered successful when the Tor circuit was established successfully, a DNS response is received, and the container is connected to the specified exit relay at the beginning and the end of the test.

The exit probability provided by the Tor API represents an approximation of the probability that a specific exit relay will be chosen. Some results are presented as absolute numbers, while others are weighted by this exit probability. For probability-weighted results, it is assumed that the number of DNS requests an exit relay receives is proportional to its exit probability.

4.5.2 Tests and Results

Tests were conducted during the period from July 3, 2024, to February 28, 2025. Due to a failure of the evaluation server on August 22, 2024, data collected between August 15, 2024, and August 24, 2024, was lost. In the result charts, this period is indicated with a dark gray background.

Apart from the general overview of successful and failed tests, all analyses considered only successful tests.

Test: Sending DNS Requests to Exit Relays

Specifically drafted DNS test requests were sent to each available exit relay during the test period to allow analysis of the corresponding DNS requests received by the authoritative name server.

Test Period: July 3, 2024 - August 14, 2024, and August 25, 2024 – February 28, 2025

Test Description: For each exit relay, three DNS requests were sent querying a subdomain of the controlled domain tordns.ovh in the form of [subdomain2].[subdomain1].tordns.ovh, where [subdomain1] and [subdomain2] contained the test ID and the internal exit relay ID, and therefore were unique per exit relay and per test. This prevented caching and enabled identifying the tested exit relay. For issuing the DNS requests the dig[8] command was employed. The exit relay performed the DNS resolution, and the command output returned the corresponding DNS response. All DNS requests received by the authoritative name server were logged, enabling the identification of exit relays and the IP addresses of the DNS resolvers that queried the specific domain names.

Assumptions and Limitations: For every test, three unique DNS requests were sent to increase the likelihood of detecting all DNS resolvers involved. A test was considered successful even if not all three DNS requests were resolved.

[8]https://manpages.ubuntu.com/manpages/jammy/man1/dig.1.html

Only DNS resolvers querying the authoritative name server could be detected. However, there may have been one or multiple forwarding DNS resolvers between the exit relay and the DNS resolver which could not be detected by this test.

Result: Tested Exit Relays

This section provides an overview of the number of exit relays tested during the test period, along with the number of successful and failed tests.

Analysis Period: July 3, 2024 - August 14, 2024, and August 25, 2024 – February 28, 2025

Results: During the test period, 4,074 exit relays were tested. Of these, 3,975 exit relays achieved at least one successful test, while 99 exit relays never completed a successful test. Table 4.1 lists the minimum, median, and maximum number of exit relays tested per day, as well as their success rates.

Figure 4.7 illustrates the numbers of tested exit relays and their success rates, showing a slight upward trend in both the number of available and successfully tested exit relays toward the end of the test period.

Result: Received Requests at the Authoritative Name Server

The logged DNS requests received at the authoritative name server were assigned to the corresponding test DNS requests. DNS requests that could not be assigned to a test were not further processed. The number, timing, and other characteristics of the recorded data were then analyzed.

Analysis Period: The period for sending test requests was from July 3, 2024, to August 14, 2024 and from August 25, 2024, to February 28, 2025. To include requests received with delays - minutes, days, or even weeks after sending the test DNS request, especially those sent near the end of the test period - DNS requests received on the name server were registered with a three-week extension. The authoritative name servers continued operating during the evaluation server outage, and all requests received in this period were also registered. This resulted in an analysis period from July 3, 2024, to March 21, 2025.

Table 4.1: Number of tested exit relays and their test success

	All Tests	**Successful Tests**	**Failed Tests**
Minimum number of exit relays per day	2,090	2,082	0
Median number of exit relays per day	2,236	2,227	7
Maximum number of exit relays per day	2,471	2,465	126

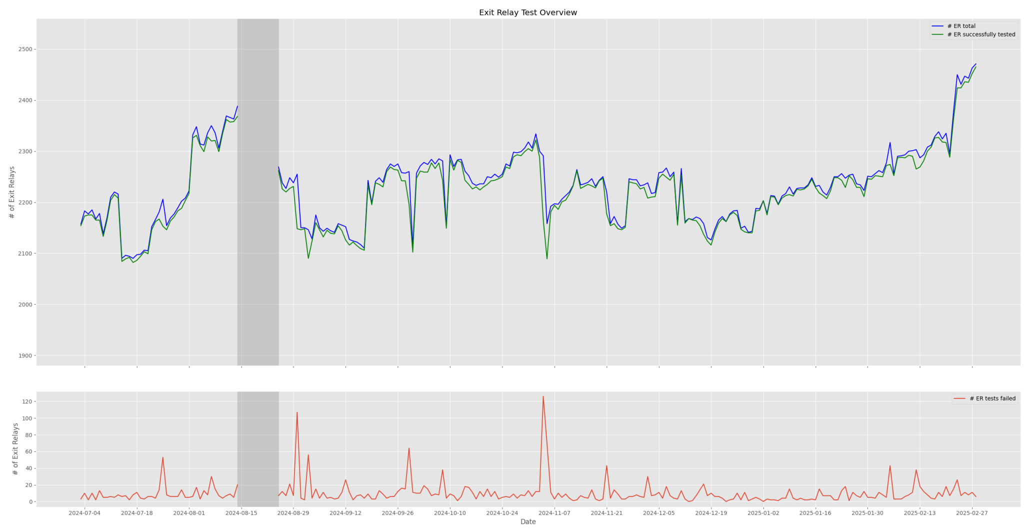

Figure 4.7: Number of tested exit relays per day

Query Types: The name server received requests for various query types. Table 4.2 shows the query types and the number of exit relays that requested them.

Number of DNS Requests received per Test DNS Request: Multiple DNS requests could be received on the authoritative name server for a single test DNS request sent to an exit relay. Depending on the DNS resolver and its configuration, DNS requests for different query types, QNAME minimization, and DNSSEC validation could result in multiple DNS requests.

Identifiable received DNS requests were assigned to the corresponding test DNS request. This included query names under the domain tordns.ovh that did not contain the test ID but were closely related in time and originated from the same IP address or AS. For measuring the number of received DNS requests, every identifiable and assignable DNS request from a successful test was counted.

Received DNS requests were grouped per test DNS request, per AS of the requesting DNS resolver, and per IP address of the DNS resolver.

Table 4.3 and Table 4.4 show the number of DNS requests received at the name server resulting from one test DNS request.

Time of received Requests: The time difference between sending a test DNS request and receiving it on the name server was measured. The applied timeout of the dig command was five seconds. Therefore, for immediate DNS resolution, no DNS requests were expected after six seconds – five seconds for the timeout plus one second for network delay and command completion. This is reflected by a notable drop in the number of requests after six seconds. Consequently, DNS requests received later than six seconds are defined as *late requests* in this thesis.

The latest DNS request was received on March 21, 2025, for a test DNS request sent on July 09, 2024. The corresponding DNS resolver was still querying the authoritative name server for that test domain after 255 days. As the analysis ended on March 21, 2025, it is likely that the authoritative name server continued receiving DNS requests for that test domain even after this date.

Figure 4.8 shows the number of DNS resolvers with different IP addresses

Table 4.2: Query types received by number of exit relays

Query Type	Number of Exit Relays
A	3,975
AAAA	3,049
DNSKEY	1,278
NS	680
MX	536
SOA	121
DS	88
TXT	33
CNAME	26
PTR	20
DNAME	4
HTTPS	4
SRV	2
SVCB	2

sending late DNS requests to the authoritative name server per day. On September 10, 2024, an unusually high number of 15,933 late DNS requests were received.

Table 4.5 lists the exit relay ASes and the corresponding DNS resolver ASes sending late requests on September 10, 2024.

Some DNS requests received from DNS resolvers in the Google AS included ECS data. The corresponding ASes of the ECS network addresses belong to Google, Inc. (AS *GOOGLE-CLOUD-PLATFORM, US*), and to Censys, Inc. (ASes *CENSYS-ARIN-01*, *CENSYS-ARIN-02*, and *CENSYS-ARIN-03*). Censys is a U.S.-based company that aims to make the Internet safer and offers its services primarily to practitioners and researchers[9]. They may have monitored the original

[9]https://censys.com/about-censys

Table 4.3: Number of different IP addresses and corresponding number of requests

Number of IP Addresses	Number of received DNS Requests
1	1,213,310
2	301,018
3	11,138
4	3,624
5	405
6	104
7	43
...	...
234	1

Table 4.4: Number of different ASes and corresponding number of requests

Number of ASes	Number of received DNS Requests
1	1,413,747
2	114,627
3	5,696
4	349
5	12
...	...
17	1

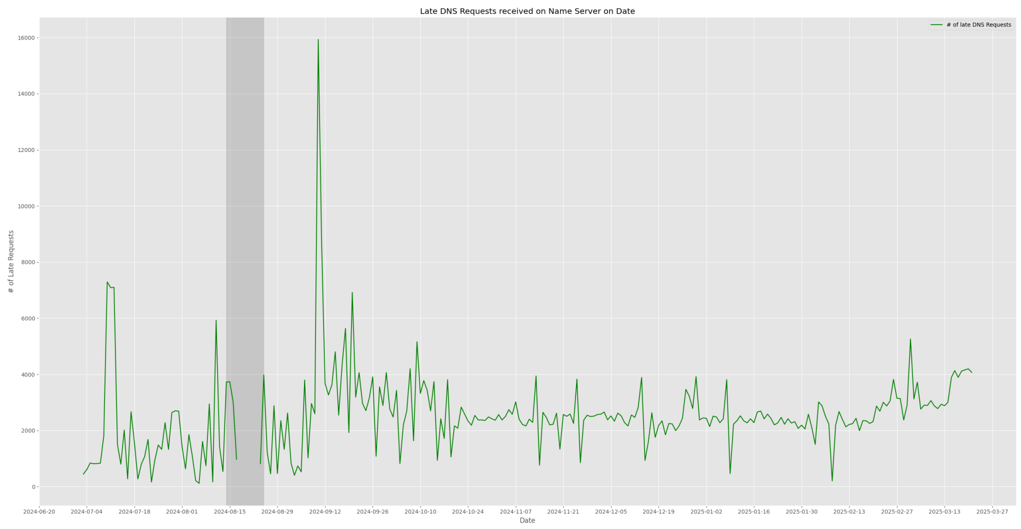

Figure 4.8: Number of received late DNS requests per day

DNS requests and included them in its Internet scanning activities, or exit relay operators may have intentionally used their services.

Result: DNS Resolver Scenarios

This analysis categorized the exit relays into the three scenarios defined in section 4.3, based on the IP addresses of the DNS resolvers.

Assumptions and Limitations: For the scenario DNS resolver in the same AS, it could not be determined who operated the DNS resolver. It may have been operated by the exit relay operator itself, the ISP, or another third party within the same AS as the exit relay.

The analysis considered all three test DNS requests per tested exit relay per day, including late requests.

Results: Figure 4.9 shows the number and probability of exit relays using DNS resolvers across the defined scenarios.

The numbers of exit relays and their probabilities correlated to a high extent.

A single exit relay could use multiple resolvers from different scenarios, even for a single test DNS request.

Table 4.5: AS information of exit relays and DNS resolvers, and the corresponding number of late requests

Exit Relay AS	Resolver AS	Number of Late Requests
AS 210558, 1337-SERVICES-GMBH-NETWORK, DE	AS 15169, GOOGLE, US	4,543
	AS 13335, CLOUDFLARENET, US	3,463
AS 53667, PONYNET, US	AS 15169, GOOGLE, US	2,709
	AS 13335, CLOUDFLARENET, US	2,299

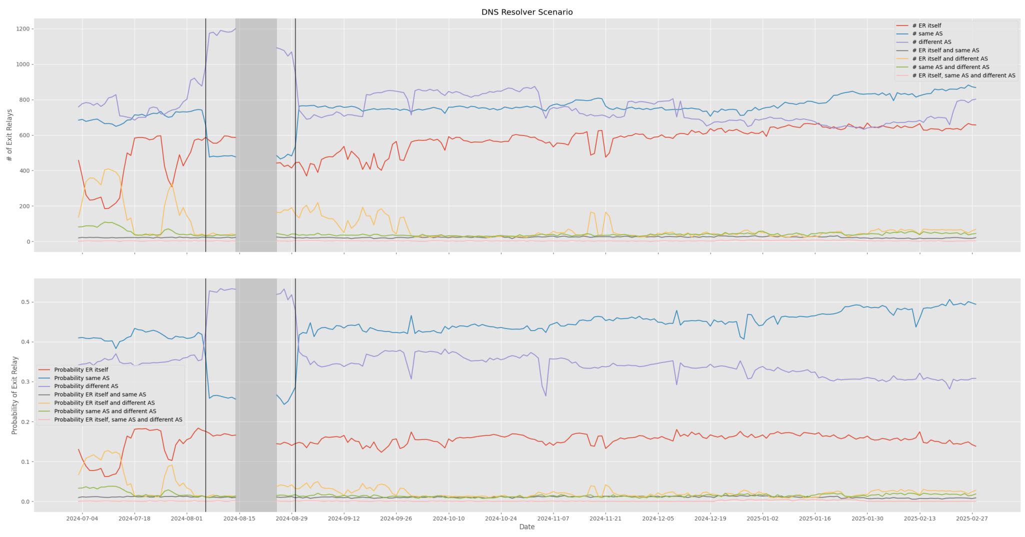

Figure 4.9: Number and probability of used scenarios by exit relays

Between 40% and 50% of exit relays used a DNS resolver located within the same AS, followed by 30% to 40% using a DNS resolver in a different AS. Approximately 15% of exit relays operated a DNS resolver using the exit relay's IP address. Table 4.6 shows the number of exit relays that always used the same scenario throughout the test period, as well as the number of exit relays that had used a given scenario at least once.

Out of 3,975 tested exit relays, 2,914 consistently used the same DNS resolver scenario, while 1,061 exit relays used more than one DNS resolver scenario during the test period.

A noticeable change in the use of the scenarios occurred between August 6, 2024, and August 30, 2024, as indicated by the vertical lines in Figure 4.9. During this period, 259 of 273 exit relays from AS 1101 switched from using a DNS resolver located within the same AS to a DNS resolver in AS 12876 (left black vertical line). On August 30, 2024, they switched back to a DNS resolver within their AS (right black vertical line). This behavior may indicate the use of a backup DNS resolver during an outage or scheduled maintenance of their primary DNS resolver. 14 exit relays from AS 1101 that performed DNS resolution using a DNS resolver with the same IP address as the exit relay were not affected.

Table 4.6: DNS resolver scenarios by exit relays

Itself	Same AS	Different AS	Number of Exit Relays only in this Scenario	Number of Exit Relays seen in this Scenario
1	0	0	426	1,013
0	1	0	797	1,285
0	0	1	1,666	2,185
1	1	0	3	69
0	1	1	3	210
1	0	1	19	483
1	1	1	0	13

Result: ASes of Resolvers

DNS requests received on the name server were categorized according to the ASes of the DNS resolvers.

Limitations: Not all DNS requests coming from an AS that operates a public DNS resolver necessarily came from that specific DNS resolver. For example, in the Cloudflare AS, independent DNS resolvers may be operated by other entities.

Results: During the analysis period, DNS requests were received from 383 different ASes on the authoritative name server. Figure 4.11 shows the seven most frequently used ASes, ranked by the number of exit relays that used a DNS resolver within each AS. Because a single exit relay could use multiple DNS resolvers across different ASes for a single test DNS request, it could be counted in multiple AS categories. The high rank of AS Online SAS can primarily be attributed to the temporary switch of exit relays from AS IP-EEND BV in August 2024.

Figure 4.10 presents ASes that hosted DNS resolvers used by more than 20 exit relays.

Table 4.7 lists the Top 7 ASes along with the probability of using an exit relay with a DNS resolver in that AS, and the number of exit relays within the defined scenarios.

DNS resolvers in the Google AS were used by 1,867 of the 3,975 exit relays, meaning that 46.97% of exit relays generated at least one DNS request received on the authoritative name server from a DNS resolver within the Google AS. The Cloudflare AS was involved for 1,295 of the 3,975 exit relays, corresponding to 32.58%.

Neither Google nor Cloudflare permit the operation of Tor exit relays within their ASes. Nevertheless, as shown in table 10, one exit relay successfully operated within the Google AS, completing a single successful test July 23, 2024.

The probability was calculated by accumulating the success probabilities of all exit relays per test and dividing the total by the number of test days. It reflects the likelihood that a random DNS request sent to an exit relay during the test

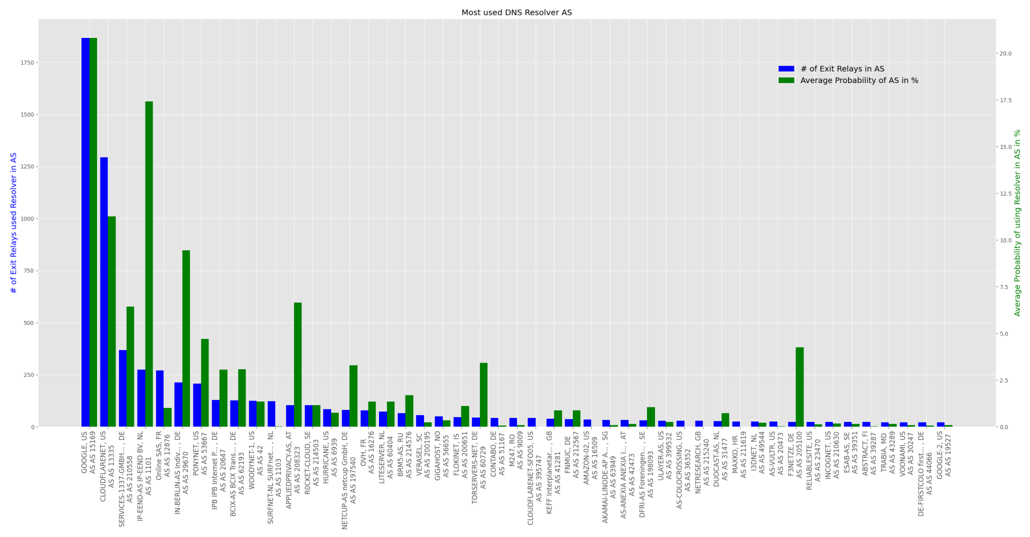

Figure 4.10: ASes used by DNS resolvers of more than 20 exit relays

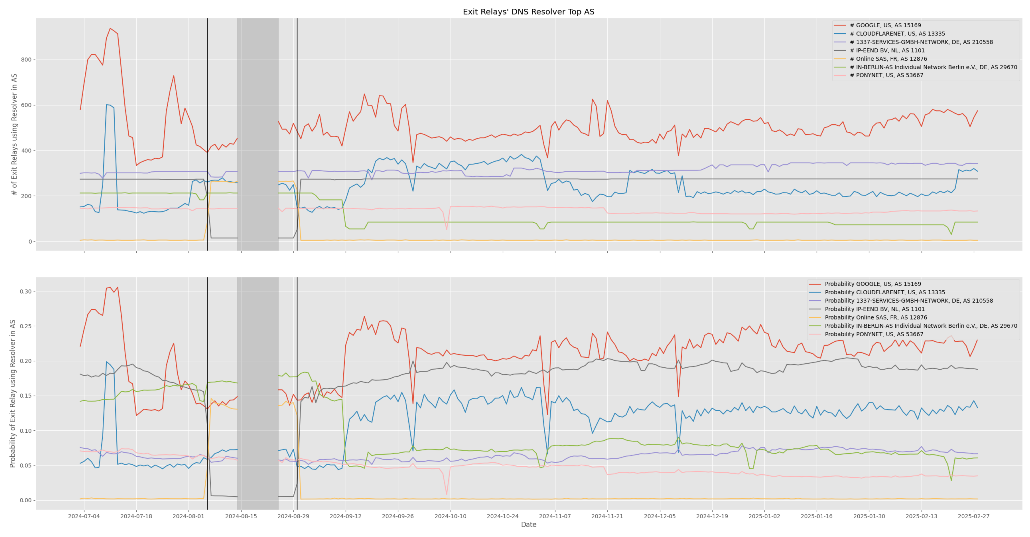

Figure 4.11: Top 7 ASes of DNS resolvers

period was resolved by a DNS resolver within the corresponding AS. Since a single test DNS request could trigger multiple DNS requests from different ASes, the sum of probabilities across all ASes may exceed one.

Result: QNAME Minimization

This analysis evaluated whether, and to what extent, the DNS resolvers used by exit relays applied QNAME minimization.

Analysis Period: July 3, 2024 - August 14, 2024, and August 25, 2024 – February 28, 2025

Analysis Description: The test DNS requests followed the pattern [subdomain2].[subdomain1].tordns.ovh, where both [subdomain1] and [subdomain2] were unique per exit relay and test. This ensured that a do-

Table 4.7: Top 7 DNS resolver ASes used by exit relays

Resolver AS	Number of Exit Relays	Average Probability	Exit Relays resolving itself	Exit Relays Resolver in same AS	Exit Relays Resolver in different AS
AS 15169, GOOGLE, US	1,867	20.81%	0	1	1,866
AS 13335, CLOUDFLARENET, US	1,295	11.27%	0	0	1,295
AS 1101, IP-EEND-AS IP-EEND BV, NL	274	17.41%	14	260	0
AS 29670, IN-BERLIN-AS Individual Network Berlin e.V., DE	214	9.46%	0	0	214
AS 210558, SERVICES-1337-GMBH 1337-SERVICES-GMBH-NETWORK, DE	369	6.43%	369	17	7
AS 12876, Online SAS, FR	270	1.02%	3	5	265
AS 53667, PONYNET, US	207	4.70%	110	100	3

main name containing one of these subdomains was identifiable and not cached.

It was analyzed whether a DNS request for [subdomain1].tordns.ovh of any query type was received on the authoritative name server before the DNS request for [subdomain2].[subdomain1].tordns.ovh of query type A, within a close time interval and coming from the same AS. This approach accounts for DNS resolver operators that distribute outgoing DNS requests across multiple servers using different IP addresses within the same AS.

