Sudheesh Singanamalla, Suphanat Chunhapanya, Jonathan Hoyland, Marek Vavruša, Tanya Verma, Peter Wu, Marwan Fayed, Kurtis Heimerl, Nick Sullivan, and Christopher Wood

# Oblivious DNS over HTTPS (ODoH): A Practical Privacy Enhancement to DNS

**Abstract:** The Internet's Domain Name System (DNS) responds to client hostname queries with corresponding IP addresses and records. Traditional DNS is unencrypted and leaks user information to on-lookers. Recent efforts to secure DNS using DNS over TLS (DoT) and DNS over HTTPS (DoH) have been gaining traction, ostensibly protecting DNS messages from third parties. However, the small number of available public large-scale DoT and DoH resolvers has reinforced DNS privacy concerns, specifically that DNS operators could use query contents and client IP addresses to link activities with identities. Oblivious DNS over HTTPS (ODoH) safeguards against these problems. In this paper we implement and deploy interoperable instantiations of the protocol, construct a corresponding formal model and analysis, and evaluate the protocols' performance with wide-scale measurements. Results suggest that ODoH is a practical privacy-enhancing replacement for DNS.

**Keywords:** DNS, privacy

DOI 10.2478/popets-2021-0085

Received 2021-02-28; revised 2021-06-15; accepted 2021-06-16.

## 1 Introduction

The Domain Name System (DNS) is the human bridge to the Internet. DNS returns machine-readable IP addresses and records in exchange for queries with human-readable hostnames [9]. This exchange is handled by resolvers that accept queries from clients and return responses from authoritative name servers. By default, DNS messages are transmitted in cleartext over the User Datagram Protocol (UDP) on port 53, hereafter referred to as Do53. As a result, Do53 is vulnerable to eavesdropping and modification by both well-intentioned and malicious third parties. Not only do these risks compromise privacy, but they can result in denial of service (DoS) and injection attacks [84]. Compared to the broader Internet and web ecosystem, wherein traffic is increasingly safeguarded by HTTPS, Do53 lags behind and remains as a weak facet of a secure Internet.

Secure variants of DNS have recently been introduced to fill that gap. DNS-over-TLS (DoT) [48] and DNS-over-HTTPS (DoH) [45] are now widely supported by browsers and increasingly supported by operating systems. In DoT and DoH, the *transport* of DNS messages between client stub resolvers and upstream recursive resolvers is encrypted over the TLS channel. However, at time of writing, DoT and DoH are supported by only a comparatively small number of providers relative to the wider resolver population.

Despite the availability of secure DNS services, a historical omission from the DNS ecosystem makes wider deployment a challenge: Although DNS benefits from auto-configuration of local services in DHCP [34], there exists no equivalent beyond the local network, nor does there exist a discovery mechanism of any kind for DNS. This has exposed two challenges with respect to deployment and privacy. First, the majority of clients leak or expose their information to on-lookers. For example, at time of writing, we observe from a large recursive DNS resolver that clients transmit 92% of their queries (254 billion/day) using the default cleartext Do53 protocol [66], and therefore leak their own information. In the absence of wider deployment, discovery, and auto-configuration supports, the only way to use secure DNS is by manual setup.

User privacy at the resolver, which is the focus of this paper, is particularly problematic: While any individual website or online service can associate requests

**Sudheesh Singanamalla:** University of Washington, and Cloudflare Inc. sudheesh@cs.washington.edu. Sudheesh was with Cloudflare Inc. while doing this work.
**Suphanat Chunhapanya:** Cloudflare Inc. pop@cloudflare.com
**Jonathan Hoyland:** Cloudflare Inc. jhoyland@cloudflare.com
**Marek Vavruša:** Cloudflare Inc. mvavrusa@cloudflare.com
**Tanya Verma:** Cloudflare Inc. tverma@cloudflare.com
**Peter Wu:** Cloudflare Inc. pwu@cloudflare.com
**Marwan Fayed:** Cloudflare Inc. marwan@cloudflare.com
**Kurtis Heimerl:** University of Washington kheimerl@cs.washington.edu
**Nick Sullivan:** Cloudflare Inc. nick@cloudflare.com
**Christopher Wood:** Cloudflare Inc. chris-wood@cloudflare.com

(and user data) with clients IP addresses for their own service, DNS resolvers' ability to observe queries makes it possible to link all client activity. One resolution to this issue is to rely on trust-based policies or legal agreements. Public resolvers, for example, publish their own privacy policies; but these are non-uniform, and require domain expertise or language proficiency to understand. Similarly, contractual agreements such as in Mozilla's Trusted Recursive Resolver (TRR) program [27, 60] can define and restrict data retention, aggregation, sale or transfer. These mechanisms, however, are hard to validate and lack any technical means of enforcement.

A better approach would be to modify DNS itself to disassociate query contents from IP addresses in a way that makes them un-linkable. Oblivious DNS (ODNS), for example, makes resolvers blind to both client IPs and their queries [72]. In ODNS, stub resolvers encrypt client queries into a new query for an `.odns` top level domain (TLD). Resolvers are then forced to direct the query to an ODNS server that presents as authoritative; the ODNS authoritative server then decrypts and resolves the query as a normal resolver, before receiving and returning an encrypted response. ODNS works in Do53, but requires registration of a new tld, changes the semantics of authoritative service, and redirects all queries to their authoritative name servers instead of local and performant recursive resolvers.

In this paper we turn attention to Oblivious DNS over HTTPS (ODoH) [51], a recent proposal at the IETF that builds on ODNS concepts to decouple queries from IP addresses and ensure privacy in DoH. ODoH works by encrypting queries into HTTP messages at the stub resolver. The encrypted queries are forwarded by an Oblivious HTTPS Proxy, to and from an Oblivious Target that decrypts queries and encrypts responses. ODoH assures server identity through TLS while also guaranteeing reliable, in-order delivery of packets, preventing the need for any state machines in ODNS. Our specific contributions are as follows:

1. We provide public, open-source, and inter-operable implementations of ODoH components; to the best of our knowledge, these are the first and only implementations of the standard.
2. We model ODoH using Tamarin [71] and provide a formal analysis to prove its security properties. Our model was able to validate a latent replay attack that emerged during analysis, as well as the change that prevents the attack.
3. We deploy ODoH at a popular large DNS resolver, and evaluate ODoH performance, at scale with a comprehensive set of wide-area measurements; we observe marked improvement over other privacy-preserving secure DNS variants, and minimal impact on page-load times compared to standard DoH.

We use these contributions to understand and inform ODoH as a practical privacy enhancement to DNS. In the sections that follow, we assess promised privacy properties against realistic threat models, identify the barriers and enablers to large-scale ODoH infrastructure, as well as use our experience to guide practical global adoption. Our presentation begins with wider context and background.

## 2 Background & Related Work

The Domain Name System (DNS) specification makes no confidentiality or integrity guarantees, which makes DNS vulnerable to eavesdropping and tampering [10, 11, 43]. The collection of such queries can also paint a revealing picture of the users' habits and interests [78]. In this section we give an overview of work to secure DNS in various forms, both technical and regulatory.

### 2.1 Encrypted DNS Protocols and Privacy

Efforts to secure DNS via message or channel encryption include DNS-Over-TLS (DoT) and DNS-Over-HTTPS (DoH) [45], a precursor in T-DNS [84] and, separately, DNSCrypt [38]. Client support for DoH is increasingly available among web browsers [5, 27, 67], mobile clients [30, 31, 52], and operating systems [54]. This has been attributed to the availability of DoH support in public recursive resolvers offered by Cloudflare and NextDNS, with their integration into Firefox as Trusted Recursive Resolvers [12, 27, 56].

Organizations that operate public DoH resolvers publish and adhere to their own privacy policies. These policies may prohibit user tracking, and state strict data use and retention practices [32, 41]. Other organizations may aggregate DNS traffic patterns into permanent logs that omit identifiable information like IP addresses [16, 77]. Reasons may include telemetry for debugging in production systems, improving response time performance, or for identifying browsing trends by geography [32, 39]. Such practices are important because IP addresses are often regarded as personally identifiable or linkable information [37, 58].

In DoT and DoH the network channel is encrypted, and serves as a secure transport for the plaintext messages that are transmitted. Conversely, DNSCrypt [38] and its predecessor DNSCurve [7] encrypt the DNS message before its transmission over TCP or UDP. A client connecting to a DNSCrypt resolver receives a public set of signed certificates that is verified by the client using a known provider public key. Each certificate contains a short-term resolver public key. Messages are encrypted with a shared key generated from the resolver public key, the client secret key, and an agreed key exchange algorithm defined in the certificate. The shared key is used to encrypt subsequent queries using an authenticated encryption algorithm.

Neither channel, nor message-based encrypted DNS protocols address privacy risks at the resolver, where client address and queries can be logged, stored, and potentially profiled or transferred. Anonymous DNSCrypt addresses this issue by introducing public non-logging proxies that forward traffic between clients and the intended DNSCrypt resolver [28, 68]. However, since the traffic is between the proxy and resolver, an adversary that is incident to a proxy's ingress and egress link could link queries to clients.

Prior proposals to address privacy included Privacy Information Retrieval (PIR) techniques. *Range queries* use random noise and PIR techniques during execution of DNS queries [82, 83]. Doing so provided confidentiality, integrity, and privacy when used in conjunction with DNSSEC. Later evaluation of these techniques revealed that substantial changes to DNS servers and clients were necessary to make it feasible, as well as that attackers with control of the channel could infer and forge queries [15].

Cryptographic mixes or mix cascades have also been leveraged to anonymize user traffic [8, 36], which is a model widely adopted in Tor. Additional attempts at privacy consist of broadcasting a desired query among a set of other decoy queries to thwart profiling attempts [36]. Doing so has practical limitations due to large bandwidth usage and long-tailed distribution of queries. These limitations led to a hybrid of mix cascades and broadcasts [36], albeit with degraded performance due to increased page load and DNS response times.

