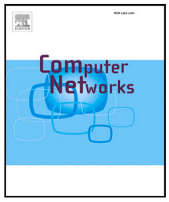
Computer Networks 237 (2023) 110078

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



μODNS: A distributed approach to DNS anonymization with collusion resistance☆

Jun Kurihara [a,b,*], Toshiaki Tanaka [a], Takeshi Kubo [b]

[a] Graduate School of Information Science, University of Hyogo, 7-1-28 Minatojima-minamimachi, Chuo-ku, Kobe, 650-0047, Hyogo, Japan
[b] Zettant Inc., 1-8-1-8F BcH, Nihonbashi-Kayabacho, Chuo-ku, 103-0025, Tokyo, Japan

ARTICLE INFO

Keywords:
Domain Name System (DNS)
Oblivious DoH
DNSCrypt
Anonymity
DNS privacy
Collusion resistance

ABSTRACT

The traditional Domain Name System (DNS) lacks fundamental security and privacy features in its design. As privacy concerns increased on the Internet, security and privacy enhancements of DNS have been actively investigated. Specifically, in the context of user privacy in DNS queries, several relay-based anonymization schemes have been recently introduced. However, these schemes are vulnerable to collusion between relays and full-service resolvers, which means user identities cannot be hidden from resolvers. This paper introduces a new concept for achieving user anonymity in DNS queries through a multiple-relay-based approach, called *μODNS* (*Mutualized Oblivious DNS*), by extending the concept of existing relay-based schemes. μODNS introduces a reasonable assumption that each user has at least one trusted or dedicated relay within the network and mutually shares the relay with other users. The user simply sets his trusted relay as the *next-hop relay* to convey his queries to the resolver and randomly chooses its *zero or more* subsequent relays shared by other entities. Under this assumption, the user's identity remains concealed from the target resolver in μODNS even if an unknown subset of relays colludes with the resolver. Namely, in μODNS, users can preserve their anonymity by paying a small cost of sharing their resources. Additionally, we extend existing protocols, Anonymized DNSCrypt and Oblivious DoH, to provide practical Proof-of-Concept specifications and implementations as instances of μODNS. These implementations are publicly available on the Internet as open-source software and *public services*. Furthermore, we demonstrate, through measurements of round-trip times for DNS messages, that our implementation can minimize the performance degradation resulting from its privacy enhancements, achieving performance levels that maintain the positive user experiences observed in existing schemes.

1. Introduction

Due to the recent increase and exposure to Internet censorship, users have been more concerned about their online activities being monitored, and demand technologies designed to protect their privacy. *Domain Name System* (DNS) is one of the components of the Internet that is being actively extended and enhanced in this context.

DNS plays a role in mapping human-readable hostnames to machine-readable information on the Internet; a user (or a *stub resolver*) exchanges a hostname with its IP address and associated resource records by querying a *full-service resolver*. In the traditional DNS called Do53, this exchange of DNS messages, i.e., a user's *query* and a resolver's *response*, is performed over UDP or TCP on port 53 in the form of plaintext. This means that in Do53, the user's Internet activity could be easily exposed to monitoring authorities.

In order to overcome such a lack of privacy in the classic design of DNS, there have been proposed some *encryption* schemes for DNS messages [2–5] to avoid them from being wiretapped. In these schemes, a secure encryption channel is established by a certain public key cryptography between a user and an encryption-enabled full-service resolver, and DNS messages are exchanged via the channel. Hence these schemes can conceal users' activities in DNS from censorship authorities monitoring messages transported. However, another privacy concern has been raised here even if we employ these encryption techniques: While DNS messages are encrypted, full-service resolvers always see

☆ The material of this paper was presented in part at arXiv [1] in 2021. A part of this work was done during the first author's stay at CyLab, Carnegie Mellon University, PA, USA in 2022. Project web page: https://junkurihara.github.io/dns/. Source codes (https://github.com/junkurihara/encrypted-dns-server-modns, https://github.com/junkurihara/dnscrypt-proxy-modns, https://github.com/junkurihara/doh-auth-proxy, https://github.com/junkurihara/doh-server) and experimental nodes (https://github.com/junkurihara/experimental-resolvers) are also publicly available.

* Corresponding author at: Graduate School of Information Science, University of Hyogo, 7-1-28 Minatojima-minamimachi, Chuo-ku, Kobe, 650-0047, Hyogo, Japan.
E-mail addresses: kurihara@ieee.org (J. Kurihara), toshi@gsis.u-hyogo.ac.jp (T. Tanaka), t-kubo@zettant.com (T. Kubo).

https://doi.org/10.1016/j.comnet.2023.110078
Received 23 May 2023; Received in revised form 22 September 2023; Accepted 27 October 2023
Available online 2 November 2023

Table 1
Terminologies and conceptual words related to anonymization in DNS.

Term	Description
Anonymization	The process of concealing or obfuscating the origin of a DNS query, i.e., the user's identity, rendering it challenging or infeasible to trace it back to its source.
Relay	A network node that simply forwards an incoming message basically without disclosing the previous node's identity to the next node. This paper supposes that when such a node colludes with a resolver, the previous node's identity is exposed to the resolver.
Mixing queries	A relay's operation that unintentionally puts messages incoming from multiple nodes to a single queue and forwards each message towards its next destination node from the queue. When the relay serves messages received from multiple nodes, each destination node is unable to uniquely identify the specific source node behind the relay for each message.
Collusion resistance	The protocol and system's ability to withstand collusion among the target resolver and compromised relays. The goal of collusion resistance is to maintain the user's privacy, i.e., conceal the origin of DNS queries, despite the potential cooperation of colluded nodes.

plaintext messages from/to users by the nature of DNS, and they can fully observe any user activities in DNS.