Assumptions and Limitations: QNAME minimization allows reducing the number of DNS requests required to resolve a single domain name by limiting how many iterations occur, with each iteration appending only one additional subdomain level. A commonly recommended limit is four, meaning that for the first four domain levels, each subdomain should be queried sequentially, e.g. ovh, tordns.ovh, subdomain1.tordns.ovh, and subdomain2.subdomain1.tordns.ovh [24]. This configuration fully covers the structure of the test domain names used in this analysis. Therefore, DNS resolvers applying QNAME minimization according to the recommendation should generate requests for each of these domain levels. It was assumed that DNS resolvers applied no stricter limit than this recommendation.

Results: A large portion of exit relays, corresponding to approximately 80% of the total exit relay selection probability, used DNS resolvers that applied QNAME minimization. Figure 4.12 illustrates these results.

Result: EDNS Client Subnet (ECS) Information

This analysis evaluated the presence and characteristics of EDNS Client Subnet information observed in DNS requests received by the authoritative name

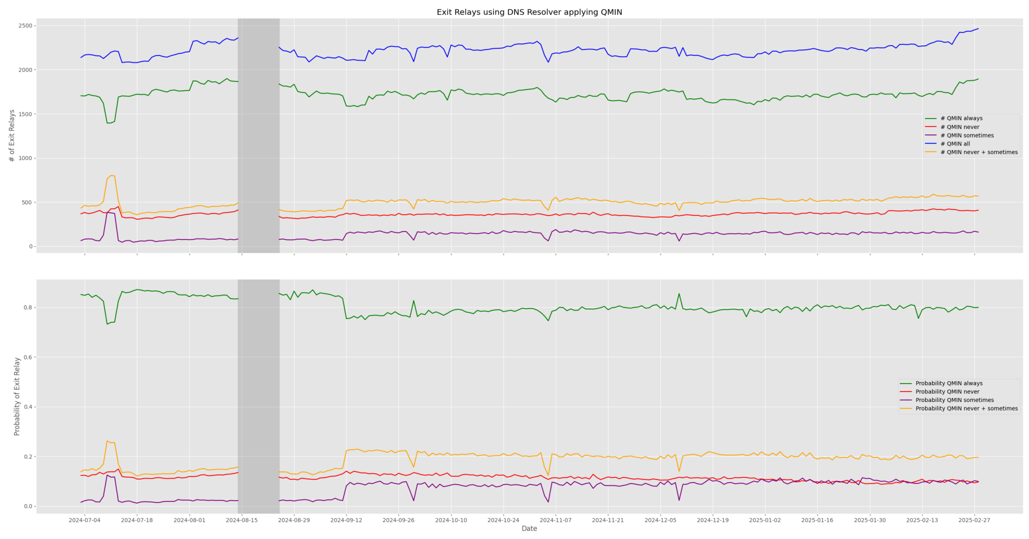

Figure 4.12: Number and probability of exit relays applying QNAME minimization

server, revealing which DNS resolvers included ECS data.

Analysis Period: July 3, 2024 – March 21, 2025

Assumptions and Limitations: A Client may omit ECS information, intentionally include ECS information, or request that the DNS resolver not forward it. Ultimately, the DNS resolver decides whether and how to include ECS information. If multiple DNS resolvers are involved in the resolution path, each can add, modify, or remove ECS information. The last DNS resolver in the chain determines the ECS information that reaches the authoritative name server. Consequently, it cannot be determined whether the client or any intermediate DNS resolver altered the ECS information.

Results: From a total of 6,161,463 DNS requests received by the name server, 814,421 DNS requests (13.22%) contained ECS information. Table 4.8 lists the ECS subnet value observed, along with the number of requests, the number of exit relays associated with these requests, and the number of distinct ASes from which they originated.

Of the 383 ASes that sent DNS requests, 358 never included ECS information, 23 occasionally included ECS data, and two consistently included it.

Google's public recursive DNS resolver, by default, includes ECS information from its clients. All DNS requests coming from the Google AS that contained ECS data could be attributed to Google's public DNS resolver, based on information containing IP addresses used by Google to query authoritative name servers[10].

In contrast, Cloudflare's public recursive DNS resolver never includes ECS information[11]. The DNS requests coming from the Cloudflare AS that included ECS information were therefore likely sent by other DNS resolvers operated within the Cloudflare AS but not by Cloudflare itself. However, this could not be verified as Cloudflare does not publish the IP addresses of its DNS resolvers used to query authoritative name servers.

[10]https://developers.google.com/speed/public-dns/faq
[11]https://developers.cloudflare.com/1.1.1.1/faq/

Table 4.8: ECS information received on the authoritative name server

ECS Information	Number of Exit Relays	Number of DNS Resolver ASes
`ECS subnet length 0`	7	6
`ECS subnet length 24`	1,778	23
`ECS subnet length 32`	1	1
`ECS subnet length 48`	2	3
`ECS subnet length 56`	197	5
`Having DNS requests with ECS information included`	1,815	25
`Having DNS requests without ECS information included`	3,281	381

All DNS requests with an ECS subnet length of zero had the ECS network address set to 0.0.0.0. An ECS subnet length of zero explicitly indicates that no ECS information should be included in the DNS request. [17]

From July 3, 2024, to September 25, 2024, ECS data with a subnet length of 32 was received from one exit relay. This exit relay consistently used a single DNS resolver, which was not observed again after this period. The ECS data included the exit relay's full IPv4 address, thereby exposing the specific exit relay responsible for the DNS request to any observer along the path beyond the recursive DNS resolver and to the authoritative name server. Sending an ECS subnet length more specific than 24 for IPv4 addresses, i.e. greater than 24, is not permitted. [17]

Furthermore, DNS requests from 1,104 exit relays included an ECS subnet length of 24 and the corresponding network address of the exit relay.

When the ECS information in a DNS request includes the exit relay's /24 network address, it may reveal that the request originated from a Tor exit relay, particularly when multiple exit relays operate within that same subnet.

Result: Case Randomization

This test evaluated whether the recursive DNS resolvers querying the authoritative name server applied case randomization to domain names, a technique that enhances protection against cache poisoning.

Analysis Period: July 3, 2024 - August 14, 2024, and August 25, 2024 – February 28, 2025

Analysis Description: The test DNS requests consisted exclusively of lowercase letters. On the authoritative name server, the received DNS requests were analyzed to determine whether the query names contained mixed-case letters, indicating the use of case randomization.

Assumptions and Limitations: Query names composed entirely of uppercase letters were assumed to be fully capitalized by chance and were therefore counted

Table 4.9: Number of mixed-case requests per domain level

Domain	Percentage Case Randomized
`tordns.ovh`	12.86%
`ns1.tordns.ovh, ns2.tordns.ovh`	55.03%
`subdomain1.tordns.ovh`	92.46%
`subdomain2.subdomain1.tordns.ovh`	98.98%

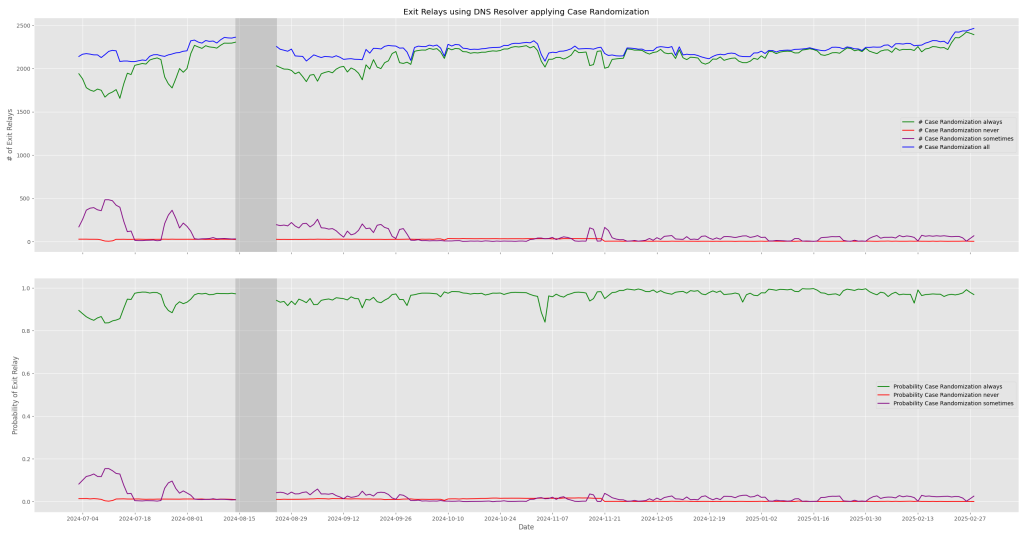

Figure 4.13: Number and probability of exit relays sendin mixed-case query names

as randomized queries. Query names containing only lowercase letters were not counted as randomized, which could lead to false negatives.

Exit relays apply case randomization by default before sending DNS to their configured DNS resolver. DNS resolvers themselves may also apply case randomization to query names. It could not be determined whether an observed case randomization originated from the exit relay or the DNS resolver. Therefore, when case randomization was detected, it was assumed to have been applied either by the exit relay alone or by both the exit relay and the DNS resolver.

On July 25, 2023, Google announced that Google's Public DNS resolver had enabled case randomization by default[12].

Results: The proportion of DNS requests containing mixed-case query names varied depending on the queried domain. DNS requests for the domain name tordns.ovh and for the name servers themselves exhibited substantially lower rates of randomized cases.

Figure 4.13 shows the number of exit relays and their combined probability of using a DNS resolver that sent DNS requests with mixed-case domain names. Only DNS requests for subdomain1.tordns.ovh and subdomain2.subdomain1.tordns.ovh were considered in this graph.

[12]https://groups.google.com/g/public-dns-discuss/c/KxIDPOydA5M

Table 4.10: Test period and the corresponding DNSSEC test domain names

Test Period	**Test Domains**
`2024-07-03 - 2024-07-11`	`sigfail.verteiltesysteme.net,` `www.rhybar.cz`
`2024-07-12 - 2024-08-14`	`sigfail.verteiltesysteme.net,` `www.rhybar.cz,` `*.invalidkey.dnssec-check.ovh`
`2024-08-25 - 2024-11-04`	`sigfail.verteiltesysteme.net,` `www.rhybar.cz,` `*.invalidkey.dnssec-check.ovh`
`2024-11-05 - 2025-01-03`	`sigfail.verteiltesysteme.net,` `www.rhybar.cz`
`2025-01-04 - 2025-02-28`	`sigfail.verteiltesysteme.net,` `www.rhybar.cz,` `*.invalidkey.dnssec-check.ovh`

Test and Result: DNSSEC Support

This test evaluated how many exit relays correctly validated DNSSEC.

Test period: July 3, 2024 - August 14, 2024, and August 25, 2024 – February 28, 2025

Test description: DNS requests were sent to domains with invalid DNSSEC signatures. If a DNS resolver returned an IP address for these domains, it was concluded that DNSSEC validation was not performed, as the response should have failed. To verify the availability of the DNS resolvers and authoritative name servers, additional DNS requests were sent to correctly signed domain names.

The following domains were used in the test:

- *Intentionally invalidly signed domain names* sigfail.verteiltesysteme.net[13], www.rhybar.cz[14], and [subdomain].invalidkey.dnssec-check.ovh
- *Correctly signed domain names* sigok.verteiltesysteme.net and [subdomain].dnssec-check.ovh

If a DNS resolver successfully resolved a domain name with an invalid DNSSEC signature, it was classified as not performing DNSSEC validation. If none of the correctly signed domain names were resolved, the test result was considered invalid.

Assumptions and Limitations: The authoritative name server for dnssec-check.ovh was not available during the entire test period, which resulted in a different use of test domain names for some periods, listed in Table 4.10.

Exit relays can use multiple DNS resolvers to resolve a single DNS request. Consequently, it was not possible to determine which specific DNS resolver performed the resolution, and exit relays might exhibit different results over time when using multiple DNS resolvers.

[13]https://wander.science/projects/dns/dnssec-resolver-test/
[14]https://www.internetsociety.org/resources/deploy360/2013/dnssec-test-sites/

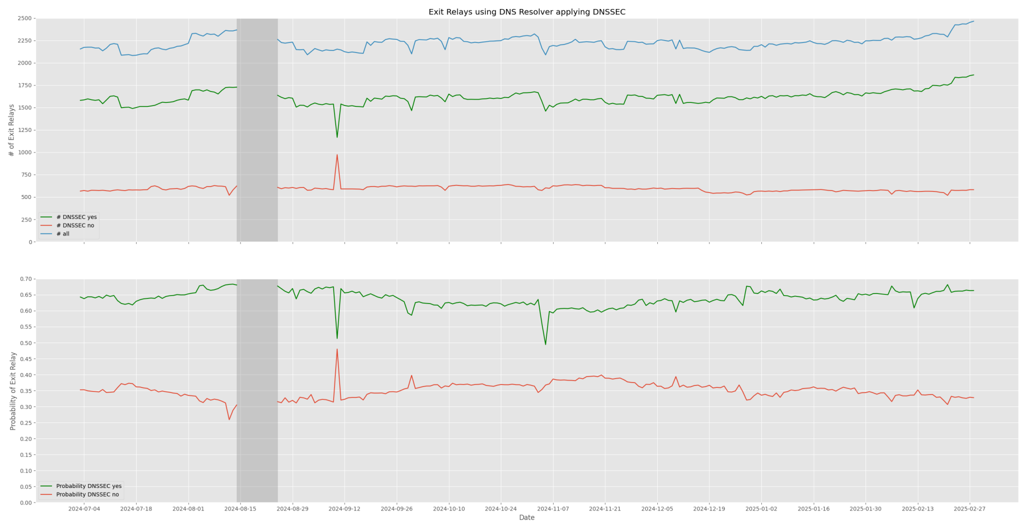

Figure 4.14: Number and probability of exit relays validating DNSSEC

Results: Figure 4.14 shows that the majority of exit relays, representing approximately 65% of the total exit relay probability, correctly validated DNSSEC.

Test and Result: Exit Relays Using Cache

This test determined whether exit relays used their own DNS cache.

Test Period: September 20, 2024 – February 28, 2025

Test Description: Two DNS requests for a domain name unique to each test and exit relay were sent within a few minutes of each other. The first DNS request was not cached at the exit relay. When an exit relay stores a DNS request in its cache, it clips the original TTL value to a fixed value and randomizes this TTL value within a fixed range. Therefore, if the TTL value in the DNS response of the second DNS request for the same domain name matches the TTL value in the first DNS response, the domain name was considered cached with a probability of 0.9979. This is due an issue of the DNS resolution by Tor exit relays, see section 4.4.4.

The test domain name followed the pattern [subdomain].tordns.ovh where [subdomain] was unique per test and exit relay.

Assumptions and Limitations: This test exploits a security issue and was applied exclusively to domain names under the control of the author's authoritative name server.

Only successful tests for which DNS responses were received for both DNS requests were evaluated.

Since the authoritative domain's TTL value was set to the maximum value of 604,800 seconds, a cached domain name on the exit relay was expected to have a TTL value between 3,360 and 3,840 seconds.

A domain name was considered cached when both DNS responses had the same TTL value within this expected range.

Results: All successful tests showed identical TTL values in both DNS responses, indicating that all successfully tested exit relays used a DNS cache.

Test and Result: Performance of DNS Resolution

This test measured the time required to resolve various test DNS domain names through the Tor network.

Test Period: July 3, 2024 - August 14, 2024, and August 25, 2024 – February 28, 2025

Test Description: The following test DNS requests were sent, and their response times were measured:

- *[subdomain2].[subdomain1].tordns.ovh* Each test DNS request sent for the test in section 4.5.2 was unique throughout the test period and therefore neither cached at the exit relay nor at any recursive DNS resolver. Three DNS requests for unique domain names were sent per test and exit relay, and the average value of the response times was calculated.
- *a.gtld-servers.net* This domain name, belonging to the authoritative name servers for the .com domain, was likely uncached at the exit relay but cached at the recursive DNS resolver. Two DNS requests for this domain name were sent per exit relay and test. After the first request, a cache hit at the exit relay's cache was expected for the second request.
- **_cache-test.tordns.ovh* The two test DNS requests sent for the test conducted and described in section 4.5.2 were unique per exit relay and test. The first DNS request was expected to be neither cached at the exit relay nor at any recursive DNS resolver, while the second was expected to be served from the exit relay's cache. Tests for this domain were conducted from 20 September 2024 until the end of the test period on 28 February 2025.

Assumptions and Limitations: The response time was measured in microseconds using the Linux dig command.

The measured time reflects not only the performance of the exit relay but also that of the Tor network, specifically the entry guard and middle node, and the test servers. As the default Tor circuit was used for connecting to exit relays, the large number of tests is assumed to provide statistically representative average values for successfully resolved DNS requests. Failed DNS requests were excluded from the analysis.

Results: Figure 4.15 presents the average time required to resolve cached and uncached domain names.

Aligned variations in response times among different test domain names indicate the performance of the Tor network and the test servers. Occasional outliers of specific test domain names, e.g. the spike on August 12, 2025 for *.tordns.ovh (459,918 µs), are likely attributable to temporary fluctuations in the performance of the authoritative name server. A gradual trend toward shorter DNS resolution times was observed during the second half of the test period.

As expected, domain names cached at the exit relay, such as a.gtld-servers.net and *_cache-test.tordns.ovh, were resolved the fastest, as no further DNS resolving outside the exit relay was required. There was no observable performance difference between these two domain names.

Two distinct patterns were observed among the domain names that were not cached at the exit relay. The domain name a.gtld-servers.net was likely cached

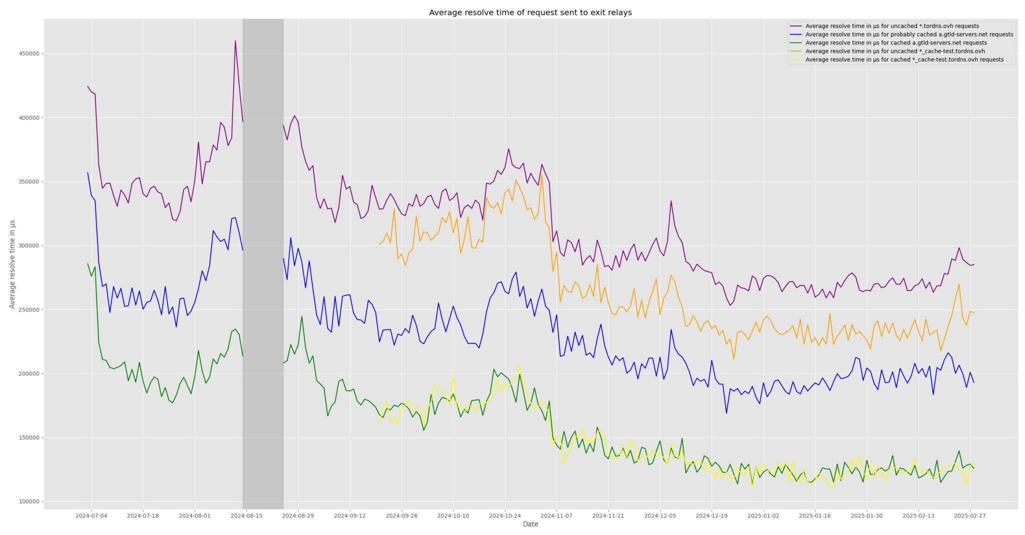

Figure 4.15: Time in µs needed for resolving test domain names

at the recursive DNS resolvers used by exit relays, meaning no additional recursive resolution was required. In contrast, the difference in resolution time between *.tordns.ovh and *_cache-test.tordns.ovh is likely attributable to the number of subdomain levels. Recursive DNS resolvers that apply QNAME minimization must perform one additional iterative DNS request to the authoritative name server for the deeper subdomain structure of *.tordns.ovh, which explains the longer resolution time.

Conclusion of Results

Not all exit relay operators follow the Tor Project's recommendations for DNS resolution. The probability that an exit relay uses a DNS resolver located within the same AS is between 40% and 50%. Not all DNS resolvers validate DNSSEC (probability approximately 65%) or apply QNAME minimization (probability approximately 80%).

For some exit relays, the ECS information contained the network address of the exit relay with a subnet length of 24, resulting in potential exposure beyond the recursive DNS resolver. Exit relays should explicitly request that recursive DNS resolvers omit ECS information, and this behavior should be tested regularly. If a resolver continues to include ECS information despite such requests, a different DNS resolver should be selected. In one observed case, an exit relay's full IP address was exposed in the ECS information, constituting a clear privacy risk and a violation of the ECS standard.

Chapter 5

Proposed Improvements

This chapter discusses and proposes improvements to the DNS, based on the threats found in chapter 3 and chapter 4, and reevaluates remaining or potentially newly introduced drawbacks or threats.

The necessity of a more secure DNS resolution mechanism for the Tor network is also addressed in a design proposal [80]. The proposal suggests shifting the responsibility for DNS resolution from the exit relay to the client, thereby keeping DNS requests confidential from exit relays and ensuring the application of DNSSEC. Users should no longer have to trust the exit relay to choose DNS resolvers.

The improvements proposed in this chapter aim to provide methods for a privacy- and security-enhanced DNS resolution that benefit Internet users in general as well as clients on the Tor network. When users apply these measures themselves, they no longer depend on exit relays for DNS resolution when using the Tor network. Conversely, exit relays implementing these measures can improve privacy and security for Tor clients who still rely on them for DNS resolution.

To enhance the privacy and security of DNS resolution, whether performed by Tor exit relays or by clients themselves, the following goals are defined:

- *Use existing systems and protocols*: Improvements should integrate into the existing DNS infrastructure. Available security enhancements such as DNSSEC, QNAME minimization, and encrypted DNS should be applied.
- *Enhance confidentiality and privacy*: Wherever feasible, information should be encrypted, and each actor should only have access to the plaintext information necessary to perform its function.
- *Provide integrity and authentication*: The integrity of information should be ensured, and communicating partners should be authenticated.
- *Prevent censorship*: Only the user should decide which domain names to block. DNS resolvers must not block any domain names.
- *Provide availability*: DNS resolution should not depend on the availability of a single recursive DNS resolver.
- *Mitigate data analysis*: The use of DNS data for creating user or behavioral profiles, which can enable reidentification of users even without direct IP address correlation, should be prevented. Furthermore, DNS data should not be usable for correlation attacks, such as the DefecTor attack on the Tor network [79].