DNS privacy may also be achieved using DoH over Tor (DoHoT), in which the encrypted DNS channel is routed over the Tor anonymous network [20, 62]. Doing so closes the linkability gap of Anonymous DNSCrypt, but also incurs substantial performance penalties [62]. However, recent work by Muffet indicates performance penalties due to the usage of Tor might not impact some users significantly and could go unnoticed by users because latency is only a small fraction of the user experience and value proposition for choosing a DNS protocol [61]. While Tor provides both anonymity and privacy guarantees, Tor nodes can be actively censored or blocked throughout the Internet [76, 81]; traffic can be subject to DNS fingerprinting attacks [42]; and operators of exit nodes may face legal liabilities [44, 57].

ODoH achieves similar privacy guarantees as Anonymous DNSCrypt and DoHoT, but the anonymity of the protocol depends on non-collusion between stakeholders in the ecosystem. ODoH incurs comparatively minimal performance cost, and make its wide adoption practical.

## 2.2 Privacy & Regulatory Considerations

DNS service provision and traffic also invokes a number of regulatory, economic, and even philosophical considerations that, while non-technical and beyond scope, are worthwhile touching upon. Among them is the manner in which a local DNS service is automatically configured by upstream ISPs, and the implications that follow [9]. We stress that any configuration mechanism is necessitated by the absence of a DNS discovery mechanism. Beyond configuration, evidence suggests that even seemingly benign activities, such as monetization of only the error traffic in DNS [80], may have unanticipated consequences.

Users and clients may instead configure their DNS to point to any of a small number of large open resolvers that promise performance and consistent quality of service across networks. The performance improvements are accompanied by an implicit shift in trust to the open resolver, and their operators to publish privacy policies and conduct audits. A trusted policy may be sufficient, but is unable to protect in all circumstances that include human error or system compromise.

## 2.3 Prior Measurements

Large scale measurements of encrypted DNS protocols show that adoption has been increasing [56]. Additional measurements reveal that DoH provides security with no significant impact on page load times [12]. Large scale evaluations from home networks indicate that latency becomes the performance bottleneck as broadband access speeds increase, making metrics like DNS response time and time to first byte more important [75]. Additional measurements suggest that the performance of DoT and DoH vary with the choice of public resolver [47]. In Section 5.2, we evaluate ODoH and other DNS protocols by directing each query to one of three public resolvers selected at random to remove bias.

# 3 Oblivious DNS over HTTPS

The broad mandate of Oblivious DNS over HTTPS (ODoH) is to address the remaining privacy concerns of encrypted protocols like DoH/DoT i.e. to prevent recursive resolvers from being able to link client IP addresses to their queries. In this section we describe ODoH features, the protocol, and our implementation.

## 3.1 Features and Properties

The design of ODoH is similar to that of DNS over HTTPS (DoH). It differs with the addition of an intermediate proxy node that forwards queries and responses between a client and target in an 'oblivious' manner. The oblivious property derives from two attributes. First, connections to and from the proxy use HTTPS to secure the message transmissions from eavesdroppers. However, a proxied-variant of DoH, alone, would expose the query to the proxy. For this reason ODoH secures the payload from the proxy with an additional layer of end-to-end encryption between the client and target for a query. The combination of HTTPS, with the intermediate proxy, and end-to-end encryption ensures that only the client knows both its identity (IP address), queries made, and responses intended for it. Beyond this, ODoH achieves the following properties:

1. Proxies know client IP addresses (i.e., identities) but cannot see actual queries and responses
2. Targets and entities upstream involved in resolution know the query, but only see the IP address of the proxy, hiding client IP information.

We discuss these properties in more detail in Section 4.4. Each of the ODoH client, proxy, target, and protocol specifics are described in detail next.

## 3.2 Protocol Description

ODoH participants, as well as the protocol, are depicted in Figure 1. We describe the design from the view of each of the ODoH components: (i) Clients that communicate using a stub resolver; (ii) an Oblivious Proxy that transmits messages over HTTPS between the client stub and (iii) the Oblivious Target that encrypts and decrypts messages between client stub and DNS resolver.

**Clients (or stub resolvers)** The protocol begins at client's stub resolver, responsible for encrypting client queries and decrypting target responses. Clients are free to use one or more proxies, as well as one or more targets. Pairs of proxy and target may be selected per-query, so long as there is no collusion between them, i.e., they are operated by separate entities. The client encapsulates an ephemeral fixed size shared secret symmetric key $K_s$ with the DNS query $Q$ which is encrypted using the public key of the target $PK$, resulting in the encrypted message $C_Q$ which is sent to the chosen proxy as shown in step (1) in Figure 1.

**Oblivious Proxies** The ODoH path consists of two HTTPS connections that meet at an Oblivious Proxy which acts as server to the client, and as a client to target servers. On path from client to target, the proxy preserves the encrypted query $C_Q$, removes client IP address information, then forwards the request to the Oblivious Target indicated by the client's message as shown in step (2) in Figure 1. Responses from the target are similarly returned to the client that originated the query as shown in step (6) in Figure 1.

**Oblivious Targets** DNS messages in ODoH are secured with Hybrid Public Key Encryption (HPKE) [6]. Targets use keying material derived from a query's HPKE encryption context to encrypt the response. Encrypted queries ($C_Q$) from the client arrive to the target via the proxy. The target decrypts and decapsulates the message received resulting in the query $Q$ the shared secret $K_s$. The target resolves the query, computes a key $K$ from the corresponding Key Derivation Function (KDF) based on the shared secret $K_s$, and then encrypts and returns the answer $A$ as an encrypted response $C_A$ via the same proxy. These communications are shown as steps 3,4, and 5 in Figure 1. The Oblivious Target publishes a public key which the clients securely retrieve and use to encrypt their DNS queries before the protocol is run and is shown by the two dashed lines between the client and target in Figure 1.

We note that the target is defined as being independent of the resolver, but is expected to be coupled

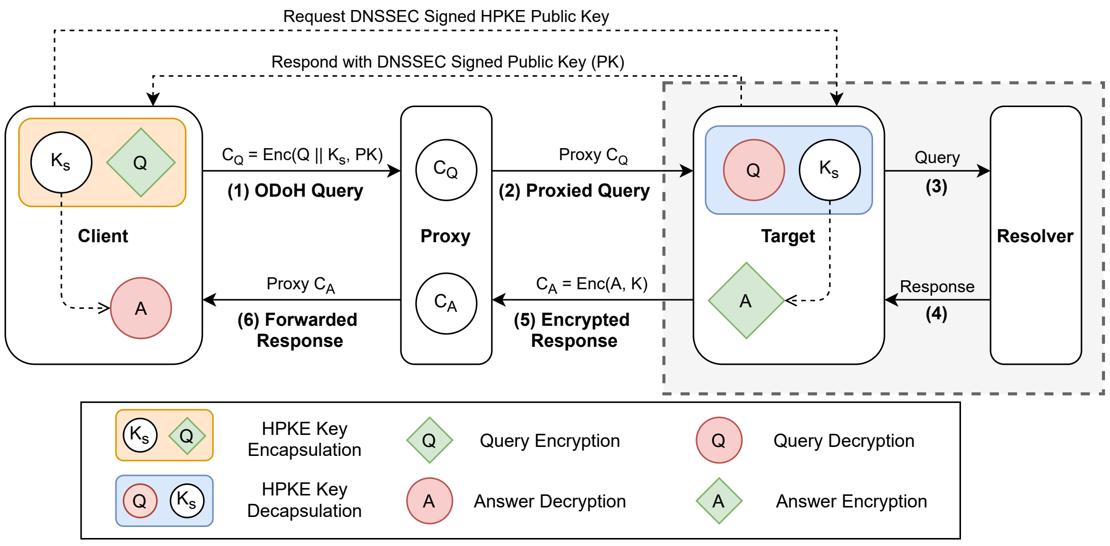


**Fig. 1.** End-to-end execution of the ODoH Protocol. Clients encrypt DNS queries using a target's public key, into an HTTP header with the target's name. Proxies forward to and from targets as indicated by the HTTP header. The grayed box indicates that targets and resolvers may be co-located, in which case messages (3) and (4) are superfluous. All channels to and from the proxy are HTTPS.

with the recursive resolver to improve performance by avoiding two additional network messages. Our implementation and evaluation focus on co-location, and we anticipate that ODoH target features could be implemented within existing resolvers.

**End-to-end Message Flow** The overall end-to-end flow of an ODoH query across clients, proxies, and targets is shown in Figure 1. The dashed box with a grey background that surrounds target and resolver indicates their possible coupling or co-location, while respecting their independence according to the definition. Co-location trades isolation for performance, without loss of privacy. A target co-located with the resolver has no need for messages (3) and (4) shown in Figure 1. We revisit & measure the benefits of co-location in Section 5.4. In the ODoH protocol, a proxy does not learn the contents of the query from the client or the response from the target, and similarly, a target resolver does not learn the identity of the client.

**Discovery, Verification, and Key Distribution** Strictly speaking, DNS in all its forms has no discovery service. Aside from manual configuration, conventional Do53 is traditionally configured (ultimately by ISPs) with DHCP. DNS over TLS (DoT) and HTTPS (DoH) are used by client stub resolvers by opportunistically probing the DNS server indicated through the DHCP configuration for support for either protocol [52]. In this respect DoT/H and ODoH are equivalent: similar probing based approaches to discover protocol support by the resolver can be reused. As a result, in the absence of DHCP support and until a DNS discovery service emerges, options for ODoH configuration are limited to hostnames listed on centralized and trusted lists that could additionally be bootstrapped into browsers or operating systems which might choose to support the protocol or through manual explicit configuration.

Once a proxy or target is known, each client stub resolver requests a DNSSEC signed resource record set (`HTTPSVC`, `SVC`, or similar special fields [73]), where the ODoH public key for the server is published. Once the signature is validated, the stub resolver adds the oblivious target to the list of verified targets and stores the public key necessary for communication. These records can be periodically retrieved and validated by the client stub resolver to check for updates. A browser or client then directs a special lookup to the resolver for `odoh.test`, which returns the necessary keys for validation and indicates support for ODoH. This mirrors the use of `doh.test` to check a resolvers support for DoH [59]. The choice to mirror DoH in this fashion is intentional, reduces barriers to adoption, and deployment.