In order to resolve this second privacy concern against full-service resolvers, there have been proposed several *anonymization* techniques [6–11] to hide a user's IP address from a full-service resolver. Table 1 summarizes the terminologies and conceptual words related to anonymization in DNS. *Oblivious DNS* (ODNS) [6] is the first anonymization scheme for DNS, which was designed to be compatible with the standard DNS architecture. In the scheme, an ODNS-specialized resolver is introduced, and an existing full-service resolver over Do53 is leveraged as a *relay* forwarding encrypted DNS messages from/to a user to/from the ODNS resolver. In other words, an encrypted channel is built between a user and the ODNS resolver, encrypted DNS messages are exchanged via the existing full-service resolver over Do53, and the ODNS resolver resolves plaintext DNS queries on behalf of the full-service resolver. Thus the user's address is concealed from the ODNS resolver thanks to the relay, and DNS messages are hidden from the relay by the encryption as well. By adopting this relay-based concept into encrypted DNS schemes [3,5], *Anonymized DNSCrypt* (ADNSCrypt) [10,11] and *Oblivious DNS over HTTPS* (ODoH) [7–9] have recently been introduced. These schemes omit the compatibility with Do53, i.e., existing Do53 resolvers are NOT reused, unlike ODNS. Namely, they simply introduce dedicated intermediate relays between users and encryption-enabled resolvers. This simplified architecture results in their good performance comparable to standard encrypted DNS schemes [9].

There exists one significant drawback in such novel DNS anonymization schemes using relays: The lack of *collusion resistance*, i.e., the privacy could be completely corrupted when the relay colludes with the target resolver. Considering the current deployment of DNS, users do not have various choices of full-service resolvers enabling DNS message encryption and usually use ones operated by large and limited entities [12], e.g., Google, Cloudflare, Quad9, etc. Much like encryption-enabled resolvers, relays would be operated by such limited big players as mentioned in [9, Section 7.1]. Indeed, as far as we know, just a few big CDN providers provide such relays in Apple iCloud Private Relay, a deployment of ODoH. Hence it may increase concerns about collusion and surveillance. Considering a case where a user does not trust such existing relays, building a relay *dedicated* to the user should be the simplest solution. However, this is completely useless since the relay's address exposed to the target resolver can be uniquely coupled with the user itself. In other words, every user must *unconditionally* trust and choose public and shared relay forwarding its message when we employ these anonymization schemes. We thus have no way to fundamentally remove the concern about the collusion in these schemes.

In this paper, our objective is to address the aforementioned issues related to collusion in existing anonymization schemes and to present a practical solution that ensures user anonymity, even when users are unable to place trust in the majority of network nodes involved in DNS. To achieve this goal, we make the following contributions in this paper:

Proposal of μODNS Concept: This paper introduces a new architectural concept, termed μ*ODNS* (*Mutualized Oblivious DNS*), aimed at preserving user anonymity in DNS messages. μODNS is designed to safeguard the user's anonymity in the context of DNS even when facing untrusted resolvers and potentially colluding network nodes. It achieves this by making a small and reasonable assumption concerning network nodes proximate to a user. Additionally, μODNS aims to minimize the performance degradation resulting from privacy enhancements. Architecturally, μODNS can be seen as an extension of existing relay-based schemes, enabling multiple relays and route randomization. In this sense, it adopts an approach akin to Tor [13], but with a highly specialized and streamlined focus on DNS. Within the μODNS framework, *each user can maintain the anonymity of DNS messages by incurring only a minor cost, even in scenarios where certain network nodes collude with the target resolver*. Furthermore, it ensures that no user can compromise the anonymity of others.

Explicit PoC Design and Implementation: This paper provides a comprehensive exposition of the design and implementation of μODNS as Proof-of-Concept (PoC). It achieves this by extending existing DNS anonymization schemes, specifically ADNSCrypt [10,11] and ODoH [7–9], to effectively realize the concept of μODNS while maintaining compatibility with the original schemes.

Internet-Based Performance Evaluation: Additionally, this paper conducts a performance evaluation of μODNS using the PoC implementation deployed on the Internet. The results of this evaluation demonstrate that μODNS can minimize the performance degradation caused by its privacy enhancements, achieving performance levels comparable to the underlying original protocol.

In summary, this paper contributes by introducing the μODNS concept and providing its explicit implementations and evaluations. Based on these contributions, we claim that our anonymization concept is practical and reasonable, addressing both its design and performance aspects in the context of preserving user privacy in the severe network environment.

The remainder of this paper is organized as follows: Section 2 introduces the background of our work, summarizes recent studies on privacy in DNS, and explains the potential problems of collusion in existing schemes. Section 3 presents assumptions and formally introduces problems considered in this paper. Under the given assumptions, Section 4 overviews our concept of μODNS and explains how it works in the presence of colluded nodes. Section 5 briefly explains instances of μODNS based on ADNSCrypt and ODoH as its Proof-of-Concept implementations. Section 6 gives some discussions on μODNS and its PoC implementations from the viewpoints of security, privacy, performance, and deployment. Section 7 finally concludes this paper.