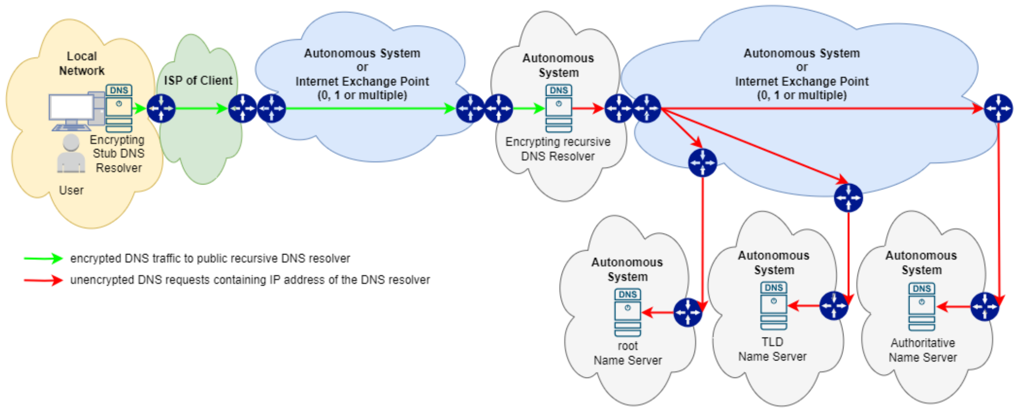
Local Network
DNS
Encrypting Stub DNS Resolver
User
ISP of Client
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Autonomous System
Encrypting recursive DNS Resolver
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Autonomous System
root Name Server
Autonomous System
TLD Name Server
Autonomous System
Authoritative Name Server
encrypted DNS traffic to public recursive DNS resolver
unencrypted DNS requests containing IP address of the DNS resolver

Figure 5.1: Architecture of encrypted DNS resolution

- *Provide anonymization*: In this thesis, anonymity is defined as preventing disclosure of the IP address of the original sender of DNS requests to any entity on the DNS resolution path beyond the client's trusted network. This includes forwarding and recursive DNS resolvers as well as authoritative name servers. None of these entities should be able to identify the originator of a DNS request.
- *Users should choose DNS resolvers*: Minimum privacy requirements for DNS resolvers should be defined, and a list of DNS resolvers meeting these requirements should be provided to users so they can decide which DNS resolvers to use.

5.1 Encrypting DNS Requests

In recent years, various encrypted DNS protocols have emerged. These protocols provide encryption of DNS traffic between the client and the DNS resolver, as well as authentication of the DNS resolver itself. To use these protocols, the client must send DNS requests through software that supports the chosen protocol. Such functionality can be integrated into applications, e.g. Firefox[1], or implemented by the operating system as a stub resolver. Correspondingly, DNS resolvers must support these protocols. Except for an experimental implementation of DoT at the root server b.root-servers.net[2], root, TLD, and authoritative name servers do not support encryption. Recursive DNS resolvers only use conventional, unencrypted DNS when querying name servers. [27, 28, 29]

Figure 5.1 illustrates the architecture of DNS when using encrypted DNS protocols.

An adversary located on the path between the client and the recursive DNS resolver can no longer monitor or manipulate DNS requests once encryption is applied. This potentially mitigates domain name blocking. In [81], the authors tested the circumvention of blocked domains by switching from conventional DNS to encrypted DNS. Using vantage points in different countries, they detected DNS traffic manipulation in five countries. Switching to encrypted DNS

[1]https://support.mozilla.org/en-US/kb/firefox-dns-over-https
[2]https://b.root-servers.org/research/tls.html

enabled access to censored domains with varying success rates depending on the country, e.g., 100% in Portugal, 50% in Denmark, and 37% in China. Changing from one encrypted DNS resolver to another did not significantly affect these results. However, some encrypted DNS resolvers have been blocked in certain countries, e.g. Google's public recursive DoH resolver was inaccessible in China.

In [82], researchers examined the accessibility of 1,600 domains and encrypted DNS resolvers (DoT and DoH) over a six-month period from more than 20,000 vantage points in various countries around the globe, including countries classified as "not free" by Freedom House[3]. Their 315,000 measurements showed that encrypted DNS enabled unblocking between 55% and 95% of censored domains depending on the country.

5.1.1 Encrypted DNS Protocols

In [83], the authors studied encrypted DNS resolvers and identified over 150 DoT and 17 DoH providers, generally offering satisfactory service quality suitable for large-scale, real-world usage. Their findings indicate that DoT and DoH are the leading encrypted DNS protocols, extensively supported by large public DNS resolvers, followed by DNSCrypt, which they classify as partially supported.

Similarly, [84] measured the adoption of encrypted DNS protocols in three large organizations and found DoH and DoT were the predominant protocols for encrypted DNS.

An online search for encrypted DNS resolvers conducted in this thesis (see section 5.4.2) revealed a substantial number of publicly available DoH, DoT, and DNSCrypt resolvers.

Based on these findings, DNS over HTTPS, DNS over TLS, and DNSCrypt were selected for encrypting DNS requests.

5.1.2 Threat Analysis Update

While encrypted DNS enhances the security of DNS resolution, certain threats persist. Moreover, its use may be perceived as problematic by intelligence agencies and police forces [2].

Blocking Encrypted DNS

The presence of encrypted DNS traffic could be blocked by ISPs or AS operators.

[84] found four times as many DoH resolvers that were hidden or not publicly listed as those listed on the most comprehensive list of well-known DoH resolvers, leading the authors to conclude that blocking DoH traffic based solely on well-known DoH resolver lists is not effective.

[82] detected blocking efforts against encrypted DNS in several countries, including China, Russia, and Saudi Arabia, and observed a significant increase in

[3]https://freedomhouse.org/country/scores

the blocking of DoT and DoH traffic in China in March 2021. Blocking encrypted DNS resolver domain names at the AS level was found only for ordns.he.net in China, blocked by the Great Firewall via DNS poisoning, and in one AS located in Thailand, where two domain names belonging to Cloudflare's encrypted DNS resolvers were blocked.

Preventing the detection of encrypted DNS traffic or concealing DNS traffic altogether is considered out of scope for this thesis.

Centralization of Data and Control

Directing all DNS requests to a single DNS resolver introduces a privacy risk through data centralization, particularly when large organizations operate the DNS resolvers. Encrypted DNS resolver operators observe both the client IP addresses and the requested domain names, creating a single point of surveillance.

The DNS resolver also determines which DNS security enhancements are applied, such as DNSSEC validation or QNAME minimization, thereby affecting all of its clients. Furthermore, DNS resolvers may block specific domain names based on undisclosed blocklists or refuse to respond to certain clients, e.g., DNS requests coming from the Tor network. A certain degree of trust in the DNS resolver is therefore required.

Reliance on a single DNS resolver and its infrastructure also introduces a single point of failure. An outage such as the incident involving Cloudflare's public DNS resolver on July 14, 2025, [69], would disrupt DNS resolution for all dependent clients.

DNS Fingerprinting

The inclusion of EDNS client subnet information that contains a client's corresponding network with subnets up to /24 by a DNS resolver can reveal a client's network address on the path between the recursive DNS resolver and the authoritative name server. Observing such DNS traffic enables adversaries to classify clients of a DNS resolver by network origin and to build DNS-based profiles, especially in cases where few clients share the same network.

Even if DNS traffic is encrypted and the requested domain names are not visible to an attacker on the path between the client and the encrypted DNS resolver, fingerprinting techniques can still be applied to infer visited websites. Using the first 50 DoH packets, [85] achieved an accuracy of 95% in identifying website domain names in a closed-world scenario and an F1-score of 93% with 100,000 websites in an open-world scenario. [86] proposed a feature set for fingerprinting attacks on encrypted DoH traffic, achieving comparable accuracy to state-of-the-art website traffic fingerprinting while requiring 124 times less data volume.

DNSCrypt exposes certain metadata, such as the total number of DNS requests and their inter-arrival times, enabling potential attacks that can disclose visited websites [2].

5.2 Distributing Encrypted DNS Requests

Distributing a client's DNS requests when accessing a single resource across multiple encrypted DNS resolvers reduces the amount of information each DNS resolver receives. To effectively distribute DNS data, the chosen DNS resolvers must be operated by independent organizations. A single organization may host multiple DNS resolvers under different domain names for different geographical locations or encryption protocols.

By ensuring that each DNS resolver receives only a fraction of the overall DNS requests, the ability of an individual DNS resolver to create or reidentify user profiles is significantly diminished. Also, fingerprinting encrypted DNS requests on the path between the client and DNS resolvers becomes more difficult, as an attacker would need to observe and identify all a client's DNS requests sent to multiple DNS resolvers.

The number of independent DNS resolvers and the algorithm for distributing DNS requests are critical factors. The minimum number of DNS resolvers required also depends on the number of DNS requests typically originating when accessing a single resource, e.g. visiting a website. According to [87], 50% of the Alexa Global Top 100,000 websites require at least 20 DNS requests to fully load their landing pages. Further research is required to determine the minimum number of DNS resolvers needed to effectively prevent DNS fingerprinting and user profiling.

Encrypted DNS resolvers should meet specific criteria to ensure reliability and security. Requirements for recursive encrypted DNS resolvers are defined, and their evaluation is presented in section 5.4. Regular assessments of DNS resolvers should be conducted, and DNS resolvers that are no longer functional or no longer meet the requirements, should be excluded. Any failing resolver should be promptly deactivated to minimize resolution delays caused by timeouts and avoid redundant transmissions of the same DNS request to multiple DNS resolvers.

For large-scale networks such as company networks or exit relays, distributing DNS requests can also reduce the load on individual DNS resolvers compared to relying on a single external DNS resolver for all DNS requests. This can be particularly beneficial for DNS resolver operators with limited resources.

However, distributing DNS requests across multiple DNS resolvers may increase the overall time required to load a website, as some DNS resolvers might respond more slowly. Using a larger number of DNS resolvers increases the likelihood of encountering a slower one [88].

Figure 5.2 illustrates a client distributing DNS requests to multiple recursive encrypted DNS resolvers.

5.2.1 Algorithm for Distribution

Several algorithms can be applied for distributing DNS requests among multiple encrypted DNS resolvers. Relevant approaches are the following:

- *Random*: Each DNS request is sent to a randomly selected DNS resolver from a set of available DNS resolvers. Over time, all DNS resolvers are expected to be chosen equally often.

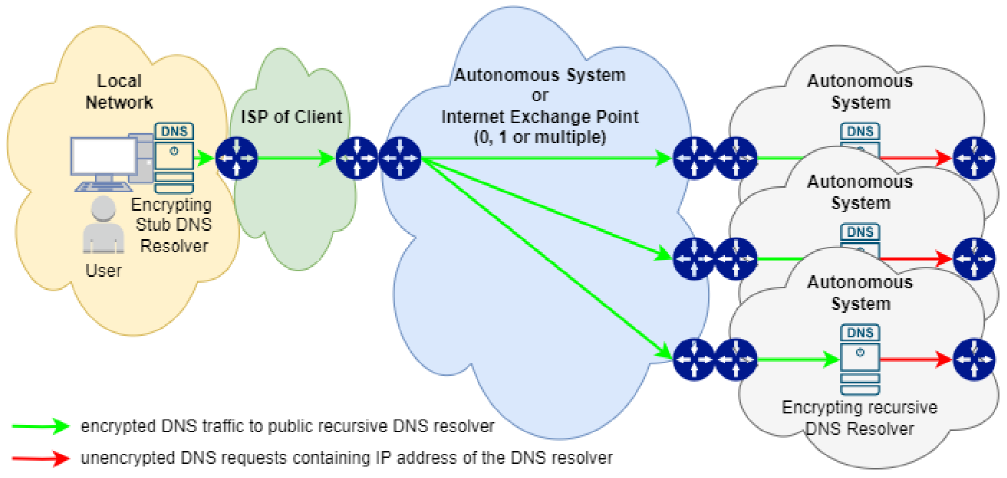
Local Network
DNS
Encrypting Stub DNS Resolver
User
ISP of Client
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Autonomous System
DNS
Autonomous System
DNS
Autonomous System
DNS
Encrypting recursive DNS Resolver
encrypted DNS traffic to public recursive DNS resolver
unencrypted DNS requests containing IP address of the DNS resolver

Figure 5.2: Distributing DNS requests to multiple encrypted DNS resolvers

- *Weighted Random*: Similar to the random approach, but DNS resolvers are assigned different selection probabilities. Some DNS resolvers will be chosen more frequently than others based on their assigned weights.
- *Round-robin*: DNS requests are sent sequentially to DNS resolvers in a list arranged in a random but fixed order, starting with the first DNS resolver in the list. After a DNS request is sent to the last DNS resolver in the list, the next DNS request is sent to the first DNS resolver in the list again. Over time, each DNS resolver receives an equal number of DNS requests.
- *Weighted Round-robin*: Similar to the round-robin approach, but some DNS resolvers appear multiple times in the list of available DNS resolvers. As a result, these DNS resolvers are selected more frequently than others according to their assigned weights, i.e. their number of appearances on the list.
- *Domain-name-based*: The DNS resolver is selected based on the domain name, e.g. a hash of the domain name. When grouping by domain names or their hash values, the number of groups likely does not match the number of available DNS resolvers. This results in an unequal distribution of DNS requests and makes it difficult to predict the fraction of DNS requests that each DNS resolver will receive.

Weight factors used to define DNS resolver selection probabilities can be determined based on factors such as the DNS resolver's available resources or its performance metrics.

5.2.2 Threat Analysis Update

To avoid organizational centralization, it is essential to use independent DNS resolvers. In [89], the authors discovered 23,960 public encrypted DNS resolvers, of which 64.83% operated as forwarding DNS resolvers, whereas 35.17% of these DNS resolvers used the same IP address for both offering DNS resolution and querying authoritative name servers. The top ten DNS providers querying authoritative name servers served 75.24% of all public encrypted DNS resolvers. These findings indicate a high degree of centralization and interdependence among public DNS resolvers.

DNS fingerprinting on encrypted DNS data may still be feasible if an attacker can observe all DNS requests, particularly when located close to the client, such as the client's ISP. Websites or software applications may also identify the DNS resolvers used by a client by issuing numerous DNS requests for domain names under their control.

When applying random or round-robin distribution algorithms, each DNS resolver receives most DNS requests for regularly requested domain names after a sufficiently long period, enabling them to create user profiles. To address this issue, [88] proposed domain-specific sharding, in which a DNS resolver receives all queries corresponding to the same second-level domain name. A domain name map assigns these domains to specific DNS resolvers. Without sharding, the authors were able to reidentify 88% of users in their tests. Implementing a sharding value of two, i.e. distributing DNS requests across two DNS resolvers, lowered reidentification to 79%, while a value of eight reduced it to 49%. They further proposed the adaptive insertion of DNS requests of popular domain names based on the uniqueness of the user's DNS stream, which further significantly lowered the probability of user reidentification in their evaluations.

5.3 Anonymizing the Original Sender of DNS Requests

In this approach, the client no longer sends DNS requests directly to encrypted DNS resolvers but instead transmits them through the Tor network. This prevents DNS resolvers on the DNS communication path beyond the exit relay from learning the client's actual IP address, as they can see only the IP address of the exit relay. All entities on the DNS communication path before the DNS requests enter the Tor network must be trusted.

The Tor network provides anonymization for its clients by routing TCP traffic through a series of Tor nodes. It also supports the resolution of conventional DNS requests. Tor software running on the client can be configured to receive DNS requests on UDP port 53 and forward them to the exit relay on an established Tor circuit. However, these DNS requests are limited to the DNS record types A, AAAA, and PTR. The exit relay determines which DNS resolver to use and sees the DNS requests in plaintext. [30, 35]

Sending DNS requests to the exit relay without using the same Tor circuit for web browsing may appear to be suspicious behavior. A client issuing numerous DNS requests without opening subsequent connections to the corresponding targets may be perceived as performing a DNS scan. To mitigate such behavior, countermeasures, such as limiting DNS requests without subsequent data traffic or introducing artificial delays, have been proposed. [90]

To implement the improvements proposed in the previous sections, encrypted DNS requests are sent via TCP through the Tor network to the selected encrypted DNS resolvers. In this configuration, the client retains control over which DNS resolvers are used, all DNS record types supported by the DNS resolver remain available, and the exit relay no longer has access to the content of

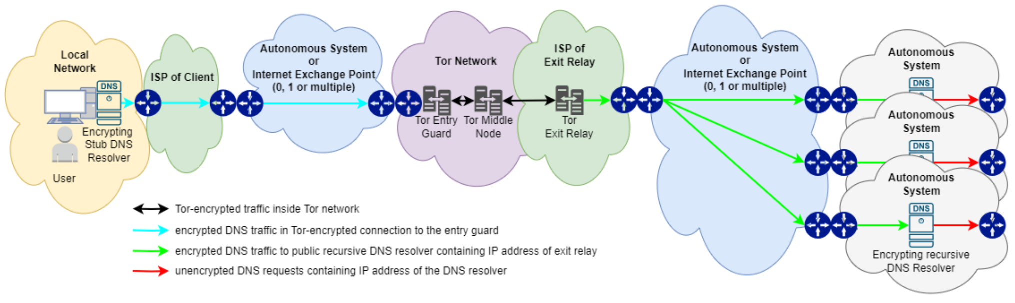
Local Network
DNS
Encrypting Stub DNS Resolver
User
ISP of Client
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Tor Network
Tor Entry Guard
Tor Middle Node
ISP of Exit Relay
Tor Exit Relay
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Autonomous System
Encrypting recursive DNS Resolver
Tor-encrypted traffic inside Tor network
encrypted DNS traffic in Tor-encrypted connection to the entry guard
encrypted DNS traffic to public recursive DNS resolver containing IP address of exit relay
unencrypted DNS requests containing IP address of the DNS resolver

Figure 5.3: Anonymizing the original sender's IP address of DNS requests by sending them over the Tor network

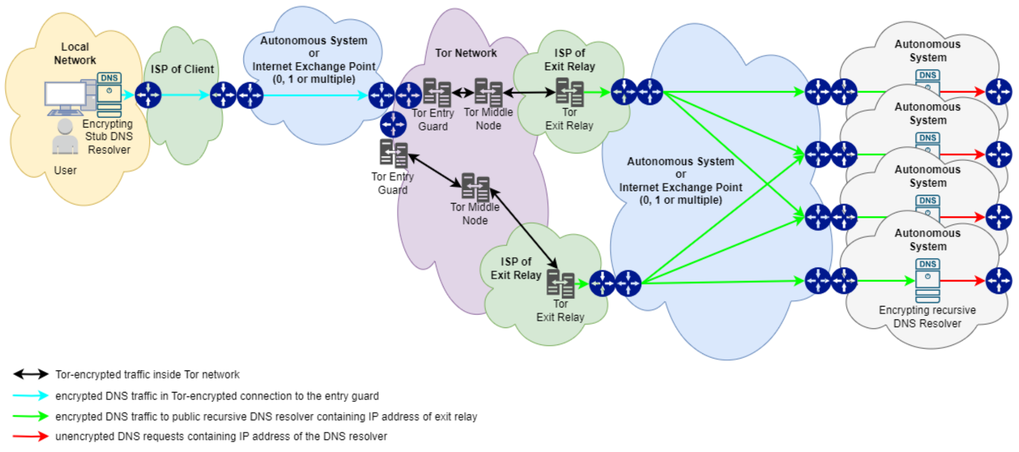
Local Network
DNS
Encrypting Stub DNS Resolver
User
ISP of Client
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Tor Network
Tor Entry Guard
Tor Middle Node
ISP of Exit Relay
Tor Exit Relay
Tor Entry Guard
Tor Middle Node
ISP of Exit Relay
Tor Exit Relay
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Autonomous System
Encrypting recursive DNS Resolver
Tor-encrypted traffic inside Tor network
encrypted DNS traffic in Tor-encrypted connection to the entry guard
encrypted DNS traffic to public recursive DNS resolver containing IP address of exit relay
unencrypted DNS requests containing IP address of the DNS resolver

Figure 5.4: Anonymizing the original sender's IP address of DNS requests by sending them over the Tor network

the DNS traffic. Conventional DNS resolution provided by exit relays may still be used for DoH bootstrap DNS requests.

In [86], the authors found that, in contrast to traffic analysis attacks on web traffic, Tor's encryption between the client and entry guard offers strong protection against traffic analysis attacks on encrypted DNS traffic, providing effective resistance to fingerprinting.

Since all clients using the same exit relay share the same exit IP address, a DNS resolver receiving requests from that exit relay can only create a single aggregated user profile representing all its clients.

Clients may apply this method to send DNS requests regardless of whether they are using the Tor network for web browsing.

Figure 5.3 illustrates an overview of a client sending encrypted DNS requests through the Tor network.

Clients may use multiple Tor circuits simultaneously and thus employ different exit IP addresses to send encrypted DNS requests. Different sets of DNS resolvers can be used for each Tor circuit. Figure 5.4 illustrates a client using two exit relays to send encrypted DNS requests to multiple DNS resolvers.

Exit relays may resolve conventional DNS requests received from their clients by forwarding them as encrypted DNS requests to encrypted DNS resolvers, or by relaying encrypted DNS requests through one or more other exit relays.

Routing encrypted DNS requests from Tor exit relays to multiple DNS resolvers would complicate the DefecTor correlation attack [79], discussed in section 4.4.5. Adversaries located on the network path between an exit relay and a DNS resolver would no longer observe plaintext DNS traffic and would be required to analyze multiple encrypted connections instead of a single unencrypted connection. Nevertheless, DNS resolvers and any participants in the upstream resolution process, e.g. forwarding DNS resolvers or authoritative name servers, remain potential adversaries, and fingerprinting of encrypted DNS traffic may be feasible. Distributing encrypted DNS requests across multiple exit relays, either by clients or by exit relays themselves, could further increase the difficulty of the attack.

5.3.1 Threat Analysis Update

DNS resolvers can observe the IP addresses of exit relays and may block DNS requests coming from them. Since concealing the use of the Tor network is not an objective of this thesis, the selected DNS resolvers must therefore accept DNS requests coming from exit relays.