A full network evaluation of ODoH performance is provided in Section 5, preceded next by a formal analysis that states and proves ODoH's security properties.

# 4 Formal Analysis

In this section we provide a formal analysis to prove that ODoH provides client query privacy. Our analysis consists of a symbolic model generated for the Tamarin theorem prover [71][1]. We make the Tamarin model public, along with its execution instructions and associated proofs accompanying the artifacts of this paper [1, 2].

## 4.1 Model Overview

Our model is kept simple, yet precise, and focuses on the security and privacy of the DNS request-response exchange in ODoH. Thus, we intentionally ignore longitudinal attacks and simple correlation-type attacks such as only one client using a given proxy are either trivially simple or impractical. Similarly, the integrity of resolver responses or their ability to de-anonymize clients is beyond scope of the protocol, and our analysis. ODoH makes no claims of resolver response integrity. A malicious target or resolver could craft answers in such a way to de-anonymize users. This is also discussed in the ODoH protocol specification.

Our model captures cryptographic protocol details, for example, HPKE-based [6] query encryption, and AEAD-based response encryption. Underlying cryptographic algorithms such as hashes are otherwise assumed ideal. Other aspects of ODoH are both impractical and unnecessary. For example, the complexity of TLS 1.3 needs hundreds of thousands of steps to prove [23], so we substitute two simplifications. First, we assume that the client and proxy can establish an authenticated shared key, which is the end-goal of TLS. This assumption relies on TLS being implemented, instantiated, and deployed in a secure way. This is reasonable, given the scale of and complexity of attack to break this key in practice [3, 63]. Moreover, we do not model precise wire-format details of ODoH, and only capture message components in an idealized way since the wire format should not affect protocol security.

As an additional simplification for the analysis, we omit the TLS layer between the proxy and target. Specifically, because we give the attacker the ability to compromise TLS sessions secrets, this simplification is equivalent to the attacker always doing so. We assert that this is a fault-preserving simplification [49], wherein any attack that succeeds on the true protocol also exists in the simplified model. Thus, a proof that the simplified model is secure also implies that ODoH as specified and deployed is secure.

## 4.2 Correlation Attacks

While ODoH provides strong cryptographic security properties for individual queries, it has limited defences against correlation attacks. Backes et al. [4] formalise this as *Sender Anonymity*, i.e., whether an attacker can identify the sender of a message with some non-negligible probability. As discussed in Das et al. [24, 25], resisting correlation attacks in the presence of an adversary that can observe both endpoints requires added latency or bandwidth overhead. In particular, they provide a lower bound on the latency required to achieve strong anonymity of this type. Moreover, the amount of latency introduced has to grow as the number of participants grows [24, Theorem. 8]. DNS has near-real-time performance requirements, so latency overhead is particularly problematic. We see this later in the comparison with DoHoT, where the use of Tor incurs a performance loss from longer circuits at the potential gain of stronger protection against correlation attacks in practice.

We thus settle for a weaker form of anonymity, where an attacker cannot identify the contents of a single session beyond statistical inference. Note, importantly, that an attacker which has compromised the target is perhaps less likely capable of observing traffic between the proxy and the client.

The results of Das et al. [24, 25] do not preclude there being any real-world gain in privacy. ODoH protects against attackers who can see large portions of the network, and because of its cryptographic guarantees even an attacker that can link the client and the target still needs to compromise the target to learn the query. DoHoT provides more anonymity protection, although it still falls short of strong anonymity [24, Table I], but at a cost of very substantial latency, see Figure 4. ODoH provides a good balance between privacy protection in the real world and usable performance. Its simplicity of design and architecture enable it to be deployed widely, and its relative performance compared to protocols with stronger guarantees encourages adoption.

**1** Tamarin has been used successfully to analyze security properties of TLS 1.3, 5G, and other protocols in the past [22, 23].

### 4.3 Adversary

The adversary in our analysis is an extended Dolev-Yao attacker [33], with the standard ability to create, drop, and modify messages. We additionally give the adversary the ability to compromise TLS sessions and the target server's long-term key (LTK). Finally we give the attacker the ability to compromise the security of AEADs if keys and nonces are reused. Obviously against an adversary such as this, ODoH is not secure. However, placing careful limits on the attackers behaviour yields tight security bounds.

Collusion is modeled by allowing the adversary to corrupt proxies and targets at will, and is achieved if the adversary is able to compromise both for any single query. An adversary that does not compromise both effectively models a non-colluding proxy and target.

### 4.4 Security Properties

In our ODoH model we prove the following statement:

**Core Lemma** *An adversary is unable to associate a connection between client and proxy with the corresponding query unless both the proxy and target are compromised.*

Our model also proves that if the adversary controls *all but one* servers, client privacy is preserved when its query is handled by the one uncorrupted server – regardless of the server being a proxy or a target. We also prove that the attacker can only learn the response by compromising the target or by having advance knowledge of the query. As we will discuss in Section 4.5 if we consider nonce reuse attacks against the AEAD this property does not hold. We suggest a fix and prove that with the fix this property holds even in the presence of such attacks.

Two additional lemmas are proved to ensure correctness: (i) the protocol runs to completion; and (ii) the client and target agree on the target's identity, the client and target's public keys, the query, and the response. The agreement property is an authentication lemma that proves either that the adversary corrupted the target, or the response was provided by the expected target. This ensures that there are no misdirection or message swap attacks, wherein the adversary reroutes messages to confuse the client about which target responded. The full details of the proof are public and made available online on GitHub [1].

### 4.5 A Replay Attack and Mitigation

Whilst performing our analysis, we discovered a replay attack in older versions of the ODoH protocol. In particular, malicious proxies can log and replay the encrypted client queries and force AEAD key and nonce reuse for the encrypted responses. This was because ODoH previously derived the target response key and nonce entirely from the client's query. If the target's response changed, as is expected with DNS, the encrypted message also changes. This allows the proxy to learn the XOR of two different response plaintexts, which breaks semantic security of response encryption. After discovering this issue, the protocol was modified such that targets include a fresh nonce in the response encryption key schedule. This nonce is sent alongside the ciphertext, allowing clients to derive the same secrets and decrypt the response. We were able to model both the attack and fix in Tamarin, proving that the fix prevents the attack. You can see the original Tamarin model in the `odoh.m4` file [1], and the fixed version in `odoh_fix.m4` [1].

## 5 Results

We furnish implementation details of the various ODoH components mentioned in Section 3.2, perform microbenchmarks and multi-point wide-area measurements to evaluate ODoH and present comparisons to other secure and anonymous DNS variants.

### 5.1 Implementation and Microbenchmarks

We implemented two interoperable variants the Oblivious Target and Proxy using both Go and Rust. The implementations were deployed and tested on Google Cloud using Google App Engine [17] and a serverless platform - Cloudflare workers [19].

We also implemented a client in Go with a command line interface similar to `dig`. The client performs ODoH queries to a chosen Oblivious Proxy and Oblivious Target. The client can optionally select proxy and target using a latency-based heuristic. For performance evaluation, a benchmarking submodule was also implemented within the client.

Our benchmarking tool launches $C$ client processes, each performing $N$ queries chosen randomly from the Tranco top million dataset [65] at a rate of $R$ DNS requests per minute.

Both the client and proxy can be configured to reuse HTTPS connections to avoid the extra costs of TCP and TLS handshakes across queries. We revisit these options on performance in Section 5.

To accelerate adoption we also integrated ODoH protocol support into popular open source client stub resolvers that support DoH or encrypted DNS protocols like DNSCrypt. We use these stub resolver implementations to perform page load time measurements presented in Section 5.7.

Our ODoH implementations across clients, proxy, and target domains are publicly available and have also been open-sourced [2].

**Cryptographic Compute Overheads** The default ciphersuite in our implementations is the HPKE ciphersuite as published in the IETF draft [6], consisting of the DHKEM(X25519, HKDF-SHA256) Key Exchange Mechanism (KEM), SHA256 based Key Derivation Function (KDF), and AES-128-GCM based Authenticated Encryption (AEAD) algorithms. This ciphersuite is performant and widely supported in production environments.

We ran our microbenchmarks on a 1 Intel Xeon 2.0 GHz CPU core with 3.75 GB of memory in a Google Compute Engine virtual machine. The compute overheads incurred by HPKE were evaluated by running microbenchmarks on the hosted client instances at sub-microsecond granularity. A set of 10,000 domains were randomly selected for this test from the Tranco million dataset [65]. For each domain the encryption and decryption operations are executed sequentially, i.e., encrypting then decrypting the query before encrypting then decrypting the response.

The main observations are summarized in Table 1, with entries at the $99^{th}$ percentile. DNS query encryption using the most performant ciphersuite (DHKEM(X25519, HKDF-SHA256), HKDF-SHA256, and AES-128-GCM) takes 360$\mu$s, while the decryption of such an encrypted query takes 246$\mu$s. The protocol also supports other cipher suites involving X448, P256, and P512 KEM; SHA384 and SHA512 KDF; and AES-256-GCM and ChaCha20Poly1305 AEADs. In some cases P256 curve might be preferred over the X25519 KEM for NIST and FIPS standards compliance [69]. For the symmetric key operations AES performs better than ChaCha20Poly1305 due to hardware acceleration in some platforms [50]. The slowest encryption times at the $99^{th}$ percentile is 430$\mu$s using the DHKEM(X25519, HKDF-SHA384) KEM with different possible permutations of the supported KDFs and AEADs; decryption is 713$\mu$s when using the SHA-384 and AES-256-GCM.