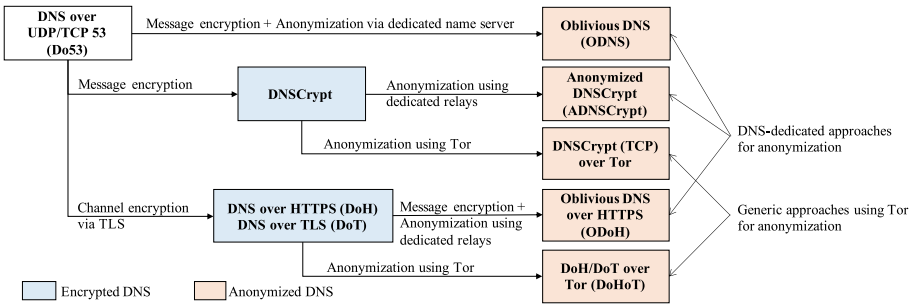


Fig. 1. Relationship among existing encrypted and anonymized DNS protocols.

2. Background and related work

Domain Name System (DNS) [14,15] was originally standardized at IETF as a system that maps human-readable hostnames to machine-readable resource records such as IP addresses. Since the traditional DNS has architectural problems causing security and privacy issues, several enhancements of DNS protocols have been investigated and incrementally deployed on the Internet. For instance, there is no guarantee of the authenticity of resource records in the original DNS protocol, and hence *DNS Security Extension* (DNSSEC), e.g., [16–18], has been proposed to protect users from attacks like DNS cache poisoning.

Recently, as increasing demands for privacy on the Internet, e.g., [19], another design problem of DNS has been drawing attention, which is the lack of *confidentiality* and *anonymity* of exchanged messages. In the following, we shall summarize recent efforts on DNS protocols for the problem. Fig. 1 briefly illustrates the relationship among such DNS protocols we shall explain in the following.

2.1. Encrypted DNS protocols

In the traditional DNS via UDP or TCP of port 53 (Do53), query and response messages are exchanged between a user and a full-service resolver in *plaintext*. This implies that with Do53, the user's activity on the Internet can be easily exposed to eavesdroppers on a channel. Namely, an authority observing the channel between the user and the resolver can immediately censor DNS messages. Thus for user privacy, several *encrypted DNS* protocols have been investigated and implemented [2–5] to protect DNS messages from being eavesdropped.

In *DNSCurve* [2] and its successor *DNSCrypt* [3], a user and a full-service resolver exchange their public keys and encrypt query and response messages with the keys. Then the encrypted messages are transported simply over UDP or TCP in a structured format. On the other hand, *DNS over TLS* (DoT) [4,20] is based on the public key infrastructure (PKI): a secure channel of transport layer security (TLS) is first established between a user and a full-service resolver, and then they securely exchange query and response messages over the secure channel. *DNS over HTTPS* (DoH) [5] leverages a TLS connection as the underlying secure channel similarly to DoT, but the DNS message exchange in DoH is executed through HTTP POST or GET methods over the secure channel, i.e., in the context of HTTPS. We note that in these protocols, query and response messages are directly exchanged between a user (a stub resolver) and a full-service resolver as well as Do53. In general, users need to utilize *public resolvers*, e.g., Google, Cloudflare, and Quad9, instead of ISPs' resolvers to enable these encrypted DNS protocols [12].

2.2. Anonymized DNS protocols

The utilization of encrypted DNS protocols helps users protect their privacy from being exposed to eavesdroppers observing messages on a transport channel. However, full-service resolvers always learn the content of DNS query messages of users to search associated resource records due to the nature of the DNS mechanisms. Also, recall that users exchange messages directly with full-service resolvers in the protocols of Do53 and even encrypted DNS given above. This means that in such protocols, the target resolver can always associate every query message uniquely with its issuer's identity, e.g., the user's IP address. Thus several anonymization techniques for DNS queries have been recently investigated as enhancements of encrypted DNS, i.e., as an extra layer for anonymization, to decouple the user's identity from queries observed at the target resolver.

We start by mentioning the employment of Tor for DNS, i.e., *DoH/DoT over Tor* (DoHoT) or *DNSCrypt over Tor*, which should be the most straightforward technique for such anonymization of DNS queries. In these schemes, we can simply anonymize DNS queries against target resolvers by leveraging Tor as the message transport channel of any TCP-based[1] encrypted DNS protocol much like HTTP messages. This simple and generic approach based on Tor usually involves a significant performance loss due to various reasons such as the overhead of multiple-layered encryption, a large volume of traffic at Tor nodes, and large round-trip time (RTT) among geographically distributed nodes. Unlike this generic approach based on Tor, there have been proposed several anonymization protocols *dedicated to DNS* as follows.

Oblivious DNS (ODNS) [6] is the first protocol specialized for DNS anonymization. The protocol introduces an authoritative name server of a special top-level domain "`.odns`". Each user generates a query to a domain of `.odns` containing an original query in encrypted form, and every query to the domain reaches the ODNS's name server. The name server decrypts the received encrypted query, dispatches the plaintext version of the original query upstream on behalf of the user, encrypts the response from the upstream, and sends it back to the user. Then, a full-service resolver located between the user and the ODNS name server can be viewed as a *relay* concealing the user's IP address from the name server observing plaintext queries. On the other hand, the full-service resolver cannot observe original DNS messages thanks to the encryption between the user and the ODNS name server.