To mitigate the risk of correlation attacks at the AS level, DNS resolvers located within the same AS as the client or the entry guard of a client's Tor circuit should be avoided.

Furthermore, a client's stub DNS resolver must not send ECS information, as doing so would reveal the client's actual network address to DNS resolvers and compromise anonymity.

5.4 Evaluation of Encrypted Recursive DNS Resolvers

5.4.1 Requirements for Encrypted DNS Resolvers

The following requirements are established for encrypted DNS resolvers:

- *DNSSEC*: DNS resolvers must validate DNSSEC for all domains that are DNSSEC-signed.
- *QNAME Minimization*: QNAME minimization should be applied for the first four labels of a domain name, as recommended in [24].
- *ECS*: DNS resolvers should not include any ECS information in DNS requests. If ECS information is included, the client's IP address must not fall within the ECS network.
- *No Blocking of Domain Names*: DNS resolvers must not block or manipulate DNS responses based on the requested domain name.
- *Acceptance of DNS Requests from the Tor Network*: DNS resolvers must respond to DNS requests coming from the Tor network in the same manner as to those coming from other networks.

- *Acceptable Performance*: DNS resolvers should provide responses within an acceptable timeframe. Since acceptable response time may vary depending on user expectations, no strict limit is defined. Instead, DNS resolvers should be ranked according to their average response time for DNS requests.
- *Low Error Rate*: The proportion of DNS responses to valid domain names that contain DNS response codes indicating an error due to resolver-side issues should not exceed the average error rate observed among other DNS resolvers.
- *High Availability*: The DNS resolver must maintain high availability and operate without significant outages.
- *Valid Certificates*: DoH and DoT resolvers must use valid TLS certificates issued by certificate authorities recognized by common operating systems. DNSCrypt resolvers must provide valid operator-issued certificates.
- *Independence of DNS Resolvers*: DNS resolvers must not be operated by the same organization.

The search for encrypted DNS resolvers, described in the following section, revealed that many DNS resolvers do not provide a privacy statement. Moreover, privacy-related claims made by DNS resolver operators cannot be verified through testing. Since the proposed improvements include anonymization of the original sender of DNS requests, logging policies are not considered a requirement for encrypted DNS resolvers in this thesis.

5.4.2 Finding Public Encrypted DNS Resolvers

A manual online search was conducted to find public encrypted DNS resolvers. Collections of such encrypted DNS resolvers are listed on various websites[4]. From all discovered DNS resolvers, 161 DoH, 40 DoT, and 133 DNSCrypt resolvers responded successfully to DNS requests during a plausibility test conducted at the time of the search. These DNS resolvers were included in the test set for further evaluation. Some DNS resolvers were found to be operated by the same organizations but offered multiple encryption protocols or operated from different geographical locations.

However, the selected set of DNS resolvers for testing may not be representative of all publicly available resolvers.

5.4.3 Test Setup

The test setup consists of three main components:

- *Evaluation Server*: This server receives information needed for testing, assigns specific test cases to the test servers, hosts a database for storing test results, and processes log files.
- *Two Test Servers*: These servers are responsible for issuing DNS requests to the encrypted DNS resolvers.

[4]https://dnscrypt.info/public-servers, https://dnsprivacy.org/public_resolvers, https://github.com/curl/curl/wiki/DNS-over-HTTPS

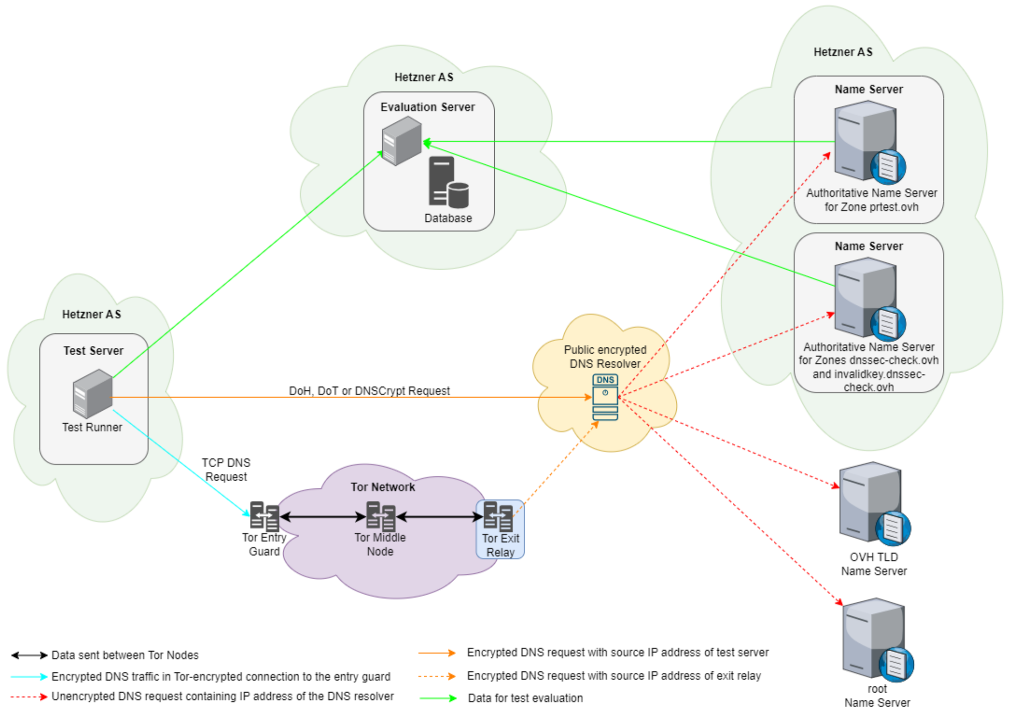
Hetzner AS
Evaluation Server
Database
Hetzner AS
Name Server
Authoritative Name Server for Zone prtest.ovh
Name Server
Authoritative Name Server for Zones dnssec-check.ovh and invalidkey.dnssec-check.ovh
Hetzner AS
Test Server
Test Runner
DoH, DoT or DNSCrypt Request
Public encrypted DNS Resolver
DNS
TCP DNS Request
Tor Network
Tor Entry Guard
Tor Middle Node
Tor Exit Relay
OVH TLD Name Server
root Name Server
Data sent between Tor Nodes
Encrypted DNS traffic in Tor-encrypted connection to the entry guard
Unencrypted DNS request containing IP address of the DNS resolver
Encrypted DNS request with source IP address of test server
Encrypted DNS request with source IP address of exit relay
Data for test evaluation

Figure 5.5: Test architecture for testing public encrypted DNS resolvers

- *Two Authoritative Name Servers*: These servers are authoritative for the domains prtest.ovh and dnssec-check.ovh, which are used throughout the testing procedures.

All servers are virtual servers rented from a hosting provider and configured with dedicated IPv4 addresses. They operate within the AS of Hetzner Online GmbH, AS number 24940, AS name *Hetzner Online GmbH*. Figure 5.5 illustrates an overview of the test setup.

Domain Name Set

The domain name set used for testing comprises domain names classified into the following categories:

- *Popular domain names*: Top one million domain names taken from The Majestic Million[5].
- *Likely blocked domain names*: A set of 18,733 domain names from four categories of blocklists (Ads[6], Malware[7], Adult[8], and Tracking[9]). This list is intended to identify DNS resolvers that perform blocking and is not exhaustive. A domain name may appear in multiple blocklist categories and may also be present in the popular domain list.

[5]https://majestic.com/reports/majestic-million, downloaded on 12.06.2024
[6]https://blocklistproject.github.io/Lists/alt-version/ads-nl.txt, downloaded on 10.06.2024
[7]https://blocklistproject.github.io/Lists/alt-version/malware-nl.txt, downloaded on 08.01.2024
[8]https://blocklistproject.github.io/Lists/alt-version/porn-nl.txt, downloaded on 08.01.2024
[9]https://blocklistproject.github.io/Lists/alt-version/tracking-nl.txt, downloaded on 08.01.2024

- *Domain names for specific testing purposes*:
  - *DNSSEC failing domain names*: The domain names *.invalidkey.dnssec-check.ovh, www.rhybar.cz, and sigfail.verteiltesysteme.net are deliberately incorrectly DNSSEC-signed. A DNSSEC validating DNS resolver must not resolve these domain names.
  - *Domain under control*: The domain prtest.ovh is authoritative on the operated name server. Domain names in the form [subdomain2].[subdomain1].prtest.ovh are used to gather information about DNS resolvers that query the authoritative name server. [subdomain1] and [subdomain2] are unique per test and per DNS resolver and include a test ID. This prevents caching and enables identifying the DNS resolver being tested.
  - *Root name servers*: The 13 root name server hostnames (a.root-servers.net through m.root-servers.net) are included as they are very likely to be cached by recursive DNS resolvers and are not expected to be blocked.

The combined set of popular and likely blocked domain names contains 1,009,251 unique domain names. Not every domain name in this set was used for testing. Domain names in the category of specific testing purposes were queried multiple times per DNS resolver and per day.

Evaluation Server

The evaluation server compiles a daily list of domain names to be tested. Each day, it randomly selects 2,500 domain names from the combined set of popular and likely blocked domain names, appends the domain names used for specific testing purposes, and distributes the resulting list to the test servers.

During the test period, every DNS resolver was tested each day using the complete daily domain list. However, the order of domain names was independently randomized for each resolver. Each DNS resolver was tested by one test server per day, with the assignment of DNS resolvers to test servers changing daily based on random allocation by the evaluation server.

The evaluation server also hosts a database used for storing test results and gets the log files from the authoritative name servers at the end of every day for further processing. These log files contain every DNS request that the authoritative name servers received. The evaluation server identifies all requests associated with the tests in the log files and inserts them into the database.

Authoritative Name Server

Both authoritative name servers are identical in terms of operating system, hardware resources, and running software. Logging is enabled and each DNS request received is recorded in a log file. Each authoritative name server is responsible for a domain.

The first server is authoritative for the domain prtest.ovh. The domain is not DNSSEC-signed and includes the wildcard record *.prtest.ovh to respond to queries for all subdomains. The TTL value is set to the maximum allowed duration of 604,800 seconds, i.e. 7 days.

The second server is authoritative for the domain name dnssec-check.ovh. Its configuration and function are described in section 4.5.1.

At the end of each day, the log files containing all received DNS requests are transmitted to the evaluation server for processing and analysis.

Test Server

Both test servers are identical in terms of operating system, hardware resources, and running software. They are hosted in the same data center and operate within the same AS. However, it cannot be guaranteed that network bandwidth and server performance remain identical at all times.

Each test server initiates the daily testing procedure by randomizing the order of the domain name list assigned for that day. For every assigned DNS resolver, the server starts two stub DNS resolvers: one for sending DNS requests directly and another for sending them through the Tor network. The test server then issues DNS requests for all test domain names via both stub DNS resolvers. Stub DNS resolvers for DoH[10], DoT[10], DNSCrypt[11], and DNS TCP[10] are implemented within containers. These containers listen on port 53 for conventional DNS requests, translate them into encrypted DNS requests, and forward them to the corresponding encrypted DNS resolver. DoH stub resolvers use Google's public DNS resolver for bootstrapping.

A dedicated container provides the Tor connection. It listens on port 53 to send DNS requests directly to exit relays and on an additional designated port to route TCP traffic through the Tor network to specified destinations. The Tor connection uses the default circuit creation algorithm.

The Tor container is restarted daily without retaining any history of previous Tor connections, thereby ensuring that entry guards do not persist across multiple days.

To manage server and network load, no more than 20 DNS resolvers are tested in parallel on each test server.

Figure 5.6 illustrates the structure of a test server.

The validity of public TLS certificates for DoH and DoT resolvers is verified several times per day, using the certificate authorities trusted by the Ubuntu operating system. Certificates provided by operators of DNSCrypt resolvers are validated by the stub DNS resolver software itself.

The test script employs the dig[12] command to send DNS requests. The command returns both the resolved IP addresses and the resolution time, measured in microseconds. This data is stored in the database on the evaluation server. If the resolution of a domain name fails, the test retries the DNS request up to four times.

[10] RouteDNS, https://github.com/folbricht/routedns
[11] dnscrypt-proxy, https://github.com/DNSCrypt/dnscrypt-proxy
[12] https://manpages.ubuntu.com/manpages/jammy/man1/dig.1.html

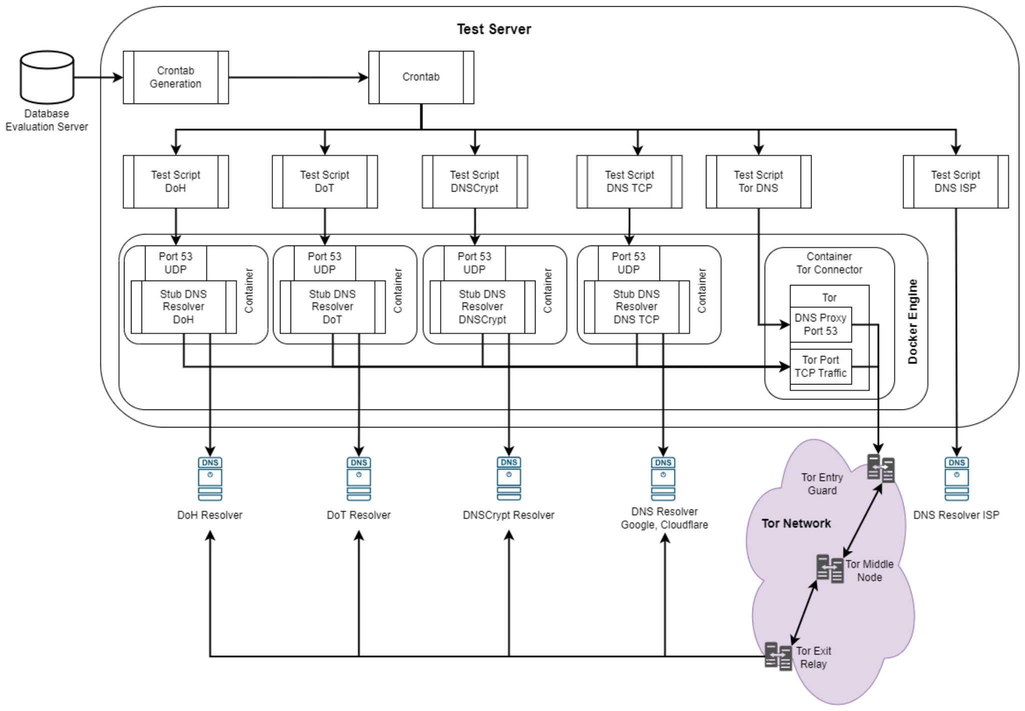

Figure 5.6: Structure of a test server

5.4.4 Tests and Results of Public Encrypted DNS Resolvers

Public encrypted DNS resolvers were tested over the period from November 24, 2024, to December 23, 2024.

All results reflect the observed behavior of the DNS resolvers during the specified testing period. AS-number data was obtained using the pyasn[13] library and updated on a daily basis.

DNS resolvers may resend DNS requests even after a query has been successfully answered and a response sent to the client - sometimes days or weeks after the original request was issued. This behavior was identified during the evaluation of DNS resolvers used by Tor exit relays, see section 4.5.2. Authoritative DNS resolvers may therefore receive such delayed DNS requests. To account for this extended exposure surface, all DNS requests received by the authoritative name server from the tested DNS resolvers were included up to January 31, 2025.

Each DNS resolver was queried both directly and via the Tor network for every test. The results were analyzed separately. However, if a test failed for either method, the entire test was considered failed.

Because no reference data was available to define acceptable thresholds for test completion, availability, and error rates, a Gaussian Mixture Model (GMM) was applied to identify statistical boundaries. This separation divided the measured data into two clusters: one representing acceptable values and another representing potentially problematic behavior.

[13]https://github.com/hadiasghari/pyasn

Tested DNS Resolvers

In addition to the encrypted DNS resolvers found in section 5.4.2, the following DNS resolvers were included in the tests as reference:

- The DNS resolver of the ISP hosting the test servers.
- Google's public DNS resolver, tested both directly and via the Tor network.
- Cloudflare's public DNS resolver, tested both directly and via the Tor network.
- Tor's conventional DNS resolver, i.e. resolution performed by exit relays.

The 334 encrypted DNS resolvers are referred to as *DNS resolver candidates*, while the four DNS resolvers listed above serve as *reference DNS resolvers*.

Test Completion

The total number of DNS test requests sent per DNS resolver during the test period was 154,830. However, not all DNS resolver candidates completed all tests, as some exhibited slow response times or a high number of DNS request timeouts. This resulted in planned DNS requests not being sent, as opposed to issues with availability, where DNS requests were sent but no responses were received.

The test completion rate was used to assess the validity of the results obtained in the subsequent analysis. By applying a Gaussian Mixture Model (GMM), the threshold for acceptable completion was determined to be 99.25%. DNS resolvers that completed fewer than 99.25% of all test DNS requests were considered below the test requirement.

All reference DNS resolvers successfully completed all DNS test requests.

Among the DNS resolver candidates, nine DoH resolvers, nine DNSCrypt resolvers, and one DoT resolver failed to reach the completion threshold.

No significant differences in completion rates were observed between DNS requests sent directly and those sent via the Tor network.

Availability

The availability was assessed by comparing the number of DNS responses received to the number of DNS test requests sent. Using a GMM applied to the timeout rates of all DNS resolver candidates, the threshold for the acceptable rate of timed-out DNS requests was determined to be 0.43%.

DNS resolver candidates with a timeout rate exceeding this value were classified as not meeting the availability requirement. This applied to eleven DoH resolvers, ten DNSCrypt resolvers, and four DoT resolvers.

Three DNS resolvers exhibited notably more timeouts when DNS requests were sent via the Tor network.

Error Rate

The error rate was assessed based on the proportion of DNS responses returning the codes SERVFAIL or REFUSED, excluding DNSSEC test requests.

DNS resolvers exhibiting more than 6.21% SERVFAIL responses or 6.55% REFUSED responses were classified as not meeting the requirement for a low error rate. These thresholds were determined using a GMM applied to the data of the corresponding response code rates of all DNS resolver candidates.

35 DoH resolvers and two DoT resolvers showed a higher error rate than these limits. No significant differences were observed between DNS requests sent directly and DNS requests sent via the Tor network.

Valid TLS Certificates

The TLS certificates of DoH and DoT resolver candidates were verified ten times per day. Certificates were accepted if they were issued by certificate authorities trusted by the Ubuntu Linux operating system and if the connection used TLS version 1.2 or 1.3.

The configuration of DNSCrypt resolvers included a public key which is used by the stub DNSCrypt resolver software to verify the certificate presented by the DNSCrypt resolver. Certificates of DNSCrypt resolver candidates that successfully returned DNS responses were considered valid.

Two categories of errors were observed: connection errors and certificate errors. These occurred for 45 DoH and four DoT resolver candidates, see Table 5.1.

The different certificate errors for each encryption protocol are listed in Table 5.2.

[83] found 25% of DoT resolvers they tested to present invalid TLS certificates. The high number of invalid TLS certificates compared to the results in this evaluation is probably due to the selection process of DNS resolver candidates. In that study, DoT resolvers were discovered by an Internet scan for open DoT ports.

DNSSEC

A DNS resolver was considered to validate DNSSEC correctly if none of the three incorrectly DNSSEC-signed domain names was resolved and if the re-

Table 5.1: Certificate check error categories per encryption protocol

DNS Resolver Type	Error	Number of DNS Resolvers
DoH	Certificate error	6
DoT	Certificate error	6
DoH	Certificate check connection error	39
Dot	Certificate check connection error	3

Table 5.2: Certificate errors per encryption protocol

DNS Resolver Type	Certificate Validation Error	Number of DNS Resolvers
DoH	Certificate expired	3
DoH	Hostname mismatch	2
DoH	Certificate self-signed	1
DoT	Certificate reading error	1

sponse code was SERVFAIL, which is the expected response code for domain names with invalid DNSSEC signatures [91]. If a domain name was resolved and an IP address was returned, the DNS resolver was classified as not validating DNSSEC.

During the test period, 28 DNS resolver candidates did not send a sufficient number of DNSSEC test requests. Two DNS resolver candidates never validated DNSSEC, and one DoH resolver resolved a single DNSSEC test request. These results are summarized in Table 5.3.

No significant differences were observed between DNSSEC requests sent directly and those sent via the Tor network.

Blocking Domain Names

In this test, domain name blocking is defined as a DNS resolver returning an incorrect or no IP address for valid domain names. To identify such behavior, the domain names from the blocklists described in section 5.4.3 were used. It was expected that a blocking DNS resolver blocks at least a subset of these domain

Table 5.3: DNSSEC validation failing DNS resolvers per encryption protocol

DNS Resolver Type	Error	Number of DNS Resolvers
DoH	No DNSSEC statement possible	20
DoT	No DNSSEC statement possible	1
DNSCrypt	No DNSSEC statement possible	7
DoH	DNSSEC validated between 90% and less than 100%	1
DoH	DNSSEC validated less than 90%	1
DoT	DNSSEC validated less than 90%	1

names. However, it is possible that a DNS resolver resolved all domain names used in this test but would have blocked other domain names.

DNS resolvers were analyzed for the following blocking behaviors:

- *Returning non-globally routable IP addresses*: Four DoH resolvers were found to block domain names by returning the IP addresses 0.0.0.0 or 127.0.0.1.
- *Returning false IP addresses*: Two DoH resolvers returned the IP address 146.112.61.108 for several likely blocked domain names, which is known to be used for DNS blocking[14].
- *Returning the response code NOERROR with an empty answer section*: No DNS resolvers were found to block domain names using this approach.
- *Returning the response code NXDOMAIN*: One DoH resolver was found to block domain names by returning the NXDOMAIN response code.

In total, blocking domain names was detected on seven DoH resolver candidates, as listed in Table 5.4. No significant differences in blocking behavior were observed between DNS requests sent directly and those sent via the Tor network.

ECS Information

In this test, DNS requests received by the authoritative name server of the domain prtest.ovh were analyzed for included ECS network information and were grouped into three categories.