**Table 1.** 99th percentile (P99) Overheads incurred by the default cryptographic suite in our HPKE implementations. HPKE uses the DHKEM(X25519, HKDF-SHA256), HKDF-SHA256, and AES-128-GCM ciphersuite

| Microbenchmark | Type | P99 Overhead |
|---|---|---|
| **Query encryption time** | **HPKE** | **360$\mu$s** |
| **Query decryption time** | **HPKE** | **246$\mu$s** |
| **Query size (Type A)** | **HPKE** | **148 bytes** |
| **Answer size (Type A)** | **AES-128-GCM** | **98 bytes** |

Our results reveal least compute overhead with HPKE query encryption and AEAD response encryption. However, all the computation overheads in the life-cycle of an ODoH query, i.e., from query encryption at the client to decryption of the response, take less than 1ms.

**Cryptographic Network Overheads** Compared to baseline cleartext DNS messages, HPKE encryption increases the size of the query message on the wire by 4x, and the and the size of the response by 1.2x. This microbenchmark was performed at the client stub resolver. A random sample of 10,000 unique domains were used to serialize cleartext DNS and ODoH query messages. The average encrypted ODoH query size for type A queries is 314% greater at 140.8 bytes. The ODoH query response sizes for the same domains were closer in size to their cleartext counterparts, with a size increase averaging 22.37% to 87.5 bytes. Note that ODoH messages must encapsulate the client public key used to encrypt the query in addition to the query, itself, and appended to an integrity hash.

## 5.2 Measurement

Our measurement setup consisted of clients, proxies, and targets deployed across the USA, with additional locations in Canada and Brazil, as depicted in Figure 2. We deployed 9 client VMs into US Google Cloud data centers. All VMs had 1 Intel Xeon 2.0 GHz CPU core with 3.75 GB of memory on the x86_64 architecture [18], with an average bandwidth of 480 Mbit/s.

For each client, corresponding proxy and target instances were geo-replicated using serverless platforms to the nearest datacenter on the Cloudflare network. An additional proxy and a target were deployed using Google app engine to a Google Cloud datacenter on the west coast of the USA.

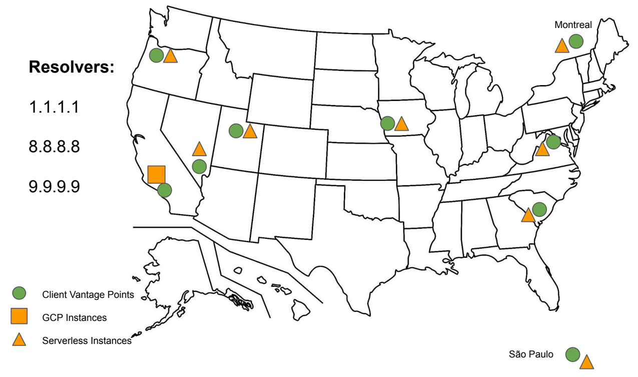


**Fig. 2.** Measurement Deployment: 9 ODoH clients deployed to Google data centers in the USA, Canada, and Brazil. Corresponding geo-replicated, serverless, proxies and targets were deployed to the nearest Cloudflare PoP. An additional proxy and target were deployed into GCP on the USA west coast. In ODoH a client can choose any combination of proxy, target, and resolver.

During the experiments, 10 query processes were launched by each of the 9 client stub resolvers. Each process mimics real client behaviour by making 200 ODoH queries at a rate of 15 requests per minute, or ∼21000 DNS requests/day. Parameter values were drawn from Wireshark packet traces of real clients [53, 70]. From each target, client queries were forwarded deterministically (`lastByte mod 3`) to open DoH resolvers (Cloudflare DNS, Google DNS and Quad9) to distribute load and mitigate against possible upstream effects of single-resolver selection.

Identical sets of queries to those launched by ODoH were also launched over 8 additional DNS variants, summarized by Table 2. We regard standard UDP DNS on port 53 (Do53) and DNS over HTTP (DoH) as our 'best-possible' measurements, in which clients direct each query randomly to one of the three resolvers.

To better characterize different aspects of the ODoH path, we implemented two artificial proxied variants that inject ODoH features to DoH in a stepwise fashion. First, in 'proxied DoH' (pDoH) the client requests proxies to forward Do53 queries as DoH queries to a chosen resolver; this is an architectural variant of the DoH protocol that introduces one additional hop between the client and recursive resolvers. The additional hop is then converted to DoH in 'cleartext ODoH', a two-hop DoH query without end-to-end encryption. Alongside, we evaluate ODoH when the target and resolver are co-located, the model that we anticipate will be dominant in practice.

The DoH and ODoH variants described above allow us to compare conventional DNS metrics, such as performance, but fail to capture properties related to privacy. For this reason we also mirror the same query measurements with DNSCrypt, which encrypts UDP queries between client and resolver. Both privacy-oriented Anyonymous DNSCrypt and DoH over the Tor network (DoHoT) introduce proxies to achieve privacy. We used dnscrypt-proxy [29, 38], an open source implementation of the DNSCrypt proxy, and restrict its use to only the DNSCrypt protocol; TCP was disabled, according to the original specification, and dnscrypt-proxy was configured to generate a new ephemeral key for each query for fair comparison against ODoH . Our stub resolver was configured to the lowest-latency DNSCrypt resolver from the public list of DNSCrypt resolvers available.

DoH over Tor (DoHoT) was granted similar performance benefits. Specifically, we allowed Tor to create its own optimal circuit, then performed DoH queries over Tor using the SOCKS5 proxy on each client instead of configuring them explicitly to use a static route. However, a limitation of our measurements with Tor are that the set of optimally chosen nodes might not all reside within the United States matching the comparisons to other protocols.

Measurements across DNS protocols are described in the remainder of this section. For additional reference, Table 2 lists the request path for each protocol with attributes and description.

## 5.3 Query Response Times

In this experiment we measure the time to resolve a query at the client, from initial request to response. We first look at various ODoH query-path selection strategies, and then compare ODoH query times with other DNS protocols and network architectural variants summarized in Table 2.

**Proxy and Target Selection** We configure the client stub resolver to select proxies and targets in three ways: (i) random selection of a proxy and a target pair; (ii) lowest-latency proxy with random target; and (iii) the lowest-latency proxy-target pair, i.e., fastest ODoH path to any target. The measured response times are shown in Figure 3, and indicate that the choice of proxy and target does impact ODoH performance. These measurements were performed with the separation of the target and resolver.

Observations from Figure 3 suggest the best overall selection strategy is lowest latency proxy-target pair. Intuitively this choice makes sense, and is most pro-

| Protocol | Query Path | Security | Privacy | Description |
|---|---|---|---|---|
| DoH - DNS over HTTPS [45] | C → R | Yes | No* | HTTPS based DNS over HTTPS |
| pDoH - Proxied DoH | C → P → R | Yes | No | Proxy performs DoH query |
| Cleartext ODoH | C → P → T → R | Yes | No | DoH query is proxied to a Target which performs DoH query |
| ODoH - Oblivious DoH [51] | C → P → T → R | Yes | Yes | Oblivious DoH protocol |
| Co-located ODoH (Section 5.4) | C → P → (T+R) | Yes | Yes | ODoH with Colocated Target and Resolver |
| DNSCrypt [38] | C → R | Yes | No* | UDP-based encrypted DNS |
| Anonymous DNSCrypt [28] | C → P → R | Yes | Yes | UDP based, Proxy routes client query |
| DoHoT - DoH over Tor [20, 62] | C → Tor → R | Yes | Yes | DoH over Tor Network |

**Table 2.** The set of *Secure* DNS platforms evaluated alongside ODoH with request path, attributes and description, for reference. Entries with citations are known and publicly available; entries without citations represent one-step at a time architectural changes towards ODoH and beyond. In the 'Query Path' column, C - Client, P - Proxy, R - Resolver, T - Target.

*-Guarantees of privacy depend on the resolver being used, and relies on legal agreements or privacy policies.

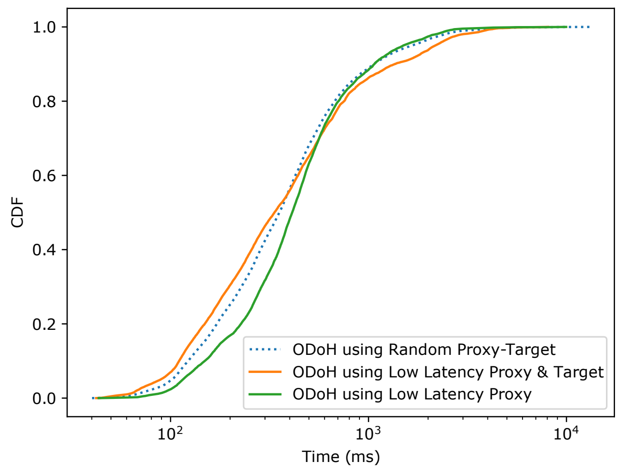


**Fig. 3.** ODoH query response times with different proxy and target selection strategies. Measurements suggest lowest total latency to target as the best strategy. The high performance of random selection is likely due to clusters of clients, proxies, and targets along eastern and western regions of North America.

nounced when the proxy is near- or on-path to the target. In these evaluations the median query to response time over the lowest latency proxy-target pair is 334.85 ms compared to 411.44 ms when solely selecting the lowest latency proxy. This marks a median improvement of 22.8%. We also observe that random selection is high-performing than averaging across all queries; we suspect this is because most clients in our deployment have a higher than average set of nearby proxies and targets from which to choose, i.e., clusters along the western- and eastern-halves of North America.

**Comparing DNS Platforms** Figure 4 shows the query performance of ODoH instantiations compared to other DNS platforms that are summarized in Table 2.

ODoH is the best performing privacy enhancing DNS protocol, ahead of DNSCrypt and DoH over Tor. Also, ODoH performance is only marginally worse than ODoH in cleartext, where a standard DoH query traverses the same logical 3-hop path as an ODoH query. This reinforces the minimal cryptographic compute overheads as presented in Section 5.1.