Although ODNS took an elegant approach compatible with the standard Do53, it involves a large RTT since all queries must be forwarded to the ODNS's authoritative name server. *Oblivious DNS over HTTPS* (ODoH) [7–9] has been designed by simplifying the ODNS's architecture and omitting the compatibility with Do53 and realizes a performance comparable to the standard DoH; The architecture of ODoH introduces its dedicated network node called *oblivious proxy* that just relays DNS messages encrypted between a user and a target resolver. Additionally in ODoH, the target resolver is allowed to be separated into two nodes: A (Do53) full-service resolver and an encryption/decryption terminal called *oblivious target*, where the oblivious target sends decrypted queries to the resolver on behalf of the user. *Anonymized DNSCrypt* (ADNSCrypt) [10,11] has been designed under the concept fundamentally the same as ODoH from the viewpoint of

[1] Tor supports only TCP. Thus, TCP-based DNSCrypt is employed for DNSCrypt over Tor.

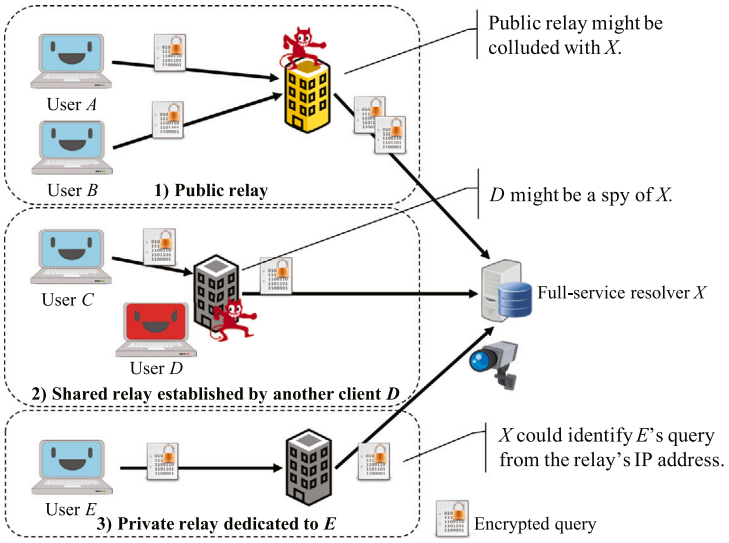


Fig. 2. Illustration of potential problems related to collusion between relays and target resolvers in existing relay-based schemes for user anonymity, considering the perspective of relay operators.

its architecture, i.e., leveraging an ADNSCrypt-dedicated relay node to hide the user's identity from the encryption-enabled target resolver. The architectural difference between ADNSCrypt and ODoH is only the existence of channel encryption based on TLS (HTTPS). We can see that an ODoH relay, i.e., oblivious proxy, simply terminates a TLS channel from a user and forwards encrypted DNS messages upstream over another TLS channel to the oblivious target much like an HTTPS proxy. On the other hand, encrypted messages in ADNSCrypt are transported directly over UDP or TCP.

In ODoH and ADNSCrypt, a user selects a single relay through which every encrypted DNS message to and from the user is routed. This relay effectively conceals the user's IP address from the observation of the target resolver, thereby decoupling the user's IP address from DNS queries. As mentioned in [9], user anonymity remains intact unless the chosen relay colludes with the target resolver.

2.3. Potential problem of collusion in existing anonymized DNS protocols

The deployment scenarios of ODoH relays, as discussed in [9], often involve operators of existing public resolvers. In the case of ADNSCrypt, relays are also primarily operated by individual entities,[2] akin to Tor relays, which may be unknown to users. Consequently, the user's anonymity in both of these schemes relies solely on third parties, and *users must place unconditional trust in these entities to preserve their anonymity*. It is worth noting, however, that such unconditional trust may not always be guaranteed, especially when considering big players and unknown operators providing public resolvers.

Additionally, it is important to recognize an implicit assumption in ODoH and ADNSCrypt concerning the mixing of queries, c.f. Table 1, at relays.: *The chosen relay is utilized by multiple users, ideally a large number of them. This requirement ensures that the relay's identity, such as its IP address, is not uniquely linked to an individual user's identity.*

From these implications, we can identify potential concerns regarding collusion between a relay and a target resolver, summarized in Fig. 2, based on the types of relay operators:

(1) Using a public relay: This is the most common scenario in existing schemes, where a public relay is expected to serve multiple users. In such cases, users must place unconditional trust in the relay's operator, as previously stated.

(2) Using a shared relay: Even when opting for a *shared* relay operated by another user, concerns related to public relays persist. Choosing a relay operated by a different entity does not eliminate the risk of potential surveillance, as the selected relay could still collude with the full-service resolver. In essence, placing unconditional trust in such a third-party entity remains a necessity for the user.

(3) Using a private relay: The only way to alleviate concerns related to public or shared relays is to choose and dedicate a relay trusted by each user. However, this approach proves completely impractical due to the risk of uniquely binding the chosen relay's identity with the user's identity by the target resolver.

In the subsequent section, we shall introduce a new anonymized DNS protocol specifically crafted to tackle these concerns and mitigate the risk of surveillance, even in cases of collusion on the Internet.

Remark 1. The problems given here might not happen in DoHoT and DNSCrypt over Tor due to the nature of Tor. However, it involves incredible performance degradation as we stated in Section 2 and will demonstrate in Section 6.2, and it is not a realistic approach for real-time processing in DNS-like systems. Indeed DNS-specific relay-based schemes have significantly outperformed a Tor-based scheme. Hence, we focus on the problem of collusion in the DNS-specific relay-based schemes.

3. Problem formulation

In Section 2.3, we mentioned the potential problem of collusion in existing anonymized DNS protocols. This section formally defines the assumed environment and the problem of the anonymity of DNS messages, which this paper will try to solve.