The first category contained DNS requests that did not include ECS data. The network 0.0.0.0/0 and other non-globally routable IP address ranges were also counted as having no ECS data.

The second category contained DNS requests that included the network address corresponding to the client's IP address. The client in this case was either the test server for DNS requests sent directly or the exit relay for DNS requests sent via the Tor network.

The third category contained all DNS requests that included ECS information that did not correspond to the client's IP address.

[14]https://docs.umbrella.com/umbrella-user-guide/docs/block-page-ip-addresses

Table 5.4: Domain name blocking DNS resolvers per blocking type and encryption protocol

DNS Resolver Type	Blocking Type	Number of DNS Resolvers
DoH	Wrong IP address resolved	2
DoH	Non globally routable IP address resolved	4
DoH	Response code NXDOMAIN returned	1

Table 5.5: DNS resolvers sending ECS data per encryption protocol

DNS Resolver Type	ECS Information	Number of DNS Resolvers
DoH	No ECS statement possible	20
DoT	No ECS statement possible	1
DNSCrypt	No ECS statement possible	7
DoH	Client IP in ECS network	7
DoT	Client IP in ECS network	1
DNSCrypt	Client IP in ECS network	0
DoH	ECS information not directly assignable to sender	15
DoT	ECS information not directly assignable to sender	4
DNSCrypt	ECS information not directly assignable to sender	1

No statement could be made for DNS resolvers that did not send a sufficient number of DNS requests to the authoritative name server.

Table 5.5 presents the number of DNS resolver candidates including ECS information, grouped by category and encryption protocol.

No different behavior in including ECS data was observed between DNS requests sent directly and those sent via the Tor network.

Among the reference DNS resolvers, Google's public DNS resolver consistently included the client's network information in the ECS data. Tor's conventional DNS resolution did occasionally include ECS data, which is expected due to using different exit relays and hence different DNS resolvers. The ISP's DNS resolver and Cloudflare's public DNS resolver did not send any ECS data.

No invalid ECS data, such as a subnet mask greater than /24 for IPv4 networks, observed in the evaluation of the Tor network in section 4.5.2, was detected.

QNAME Minimization

This test evaluated whether the DNS resolvers applied QNAME minimization at the subdomain level. The test used domain names following the pattern [subdomain2].[subdomain1].prtest.ovh, where [subdomain1] and [subdomain2] were unique for each DNS resolver and test DNS request and included a test ID. This ensured that caching could not occur and allowed unambiguous identification of the tested DNS resolver.

A DNS resolver was considered to apply QNAME minimization if it queries the authoritative name server for [subdomain1].prtest.ovh of any query type before querying the full domain name for query type A within a short time interval and using an IP address belonging to the same AS.

DNS resolver candidates that did not apply QNAME minimization for each DNS request sent to the authoritative name server are listed in Table 5.6.

No significant differences in QNAME minimization behavior were observed between DNS requests sent directly and those sent via the Tor network.

In this test, Google's public DNS resolver was found not to apply QNAME minimization. According to [92], Google's public DNS resolver was also classified as not applying QNAME minimization when tested using the queries "a.b.qnamemin-test.nlnetlabs.nl TXT" and "a.b.qnamemintest.net TXT", but was classified as applying QNAME minimization for the domain name "a.b.qnamemin-test.internet.nl TXT". Google explained to NLnet Labs, the operator of the domain nlnetlabs.nl, that its public DNS resolver applies QNAME minimization only at the root and TLD levels, and that an exception has been added for the test domain "a.b.qnamemin-test.internet.nl TXT".

Performance

The performance of DNS resolvers in this thesis is defined as the time required to resolve a DNS request, based on the output of the dig command. DNS requests

Table 5.6: DNS resolvers not applying QNAME minimization for all requests per encryption protocol

DNS Resolver Type	QNAME Minimization	Number of DNS Resolvers
DoH	No QMIN statement possible	20
DoT	No QMIN statement possible	1
DNSCrypt	No QMIN statement possible	7
DoH	QMIN not applied	11
DoT	QMIN not applied	1
DoH	QMIN rarely applied (> 0% and < 25%)	1
DoH	QMIN sometimes applied (>= 25% and < 90%)	7
DoT	QMIN sometimes applied (>= 25% and < 90%)	1
DoH	QMIN mostly applied (>= 90% and < 100%)	15
DoT	QMIN mostly applied (>= 90% and < 100%)	5
DNSCrypt	QMIN mostly applied (>= 90% and < 100%)	23

Table 5.7: Average DNS resolution time per DNS resolver category and tested domains

DNS Resolver	**Average Time needed for DNS Resolution in Milliseconds**		
ISP DNS	2	29	65
Google DNS	12	19	55
Google DNS TCP via Tor	145	170	233
Cloudflare DNS	7	7	82
Cloudflare DNS TCP via Tor	152	137	242
Tor DNS (various exit relays)	146	212	233
DoH	139	185	300
DoH via Tor	280	354	516
DoT	67	117	155
DoT via Tor	210	274	345
DNSCrypt	129	177	316
DNSCrypt via Tor	460	531	707

that resulted in timeouts were excluded in this test.

DNS requests sent through the Tor network are influenced by the performance of the underlying Tor circuit. All tests used the same circuit, and it is assumed that, over the entire test period, variations in circuit performance affected all DNS resolvers equally on average.

Further research is required to determine the maximum tolerable DNS resolution time from a user perspective. In this thesis, no fixed time limit is defined. Instead, DNS resolver candidates were ranked according to their average response time, which may serve as a selection criterion when only a single DNS resolver from a group of suitable candidates can be chosen to ensure DNS resolver independence.

The performance comparison includes three categories of domain names. Domain names under *.prtest.ovh are unique per test and therefore not cached. Domain names among the top 1,000 of the Majestic Million dataset are expected to be cached to a high extent and to have their corresponding authoritative name servers located at various geographical locations. The domain names of the root name servers are assumed to be cached by all DNS resolvers.

Table 5.7 presents the average resolution times grouped by domain category, DNS resolver type, and whether the DNS request was sent directly or via the Tor network.

Figure 7 and figure 8 compare the performance between different DNS encryption protocols for the root name server domain and the *.prtest.ovh domain.

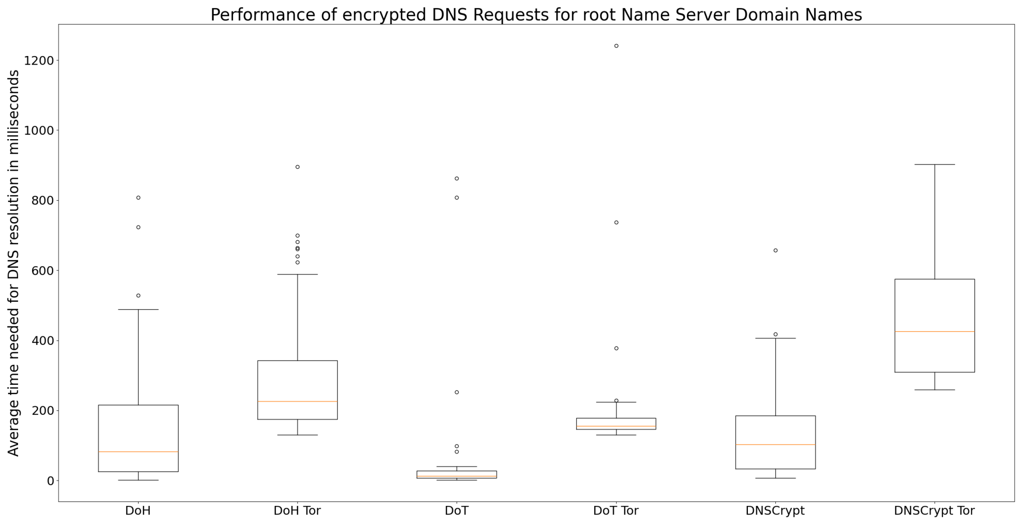

Figure 5.7: Performance of DNS requests for root name server domain names

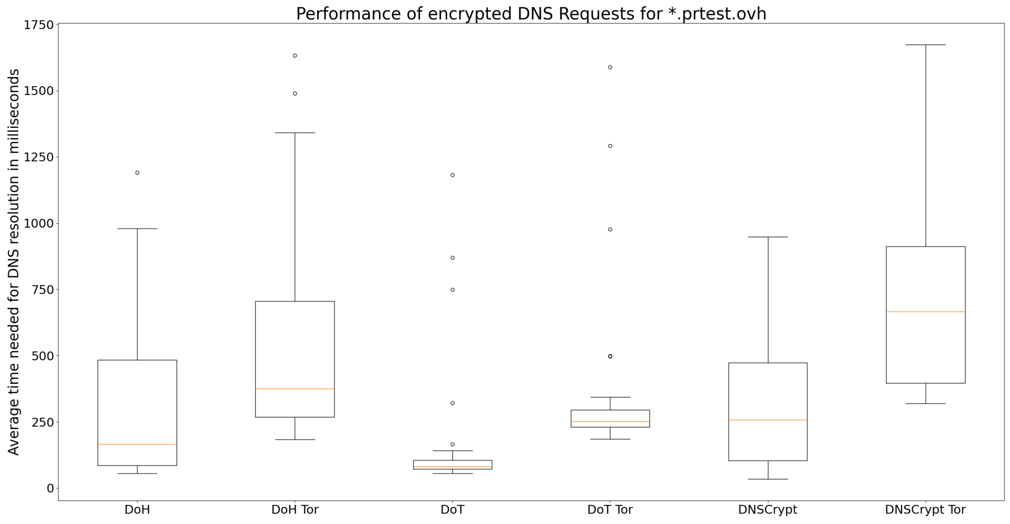

Figure 5.8: Performance of DNS requests for *.prtest.ovh domain names

As expected, encryption protocols and the Tor network introduce additional latency to the resolution process. Among the tested DNS resolvers, DoT resolvers exhibited the fastest average response times, followed by DoH resolvers, while DNSCrypt resolvers were the slowest.

5.4.5 Summary of public encrypted DNS Resolver Tests

Sending requests through the Tor network had no significant impact on the domain name blocking or the error rate. Higher response times were experienced which was to be expected.

The only significant difference observed between DNS requests sent directly and those sent via the Tor network was in performance. DNS resolvers that showed the following results were considered to satisfy the defined requirements:

- Achieved a test completion rate in a range of acceptable values.
- Demonstrated availability in a range of acceptable values.
- Exhibited an error rate in a range of acceptable values.
- Consistently presented a valid certificate.
- Validated DNSSEC for more than 90% of test requests.
- Did not block domain names.
- Did not include ECS information related to the client.
- Applied QNAME minimization for more than 90% of the test requests.

Based on these criteria, 81 of 161 DoH resolvers, 30 of 40 DoT resolvers, and 123 of 133 DNSCrypt resolvers met the requirements. However, 108 of the compliant DNSCrypt resolvers were operated by only two organizations, which explains the high overall compliance among DNSCrypt resolvers.

Among the reference DNS resolvers, the ISP's and Cloudflare's DNS resolver met the defined criteria, whereas Google's public DNS resolver and Tor's conventional DNS resolution did not. The results are summarized in Table 5.8.

5.4.6 Independence of Encrypted DNS Resolvers

This section presents the evaluation of independence for DNS resolver candidates that met the requirements.

In [89], the encrypted DNS infrastructure is conceptually divided into two categories: ingress servers, which act as publicly accessible encrypted DNS resolvers serving as an interface to the client, and egress servers, which perform

Table 5.8: Summary of DNS resolver rest

	DNS Resolvers complying with Criteria		
DNS Resolver Type	**Number of DNS Resolvers**	**Percentage of DNS Resolvers**	**Different Organizations**
DNS	2	50.00%	2
DoH	81	50.31%	35
DoT	30	75.00%	20
DNSCrypt	123	92.48%	13
All	236	69.82%	54
	DNS Resolvers not complying with Criteria		
DNS Resolver Type	**Number of DNS Resolvers**	**Percentage of DNS Resolvers**	**Different Organizations**
DNS	2	50.00%	2*
DoH	80	49.69%	50
DoT	10	25.00%	9
DNSCrypt	10	7.52%	4
All	102	30.18%	57*

* All DNS resolvers used by exit relays are counted as one organization

DNS resolution by querying the authoritative name servers. This concept was also used here to provide a separation of servers involved in the DNS resolution process. Accordingly, independence was assessed on both the ingress and the egress side, as well as for their interconnections.

Figure 5.9 illustrates the structure of an encrypted DNS infrastructure. DNS resolver candidates (RCs) on the left represent the ingress side and use DNS resolvers on the right to perform DNS resolution toward the authoritative name servers. Between the ingress and the egress side, one or more forwarding DNS resolvers may exist, which could not be evaluated by the conducted tests. On each side, DNS resolvers were grouped into sets, with each set containing DNS resolvers found to belong to the same organization or provider.

On the ingress side, the organization was determined primarily by the domain name in the certificate. Additionally, DNS resolvers that shared an IP address within the same /24 subnet were assigned to the same organization. Using these classification criteria, 234 DNS resolver candidates were grouped into 52 independent ingress sets.

On the egress side, all identifiable test DNS requests received by the authoritative name server of prtest.ovh were analyzed. Each learned IP address was associated with its corresponding AS number and the DNS resolver candidate that initiated this DNS request. To identify relationships between egress DNS resolvers, their IP addresses were merged into egress sets as follows:

All egress-side IP addresses used by an ingress-side DNS resolver are merged with those used by other ingress-side DNS resolvers whenever they share at least one common IP address used by egress-side DNS resolvers. This merging process is applied recursively, so that any overlap in IP addresses links the involved egress-side DNS resolvers into a single egress set. However, IP addresses belonging to different AS numbers are kept in separate egress sets, even if they are used by the same ingress-side DNS resolver. As a result, a single ingress-side DNS resolver may be associated with multiple egress sets corresponding to different ASes. Conversely, distinct egress sets may contain IP addresses from the same AS, provided that these IP addresses are not shared by any single ingress-side DNS resolver.

Overall, 2,703 IP addresses from 157 different ASes were grouped into 232 egress sets.

A single DNS resolver candidate may use multiple egress sets (e.g. RC_3 in Figure 5.9), and conversely, one egress set may be shared by multiple DNS resolver candidates (e.g. ES_1 in Figure 5.9).

The number of independent DNS resolvers is constrained by the total number of ingress sets. Furthermore, only one DNS resolver can be selected from each ingress set. To determine this DNS resolver, all DNS resolvers within an ingress set are prioritized according to the number of corresponding egress sets they use. DNS resolvers using fewer egress sets are given a higher priority, as this minimizes exposure on the egress side and preserves more egress sets for other ingress sets. If multiple DNS resolvers have the same lowest number of egress sets, the average DNS resolution time remains as the deciding criterion, with the fastest DNS resolver being selected.

The ingress set and all egress sets associated with the highest-priority DNS resolver are then excluded from selection for lower-priority DNS resolver candidates. This ensures that only the fastest DNS resolver per ingress set is selected and that each egress set is used by only one DNS resolver.

Following this procedure, 48 of the 52 ingress sets could be assigned to egress sets without reuse. The remaining four ingress sets could not be used due to conflicting egress sets.

In total, 48 DNS resolvers from distinct ingress sets were mapped to 70 egress sets.

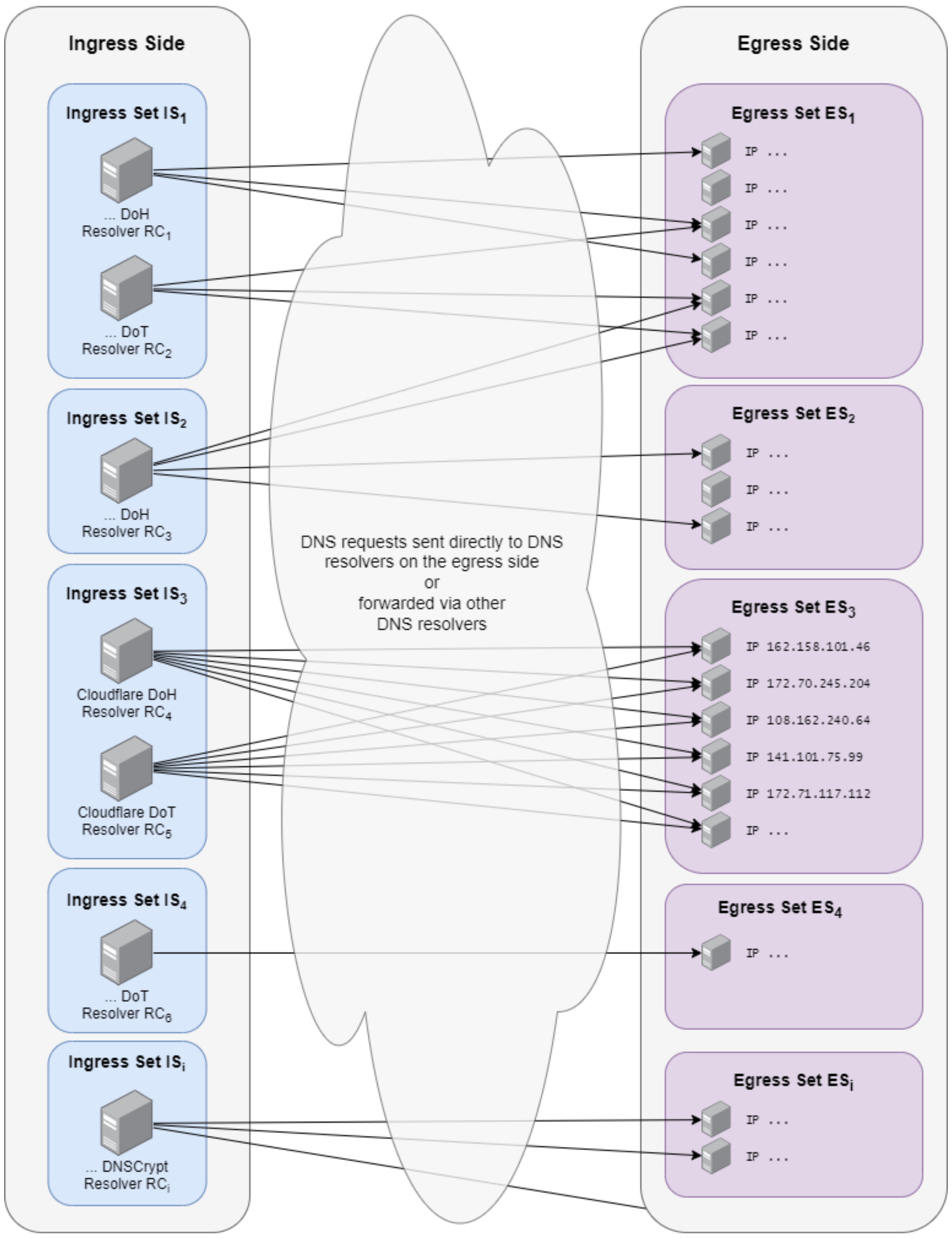

Figure 5.9: Schema of encrypted DNS infrastructure

Chapter 6

Implementation and Evaluation

6.1 Consolidated Improvements

This section consolidates the proposed improvements discussed in chapter 5. Clients send DNS requests to an application running on their device or on a server within a local network. The application listens for conventional unencrypted DNS requests and supports DNS blocking based on a domain name list specified by the operator. Additionally, DNS entries for hostnames available only within the local network can be defined in a hosts file and answered directly without further processing.

Query names for the Special-Use Top-Level Domain .onion should be blocked by the application, as onion services do not use the global DNS. Such requests must instead be handled by Tor software and routed within the Tor network. [93]

The operator may enable caching for all received DNS requests. DNS requests that are neither blocked nor answered from local entries or the cache are translated into encrypted DNS requests. Based on the distribution algorithm and the list of available encrypted DNS resolvers, an encrypted DNS resolver is selected. The DNS request is then transmitted through one of several available Tor circuits to the selected DNS resolver. If a bootstrap DNS request is required, it is sent conventionally to an exit relay for resolution.

Tor circuits should ideally be established at application startup and kept open to avoid the circuit-creation overhead for each DNS request. The application determines the IP addresses and ASes of the client, the entry guard, and the configured DNS resolvers, and DNS resolvers located in the same AS as the client or the entry guard are excluded from selection.

For each DNS request, the public certificate of the selected encrypted DNS resolver is validated. Failing encrypted DNS resolvers are detected and temporarily excluded. DNS resolvers that fail repeatedly are permanently removed from selection.

Figure 6.1 illustrates the process of a DNS request entering as a conventional, unencrypted DNS request and leaving as encrypted DNS request directed to one of multiple encrypted DNS resolvers.

Clients configure their operating system or applications to send DNS requests to this application. Unencrypted DNS traffic remains within the user's operating system or a trusted local network, and no additional software is required on the client side.

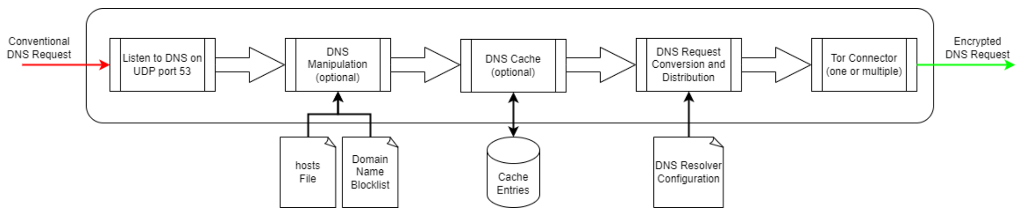
Conventional DNS Request
Listen to DNS on UDP port 53
DNS Manipulation (optional)
DNS Cache (optional)
DNS Request Conversion and Distribution
Tor Connector (one or multiple)
Encrypted DNS Request
hosts File
Domain Name Blocklist
Cache Entries
DNS Resolver Configuration

Figure 6.1: Process flow of a DNS requests

The operator of the application is responsible for selecting suitable encrypted DNS resolvers and maintaining the list of encrypted DNS resolvers. In the following, this application combining the proposed improvements is referred to as the *Private DNS Resolver*.