Observations from Figure 4 also reinforce that the proxy and target selection dominate performance. This can be seen by comparing standard DoH as the baseline to pDoH and ODoH variants. The median DoH response of 146.12ms increases by 49.5% to 218.52ms when the same DoH query is proxied (labeled as pDoH with the dotted black line). The same DoH queries increases the median response time by 109% when using the ODoH protocol to 305.11ms and in the case of Cleartext ODoH, when sent as DoH through both the proxy and the target, before being resolved by the resolver, we see an increase in median DNS response times by 103.2% to 297.015ms. We estimate, however, that the additional penalty of ODoH over pDoH is mitigated by co-locating the oblivious target with the resolver, as explained in Section 5.4.

If we focus on privacy-performance trade-offs, the ability of ODoH to match proxied DoH variants is particularly noteworthy: Compared to a median ∼50% increase in co-located ODoH queries, the median DNSCrypt query time of 487.68ms is 233.7% greater than DoH, while DoHoT is 376.5% greater at 696.40ms.

## 5.4 Co-locating Targets with Resolvers

The specification defines Oblivious Targets as being independent from resolvers, but allows for their co-location or integration. While we envision direct ODoH support in popular DoH providers, in practice, we make the same distinction as the specification in our evaluations – measurements labeled as ODoH consist of targets

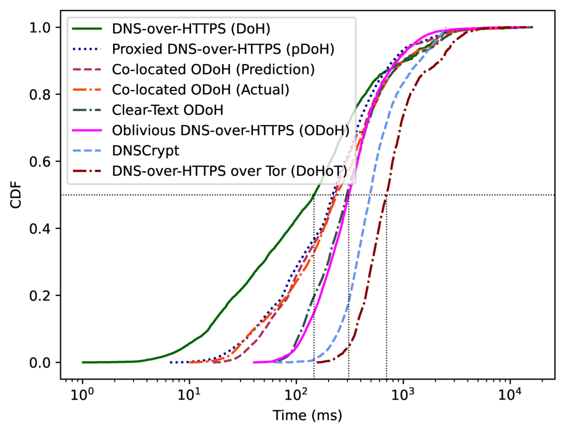


**Fig. 4.** Comparison of ODoH to other DNS Protocols. ODoH performance matches proxied DoH (pDoH) when the target and resolver are co-located. Note that both colocated ODoH and non-located ODoH variants provide privacy at a substantially reduced cost over other privacy-oriented protocols. (Left to Right order of curves maintained in legend order)

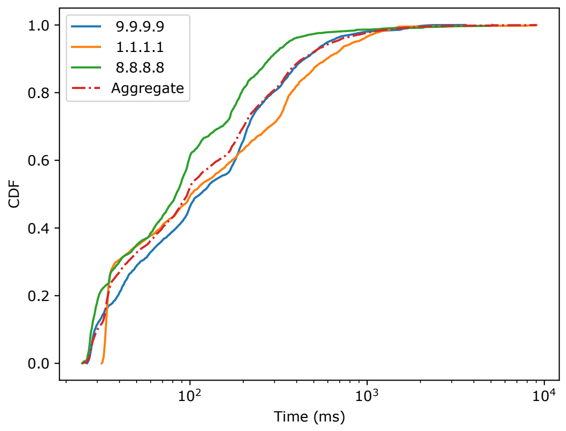


**Fig. 5.** The time taken to resolve DoH query from target instances to resolvers points to colocation benefits. Here, almost all queries from the target deployed in Google Cloud are fastest when directed to Google's own DNS resolver.

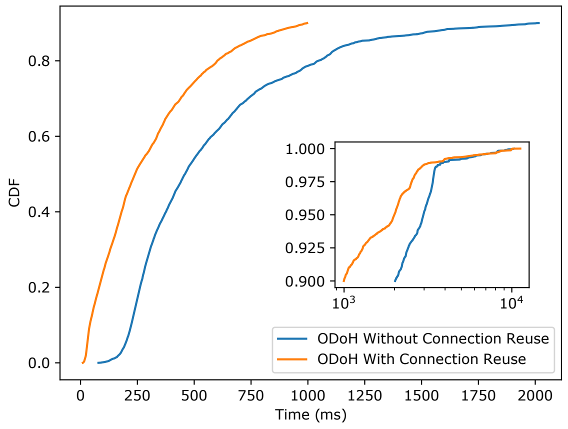


**Fig. 6.** ODoH Connection reuse: ODoH query resolution times increase if a new connections is used for each query.

that are physically separate and hosted individually, while *co-located ODoH* targets are implemented within resolvers' datacenter facilities or in the resolver services directly. Crucially, either instantiation preserves all previously stated security and privacy properties.

The effects of target location on performance are hinted at by closer inspection of the time it takes to resolve queries from any single target. For example, Figure 5 shows query resolution times from our target deployed inside Google Cloud datacenters. From this target, queries to Google's DNS service consistently resolve in less time than queries sent to resolvers not located within Google's datacenters.

We use this insight to estimate the values for co-located ODoH service in Figure 4 (indicated by the dashed blue line). Doing so allows us to isolate query resolution time at the resolver from the cost of the last target-to-resolver hop. We estimate the improvement by measuring a 'toy' recursive DNS resolver, in which queries resolved by a cache hit take less than 1ms at the $99^{th}$ percentile, and take 120ms with a cache miss. We conservatively assume a cache hit ratio of 70%, and use this value to estimate the lower-bound median response times of 228.52ms if Oblivious Targets were always co-located with the resolver.

We further validate these estimates with live measurements. The ODoH services were hardened for scale and implemented into a large public recursive resolver operated by Cloudflare, enabling us to measure ODoH with production targets. Both the predicted and measured values (purple dashed and dot line) are indicated in Figure 4. Their performance is similar to proxied DoH, with the median response time from co-located ODoH being 238.60ms, suggesting that predicted median value 228.52ms are reasonable, as are the potential performance improvements of co-location.

## 5.5 Connection Reuse

Persistent connections, or connection reuse, are HTTPS optimizations that enable clients to establish and re-use a connection to transmit multiple requests to the same host, rather than to open a new connection for each request.

In all of our experiments described from here, the proxy-target connections are persistent and use the co-located target case. However, the measurements pre-

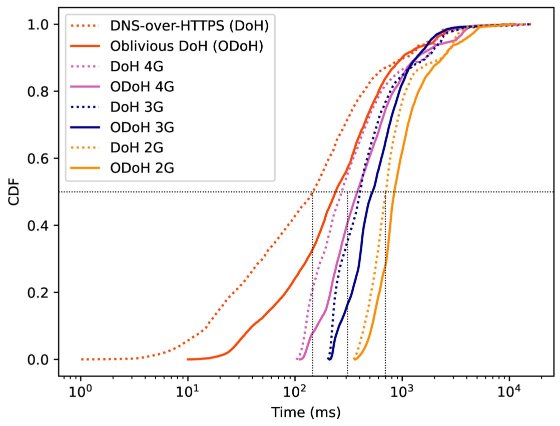
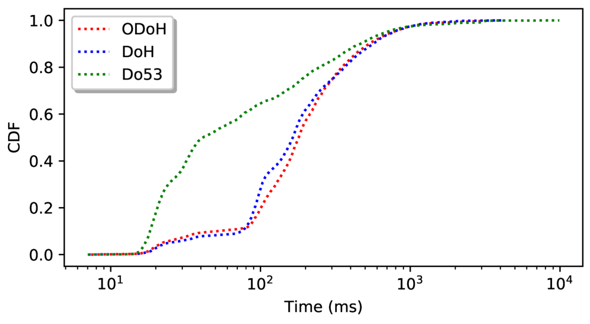


**Fig. 7.** Time to resolve a ODoH query over different network conditions. The Figure shows the comparison of the usage of ODOH and DOH due to various network types. (Left to Right order of curves maintained in legend order)

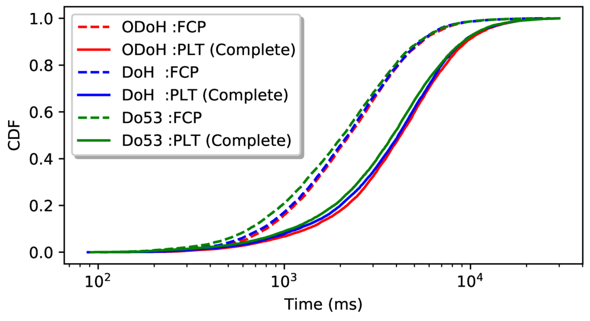
sented in Figures 3 and 4 benefit from persistent client-proxy connections for best performance. The use of persistent connections does not limit the ODoH security properties; however, we consider the impact of initiating a new HTTPS connection for each query in Figure 6. In our measurements the median query response times with a client creating a new connection for each request without reuse, increase by 92.5% from 238.60ms to 459.42ms. We revisit concerns due to connection reuse and re-iterate its benefits in Section 5.8.

## 5.6 Impact of Network Type

Previous work suggests that DoH performance is significantly impacted by poor mobile network connectivity [9]. We emulate similar network conditions to understand their effect on ODoH, with results summarized by Figure 7. Using the `qdisc` scheduler we configure and shape our network traffic by throttling egress bandwidth to (i) 0.56 Mbps with 350 ms latency for 2G, (ii) 1.25 Mbps with 200 ms latency for 3G, and (iii) 12 Mbps with 100 ms latency for 4G [55, 74]. Compared to the unthrottled colocated ODoH median response time of 238.06ms, the throttled ODoH response times increase by 59.3%, 121.5% and 252.8% for 4G, 3G and 2G network connections, respectively. DoH in the unthrottled case has a median response time of 146.12ms, and the throttled variants increase by 86.6%, 177.1%, and 382.6% respectively. However, in comparing ODoH against DoH under identical conditions, the performance gap diminishes. ODoH response times are 39.4%, 30.5%, and 19.4% higher compared to DoH in



**(a)** DNS response times for 10000 Page Loads from Local Stub Resolver



**(b)** First Contentful Paint (FCP) and Page Load Time (PLT) measurements for 10000 webpages based on DNS protocol used

**Fig. 8.** Impact of page load time due to usage of ODoH compared to other DNS protocols

the same 4G, 3G and 2G networks. This suggests that the network effects dominate over protocol differences.