Firstly, we introduce formal assumptions of the network considered in the following sections. Most parts of our assumptions are almost the same as those in ODoH [7–9] and ADNSCrypt [10,11]. Assume that a user will use a *public resolver* as a full-service resolver such as Google DNS and Quad9, and that multiple (ideally large number of) users join the network. Also, assume that there are network nodes called *relays* that simply forward incoming encrypted DNS messages upstream or downstream. As in ODoH and ADNSCrypt, relays are *semi-honest*, i.e., they work correctly but they may try to observe the content of messages. Hence, to avoid relays from observing messages,

[2] c.f., the list of public DNSCrypt relays: https://github.com/DNSCrypt/dnscrypt-resolvers/blob/master/v3/relays.md in [21].

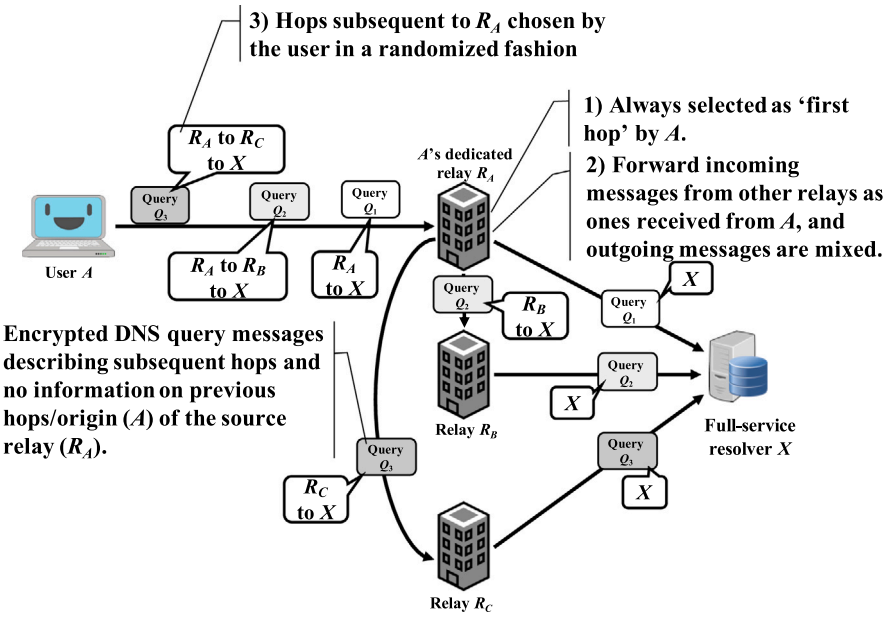


Fig. 3. Key features (1), (2), and (3) of μODNS in an exemplary scenario, where each feature corresponds to features (1), (2), and (3) enumerated in Section 4. This example supposes that user A issues queries Q_1, Q_2, and Q_3, which are routed towards the full-service resolver X through several relays, R_A, R_B, and R_C, along their path.

the user exchanges DNS messages with the target resolver encrypted in an end-to-end manner. This paper mainly focuses on attackers observing messages at public resolvers and relays, and we do not consider *wiretappers* who observe transit channels.

In addition to the above assumptions inherited from those in ODoH and ADNSCrypt, this paper supposes that for a user, *a relay operated by another entity can collude with the public resolver*. Also, suppose that the location of the colluded relay(s) is unknown to the user. Here we suppose, as a realistic assumption, that a tiny subset of relays potentially colludes with the target resolver. Unlike the settings of ODoH and ADNSCrypt, we do NOT assume relays are unconditionally trusted by a number of users. This means that, for example, even if a relay is trusted by a user, it is NOT usually trusted by others.

The problem we aim to address in this paper is how to maintain user anonymity in DNS even in the severe environment we assumed above. In this environment, each user should be able to exchange encrypted DNS messages with a targeted public resolver in a manner that prevents the resolver from identifying the user based on received messages and information provided by colluding relays. To achieve this, our aim is to design a scheme that practically conceals the user's identity from the target resolver in the assumed environment. In developing such a scheme, we also strive to minimize performance degradation and attain performance levels comparable to ODoH and ADNSCrypt.

4. Overview of μODNS

The previous section has assumed a severe environment where an unknown subset of relays has colluded with public resolvers. The design goal of our scheme, μODNS (Mutualized Oblivious DNS), is to practically anonymize each user's DNS message in the environment. To achieve the goal, μODNS is designed as a relay-based scheme leveraging *multiple hops of relays* with several key features summarized in Fig. 3. Our scheme takes an approach that is analogous to the private relay depicted in option (3) of Fig. 2 and explained in Section 2.3. Namely, we suppose that other than potentially-colluded relays in the network, *every user (or every organization of a set of users) has at least one trusted instance of relays*, e.g., a relay deployed by himself. By fully and mutually utilizing such a dedicated and trusted relay of each user, our scheme realizes collusion resistance and protects the user's identity from being leaked to resolvers. Note that we do not limit the network to exclusively comprise users' trusted relays. We permit the inclusion of public and shared relays that serve multiple-hop messages, corresponding to options (1) and (2) in Fig. 2, i.e., relays that do not belong to μODNS users. In the following, we shall describe the key features of our scheme and briefly explain how it achieves anonymity against relays colluding with target resolvers.

(1) Employment of a user's dedicated relay(s) as his next-hop: Essentially, in this scheme, every DNS query message issued by a user is routed to the target resolver through one or more relays, involving multiple hops. The first key feature of μODNS is to require each user to select their dedicated relay as their *next-hop*. Alternatively, the user can become their trusted relay, or opt for a fully-trusted shared relay provided by their organization, in contrast to using public relays. The use of a dedicated and trusted relay as the next-hop simply precludes collusion between the target resolver and a node that possesses direct knowledge of the user's IP address.