6.2 Use Cases

The following use cases are defined based on who operates the *Private DNS Resolver*:

- *User*: The user runs the *Private DNS Resolver* locally on their device and configures the operating system or applications to use it for DNS resolution.
- *Network operator*: A network operator hosts the *Private DNS Resolver* on a server within a local network and uses it to provide DNS resolution to clients on that network.
- *Exit relay operator*: An exit relay operator runs the *Private DNS Resolver* and uses it to resolve conventional DNS requests received from Tor clients on its connected circuits.

6.2.1 User-Operated Private DNS Resolver

The *Private DNS Resolver* is operated by the user on their device. DNS requests originating on their operating system are directed to this application. The user defines the list of encrypted DNS resolvers, blocked domain names, and local host entries, and defines the decision on enabling DNS caching.

DNS requests leaving the client's device are encrypted both by the encrypted DNS protocol and Tor's circuit-level encryption. Exit relays can observe which encrypted DNS resolvers are contacted but cannot assess the content of the DNS requests.

If the user accesses the web over Tor, DNS requests for this web traffic can be sent to the *Private DNS Resolver*. These DNS requests use a separate Tor circuit and likely a different exit relay than the web traffic. All query types supported by the selected encrypted DNS resolvers are supported. Proper configuration is required to ensure that these DNS requests are routed exclusively through the Tor network. Leaking DNS requests in that scenario would enable deanonymization of the client.

Figure 6.2 illustrates the architecture of a client running the *Private DNS Resolver* on their device.

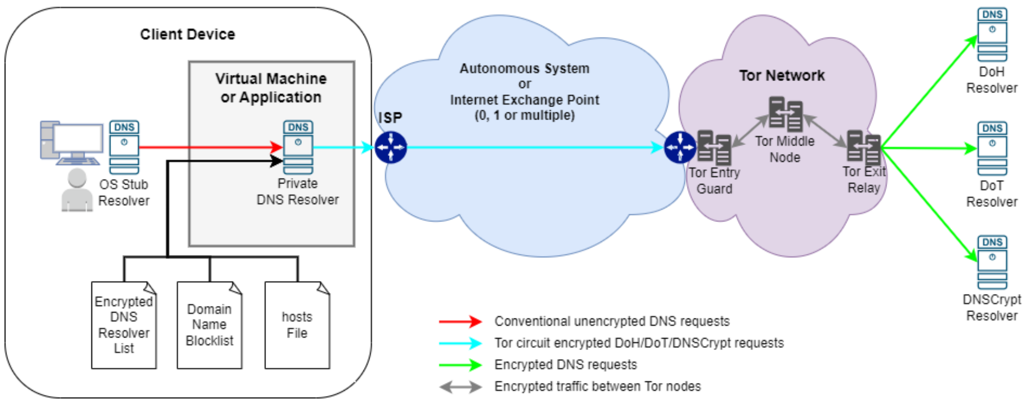
Client Device
Virtual Machine or Application
OS Stub Resolver
Private DNS Resolver
Encrypted DNS Resolver List
Domain Name Blocklist
hosts File
ISP
Autonomous System or Internet Exchange Point (0, 1 or multiple)
Tor Network
Tor Entry Guard
Tor Middle Node
Tor Exit Relay
DoH Resolver
DoT Resolver
DNSCrypt Resolver
Conventional unencrypted DNS requests
Tor circuit encrypted DoH/DoT/DNSCrypt requests
Encrypted DNS requests
Encrypted traffic between Tor nodes

Figure 6.2: Architecture of a user operated use case

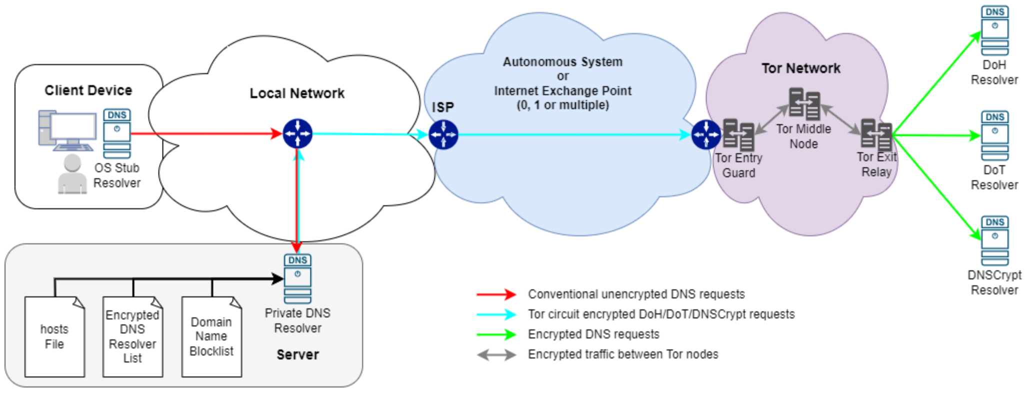

Figure 6.3: Architecture of network operator use case

6.2.2 Network Operator-Provided Private DNS Resolver

The *Private DNS Resolver* is hosted on a server within a local network by the network operator. Clients on this network may be configured, either manually or via DHCP, to direct their DNS requests to this server.

The network operator defines the list of encrypted DNS resolvers, blocked domain names, and local host entries, and determines whether DNS caching is enabled. DNS traffic may be logged, and information about blocked domain names or domain categories can be made available to clients through policy statements. DNS requests that require external DNS resolution are distributed through the Tor network to encrypted DNS resolvers.

The local network and its operator must be trusted, as DNS traffic within the network is transmitted unencrypted. Clients outside the local network may connect via a virtual private network to access the *Private DNS Resolver*.

Clients must not send DNS requests to a *Private DNS Resolver* provided by their network operator when accessing the web over Tor, as the network operator could observe these DNS requests, and DNS traffic cannot be assumed to be routed through the Tor network.

Figure 6.3 illustrates the architecture of a network operator providing the *Private DNS Resolver*.

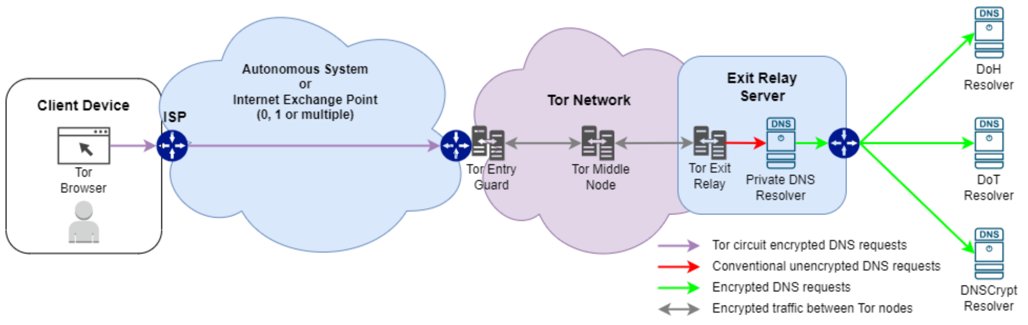

Figure 6.4: Architecture of exit relay operator use case

6.2.3 Exit Relay Operator-Provided Private DNS Resolver

The exit relay operator runs the *Private DNS Resolver* using the exit relay's IP address and employs it for external DNS resolution.

DNS requests from Tor clients that are sent conventionally over a Tor circuit for DNS resolution by the exit relay are encrypted by the Tor circuit encryption up to the exit relay. There, they are received as unencrypted DNS requests, and the Tor's default DNS resolution process is applied, which includes DNS caching and case randomization by default. The exit relay limits query types to A, AAAA, and PTR, can observe the DNS requests of Tor clients, and may determine which encrypted DNS resolvers are used.

DNS requests that cannot be answered from the exit relay's cache are forwarded to the *Private DNS Resolver* for external DNS resolution. In this use case, the *Private DNS Resolver* does apply caching and forwards DNS requests directly to encrypted DNS resolvers, as the original sender's IP address is no longer present in these DNS requests.

Figure 6.4 illustrates the architecture of an exit relay providing the *Private DNS Resolver*.

6.3 Proof-of-Concept (PoC) Implementation

The proof-of-concept implementation of the *Private DNS Resolver* runs on an Ubuntu 24.04 server inside a virtual machine. Docker is used to containerize components and to place them on an internal, non-exposed network.

6.3.1 Docker Images

The following Docker images are used:

- *folbricht/routedns*[1]: Runs RouteDNS v0.1.118, which forms the core of the PoC. It supports domain name blocking, local host entries, DNS caching, translation of conventional DNS into DoH and DoT, and DNS request distribution.

[1]https://hub.docker.com/r/folbricht/routedns

- *torconnector*: Built from Dockerfile Listing A.1. Runs Tor v0.4.8.19. It establishes Tor circuits and forwards TCP streams and UDP DNS requests to exit relays via dedicated ports.
- *dnscrypt*: Built from Dockerfile Listing A.2. Runs dnscrypt-proxy[2] v2.1.14. RouteDNS does not natively support DNSCrypt, so a separate container is started per DNSCrypt resolver. Each container is configured with the resolver's stamp and the *torconnector*'s IP address so that DNSCrypt traffic is also routed through the Tor network.

6.3.2 Data Flow and RouteDNS Configuration

Conventional DNS requests that must be handled by the *Private DNS Resolver* are forwarded to the VM and then to *RouteDNS*, which listens on UDP and TCP port 53. The configuration file Listing A.5 specifies:

- *Listening ports*: UDP and TCP port 53.
- *Domain name blocking*: Listing A.3 contains domain names to block and is loaded at startup, and reloaded every 86,400 seconds (i.e. 24 hours).
- *Local host entries*: Listing A.4 contains local host entries and is loaded at startup, and reloaded every 86,400 seconds (i.e. 24 hours).
- *DNS cache*: Cache capacity: 4,096 entries. TTL values greater than 86,400 seconds are truncated to 86,400 seconds (i.e. 24 hours).
- *Distribution algorithm*: Random selection among available DNS resolvers.
- *DoH and DoT resolvers*: The configuration of DoH and DoT resolvers is included directly in the *RouteDNS* configuration file. DoH resolvers use the *torconnector* as bootstrap DNS resolver. For DoT resolvers the expected TLS server name is specified for certificate validation.
- *DNSCrypt resolvers*: Each DNSCrypt resolver is defined as a DNS resolver in *RouteDNS* and mapped to its dedicated *dnscrypt* container. *RouteDNS* sends a conventional DNS request to that container, which converts it to DNSCrypt and forwards it through *torconnector*. An example configuration and the stamps of the selected DNSCrypt resolvers is presented in Listing A.6 and Listing A.7.

6.3.3 Operational Behavior

RouteDNS receives incoming DNS requests, applies domain name blocking, serves local host entries, consults the cache, and forwards the DNS request on cache misses:

- as DoH/DoT request via *torconnector* directly to the selected encrypted DNS resolver, or
- as a conventional DNS requests to the corresponding *dnscrypt* container, which sends them as DNSCrypt requests via *torconnector*.

[2]https://github.com/DNSCrypt/dnscrypt-proxy

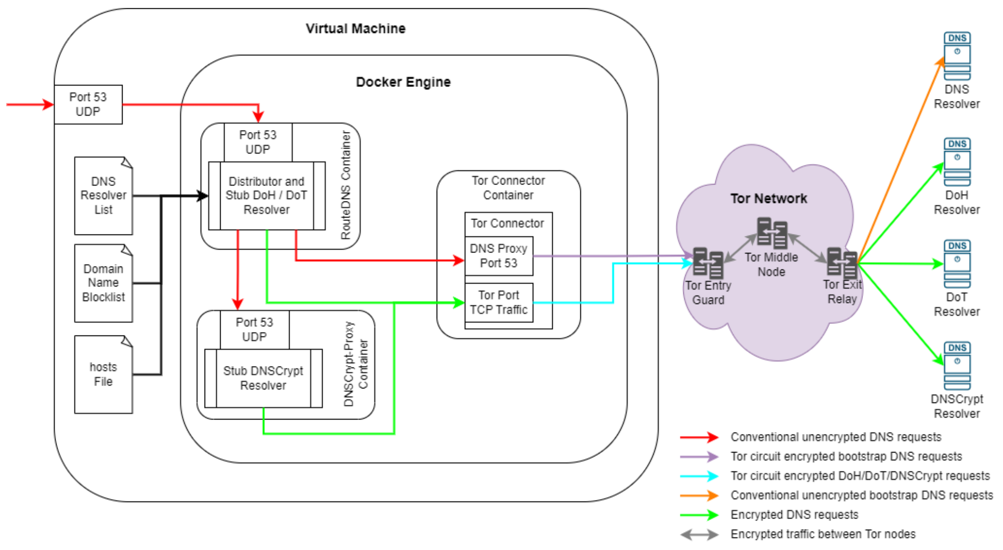

Figure 6.5: Proof-of-concept system architecture

Bootstrap DNS requests required to establish connections to DoH resolvers are sent as conventional DNS requests directly to the *torconnector*.

Figure 6.5 illustrates the architecture: a conventional DNS request enters the system, is processed by the *Private DNS Resolver*, and leaves as an encrypted DNS request toward one of multiple encrypted DNS resolvers.

The containers are started in the following order to ensure proper initialization and connectivity (see Listing A.8):

1. *torconnector*
2. *DNSCrypt resolvers*
3. *RouteDNS*

6.4 Evaluation

6.4.1 Practical Evaluation

The PoC implementation of the Private DNS Resolver was started, and DNS requests for various test purposes were issued using the dig[3] command directed to the application.

Resolution Times and Errors

A total of 10,000 DNS requests for test domain names in the form pdr[number].prtest.ovh and isp[number].prtest.ovh were sent to the *Private DNS Resolver* and to the ISP's DNS resolver of the test server, respectively. The variable [number] ranges from 0 to 9,999, ensuring that each domain name was unique and preventing caching by the application and external DNS

[3]https://manpages.ubuntu.com/manpages/jammy/man1/dig.1.html

Table 6.1: DNS resolution times of *Private DNS Resolver* and ISP DNS resolver

	Response Code	**Number of DNS Requests**	**Average DNS Resolution Time**
`ISP DNS resolver`	`NOERROR`	10,000	25.74 ms
`Private DNS Resolver`	`NOERROR`	9,995	321.70 ms
	`SERVFAIL`	3	1231.33 ms
	`REFUSED`	2	398.00 ms

resolvers. As expected, the *Private DNS Resolver* introduced a measurable delay. Of the 10,000 DNS requests, three returned the response code SERVFAIL, and two returned REFUSED, corresponding to an overall error rate of 0.05%. The results are presented in Table 6.1.

DNSSEC Validation

To evaluate DNSSEC validation, 1,000 DNS requests were sent for test domain names in the form of pdr[number].invalidkey.dnssec-check.ovh and isp[number].invalidkey.dnssec-check.ovh to the *Private DNS Resolver* and to the ISP's DNS resolver, respectively. The domain was incorrectly DNSSEC-signed, and DNS resolution was therefore expected to fail for DNSSEC-validating DNS resolvers. The variable [number] ranged from 0 to 999, again ensuring uniqueness and preventing caching. Both DNS resolvers returned the expected response code SERVFAIL for all 1,000 test DNS requests, indicating proper DNSSEC validation for invalidly signed domains.

DNS Cache

Multiple DNS requests were issued for the domain name cachetest.prdns.ovh. The first DNS request required approximately 300 ms, while subsequent DNS requests for the same domain name were answered instantly, returning identical IP addresses and a resolution time of 0 ms. This confirms the effective use of a DNS cache by the *Private DNS Resolver*.

Domain Name Blocking

DNS requests for domain names included in the configured blocklist were not resolved to IP addresses. Instead, the application returned a DNS response with the response code NXDOMAIN, confirming correct blocking functionality.

Local Host Entries

DNS requests for domain names listed in the local hosts file were resolved according to their defined IP addresses, verifying the proper functioning of custom host mappings.

6.4.2 Remaining Threats

- *Blocking of encrypted DNS traffic*: Encrypted DNS traffic may be detected and intentionally blocked by intermediary entities, such as network or AS operators.
- *ECS information included in DNS requests*: The network address of the client or server running the Private DNS Resolver must never be included as ECS network information in outgoing DNS requests. While this can be controlled locally by the Private DNS Resolver, external DNS resolvers might still append ECS data, potentially exposing the network address of the exit relay.
- *Detection of selected DNS resolvers*: External entities may be able to identify the encrypted DNS resolvers used by the Private DNS Resolver. For instance, websites could request numerous embedded resources under a controlled domain and observe which DNS resolvers query their name server. The set of selected DNS resolvers should not be published, although it cannot be considered secret.
- *Information embedded in domain names*: Applications or websites may systematically encode identifying information within subdomains of a controlled domain. This enables the correlation of DNS resolvers, IP addresses used to access resources, and user-supplied data. Applications may further access system information, such as the operating system or the client's external IP address prior to the application of anonymization measures, such as routing traffic through the Tor network.
- *Leaking DNS requests*: The Private DNS Resolver must be configured correctly, and its operation should be verified through testing to ensure that no DNS requests intended to be routed through the Private DNS Resolver are transmitted otherwise.
- *Cache snooping*: Clients of the Private DNS Resolver, as well as external entities such as websites, may infer the presence of domain names in the cache by observing DNS response timing behavior. Introducing randomized delays for cache hits based on measured DNS resolution times may mitigate cache snooping. This requires further investigation.

Chapter 7

Conclusion and Outlook

The Domain Name System possesses inherent weaknesses that affect both privacy and security. The evaluation of DNS behavior on the Tor network revealed that a considerable number of exit relay operators do not fully adhere to the Tor Project's recommendations. Compliance with these recommendations would already mitigate DNS threats through existing DNS security enhancements.

The proposed improvements in this thesis apply available protocols for encrypted DNS to provide confidentiality and integrity for specific segments of the DNS resolution path. To address the issue of centralization introduced by these protocols, a different approach was presented: sharding DNS requests across multiple DNS resolvers. While this increases the number of parties involved in DNS resolution, it prevents any single entity from obtaining sufficient information to compromise user privacy. The analysis further identified a promising number of independent encrypted DNS resolvers that meet the defined criteria for privacy, security, and stability.

However, some challenges persist, and future work is proposed to further improve DNS privacy and security.

7.0.1 Future Work

While this thesis assessed the compliance and independence of all encrypted DNS resolvers identified through an online search, the total number of suitable DNS resolvers may still be insufficient. Future research should determine the minimum number of encrypted DNS resolvers required to effectively prevent DNS fingerprinting and user reidentification.

Egress sets may be reused and shared among multiple DNS resolvers belonging to independent ingress sets, allowing for the inclusion of a greater number of encrypted DNS resolvers. Further investigation is needed to evaluate whether the reuse of egress sets compromises independence and weakens privacy.

The current algorithm for selecting independent DNS resolvers prioritizes those with the smallest number of occupied egress sets. An improved selection algorithm might identify combinations that use a higher number of DNS resolvers while maintaining independence.

A user study should determine the maximum tolerable DNS resolution time from a user perspective across common use cases, and how this limit relates to response times of individual DNS resolvers. The results could serve as a basis for defining performance criteria for compliant DNS resolvers.

The DNS cache remains vulnerable to cache snooping attacks from adversaries on the local network, as well as remote adversaries capable of provoking DNS requests, such as websites loading embedded content. Possible mitigations, such as introducing randomized delays for cached domain names, should be further investigated and evaluated for implementation.

DNS traffic within local networks may be further protected. When a network operator deploys the Private DNS Resolver on a local server, clients could use encrypted DNS to transmit their DNS requests to it. This would require a stub DNS resolver on client devices and support for receiving encrypted DNS by the Private DNS Resolver.

Since the number and characteristics of encrypted DNS resolvers might change over time, maintaining an up-to-date list of compliant DNS resolvers requires continuous reevaluation. DNS resolvers that no longer meet privacy or security requirements should be removed, while newly identified DNS resolvers should be added. Each time the list is updated, DNS resolver independence should be reassessed, and the subset of DNS resolvers selected for active use should be reevaluated accordingly.

7.0.2 Remaining Challenges

Authoritative Name Servers

DNS requests from recursive DNS resolvers to root, top-level, and authoritative name servers remain unencrypted, as these servers currently do not support encrypted DNS protocols. Although QNAME minimization reduces the exposure of full domain names to some extent, it does not eliminate it entirely. DNSSEC provides integrity and authenticity for signed domain names, yet its adoption remains limited [72].

Unencrypted SNI in the TLS Handshake

While the measures proposed in this thesis enhance the privacy and security of DNS, domain names may still be exposed to intermediate entities during TLS-encrypted communications between the client and the resource. When establishing a TLS session, the client transmits the domain name in plaintext within the Server Name Indication (SNI) field of the TLS handshake. This issue is expected to be mitigated by the upcoming Encrypted Client Hello (ECH) extension, which encrypts the initial message exchanged during the TLS handshake, called ClientHello. However, as ECH is still in draft status, it may not be adopted widely - or at all. [10, 94, 95]

Bibliography

[1] Stéphane Bortzmeyer. 2015. DNS Privacy Considerations. RFC 7626. (August 2015). DOI: 10.17487/RFC7626. https://www.rfc-editor.org/info/rfc7626.

[2] Giovanni Schmid. 2021. Thirty years of DNS insecurity: Current issues and perspectives. *IEEE Communications Surveys & Tutorials*, 23, 4, 2429–2459.

[3] Christian Grothoff, Matthias Wachs, Monika Ermert, and Jacob Appelbaum. 2017. NSA's MORECOWBELL: knell for DNS. *Unpublished technical report*.

[4] Wouter B De Vries, Roland van Rijswijk-Deij, Pieter-Tjerk De Boer, and Aiko Pras. 2019. Passive observations of a large DNS service: 2.5 years in the life of Google. *IEEE transactions on network and service management*, 17, 1, 190–200.

[5] Spyridon Samonas and David Coss. 2014. The CIA strikes back: Redefining confidentiality, integrity and availability in security. *Journal of Information System Security*, 10, 3.

[6] Lee Kim. 2022. Cybersecurity: Ensuring confidentiality, integrity, and availability of information. In *Nursing Informatics: A Health Informatics, Interprofessional and Global Perspective*. Springer, pp. 391–410.