## 5.7 Page Load Times with ODoH

Previous DoH evaluations concluded that the performance penalty of DNS-over-HTTPS over clear text DNS is absorbed into page load times with no significant perceived effect [12]. We similarly evaluate page load times with ODoH by replacing the DoH stub resolver in a popular open source DNS proxy with an ODoH compatible stub resolver.

In this experiment a set of 10000 websites selected, composed of the top 5000 from the top million Tranco dataset [65], and another 5000 randomly selected from the remainder of the list. Page load times are captured across three configurations for both the top 5000 and the randomly chosen 5000 websites: (1) Do53 to Cloudflare's 1.1.1.1; (2) DoH to 1.1.1.1; (3) ODoH with an off-path proxy to a colocated ODoH resolver; The client stub resolver reuses connections when possible to avoid additional TCP and TLS handshakes.

To measure page load times we use Google Chrome and capture the page load session into an HTTP Archive

(HAR) file additionally recording the results of the `window.performance` API [64]. Each experimental run is launched with a new profile to remove caching effects and uses the same set of 10000 websites loaded in the same order. This is done to ensure that the results captured during the page load time measurement does not include order effects. We believe that loading the same set of pages, from the same starting state, in the same sequence would balance out additional effects due to browser artifacts such as caching. Maintaining the sequence of page loads across all the experimental settings indicates the same cache state of the browser after each web page load with a high probability. Additionally, the local DNS stub resolver cache is flushed for each configuration forcing an empty starting state for each evaluation. Page load time is calculated from page load event information in the defined HAR format, and measure the DNS lookup times as the difference between `domainLookupEnd` and `domainLookupStart` events recorded by the browser. These values are validated with the DNS time for the request captured in the HAR file. This is recommended by the W3C standard for page navigation time and perceived page load time [79].

We present our results on page load times with different DNS protocols in Figure 8. Figure 8a shows the time taken to resolve queries for the websites. The median DNS response time when using Do53 is 41.28ms and increases by 298.01% to 164.30ms for DoH and 335.70% to 179.86ms for ODoH indicating a 9.47% difference between usage of DoH and ODoH. The CDF results for DoH and ODoH indicate a sharp increase close to 100ms possibly indicating an artifact of the local stub resolver.

Page load measurements for the websites are shown in Figure 8b. The median page load time when using Cloudflare DNS in cleartext UDP Do53 packets is 3830.22ms. We consider this the baseline and find that the median page load times increase by 6.76% when using the DoH protocol to Cloudflare resulting in 4089.35ms median page load time respectively. The usage of colocated ODoH with the resolver increases median page load times by 9.83% compared to the cleartext baseline usage to 4207.10ms. To understand the impact of colocation in page load performance, the usage of ODoH with separation between the target and the resolver increases the median page load time by 20.12% to 4601.03ms. At the 95th percentile, we notice that the difference between DoH and ODoH protocols compared to the baseline cleartext UDP based DNS protocol reduces to 0.66% and 5.86% respectively resulting in page load times of 11524.59ms and 12113.84ms when compared to 11448.89ms with Do53 Cloudflare DNS.

The performance difference between ODoH and DoH (and Do53) can be attributed to the network path topology differences. With an on-path proxy to the resolver, clients can benefit from slightly better page load times for the ODoH protocol than the measurements described in this section. The complete page load time measurements might skew results due to intensive client side javascript operations. To mitigate, in Figure 8b we also present our results for the First Contentful Paint (FCP) measurements for the website loads and identify that the median FCP time for client using the Do53 protocol is 2065.17ms and increased with the usage of DoH protocol by 6.15% to 2192.33, and ODoH by 8.03% to 2231.12ms which are in-line with the total page load time measurements indicating the impact of protocol choice.

Our results indicate that client choice of proxy and targets strongly impacts the user-facing performance of the ODoH protocol and the corresponding page load time measurements. Additionally, we show that co-location of the oblivious target services with the recursive resolver improves performance. Our experiments show that the median page load times increase by 2% when using ODoH compared to secure DNS protocols like DoH while providing both security and privacy guarantees making it a practical replacement for the Do53 DNS protocol and could be used by clients adopting DoH needing additional privacy guarantees. We further believe that the usage of ODoH will have no adverse impact on video streams or gaming since there is no constant need to rely on DNS once an IP address is obtained and a successful connection is established.

## 5.8 Discussion

**HTTPS Reuse** In our experimental evaluations, client stub resolvers try and reuse the HTTPS connection for multiple queries to amortize the cost of connection establishment. This is acceptable in practice since there is no correlation between client-proxy connections and proxy-target connections. In fact, if a proxy established and reused a per-client tunnel to upstream target resolver, it would be possible for the resolver to link all queries on that session to the same client. Additionally, the proxy is a TLS terminating proxy indicating that the TLS connection from the client is not passed through to the target resolver to establish an end to end

TLS connection. The usage of TLS passthrough proxies could leak client IP addresses. Clients could disable connection reuse and create new HTTPS connections for every query to the proxy, similar to DNSCrypt users. However, this would introduce substantial performance overhead as shown in Section 5.5, with no practical privacy improvement.

**Page Load Time Triggers** For the page load time measurements in Section 5.7, we use the full page load events obtained from the page load session timeline that is triggered after all the sub resources of the page have been obtained. This represents the worst case performance compared to other metrics such as First Meaningful Paint (FMP), and Time to Interactive (TTI) metrics which could be used for page load times. We believe and show that ODoH with colocation of the target with the resolver services, in addition to the existing usage of DNS resolvers like Cloudflare, NextDNS, or Google will result in better page load and DNS response times.

## 6 Wider Deployment Considerations

Our controlled experiments and experimental setup in Section 5 overcome or ignore a wider set of deployment challenges. Service availability and discovery, potential abuse or attack vectors, as well as organizational considerations are discussed in further detail below.

### 6.1 A Practical Roadmap for Adoption

ODoH activities at the IETF increase the likelihood of adoption, and are facilitated by a substantial overlap with DoH, which is increasingly being supported post-standardization. At time of writing, DoH is natively supported in iOS and Windows [31, 54], in applications such as Firefox and Chrome, and with client stub resolvers across multiple platforms. We share the following set of insights and experience from our own deployment.

**Target Integration into DNS Resolvers** As open anycast DNS resolvers continue to grow, it would make sense that operators reinforce promises of user privacy by supporting ODoH in their resolvers. Support begins with a test domain like `odoh.test`, and the corresponding DNSSEC signed HPKE public key information to be registered in the resolver for client discovery. In addition, the resolver should support ODoH messages with content type `application/oblivious-dns-message` on the DoH API interface, respecting the DNS wireformat. Lastly services are needed to decrypt queries and encrypt answers. Support for HPKE is available in popular cryptographic libraries like BoringSSL [40] and in language specific libraries written in Go or Rust [2].

**Proxy Operators** We expect that ODoH proxy service will primarily flourish as a value-added service offered to the customers of publicly funded national networks, and of data-center operators that provide infrastructure for private cloud and CDN networks; Internet Exchanges may also recognize value in offering ODoH proxy service to tenants and customers. The integration of ODoH into the tooling of DNSCrypt and its ecosystem also increases protocol adoption. Since having an on-path proxy improves performance, ISPs could provide ODoH proxy services to privacy-focused customers. On-path proxy service could run contrary to some ISP business models, however we believe that this may be tractable given recent efforts by Mozilla in onboarding Comcast, a large ISP in the USA, to Mozilla's Trusted Recursive Resolver list [59]. An additional incentive for ISPs may present in the form of regulatory backlash [9, 80]. We make it easy for users to deploy their own proxies to popular platform as a service (PaaS), and to deploy and provide ODoH proxy services to private home networks or communities [2].

**Client Stub Resolver Modifications** Clients can use ODoH in multiple ways. Support for the protocol could be implemented directly in the operating system stub resolver. Alternatively, ODoH service could be provided by a DoH local proxy DNS service that users install on their devices, such as dnscrypt-proxy, Acrylic (the experimental DoH-Proxy from Facebook), or applications like BraveDNS on Android devices [13, 21, 35]. Support for ODoH in these tools requires minimal changes, given existing support for DoH. We implemented the protocol in one such popular open source tool (`cloudflared`) and used it to perform page load time measurements. Firefox and Chrome browser support for DoH make them candidates for ODoH support, bypassing the need for users to rely on the operating system or a system resolver. In the case of both browser and operating system support, additional considerations need to be made regarding usability, ease of configuration for users, and the manner in which local proxy and target lists are populated.

**Proxy and Target Service Discovery** One open problem that is as old as DNS is service discovery. The

DNSCrypt community resorts to a published list of hostnames, which is difficult to maintain, and unlikely to scale. Another solution consists of sets of trusted proxies and targets that are curated by developers and presented to users in a manner akin to Mozilla's Trusted Recursive Resolvers program [60]. Alternatively clients might automatically upgrade DNS to ODoH and choose a public list of proxies if support is detected by the resolver, as recently introduced into Android for DoT [52].

## 6.2 Potential Attacks

Beyond the boundaries of ODoH privacy properties, certain attacks may be possible. These attacks are either prohibitively expensive, or are due to known vulnerabilities of the underlying mechanisms, as described below.

**Association Attacks** A requirement of ODoH is that per-query proxy and target pairs are selected from non-colluding parties. There remains, however, a few scenarios in which it becomes possible to link or infer queries with clients' subsequent request activity. For example, a selected target within the clients' ISP could conceivably be used by the ISP to link IP addresses returned by the target with subsequent requests to those addresses. An Oblivious Proxy selected within the ISP could be used in a similar manner. Specifically, an ISP might be able to deduce the contents of a query, and link that query to a client, if the subsequent request or its destination IP address reveal any information about the activity. The usage of ODoH in these scenarios increases the effort needed to reliably track clients and is no more or a risk than what already exists with DoH.