Remark 2. Notably, this concept of the dedicated next-hop resembles the *entry guard* concept in Tor [13]. In Tor, an entry node of a Tor circuit is chosen from relatively trusted (stable and long-running) nodes, and it is fixedly used for a period to hide the user's identity from randomly-chosen subsequent nodes.

(2) Mutually sharing the user's dedicated relays to mix query messages: As we mentioned in Section 3, the identity of the user is uniquely associated with his dedicated relay in the use of a single relay, i.e., ODoH and ADNSCrypt, since the relay dedicated to a user uniquely couples with the user himself. Even in the case of multiple relays, this occurs as well when a hop after the dedicated next-hop of the user has colluded with the target resolver. Hence the second key feature designed in μODNS is to enforce the dedicated next-hop of a user to accept incoming messages from other relays or resolvers and forward them in addition to DNS messages from/to the user. Namely, users mutually share and leverage their dedicated relays each other as hops after their next-hop. This allows the next-hop of a user to mix query messages, c.f., Table 1, from its user with ones from other relays. Thus this feature makes it difficult for other relays receiving messages from the next-hop to correctly identify the user's messages. Namely in our concept of μODNS, when one shares its resource, i.e., allowing one's dedicated

relay to forward messages issued by others, the privacy of one's identity is protected.

(3) Randomization of subsequent hops by the user: To fully leverage the first and second features to hide the user's identity, the choice of relays after the next-hop is quite important. Consider a scenario where the relays involved in receiving/sending messages to/from the dedicated relay have been fixed, and unfortunately, all of them have colluded with the target resolver. In such a case, the user's messages could potentially be easily identified at the target resolver. This might be unrealistic in general but could be realistic if the target resolver selectively increases colluded nodes when the route is always fixed. Thus to disallow such a case and minimize the identity leakage even if we meet the case, we introduce the third key feature: The user randomizes choices of hops after his dedicated relay. This means that the number of subsequent hops, the selection of hops, and the order of hops are all randomized. That is, the μODNS allows messages to be conveyed via one or more relays or *directly* to the target resolver after the dedicated next-hop. Note that in our motivation and in the nature of DNS, only the origin of a query message must be anonymized to its target resolver and its colluded relays, but *the target resolver itself is NOT required to be hidden to relays.* Hence in choices of subsequent relays, the direct forwarding from the dedicated relay to the resolver is also allowed like ODoH [7–9] and unlike Tor [13].

These three key features are also illustrated in Fig. 3. In an exemplary scenario depicted in Fig. 3, user A initiates three query messages, namely Q_1, Q_2, and Q_3 directed towards the full-service resolver X. These messages follow a multi-hop path through several relays, including R_A, R_B, and R_C. Key Feature (1) ensures that relay R_A is the user's trusted relay and is consistently selected as the first hop for all queries. Furthermore, R_A not only handles messages originating from A but also messages forwarded from other relays, such as R_B, as facilitated by key Feature (2). Key Feature (3) randomizes the path for each query Q_A, Q_B, and Q_C to X, by A as shown in Fig. 3.

Additionally from the mechanical viewpoint, we see that in μODNS, every DNS query message conveyed on a path includes its forwarding information only about subsequent relays, and the origin of paths is hidden from the target resolver. For instance in Fig. 3, a query message Q_3 on the path between R_A and R_C includes forwarding information that will be forwarded to X after R_C and does not include previous paths, i.e., A to R_A. That is, each relay simply strips the previous hop's information of an incoming query message. But as with ODoH and ADNSCrypt, although DNS response messages from the target resolver include no information of their returning path, they are correctly returned to the user just by trailing L4 connections[3] between adjacent nodes. We thus see that subsequent relays and target resolvers do not learn the information on the origin of paths from DNS messages as network packets.

Fig. 4 illustrates an exemplary scenario of the μODNS showcasing how it preserves user anonymity in an environment where a relay subsequent to the user's dedicated relay colludes with a target resolver. In this example, user A issues two query messages, referred to as Query A's, directed to the full-service resolver X. One of these queries is routed through R_A and then R_B to reach X (involving two relays), while the other traverses R_A, R_B, and then R_C before reaching X (involving three relays). Simultaneously, user B issues two queries, referred to as Query B's, towards X, following paths $R_A \rightarrow X$ and $R_C \rightarrow X$. Consider a scenario in which relay R_C colludes with X, attempting to identify the source of received queries. However, for R_C, there is no guarantee that any queries forwarded from R_B originated from user B or A. This uncertainty arises because R_B conceals the previous hops of outgoing messages, mixes queries from multiple sources, and randomizes query paths before R_B, thanks to the key features of μODNS. Consequently, the origins of received queries cannot be uniquely identified at R_C, and the same holds for those received from R_A at X.

A detailed discussion and analysis of the security of our scheme will be given in Section 6.1.1. In the next section, we introduce a proof-of-concept implementation of μODNS with the key features.

5. Proof-of-concept of μODNS

As proof-of-concept (PoC) of μODNS, we have developed practical instances of μODNS based on existing relay-based anonymized DNS protocols, namely, ADNSCrypt [10,11] and ODoH [7–9]. These instances consist of client proxy modules that translate Do53 to μODNS and relay modules.