[7] ISO ISO. 2015. IEC/IEEE International Standard-Systems and software engineering–System life cycle processes. *ISO/IEC/IEEE 15288 First edition 2015–05–15, Technical report*. DOI: 10.1109/IEEESTD.2017.8016712.

[8] Cambridge University Press. 2025. Trust. Definition entry. Cambridge Dictionary. Retrieved 11/12/2025 from https://dictionary.cambridge.org/dictionary/english/trust.

[9] Donald E. Eastlake 3rd. 2011. Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066. (January 2011). DOI: 10.17487/RFC6066. https://www.rfc-editor.org/info/rfc6066.

[10] Eric Rescorla. 2018. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. (August 2018). DOI: 10.17487/RFC8446. https://www.rfc-editor.org/info/rfc8446.

[11] 1983. Domain names: Concepts and facilities. RFC 882. (November 1983). DOI: 10.17487/RFC0882. https://www.rfc-editor.org/info/rfc882.

[12] 1987. Domain names - concepts and facilities. RFC 1034. (November 1987). DOI: 10.17487/RFC1034. https://www.rfc-editor.org/info/rfc1034.

[13] 1987. Domain names - implementation and specification. RFC 1035. (November 1987). DOI: 10.17487/RFC1035. https://www.rfc-editor.org/info/rfc1035.

[14] Paul E. Hoffman and Kazunori Fujiwara. 2024. DNS Terminology. RFC 9499. (March 2024). DOI: 10.17487/RFC9499. https://www.rfc-editor.org/info/rfc9499.

[15] Vladimir Ksinant, Christian Huitema, Dr. Susan Thomson, and Mohsen Souissi. 2003. DNS Extensions to Support IP Version 6. RFC 3596. (October 2003). DOI: 10.17487/RFC3596. https://www.rfc-editor.org/info/rfc3596.

[16] Scott Rose, Matt Larson, Dan Massey, Rob Austein, and Roy Arends. 2005. Resource Records for the DNS Security Extensions. RFC 4034. (March 2005). DOI: 10.17487/RFC4034. https://www.rfc-editor.org/info/rfc4034.

[17] Carlo Contavalli, Wilmer van der Gaast, David C Lawrence, and Warren Kumari. 2016. Client Subnet in DNS Queries. RFC 7871. (May 2016). DOI: 10.17487/RFC7871. https://www.rfc-editor.org/info/rfc7871.

[18] Paul E. Hoffman, Andrew Sullivan, and Kazunori Fujiwara. 2019. DNS Terminology. RFC 8499. (January 2019). DOI: 10.17487/RFC8499. https://www.rfc-editor.org/info/rfc8499.

[19] Joao Luis Silva Damas, Michael Graff, and Paul A. Vixie. 2013. Extension Mechanisms for DNS (EDNS(0)). RFC 6891. (April 2013). DOI: 10.17487/RFC6891. https://www.rfc-editor.org/info/rfc6891.

[20] Scott Rose, Matt Larson, Dan Massey, Rob Austein, and Roy Arends. 2005. DNS Security Introduction and Requirements. RFC 4033. (March 2005). DOI: 10.17487/RFC4033. https://www.rfc-editor.org/info/rfc4033.

[21] Roy Arends, Geoffrey Sisson, David Blacka, and Ben Laurie. 2008. DNS Security (DNSSEC) Hashed Authenticated Denial of Existence. RFC 5155. (March 2008). DOI: 10.17487/RFC5155. https://www.rfc-editor.org/info/rfc5155.

[22] Jacob Schlyter. 2004. DNS Security (DNSSEC) NextSECure (NSEC) RDATA Format. RFC 3845. (August 2004). DOI: 10.17487/RFC3845. https://www.rfc-editor.org/info/rfc3845.

[23] Jan Včelák, Sharon Goldberg, Dimitrios Papadopoulos, Shumon Huque, and David C Lawrence. 2018. NSEC5, DNSSEC Authenticated Denial of Existence. Internet-Draft draft-vcelak-nsec5-08. Work in Progress. Internet Engineering Task Force, (December 2018). 35 pages. https://datatracker.ietf.org/doc/draft-vcelak-nsec5/08/.

[24] Stéphane Bortzmeyer, Ralph Dolmans, and Paul E. Hoffman. 2021. DNS Query Name Minimisation to Improve Privacy. RFC 9156. (November 2021). DOI: 10.17487/RFC9156. https://www.rfc-editor.org/info/rfc9156.

[25] Paul A. Vixie and David Dagon. 2008. Use of Bit 0x20 in DNS Labels to Improve Transaction Identity. Internet-Draft draft-vixie-dnsext-dns0x20-00. Work in Progress. Internet Engineering Task Force, (March 2008). 8 pages. https://datatracker.ietf.org/doc/draft-vixie-dnsext-dns0x20/00/.

[26] David Dagon, Manos Antonakakis, Paul Vixie, Tatuya Jinmei, and Wenke Lee. 2008. Increased DNS forgery resistance through 0x20-bit encoding: security via leet queries. In *Proceedings of the 15th ACM conference on Computer and communications security*, pp. 211–222.

[27] Paul E. Hoffman and Patrick McManus. 2018. DNS Queries over HTTPS (DoH). RFC 8484. (October 2018). DOI: 10.17487/RFC8484. https://www.rfc-editor.org/info/rfc8484.

[28] Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul E. Hoffman. 2016. Specification for DNS over Transport Layer Security (TLS). RFC 7858. (May 2016). DOI: 10.17487/RFC7858. https://www.rfc-editor.org/info/rfc7858.

[29] DNSCrypt Project. 2025. DNSCrypt Protocol Specification. Technical documentation. DNSCrypt Project. Retrieved 11/12/2025 from https://dnscrypt.info/protocol/.

[30] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-Generation Onion Router. Design Paper. Originally presented at the 13th USENIX Security Symposium (2004). The Tor Project, (August 2004). Retrieved 11/12/2025 from https://svn-archive.torproject.org/svn/projects/design-paper/tor-design.pdf.

[31] The Tor Project. 2025. Tor — Core Repository. GitLab source code repository. The Tor Project. Retrieved 11/12/2025 from https://gitlab.torproject.org/tpo/core/tor.

[32] The Tor Project. 2025. Types of Relays. Community documentation page. The Tor Project. Retrieved 11/12/2025 from https://community.torproject.org/relay/types-of-relays/.

[33] Kyle Hogan, Sacha Servan-Schreiber, Zachary Newman, Ben Weintraub, Cristina Nita-Rotaru, and Srinivas Devadas. 2022. Shortor: Improving tor network latency via multi-hop overlay routing. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, pp. 1933–1952.

[34] Whonix Project. 2025. Alternative DNS Resolver. Whonix documentation page. Whonix Project. Retrieved 11/12/2025 from https://www.whonix.org/wiki/Alternative_DNS_Resolver.

[35] The Tor Project. 2025. *tor(1) — The Second-Generation Onion Router*. Manual page. The Tor Project. Retrieved 11/12/2025 from https://manpages.org/tor.

[36] The Tor Project. 2025. Glossary. Support documentation page. The Tor Project. Retrieved 11/12/2025 from https://support.torproject.org/glossary/.

[37] The Tor Project. 2025. Arti — A complete rewrite of the C Tor codebase in Rust. Project overview. The Tor Project. Retrieved 11/12/2025 from https://tpo.pages.torproject.net/core/arti/.

[38] The Tor Project. 2025. The Tor Project — Privacy & Freedom Online. Official project website. The Tor Project. Retrieved 11/12/2025 from https://www.torproject.org/.

[39] The Tor Project. 2025. About — Tor Browser Manual. Tor Browser Manual documentation page. The Tor Project. Retrieved 11/12/2025 from https://tb-manual.torproject.org/about/.

[40] The Tor Project. 2025. Tor Browser – „Installation“ (de) – Getting started. Support documentation – German language version. The Tor Project. Retrieved 11/11/2025 from https://support.torproject.org/tor-browser/getting-started/installing/.

[41] Guardian Project. 2025. FAQs – Orbot. Frequently asked questions page. Guardian Project. Retrieved 11/12/2025 from https://orbot.app/en/faqs/.

[42] The Tor Project. 2025. Mobile Tor — Tor Browser Manual. Tor Browser Manual documentation page. The Tor Project. Retrieved 11/12/2025 from https://tb-manual.torproject.org/mobile-tor/.

[43] Onion Browser. 2025. Onion Browser — Super fast, super secure access to popular sites. Official product website. Retrieved 11/12/2025 from https://onionbrowser.com/.

[44] mtigas. 2016. Tor at the Heart: Onion Browser (and more iOS Tor). Blog post. The Tor Project. (December 2016). Retrieved 11/12/2025 from https://blog.torproject.org/tor-heart-onion-browser-and-more-ios-tor/.

[45] Tails Project. 2025. About – Tails OS. Official information page. Tails Project. Retrieved 11/12/2025 from https://tails.net/about/index.en.html.

[46] Whonix Project. 2025. Whonix - Overview. Project overview page. Whonix Project. Retrieved 11/12/2025 from https://www.whonix.org/wiki/About.

[47] Romain Fouchereau and *IDC Research*. 2023. 2023 Global DNS Threat Report: Augmenting Cyber Threat Intelligence. Technical report. InfoBrief sponsored by EfficientIP, IDC #EUR150930923. International Data Corporation (IDC), (August 2023). Retrieved 11/12/2025 from https://efficientip.com/wp-content/uploads/2023/09/IDC-2023-DNS-Threat-Report.pdf.

[48] JHC Van Heugten. 2018. Privacy analysis of DNS resolver solutions. *Master of System Network Engineering University of Amsterdam*, 1–17.

[49] Tim Wicinski. 2021. DNS Privacy Considerations. RFC 9076. (July 2021). DOI: 10.17487/RFC9076. https://www.rfc-editor.org/info/rfc9076.

[50] William B Norton. 2001. Internet service providers and peering. In *Proceedings of NANOG*. Volume 19, pp. 1–17.

[51] Roxana Radu and Michael Hausding. 2020. Consolidation in the DNS resolver market–how much, how fast, how dangerous? *Journal of Cyber Policy*, 5, 1, 46–64.

[52] Despoina Farmaki. 2021. The effectiveness of blocking injunctions against ISPs in respect of online copyright infringement in Europe: a comparative analysis from the UK, Greece and the Nordic countries. *Stockholm Intellectual Property Law Review*, 2, 6–17.

[53] Oliver Borchert, Kyehwan Lee, Kotikalapudi Sriram, Douglas Montgomery, Patrick Gleichmann, and Mehmet Adalier. 2021. BGP Secure Routing Extension (BGP-SRx): Reference Implementation and Test Tools for Emerging BGP Security Standards. NIST Technical Note TN 2060 TN 2060. Final version, published September 15 2021. National Institute of Standards and Technology, (September 2021). Retrieved 11/12/2025 from https://nvlpubs.nist.gov/nistpubs/TechnicalNotes/NIST.TN.2060.pdf.

[54] Kuai Xu, Zhenhai Duan, Zhi-Li Zhang, and Jaideep Chandrashekar. 2004. On properties of internet exchange points and their impact on as topology and relationship. In *International Conference on Research in Networking*. Springer, pp. 284–295.

[55] Roque Gagliano. 2010. IPv6 Deployment in Internet Exchange Points (IXPs). RFC 5963. (August 2010). DOI: 10.17487/RFC5963. https://www.rfc-editor.org/info/rfc5963.

[56] Stephen Farrell and Hannes Tschofenig. 2014. Pervasive Monitoring Is an Attack. RFC 7258. (May 2014). DOI: 10.17487/RFC7258. https://www.rfc-editor.org/info/rfc7258.

[57] Mahmoud Sammour, Burairah Hussin, Mohd Fairuz Iskandar Othman, Mohamed Doheir, Basel AlShaikhdeeb, and Mohammed Saad Talib. 2018. DNS tunneling: A review on features. *International Journal of Engineering & Technology*, 7, 3.20, 1–5.

[58] Jawad Ahmed, Hassan Habibi Gharakheili, Qasim Raza, Craig Russell, and Vijay Sivaraman. 2019. Monitoring enterprise DNS queries for detecting data exfiltration from internal hosts. *IEEE Transactions on Network and Service Management*, 17, 1, 265–279.

[59] Steven J Murdoch and Piotr Zieliński. 2007. Sampled traffic analysis by internet-exchange-level adversaries. In *International workshop on privacy enhancing technologies*. Springer, pp. 167–183.

[60] Baojun Liu, Chaoyi Lu, Haixin Duan, Ying Liu, Zhou Li, Shuang Hao, and Min Yang. 2018. Who is answering my queries: Understanding and characterizing interception of the {DNS} resolution path. In *27th USENIX Security Symposium (USENIX Security 18)*, pp. 1113–1128.

[61] Anne Edmundson, Roya Ensafi, Nick Feamster, and Jennifer Rexford. 2016. Characterizing and avoiding routing detours through surveillance states. *arXiv preprint arXiv:1605.07685*.

[62] Zhiwei Yan and Jong-Hyouk Lee. 2020. The road to DNS privacy. *Future Generation Computer Systems*, 112, 604–611.

[63] Geoff Huston. 2019. DNS resolver centrality. Blog post. APNIC Labs. (September 2019). Retrieved 11/12/2025 from https://blog.apnic.net/2019/09/23/dns-resolver-centrality.

[64] Deliang Chang, Qianli Zhang, and Xing Li. 2015. Study on os fingerprinting and nat/tethering based on dns log analysis. In *IRTF & ISOC Workshop on Research and Applications of Internet Measurements (RAIM)*, pp. 1–4.

[65] Dae Wook Kim and Junjie Zhang. 2015. You are how you query: Deriving behavioral fingerprints from DNS traffic. In *International Conference on Security and Privacy in Communication Systems*. Springer, pp. 348–366.

[66] Matthias Kirchler, Dominik Herrmann, Jens Lindemann, and Marius Kloft. 2016. Tracked without a trace: linking sessions of users by unsupervised learning of patterns in their DNS traffic. In *Proceedings of the 2016 ACM workshop on artificial intelligence and security*, pp. 23–34.

[67] Alain Durand. 2022. DNS Resolvers Used in the EU. *Small*, 2, 19.

[68] Geoff Huston. 2021. Centrality and the Internet. Technical Note. Published June 2021. The ISP Column, APNIC Labs, (June 2021). Retrieved 11/12/2025 from https://www.potaroo.net/ispcol/2021-06/centrality.pdf.

[69] Ash Pallarito and Joe Abley. 2025. Cloudflare 1.1.1.1 incident on July 14, 2025. Blog post. Cloudflare, Inc. (July 2025). Retrieved 11/12/2025 from https://blog.cloudflare.com/cloudflare-1-1-1-1-incident-on-july-14-2025/.

[70] Trinh Viet Doan, Justus Fries, and Vaibhav Bajpai. 2021. Evaluating public DNS services in the wake of increasing centralization of DNS. In *2021 IFIP Networking Conference (IFIP Networking)*. IEEE, pp. 1–9.

[71] Giovane CM Moura, Sebastian Castro, Wes Hardaker, Maarten Wullink, and Cristian Hesselman. 2020. Clouding up the internet: How centralized is dns traffic becoming? In *Proceedings of the ACM Internet Measurement Conference*, pp. 42–49.

[72] Taejoong Chung, Roland van Rijswijk-Deij, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. 2017. Understanding the role of registrars in DNSSEC deployment. In *Proceedings of the 2017 Internet Measurement Conference*, pp. 369–383.

[73] Elisa Tsai, Deepak Kumar, Ram Sundara Raman, Gavin Li, Yael Eiger, and Roya Ensafi. 2023. Certainty: Detecting dns manipulation at scale using tls certificates. *arXiv preprint arXiv:2305.08189*.

[74] ICANN Security and Stability Advisory Committee (SSAC). 2025. DNS Blocking Revisited: A Report from the ICANN Security and Stability Advisory Committee (SAC127). SSAC Report No. 127 SAC127. Final report published 16 May 2025. Internet Corporation for Assigned Names and Numbers (ICANN), (May 2025). Retrieved 11/12/2025 from https://itp.cdn.icann.org/en/files/security-and-stability-advisory-committee-ssac-reports/sac127-dns-blocking-revisited-16-05-2025-en.pdf.

[75] Mingxuan Liu, Yiming Zhang, Xiang Li, Chaoyi Lu, Baojun Liu, Haixin Duan, and Xiaofeng Zheng. 2024. Understanding the Implementation and Security Implications of Protective DNS Services. In *Proceedings of the 31st Annual Network and Distributed System Security Symposium, NDSS*. Volume 24.

[76] Internet Corporation for Assigned Names and Numbers (ICANN) Office of the CTO Security, Stability and Resiliency Team. 2024. Domain Abuse Activity Reporting (DAAR) Monthly Report: Data for September 2024. Monthly Report. Internet Corporation for Assigned Names and Numbers (ICANN), (September 2024). Retrieved 11/12/2025 from https://www.icann.org/en/system/files/files/daar-monthly-report-30sep24-en.pdf.

[77] Derek Atkins and Rob Austein. 2004. Threat Analysis of the Domain Name System (DNS). RFC 3833. (August 2004). DOI: 10.17487/RFC3833. https://www.rfc-editor.org/info/rfc3833.

[78] Rasmus Dahlberg and Tobias Pulls. 2023. Timeless Timing Attacks and Preload Defenses in Tor's {DNS} Cache. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 2635–2652.

[79] Benjamin Greschbach, Tobias Pulls, Laura M Roberts, Philipp Winter, and Nick Feamster. 2016. The effect of DNS on Tor's anonymity. *arXiv preprint arXiv:1609.08187*.

[80] Christian Hofer. 2020. Proposal 317: Improve security aspects of DNS name resolution. Technical report 317. Tor design proposal; status: Needs-Revision. The Tor Project, (March 2020). Retrieved 11/12/2025 from https://spec.torproject.org/proposals/317-secure-dns-name-resolution.html.

[81] Lin Jin, Shuai Hao, Haining Wang, and Chase Cotton. 2021. Understanding the impact of encrypted DNS on internet censorship. In *Proceedings of the Web Conference 2021*, pp. 484–495.

[82] Nguyen Phong Hoang, Michalis Polychronakis, and Phillipa Gill. 2022. Measuring the accessibility of domain name encryption and its impact on internet filtering. In *International Conference on Passive and Active Network Measurement*. Springer, pp. 518–536.

[83] Chaoyi Lu, Baojun Liu, Zhou Li, Shuang Hao, Haixin Duan, Mingming Zhang, Chunying Leng, Ying Liu, Zaifeng Zhang, and Jianping Wu. 2019. An end-to-end, large-scale measurement of DNS-over-encryption: How far have we come? In *Proceedings of the Internet Measurement Conference*, pp. 22–35.

[84] Sebastián García, Karel Hynek, Dmtrii Vekshin, Tomáš Čejka, and Armin Wasicek. 2021. Large scale measurement on the adoption of encrypted DNS. *arXiv preprint arXiv:2107.04436*.

[85] Yong Shao, Kenneth Hernandez, Kia Yang, Eric Chan-Tin, and Mohammed Abuhamad. 2023. Lightweight and Effective Website Fingerprinting over Encrypted DNS. In *2023 Silicon Valley Cybersecurity Conference (SVCC)*.

[86] Sandra Siby, Marc Juarez, Claudia Diaz, Narseo Vallina-Rodriguez, and Carmela Troncoso. 2019. Encrypted DNS–> privacy? A traffic analysis perspective. *arXiv preprint arXiv:1906.09682*.

[87] Timm Böttger, Felix Cuadrado, Gianni Antichi, Eder Leão Fernandes, Gareth Tyson, Ignacio Castro, and Steve Uhlig. 2019. An Empirical Study of the Cost of DNS-over-HTTPS. In *Proceedings of the Internet Measurement Conference*, pp. 15–21.

[88] Rashna Kumar and Fabián E Bustamante. 2023. Reclaiming privacy and performance over centralized DNS. *arXiv preprint arXiv:2302.13274*.

[89] Baiyang Li, Yujia Zhu, Yong Ding, Yong Sun, Yuedong Zhang, Qingyun Liu, and Li Guo. 2024. From Fingerprint to Footprint: Characterizing the Dependencies in Encrypted DNS Infrastructures. In *European Symposium on Research in Computer Security*. Springer, pp. 45–64.

[90] Michael Sonntag. 2019. Malicious DNS Traffic in Tor: Analysis and Countermeasures. In *ICISSP*, pp. 536–543.

[91] Scott Rose, Matt Larson, Dan Massey, Rob Austein, and Roy Arends. 2005. Protocol Modifications for the DNS Security Extensions. RFC 4035. (March 2005). DOI: 10.17487/RFC4035. https://www.rfc-editor.org/info/rfc4035.

[92] Jonathan Magnusson, Moritz Müller, Anna Brunstrom, and Tobias Pulls. 2023. A second look at DNS QNAME minimization. In *International Conference on Passive and Active Network Measurement*. Springer, pp. 496–521.

[93] Jacob Appelbaum and Alec Muffett. 2015. The ".onion" Special-Use Domain Name. RFC 7686. (October 2015). DOI: 10.17487/RFC7686. https://www.rfc-editor.org/info/rfc7686.

[94] Yaron Sheffer, Peter Saint-Andre, and Thomas Fossati. 2022. Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). RFC 9325. (November 2022). DOI: 10.17487/RFC9325. https://www.rfc-editor.org/info/rfc9325.

[95] Eric Rescorla, Kazuho Oku, Nick Sullivan, and Christopher A. Wood. 2025. TLS Encrypted Client Hello. Internet-Draft draft-ietf-tls-esni-25. Work in Progress. Internet Engineering Task Force, (June 2025). 53 pages. https://datatracker.ietf.org/doc/draft-ietf-tls-esni/25/.