**Denial-of-Service** Resolvers may throttle IP addresses associated with malicious clients attempting to perform denial of service (DoS) attacks. ODoH would seem to shift this responsibility downstream to the Oblivious Proxy or Target that is receiving the malicious queries. Proxies should therefore implement rate limiting or other mitigation strategies like the usage of an allow list of targets to prevent clients from launching large numbers of DNS requests and choosing arbitrary targets in an attempt to perform DNS Tunneling attacks or as a form of abuse. Malicious clients can however distribute their requests to different proxies destined to the same targets. Targets should implement similar strategies to protect against malicious proxies, and contact proxy operators. Well-behaved clients that are negatively impacted by target rate-limiting of the selected proxy could shift queries to different proxies.

Alternatively, malicious clients could also encrypt malformed DNS queries or check for non-existent domains in an effort to make the resolver spend compute cycles in decryption of the message or spend time communicating with various name servers [14]. We argue that with reasonable DoS prevention in place for deployed artifacts, such amplification or denial of service attacks can be made impractical in practice.

**Public Key Linkability** ODoH targets may give each client a unique HPKE key for query encryption in an attempt to deanonymize them. This is a common problem with protocols of this form, including Privacy Pass [26]. Multiple mitigations exist against this type of attack. For example, a trusted third party may fetch target keys on behalf of multiple clients, so that a target is unable to uniquely tag clients with public keys. The exact mechanism for mitigating against this problem is a general problem that is far from unique to ODoH.

**Compromised Proxies and Targets** An adversary that wants to link queries to clients must compromise both proxy and target (see formal analysis in Section 4). Given the proxy's (IP, $C_Q$) tuples and the target's $(C_Q, Q)$ tuples, the attacker in possession can perform a `JOIN` to map client IP addresses to their queries. In a practical setting, a malicious proxy can correlate encrypted queries to encrypted responses $(C_Q, C_A)$ but does not gain much due to replay attack resistance. Network adversaries at the ingress and egress could perform correlation attacks but are practically limited due to encrypted transport enabled by TLS [3]. Like DoT and DoH, ODoH was never designed to prevent such attacks.

## 6.3 Organizational Implications

Encrypted DNS protocols pose challenges to enterprise system administrators that rely on DNS to keep organizations safe or provide controls that protect users inside the network from content deemed harmful or malicious. In these situations, we believe ODoH can be used to increase organizational safety. For example, an enterprise could use DoH for queries to the internal resolver, and use ODoH for any outbound queries from the organization's gateway. This protects potentially sensitive organization web traffic information from being recognized by upstream ISPs and other DNS name servers.

Techniques to protect Internet traffic while maintaining local control over DNS can also be implemented in a home network with tools such as PiHole [46]. For example, the Internet gateway for all web traffic from the home can convert insecure Do53 queries to ODoH.

## 7 Concluding Remarks

Cleartext DNS queries continue to be the most popular way in which users communicate with DNS ecosystem. In this paper we implement, analyze, and evaluate Oblivious DNS over HTTP. It is instructive to compare ODoH to its secure and private counterparts. Where DoT and DoH secure the DNS messages in transit, ODoH additionally guarantees that only the client has knowledge of both its query and its IP address. Our evaluations show that this level of privacy incurs marginal performance cost relative to DoH, while substantially out-performing privacy-oriented DNSCrypt and DoH over Tor. Our formal analysis finds and fixes an attack on a previous version of ODoH and demonstrates that the new version of ODoH achieves the desired privacy goals. We use our experience to identify ecosystem challenges and inform a roadmap to wide adoption.

Our implementations are available, open-source [2], and being used on the Internet to provide ODoH service. Overall, we find that ODoH has the potential to be deployed at scale with good performance, at the same time contributing to a safe and secure Internet.

## Acknowledgements

We would like to thank our shepherd, Tobias Pulls, and the anonymous reviewers for their valuable feedback. We would also like to thank Eric Kinnear, Tommy Pauly, Paul McManus, Edo Ryker, Alissa Starzak, and John Graham-Cumming for their suggestions throughout. In addition, Anbang Wen for the work to harden and transition ODoH services to production at Cloudflare, Joost Van Dijk for operating a production ODoH proxy at SURF NL, Stephen Spencer at UW CSE for provisioning measurement infrastructure. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

## References

[1] ODoH Analysis Tamarin Model. https://github.com/cloudflare/odoh-analysis.

[2] ODoH Artifacts. https://github.com/sudheesh001/ODoH-Artifacts.

[3] N Aifardan, D Bernstein, K Paterson, B Poettering, and J Schuldt. On the security of RC4 in TLS and WPA. In USENIX Security, 2013.

[4] Michael Backes, Aniket Kate, Praveen Manoharan, Sebastian Meiser, and Esfandiar Mohammadi. AnoA: A Framework for Analyzing Anonymous Communication Protocols. In 2013 IEEE 26th Computer Security Foundations Symposium, pages 163–178, 2013.

[5] Kenji Baheux. Chromium blog: A safer and more private browsing experience with secure DNS. https://blog.chromium.org/2020/05/a-safer-and-more-private-browsing-DoH.html, 05 2020. (Accessed on 09/15/2020).

[6] Richard Barnes, Karthikeyan Bhargavan, Benjamin Lipp, and Christopher A. Wood. Hybrid Public Key Encryption. Internet-Draft draft-irtf-cfrg-hpke-08, Internet Engineering Task Force, February 2021. Work in Progress.

[7] Daniel J Bernstein. DNSCurve: Usable security for DNS. dnscurve.org, 4, 2009.

[8] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In Designing privacy enhancing technologies, pages 115–129. Springer, 2001.

[9] Kevin Borgolte, Tithi Chattopadhyay, Nick Feamster, Mihir Kshirsagar, Jordan Holland, Austin Hounsel, and Paul Schmitt. How DNS over HTTPS is Reshaping Privacy, Performance, and Policy in the Internet Ecosystem. Performance, and Policy in the Internet Ecosystem (July 27, 2019), 2019.

[10] Stephane Bortzmeyer. DNS privacy considerations. Work in Progress, draft-ietf-dprive-problem-statement-06, 1, 2015.

[11] Stephane Bortzmeyer. Dns query name minimisation to improve privacy. RFC7816, 2016.

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

[13] BraveDNS. BraveDNS - A fast, secure, configurable, private DNS + Firewall for Android. https://www.bravedns.com/. (Accessed on 09/16/2020).

[14] Nevil Brownlee, Kimberly C Claffy, and Evi Nemeth. DNS measurements at a root server. In GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No. 01CH37270), volume 3, pages 1672–1676. IEEE, 2001.

[15] Sergio Castillo-Perez and Joaquin Garcia-Alfaro. Evaluation of two privacy-preserving protocols for the DNS. In 2009 Sixth International Conference on Information Technology: New Generations, pages 411–416. IEEE, 2009.

[16] A Chau and S Hertzberg. California Consumer Privacy Act of 2018 1798.140 (v). https://leginfo.legislature.ca.gov/faces/codes_displaySection.xhtml?lawCode=CIV&sectionNum=1798.140., 2018. (Accessed on 02/27/2021).

[17] Google Cloud. App Engine Application Platform - Google Cloud. https://cloud.google.com/appengine. (Accessed on 02/27/2021).

[18] Google Cloud. Google Compute Engine - Machine Types. https://cloud.google.com/compute/docs/machine-types. (Accessed on 09/16/2020).

[19] Cloudflare. Cloudflare Workers®. https://workers.cloudflare.com/. (Accessed on 09/15/2020).

[20] Cloudflare. DNS over Tor | Cloudflare Developer Docs. https://developers.cloudflare.com/1.1.1.1/fun-stuff/dns-over-tor/. (Accessed on 09/15/2020).

[21] Cloudflare. Argo Tunnel Client. https://github.com/cloudflare/cloudflared, 2020.

[22] Cas Cremers and Martin Dehnel-Wild. Component-based formal analysis of 5G-AKA: Channel assumptions and session confusion. In Network and Distributed Systems Security (NDSS) Symposium 2019, February 2019.

[23] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A comprehensive symbolic analysis of TLS 1.3. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 1773–1788, 2017.

[24] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two. In 2018 IEEE Symposium on Security and Privacy (SP), pages 108–126, 2018.

[25] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Comprehensive anonymity trilemma: User coordination is not enough. Proceedings on Privacy Enhancing Technologies, 2020(3):356–383, 2020.

[26] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. Proceedings on Privacy Enhancing Technologies, 2018(3):164–180, 2018.

[27] Selena Deckelmann. Firefox continues push to bring DNS over HTTPS by default for US users - The Mozilla Blog. https://blog.mozilla.org/blog/2020/02/25/firefox-continues-push-to-bring-dns-over-https-by-default-for-us-users/, 02 2020. (Accessed on 09/15/2020).

[28] Frank Denis. Anonymized DNSCrypt specification. https://github.com/DNSCrypt/dnscrypt-protocol/blob/master/ANONYMIZED-DNSCRYPT.txt, 06 2020. (Accessed on 09/15/2020).

[29] Frank Denis and Contributors. A flexible DNS proxy, with support for encrypted DNS protocols. https://github.com/DNSCrypt/dnscrypt-proxy/. (Accessed on 09/17/2020).

[30] Apple Developer. DNS Proxy Provider | Apple Developer Documentation. https://developer.apple.com/documentation/networkextension/dns_proxy_provider. (Accessed on 09/15/2020).

[31] Apple Developer. Enable encrypted DNS - WWDC 2020. https://developer.apple.com/videos/play/wwdc2020/10047/. (Accessed on 09/15/2020).

[32] Google DNS. Your Privacy - Public DNS - Google Developers. https://developers.google.com/speed/public-dns/privacy. (Accessed on 02/27/2021).

[33] Danny Dolev and Andrew Yao. On the security of public key protocols. IEEE Transactions on information theory, 29(2):198–208, 1983.

[34] Ralph Droms. RFC2131: Dynamic Host Configuration Protocol, 1997.