For the extension of ADNSCrypt, we forked and modified the existing open-source software of client proxy[4] and relay.[5] The implemented client and relay modules are available on GitHub.[6] On the other hand, for the ODoH-based implementation, we built the client and relay modules from scratch, also available on GitHub.[7] It is important to note that in our PoC, the target resolvers are fully compatible with standard ADNSCrypt and ODoH. Therefore, we deployed existing implementations of ADNSCrypt and ODoH servers[8] as the target resolvers in our PoC for μODNS. For the sake of simplicity, we will now refer to our PoC protocols and implementations of μODNS as *μADNSCrypt* for the ADNSCrypt-based implementation and *μODoH* for the ODoH-based one.

In the following subsections, we will provide detailed protocols for μADNSCrypt and μODoH, along with brief introductions to their implementations. To facilitate the reader's understanding, we will start with a concise explanation of the original ADNSCrypt and ODoH protocols. Subsequently, we will elucidate how these original protocols have been adapted to realize the concept of μODNS. Table 2 offers a succinct comparison of protocol specifications for ADNSCrypt, ODoH, μADNSCrypt, and μODoH, covering aspects such as the underlying protocol, protocol identification at relays and resolvers, route description of queries, and the number of relays. These details will be elaborated upon in the following subsections.

5.1. Protocols of existing relay-based anonymized DNS

5.1.1. ADNSCrypt protocol

In the ADNSCrypt protocol [10,11], a user and a target resolver exchange all encrypted DNS messages, i.e., queries and responses, via a single relay specified by the user. The message encryption is done in the end-to-end manner of DNSCrypt version 2 protocol [29,30] between the user and resolver, and the relay only passively forwards messages upstream and downstream.

Query messages in ADNSCrypt simply consist of two parts: ADNSCrypt header and ANDSCrypt payload. The header part lets a relay know the target resolver. The payload part is exactly the encrypted query in the *non-anonymized* DNSCrypt v2, and only this part is forwarded to the target resolver by the relay. Thus from the viewpoint of the target resolver, the relay is regarded as a user in the context of

[3] UDP/TCP connections are managed at each node.

[4] dnscrypt-proxy: https://github.com/DNSCrypt/dnscrypt-proxy [22].

[5] encrypted-dns-server: https://github.com/DNSCrypt/encrypted-dns-server [23].

[6] μADNSCrypt client: https://github.com/junkurihara/dnscrypt-proxy-modns [24], and relay: https://github.com/junkurihara/encrypted-dns-server-modns [25].

[7] μODoH client: https://github.com/junkurihara/doh-auth-proxy [26], and relay: https://github.com/junkurihara/doh-server [27]. The repository of relay implementation is actually the fork of an existing ODoH target implementation (doh-server [28]), but its relay function was completely built from the scratch.

[8] ADNSCrypt server: https://github.com/DNSCrypt/encrypted-dns-server [23], ODoH server: https://github.com/DNSCrypt/doh-server [28].

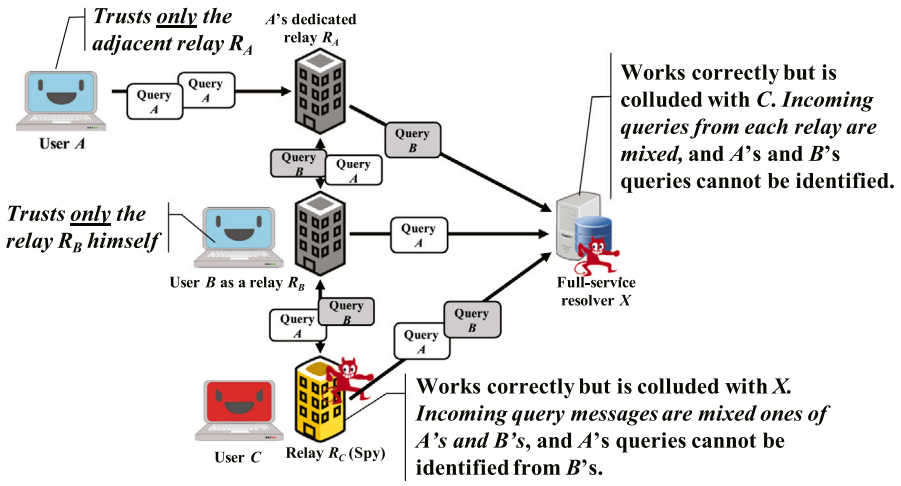


Fig. 4. An exemplary scenario of three parties case in μODNS, where the user A issues DNS query messages (Query A's) routed as $R_A \to R_B \to X$ and $R_A \to R_B \to R_C$, and user B issues ones (Query B's) routed as $R_A \to X$ and $R_C \to X$. The resolver X cannot see that the origin of messages received from R_A is B, and also cannot identify origins of messages from R_C even by utilizing the colluded spy R_C.

Table 2
Protocol specification comparison of existing single relay-based DNS anonymization, and μODNS instances as their extension, where in μADNSCrypt and μODoH, the number of relays following the user's next-hop, denoted as α, is chosen randomly.