Appendix A

Proof-of-Concept

A.1 Dockerfiles

A.1.1 Dockerfile for torconnector

```
FROM ubuntu:24.04

RUN apt-get update
RUN apt-get install -y gnupg
RUN apt-get install -y curl

RUN echo "deb https://deb.torproject.org/torproject.org noble main" >>
↪ /etc/apt/sources.list.d/tor.list
RUN echo "deb-src https://deb.torproject.org/torproject.org noble main" >>
↪ /etc/apt/sources.list.d/tor.list

RUN curl
↪ https://deb.torproject.org/torproject.org/A3C4F0F979CAA22CDBA8F512EE8CBC9E886DDD89.asc |
↪ gpg --import
RUN gpg --export A3C4F0F979CAA22CDBA8F512EE8CBC9E886DDD89 | apt-key add -

RUN apt-get update
RUN apt-get install -y tor deb.torproject.org-keyring

RUN echo "Log notice stdout" >> /etc/tor/torrc
RUN echo "SocksPort 0.0.0.0:9150" >> /etc/tor/torrc
RUN echo "DNSPort 0.0.0.0:53" >> /etc/tor/torrc

EXPOSE 9150

ENTRYPOINT ["tor", "-f", "/etc/tor/torrc"]
```

Listing A.1: torconnector Dockerfile

A.1.2 Dockerfile for dnscrypt Resolver

```
FROM ubuntu:24.04

RUN apt-get update
RUN apt-get install -y wget

RUN wget https://github.com/DNSCrypt/dnscrypt-proxy/releases/download/2.1.14/dnscrypt-proxy
 ↪  -linux_x86_64-2.1.14.tar.gz
RUN tar -vxf dnscrypt-proxy-linux_x86_64-2.1.14.tar.gz

ENTRYPOINT ["bash", "-c", "cd linux-x86_64 && exec ./dnscrypt-proxy"]
```

Listing A.2: dnscrypt Dockerfile

A.2 Lists for Domain Name Blocking and Local Host Entries

A.2.1 Domain Name Blocklist

```
# domain-name-blocklist.txt
# domain names that are blocked by the Private DNS Resolver

trackmeifyoucan.com
ads.com
adultstuff.com
dangeroussite.com
```

Listing A.3: Blocked domain names file

A.2.2 List of Local Host Entries

```
# local-host-entries.txt
# domain names of local host entries

192.168.10.11 nextcloud.lan
192.168.10.12 vaultwarden.lan
```

Listing A.4: Local host names file

A.3 Configuration Files

A.3.1 RouteDNS Configuration File

```
# listening ports
[listeners.local-udp]
address = "0.0.0.0:53"
protocol = "udp"
resolver = "blocklist-domains"

[listeners.local-tcp]
address = "0.0.0.0:53"
```

```
protocol = "tcp"
resolver = "blocklist-domains"

# blocked domain names
[groups.blocklist-domains]
type              = "blocklist-v2"
resolvers         = ["localhosts"]
blocklist-format  = "domain"
blocklist-refresh = 86400
blocklist-source  = [
  {format = "domain", source = "/etc/routedns/domain-name-blocklist.txt"},
]

# local host entries
[groups.localhosts]
type                = "blocklist-v2"
resolvers           = ["cache_wrap"]
blocklist-refresh = 86400
blocklist-source  = [
  {format = "hosts", source = "/etc/routedns/local-host-entries.txt"},
]

# DNS cache
[groups.cache_wrap]
type = "cache"
resolvers = ["randomize"]
backend = {type = "memory"}
size = 4096
min-ttl = 0
max-ttl = 86400
prefetch = false

# distribution algorithm
[groups.randomize]
type = "random"
resolvers = ["serbica-cry", "cryptostorm_is_dus3-cry", "dnscrypt_org-cry",
 ↪  "scaleway-ams-cry", "dnscrypt_uk_vultr-cry", "dnscrypt_ca-cry", "saldns02-conoha-cry",
 ↪  "pl-guardian-cry", "ksol_io-ns2-dnscrypt-cry", "fluffycat-fr-02-cry", "seby_io-doh",
 ↪  "dnscry_pt_nue01-doh", "lacontrevoie_fr-doh", "a47_me-doh", "belnet_be-doh",
 ↪  "00dani_me-doh", "kernel-error_de-doh", "kescher_at-doh", "kooman-doh",
 ↪  "mnet-online_de-doh", "novg_net-doh", "sunet_se-doh", "waringer-atg_de-doh", "yarp-doh",
 ↪  "nextdns_io-doh", "applied_privacy_net-doh", "fdn_fr-doh", "hurricane_electric-doh",
 ↪  "mullvad_net-doh", "wikimedia-doh", "tiar_app_jp-doh", "njal_la-doh",
 ↪  "adguard_unfilterd_2-dot", "cloudflare-dot", "dns_sb_1-dot",
 ↪  "digitale_gesellschaft_2-dot", "uncensoreddns_org-dot", "getdnsapinet443-dot",
 ↪  "cisco_opendns_sandbox_1-dot", "dns-ga_de_3-dot", "controld-dot",
 ↪  "andrews_arnold_1-dot", "bortzmeyer_fr-dot", "digitalize_net-dot", "dnshome_de_2-dot",
 ↪  "cgnat_net-dot", "freifunk_munich_2-dot", "digitalcourage_de-dot"]

# DoH resolver configurations
[resolvers.seby_io-doh]
address = "https://doh.seby.io/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
```

```toml
socks5-address = "172.18.0.2:9150"

[resolvers.dnscry_pt_nue01-doh]
address = "https://nue01.dnscry.pt/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.lacontrevoie_fr-doh]
address = "https://doh.lacontrevoie.fr/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.a47_me-doh]
address = "https://dns.a47.me/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.belnet_be-doh]
address = "https://dns.belnet.be/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.00dani_me-doh]
address = "https://ns.00dani.me/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.kernel-error_de-doh]
address = "https://dns.kernel-error.de/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.kescher_at-doh]
address = "https://dns.kescher.at/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"
```

```

[resolvers.kooman-doh]
address = "https://doh.kooman.org/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.mnet-online_de-doh]
address = "https://dns.mnet-online.de/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.novg_net-doh]
address = "https://dns.novg.net/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.sunet_se-doh]
address = "https://resolver.sunet.se/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.waringer-atg_de-doh]
address = "https://abel.waringer-atg.de/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.yarp-doh]
address = "https://yarp.lefolgoc.net/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.nextdns_io-doh]
address = "https://anycast.dns.nextdns.io/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

```

```toml
[resolvers.applied_privacy_net-doh]
address = "https://doh.applied-privacy.net/query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.fdn_fr-doh]
address = "https://ns0.fdn.fr/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.hurricane_electric-doh]
address = "https://ordns.he.net/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.mullvad_net-doh]
address = "https://dns.mullvad.net/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.wikimedia-doh]
address = "https://wikimedia-dns.org/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.tiar_app_jp-doh]
address = "https://jp.tiar.app/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

[resolvers.njal_la-doh]
address = "https://dns.njal.la/dns-query"
protocol = "doh"
query-timeout = 5
retry-delay = 300
bootstrap-addr = "172.18.0.2"
socks5-address = "172.18.0.2:9150"

# DoT resolver configurations
```

```
[resolvers.adguard_unfilterd_2-dot]
address = "94.140.14.140"
protocol = "dot"
query-timeout = 5
retry-delay = 300
server-name = "unfiltered.adguard-dns.com"
socks5-address = "172.18.0.2:9150"

[resolvers.cloudflare-dot]
address = "1.1.1.1"
protocol = "dot"
query-timeout = 5
retry-delay = 300
server-name = "1dot1dot1dot1.cloudflare-dns.com"
socks5-address = "172.18.0.2:9150"

[resolvers.dns_sb_1-dot]
address = "185.222.222.222"
protocol = "dot"
query-timeout = 5
retry-delay = 300
server-name = "dot.sb"
socks5-address = "172.18.0.2:9150"

[resolvers.digitale_gesellschaft_2-dot]
address = "185.95.218.43"
protocol = "dot"
query-timeout = 5
retry-delay = 300
server-name = "dns.digitale-gesellschaft.ch"
socks5-address = "172.18.0.2:9150"

[resolvers.uncensoreddns_org-dot]
address = "91.239.100.100"
protocol = "dot"
query-timeout = 5
retry-delay = 300
server-name = "anycast.uncensoreddns.org"
socks5-address = "172.18.0.2:9150"

[resolvers.getdnsapinet443-dot]
address = "185.49.141.37:443"
protocol = "dot"
query-timeout = 5
retry-delay = 300
server-name = "getdnsapi.net"
socks5-address = "172.18.0.2:9150"

[resolvers.cisco_opendns_sandbox_1-dot]
address = "208.67.222.2"
protocol = "dot"
query-timeout = 5
retry-delay = 300
server-name = "sandbox.opendns.com"
socks5-address = "172.18.0.2:9150"

[resolvers.dns-ga_de_3-dot]
```

```
address = "138.201.81.119"
protocol = "dot"
query-timeout = 5
retry-delay = 300
server-name = "dot.dns-ga.de"
socks5-address = "172.18.0.2:9150"

[resolvers.controld-dot]
address = "76.76.2.11"
protocol = "dot"
query-timeout = 5
retry-delay = 300
server-name = "p0.freedns.controld.com"
socks5-address = "172.18.0.2:9150"

[resolvers.andrews_arnold_1-dot]
address = "217.169.20.22"
protocol = "dot"
query-timeout = 5
retry-delay = 300
server-name = "dns.aa.net.uk"
socks5-address = "172.18.0.2:9150"

[resolvers.bortzmeyer_fr-dot]
address = "193.70.85.11"
protocol = "dot"
query-timeout = 5
retry-delay = 300
server-name = "dot.bortzmeyer.fr"
socks5-address = "172.18.0.2:9150"

[resolvers.digitalize_net-dot]
address = "94.130.135.203"
protocol = "dot"
query-timeout = 5
retry-delay = 300
server-name = "dns.digitalsize.net"
socks5-address = "172.18.0.2:9150"

[resolvers.dnshome_de_2-dot]
address = "45.86.125.59"
protocol = "dot"
query-timeout = 5
retry-delay = 300
server-name = "dns.dnshome.de"
socks5-address = "172.18.0.2:9150"

[resolvers.cgnat_net-dot]
address = "144.22.247.219"
protocol = "dot"
query-timeout = 5
retry-delay = 300
server-name = "ibuki.cgnat.net"
socks5-address = "172.18.0.2:9150"

[resolvers.freifunk_munich_2-dot]
address = "185.150.99.255"
```

```
protocol = "dot"
query-timeout = 5
retry-delay = 300
server-name = "dot.ffmuc.net"
socks5-address = "172.18.0.2:9150"

[resolvers.digitalcourage_de-dot]
address = "5.9.164.112"
protocol = "dot"
query-timeout = 5
retry-delay = 300
server-name = "dns3.digitalcourage.de"
socks5-address = "172.18.0.2:9150"

# DNSCrypt resolver definitions
[resolvers.serbica-cry]
address = "cry-serbica:53"
protocol = "udp"
query-timeout = 5
retry-delay = 300

[resolvers.cryptostorm_is_dus3-cry]
address = "cry-cryptostorm-is-dus3:53"
protocol = "udp"
query-timeout = 5
retry-delay = 300

[resolvers.dnscrypt_org-cry]
address = "cry-dnscrypt-org:53"
protocol = "udp"
query-timeout = 5
retry-delay = 300

[resolvers.scaleway-ams-cry]
address = "cry-scaleway-ams:53"
protocol = "udp"
query-timeout = 5
retry-delay = 300

[resolvers.dnscrypt_uk_vultr-cry]
address = "cry-dnscrypt-uk-vultr:53"
protocol = "udp"
query-timeout = 5
retry-delay = 300

[resolvers.dnscrypt_ca-cry]
address = "cry-dnscrypt-ca:53"
protocol = "udp"
query-timeout = 5
retry-delay = 300

[resolvers.saldns02-conoha-cry]
address = "cry-saldns02-conoha:53"
protocol = "udp"
query-timeout = 5
retry-delay = 300

```

```
[resolvers.pl-guardian-cry]
address = "cry-pl-guardian:53"
protocol = "udp"
query-timeout = 5
retry-delay = 300

[resolvers.ksol_io-ns2-dnscrypt-cry]
address = "cry-ksol-io-ns2-dnscrypt:53"
protocol = "udp"
query-timeout = 5
retry-delay = 300

[resolvers.fluffycat-fr-02-cry]
address = "cry-fluffycat-fr-02:53"
protocol = "udp"
query-timeout = 5
retry-delay = 300
```

Listing A.5: RouteDNS configuration file

A.3.2 Example Configuration File of DNSCrypt Resolver dnscrypt_org

```
# IP address of torconnector: 172.18.0.2
# TCP port of torconnector: 9150

listen_addresses = ["0.0.0.0:53"]
max_clients = 250
ipv4_servers = true
dnscrypt_servers = true
force_tcp = true
proxy = "socks5://172.18.0.2:9150"
timeout = 5000
keepalive = 30
cert_refresh_delay = 240
dnscrypt_ephemeral_keys = true
ignore_system_dns = true
netprobe_timeout = 60
netprobe_address = "172.18.0.2:53"
block_ipv6 = false
block_unqualified = true
block_undelegated = true
cache = false
[static]
[static."dnscrypt_org"]
stamp = "sdns://AQcAAAAAAAAAADjIxMi40Ny4yMjguMTM2IOgBuE6mBr-wusDOQ0RbsV66ZLAvo8SqMa4QY2oHkDJNHzIuZG5zY3J5cHQtY2VydC5mci5kbnNjcnlwdC5vcmc"
```

Listing A.6: DNSCrypt resolver example configuration file

A.3.3 Stamps of selected DNSCrypt Resolver Configurations

```
# stamp of serbica
stamp = "sdns://AQcAAAAAAAAAEzE4NS42Ni4xNDMuMTc4OjUzNTMg-Y2MQmGOXiggAEKulN-ITGEn_Kj3TIP1UK1X2wh3o7wXMi5kbnNjcnlwdC1jZXJ0LnNlcmJpY2E"

# stamp of cryptostorm_is_dus3
stamp = "sdns://AQYAAAAAAAAADjg5LjE2My4yMjEuMTgxIDEzcq1ZVjLCQWuHLwmPhRvduWUoTGy-mk8ZCWQw26laHjIuZG5zY3J5cHQtY2VydC5jcnlwdG9zdG9ybS5pcw"

# stamp of dnscrypt_org
stamp = "sdns://AQcAAAAAAAAADjIxMi40Ny4yMjguMTM2IOgBuE6mBr-wusDOQ0RbsV66ZLAvo8SqMa4QY2oHkDJNHzIuZG5zY3J5cHQtY2VydC5mci5kbnNjcnlwdC5vcmc"

# stamp of scaleway-ams
stamp = "sdns://AQcAAAAAAAAADTUxLjE1LjEyMi4yNTAg6Q3ZfapcbHgiHKLF7QFoli0Ty1Vsz3RXs1RUbxUrwZAcMi5kbnNjcnlwdC1jZXJ0LnNjYWxld2F5LWFtcw"

# stamp of dnscrypt_uk_vultr
stamp = "sdns://AQcAAAAAAAAAEzEwNC4yMzguMTg2LjE5Mjo0NDMg7Uk9jOrXkGZPBjxHt5WaI2ktfJA2PJ5DzLWRe-W0HuUdMi5kbnNjcnlwdC1jZXJ0LnYuZG5zY3J5cHQudWs"

# stamp of dnscrypt_ca
stamp = "sdns://AQcAAAAAAAAAEzE4NS4xMTEuMTg4LjQ2Ojg0NDMgC-tbTwd-08e_JtBJmgsvjAG9i10itE-LBNCwjTflezQiMi5kbnNjcnlwdC1jZXJ0LmRuc2NyeXB0LmNhLTEtaXB2NA"

# stamp of saldns02-conoha
stamp = "sdns://AQcAAAAAAAAAFTEzMy4xMzAuMTE4LjEwMzo1MDQ0MyB7SI0q4_Ff8lFRUCbjPtcAQ3HfdWlLxyGDUUNc3NUZdiIyLmRuc2NyeXB0LWNlcnQuc2FsZG5zMDIudHlwZXEub3Jn"

# stamp of pl-guardian
stamp = "sdns://AQMAAAAAAAAAFDE3OC4yMTYuMjAxLjEyODoyMDU0IH9hfLgepVPSNMSbwnnHT3tUmAUNHb8RGv7mmWPGR6FpGzIuZG5zY3J5cHQtY2VydC5kbnNjcnlwdC5wbA"

# stamp of ksol_io-ns2-dnscrypt
stamp = "sdns://AQcAAAAAAAAADjE5My4yMDEuMTg4LjQ4IBERKdQJgLSjqCSK99e2f_WRTQzEq9__DeXlQFvxxhZ6GzIuZG5zY3J5cHQtY2VydC5uczIua3NvbC5pbw"

# stamp of fluffycat-fr-02
stamp = "sdns://AQcAAAAAAAAAFDEyOS4xNTEuMjQzLjE0Mzo1MzUzICaU0eKbEtcmoE0ljHjvADPHNyZBX23wJ4owxhVprIpFHzIuZG5zY3J5cHQtY2VydC5mbHVmZnljYXQtZnItMDI"
```

Listing A.7: Stamps of selected DNSCrypt resolvers

A.4 Startup Script

A.4.1 Bash Script containing Startup Commands for Launching the Private DNS Resolver

```
#!/bin/bash

```

```
docker run --rm --name connector --network networktor --mount
 ↪  type=bind,source=/etc/localtime,target=/etc/localtime,readonly -d connector:latest

docker run --rm --name cry-serbica --network networktor --mount
 ↪  type=bind,source=/etc/localtime,target=/etc/localtime,readonly --mount
 ↪  type=bind,source=/home/privatednsresolver/config/dnscrypt/CRY_serbica/dnscrypt-proxy_t.⌋
 ↪  toml,target=/linux-x86_64/dnscrypt-proxy.toml,readonly -d dnscrypt:latest
docker run --rm --name cry-cryptostorm-is-dus3 --network networktor --mount
 ↪  type=bind,source=/etc/localtime,target=/etc/localtime,readonly --mount
 ↪  type=bind,source=/home/privatednsresolver/config/dnscrypt/CRY_serbica/dnscrypt-proxy_t.⌋
 ↪  toml,target=/linux-x86_64/dnscrypt-proxy.toml,readonly -d dnscrypt:latest
docker run --rm --name cry-dnscrypt-org --network networktor --mount
 ↪  type=bind,source=/etc/localtime,target=/etc/localtime,readonly --mount
 ↪  type=bind,source=/home/privatednsresolver/config/dnscrypt/CRY_serbica/dnscrypt-proxy_t.⌋
 ↪  toml,target=/linux-x86_64/dnscrypt-proxy.toml,readonly -d dnscrypt:latest
docker run --rm --name cry-scaleway-ams --network networktor --mount
 ↪  type=bind,source=/etc/localtime,target=/etc/localtime,readonly --mount
 ↪  type=bind,source=/home/privatednsresolver/config/dnscrypt/CRY_serbica/dnscrypt-proxy_t.⌋
 ↪  toml,target=/linux-x86_64/dnscrypt-proxy.toml,readonly -d dnscrypt:latest
docker run --rm --name cry-dnscrypt-uk-vultr --network networktor --mount
 ↪  type=bind,source=/etc/localtime,target=/etc/localtime,readonly --mount
 ↪  type=bind,source=/home/privatednsresolver/config/dnscrypt/CRY_serbica/dnscrypt-proxy_t.⌋
 ↪  toml,target=/linux-x86_64/dnscrypt-proxy.toml,readonly -d dnscrypt:latest
docker run --rm --name cry-dnscrypt-ca --network networktor --mount
 ↪  type=bind,source=/etc/localtime,target=/etc/localtime,readonly --mount
 ↪  type=bind,source=/home/privatednsresolver/config/dnscrypt/CRY_serbica/dnscrypt-proxy_t.⌋
 ↪  toml,target=/linux-x86_64/dnscrypt-proxy.toml,readonly -d dnscrypt:latest
docker run --rm --name cry-saldns02-conoha --network networktor --mount
 ↪  type=bind,source=/etc/localtime,target=/etc/localtime,readonly --mount
 ↪  type=bind,source=/home/privatednsresolver/config/dnscrypt/CRY_serbica/dnscrypt-proxy_t.⌋
 ↪  toml,target=/linux-x86_64/dnscrypt-proxy.toml,readonly -d dnscrypt:latest
docker run --rm --name cry-pl-guardian --network networktor --mount
 ↪  type=bind,source=/etc/localtime,target=/etc/localtime,readonly --mount
 ↪  type=bind,source=/home/privatednsresolver/config/dnscrypt/CRY_serbica/dnscrypt-proxy_t.⌋
 ↪  toml,target=/linux-x86_64/dnscrypt-proxy.toml,readonly -d dnscrypt:latest
docker run --rm --name cry-ksol-io-ns2-dnscrypt --network networktor --mount
 ↪  type=bind,source=/etc/localtime,target=/etc/localtime,readonly --mount
 ↪  type=bind,source=/home/privatednsresolver/config/dnscrypt/CRY_serbica/dnscrypt-proxy_t.⌋
 ↪  toml,target=/linux-x86_64/dnscrypt-proxy.toml,readonly -d dnscrypt:latest
docker run --rm --name cry-fluffycat-fr-02 --network networktor --mount
 ↪  type=bind,source=/etc/localtime,target=/etc/localtime,readonly --mount
 ↪  type=bind,source=/home/privatednsresolver/config/dnscrypt/CRY_serbica/dnscrypt-proxy_t.⌋
 ↪  toml,target=/linux-x86_64/dnscrypt-proxy.toml,readonly -d dnscrypt:latest

docker run --rm --name routedns -p 5353:53/udp -p 5353:53/tcp --network networktor --mount
 ↪  type=bind,source=/etc/localtime,target=/etc/localtime,readonly --mount
 ↪  type=bind,source=/home/privatednsresolver/config/routedns/routedns.toml,target=/etc/rou⌋
 ↪  tedns/config.toml,readonly --mount
 ↪  type=bind,source=/home/privatednsresolver/config/routedns/domain-name-blocklist.txt,tar⌋
 ↪  get=/etc/routedns/domain-name-blocklist.txt,readonly --mount
 ↪  type=bind,source=/home/privatednsresolver/config/routedns/local-host-entries.txt,target⌋
 ↪  =/etc/routedns/local-host-entries.txt,readonly -d folbricht/routedns:latest
 ↪  /etc/routedns/config.toml
```

Listing A.8: Startup script