[35] Facebook. DNS Over HTTPS Proxy | Facebook. https://github.com/facebookexperimental/doh-proxy, 2020.

[36] Hannes Federrath, Karl-Peter Fuchs, Dominik Herrmann, and Christopher Piosecny. Privacy-preserving DNS: analysis of broadcast, range queries and mix-based protection methods. In European Symposium on Research in Computer Security, pages 665–683. Springer, 2011.

[37] Michèle Finck and Frank Pallas. They who must not be identified—distinguishing personal from non-personal data under the GDPR. International Data Privacy Law, 10(1):11–36, 03 2020.

[38] Frank Denis and Yecheng Fu. DNSCrypt: A protocol to improve DNS security. https://www.dnscrypt.org/, 02 2021. (Accessed on 02/20/2021).

[39] Google. DNS-over-HTTPS (DoH) | Public DNS | Google Developers. https://developers.google.com/speed/public-dns/docs/doh. (Accessed on 09/15/2020).

[40] Google. crypto/hpke - boringssl - Git at Google. https://boringssl.googlesource.com/boringssl/+/refs/heads/master/crypto/hpke/, 07 2020. (Accessed on 09/17/2020).

[41] John Graham-Cumming. Announcing the Results of the 1.1.1.1 Public DNS Resolver Privacy Examination. https://blog.cloudflare.com/announcing-the-results-of-the-1-1-1-1-public-dns-resolver-privacy-examination/, 03 2020. (Accessed on 09/15/2020).

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

[43] Christian Grothoff, Matthias Wachs, Monika Ermert, and Jacob Appelbaum. NSA's morecowbell: Knell for dns, 2015.

[44] Ansel Herz. Judge Who Authorized Police Search of Seattle Privacy Activists Wasn't Told They Operate Tor Network. https://web.archive.org/web/20191210114929/https://www.thestranger.com/slog/2016/04/08/23914735/judge-who-authorized-police-search-of-seattle-privacy-activists-wasnt-told-they-operate-tor-network/, 04 2016. (Accessed on 09/15/2020).

[45] Paul Hoffman and Patrick McManus. DNS queries over HTTPS (DoH). Internet Requests for Comments, IETF, RFC, 8484, 2018.

[46] Pi Hole. Pi-hole – A black hole for Internet advertisements. https://pi-hole.net/. (Accessed on 09/16/2020).

[47] Austin Hounsel, Paul Schmitt, Kevin Borgolte, and Nick Feamster. Measuring the Performance of Encrypted DNS Protocols from Broadband Access Networks, 2020.

[48] Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul Hoffman. Specification for DNS over transport layer security (TLS). IETF RFC7858, May, 2016.

[49] Mei Lin Hui and Gavin Lowe. Fault-preserving simplifying transformations for security protocols. Journal of Computer Security, 9(1-2):3–46, 2001.

[50] Franziskus Kiefer. Improving AES-GCM Performance - Mozilla Security Blog. https://blog.mozilla.org/security/2017/09/29/improving-aes-gcm-performance/, 09 2017. (Accessed on 09/16/2020).

[51] E. Kinnear, P. McManus, T. Pauly, and C. Wood. Oblivious DNS Over HTTPS–IETF Draft. https://tools.ietf.org/html/draft-pauly-dprive-oblivious-doh-01, 2019.

[52] Erik Kline. DNS over TLS support in Android P Developer Preview. https://android-developers.googleblog.com/2018/04/dns-over-tls-support-in-android-p.html, 04 2018. (Accessed on 09/15/2020).

[53] Ulf Lamping and Ed Warnicke. Wireshark user's guide. Interface, 4(6):1, 2004.

[54] Brandon LeBlanc. Announcing windows 10 insider preview build 20185. https://blogs.windows.com/windows-insider/2020/08/05/announcing-windows-10-insider-preview-build-20185/, 08 2020. (Accessed on 09/15/2020).

[55] Ken Lo. Download Speeds: Comparing 2G, 3G, 4G & 5G Mobile Networks. https://kenstechtips.com/index.php/download-speeds-2g-3g-and-4g-actual-meaning, 11 2018. (Accessed on 09/16/2020).

[56] Chaoyi Lu, Baojun Liu, Zhou Li, Shuang Hao, Haixin Duan, Mingming Zhang, Chunying Leng, Ying Liu, Zaifeng Zhang, and Jianping Wu. An End-to-End, Large-Scale Measurement of DNS-over-Encryption: How Far Have We Come? In Proceedings of the Internet Measurement Conference, pages 22–35, 2019.

[57] Electronic Frontier Foundation Marcia Hoffmann. Why IP Addresses Alone Don't Identify Criminals. https://www.eff.org/deeplinks/2011/08/why-ip-addresses-alone-dont-identify-criminals, 08 2011. (Accessed on 09/15/2020).

[58] Erika McCallister, Tim Grance, and Karen Scarfone. Guide to protecting the confidentiality of Personally Identifiable Information (PII): Recommendations of the National Institute of Standards and Technology. NIST special publication ; 800-122. Computer security. U.S. Dept. of Commerce, National Institute of Standards and Technology, Gaithersburg, MD, 2010.

[59] Mozilla. Comcast's Xfinity Internet Service Joins Firefox's Trusted Recursive Resolver Program - The Mozilla Blog. https://blog.mozilla.org/blog/2020/06/25/comcasts-xfinity-internet-service-joins-firefoxs-trusted-recursive-resolver-program/, 06 2020. (Accessed on 09/15/2020).

[60] Mozilla. Mozilla Policy Requirements for DNS over HTTPs Partners. https://wiki.mozilla.org/Security/DOH-resolver-policy, 09 2020. (Accessed on 09/15/2020).

[61] Alec Muffet. No Port 53, Who Dis?; A Year of DNS over HTTPS over Tor. In NDSS DNS Privacy Workshop, 02 2021.

[62] Alec Muffett. DoHoT: making practical use of DNS over HTTPS over Tor. https://github.com/alecmuffett/dohot, 07 2020. (Accessed on 09/15/2020).

[63] NIST. NVD - CVE-2013-2566. https://nvd.nist.gov/vuln/detail/CVE-2013-2566, 03 2013. (Accessed on 09/16/2020).

[64] Jan Odvarko. HAR 1.2 Spec. http://www.softwareishard.com/blog/har-12-spec/. (Accessed on 02/28/2021).

[65] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. Network and Distributed Systems Security (NDSS) Symposium, 2019.

[66] Matthew Prince. Introducing 1.1.1.1 for Families. https://blog.cloudflare.com/introducing-1-1-1-1-for-families/, 2020.

[67] Chromium Projects. DNS over HTTPS (aka DoH). https://www.chromium.org/developers/dns-over-https. (Accessed on 09/15/2020).

[68] DNSCrypt Proxy. Anonymized DNS Wiki. https://github.com/DNSCrypt/dnscrypt-proxy/wiki/Anonymized-DNS. (Accessed on 09/15/2020).

[69] FIPS PUB. Security Requirements for Cryptographic Modules. FIPS PUB, 140, 1994.

[70] Reddit Communities. DNS query average : PiHole. https://www.reddit.com/r/pihole/comments/a8ngnu/dns_query_average/, 12 2018. (Accessed on 09/15/2020).

[71] Benedikt Schmidt, Simon Meier, Cas Cremers, and David Basin. Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties. In Stephen Chong, editor, 25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012, pages 78–94. IEEE, 2012.

[72] Paul Schmitt, Anne Edmundson, Allison Mankin, and Nick Feamster. Oblivious DNS: Practical Privacy for DNS Queries: Published in PoPETS 2019. In Proceedings of the Applied Networking Research Workshop, ANRW '19, page 17–19, New York, NY, USA, 2019. Association for Computing Machinery.

[73] Benjamin M. Schwartz, Mike Bishop, and Erik Nygren. Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs). Internet-Draft draft-ietf-dnsop-svcb-https-03, Internet Engineering Task Force, February 2021. Work in Progress.

[74] Milan P Stanic. TC–Traffic Control. Linux QOS Control Tool, 2001.

[75] Srikanth Sundaresan, Nazanin Magharei, Nick Feamster, Renata Teixeira, and Sam Crawford. Web performance bottlenecks in broadband access networks. In Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems, pages 383–384, 2013.

[76] TracBot. Tor blocked in UAE (#25137) · Issues · Legacy / Trac · GitLab. https://gitlab.torproject.org/legacy/trac/-/issues/25137, 02 2018. (Accessed on 09/15/2020).

[77] European Union. What is considered personal data under EU GDPR. https://gdpr.eu/eu-gdpr-personal-data/. (Accessed on 02/27/2021).

[78] Upturn. What ISPs Can See. https://www.upturn.org/reports/2016/what-isps-can-see/, 03 2016. (Accessed on 09/15/2020).

[79] Zhiheng Wang. Navigation Timing - World Wide Web Consortium (W3C). https://www.w3.org/TR/navigation-timing/, 12 2012. (Accessed on 09/17/2020).

[80] Nicholas Weaver, Christian Kreibich, and Vern Paxson. Redirecting DNS for Ads and Profit. FOCI, 2:2–3, 2011.

[81] Xynou, Maria, and Filasto, Arturò. Iran Protests: OONI data confirms censorship events (Part 1) | OONI. https://ooni.org/post/2018-iran-protests/. (Accessed on 09/15/2020).

[82] Fangming Zhao, Yoshiaki Hori, and Kouichi Sakurai. Analysis of privacy disclosure in DNS query. In 2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07), pages 952–957. IEEE, 2007.

[83] Fangming Zhao, Yoshiaki Hori, and Kouichi Sakurai. Two-servers PIR based DNS query scheme with privacy-preserving. In The 2007 International Conference on Intelligent Pervasive Computing (IPC 2007), pages 299–302. IEEE, 2007.

[84] Liang Zhu, Zi Hu, John Heidemann, Duane Wessels, Allison Mankin, and Nikita Somaiya. T-DNS: Connection-oriented DNS to improve privacy and security. ACM SIGCOMM Computer Communication Review, 44(4):379–380, 2014.