	Underlying protocol	Identification at relays and resolvers	Description of query's route	# Relays	Extended from
ADNSCrypt [10,11] (Section 5.1.1)	UDP/TCP	Dedicated header	Dedicated header (Target)	1	-
ODoH [6,7] (Section 5.1.2)	HTTPS	URL path, HTTP header	URL query string (Target)	1	–
μADNSCrypt (Section 5.2.1)	UDP/TCP	Dedicated header	Dedicated header (Target and relays)	$1+\alpha$ ($\alpha \geq 0$)	ADNSCrypt
μODoH (Section 5.2.2)	HTTPS	URL path, HTTP header	URL query string (Target and relays)	$1+\alpha$ ($\alpha \geq 0$)	ODoH

the non-anonymized DNSCrypt v2. The structure of the header part is given as follows[9]:

ADNSCrypt header

$$:= |\text{Anonymized query magic}|\underbrace{|\texttt{192.168.1.1}|}_{\text{Target address}}\underbrace{|\texttt{8443}|}_{\text{Target port}}, \quad (1)$$

where the anonymized query magic is a special constant indicating the message is an ADNSCrypt query. The user simply dispatches this query message over UDP or TCP to its specified relay instead of the target resolver. The response to the query is structured exactly in the form of the non-anonymized DNSCrypt response, and it is just inversely forwarded from the resolver to the user via the relay with no modification on the path.

5.1.2. ODoH protocol

In ODoH [6,7], DNS query and response messages are encrypted in an end-to-end manner between a user and a target resolver exactly similar to the ADNSCrypt protocol. The encrypted query and response are transmitted as HTTP request and response messages through a relay specified in a request URL. The architectural difference between ODoH from ADNSCrypt is only the employment of the secure channel, i.e., HTTPS, as the transmission channel of encrypted messages.

Specifically, the user encrypts a plaintext query message of Do53 by leveraging the hybrid public key encryption (HPKE) [31–33] and sends the encrypted query to a relay using the POST method as an HTTP request message. Then the request URL to the relay contains the information on the target resolver in the form of a query string with parameters: `targethost` and `targetpath`. The relay that receives such an HTTP request composes a new URL from the request URL given by the user and dispatches another HTTP request with the new URL to the target resolver. Note that each HTTP request has a specific header indicating its content-type is an encrypted query.[10] For instance, when a user sends an encrypted query as an HTTP request with a URL

```
https://relay.example/proxy?
    targethost=target.example
    &targetpath=/dns-query,
```
(2)

the relay just forwards the received request to the target resolver by replacing its request URL with

```
https://target.example/dns-query,
```

via the HTTP POST method. The response to the query is also encrypted with HPKE at the target resolver and sent back as an HTTP response to the user through the relay.

5.2. Protocols of μADNSCrypt and μODoH

Both in μADNSCrypt and μODoH, the target resolvers are respectively still the DNSCrypt v2 resolver and the ODoH target, and hence our PoC protocols are compatible with existing (A)DNSCrypt and ODoH infrastructures. However, in order to realize the concept of μODNS, we rebuilt the construction and format of query messages and modified the relaying protocol of ADNSCrypt and ODoH. In the following, we shall describe such modifications to each protocol.

[9] Values in target address and port are just examples.

[10] Content-Type: application/oblivious-dns-message.

5.2.1. μADNSCrypt protocol overview

A μADNSCrypt query message is composed of two parts: μADNSCrypt header and payload. The payload part is exactly an encrypted query of DNSCrypt v2 as an inherited structure from the original ADNSCrypt. On the other hand, since each query message itself has to describe the path on which it will follow to the target resolver, the header part is designed so as to include an ordered list of subsequent relays and a target resolver by extending (1). In particular, the μADNSCrypt header is structured as follows.

$$\mu\text{ADNSCrypt header} := |\mu\text{ADNSCrypt query magic}| \underbrace{|n|}_{\#\text{ nodes}} \underbrace{|\texttt{192.168.1.1}||\texttt{8443}|}_{\text{Node 1 address and port}} \underbrace{|\texttt{192.168.2.4}||\texttt{8443}|}_{\text{Node 2 address and port}} \ldots \ldots \underbrace{|\texttt{192.168.128.32}||\texttt{8443}|}_{\text{Node } n \text{ (target resolver) address and port}}, \quad (3)$$

where the μADNSCrypt query magic is a constant indicating the μADNSCrypt query message much like the anonymized query magic in ADNSCrypt. This simply describes the path information by the number of nodes, n, and an ordered list of n tuples comprising IP addresses and ports of intermediate relays and a target resolver. Notably, this information excludes the user's trusted next-hop. After the user's trusted next-hop, the query message is routed in order from the beginning of the list and finally reaches the target resolver specified as the end of the list. We note here that the intermediate relays, their number, and their order are chosen by the user at random as explained in Section 4.

When a relay including the user's next-hop receives a message starting with the μADNSCrypt query magic, it first checks the next node to which the relay forwards the message from the first element of the list. Then, it composes the updated query by decrementing the number of subsequent nodes and peeling off the first element of the list, and it sends the updated query out to the next node. Note that every relay works in this manner for the received μADNSCrypt query message with $n > 1$. On the other hand, if $n = 1$, the list includes only the information of the target resolver of DNSCrypt v2, and hence the relay that receives such messages just strips off all the header part and simply dispatches only the payload part, i.e., DNSCrypt v2 query, to the target resolver. As in the ADNSCrypt, the response to the query is that of the vanilla DNSCrypt v2 and is inversely forwarded from the resolver to the user.

5.2.2. μODoH protocol overview

The design principle of μODoH is the same as the μADNSCrypt. Namely, every plaintext query message itself is composed and encrypted in the manner of the original protocol, ODoH, and the path from a user to a target resolver is randomly configured for the message. In μODoH, we describe the randomized list of nodes that a query message will visit in the query string of the HTTP request URL instead of the ordered list of nodes in μADNSCrypt header. In particular, we introduced new parameters in the query string of the URL, `relayhost[`i`]` and `relaypath[`i`]` ($i \geq 1$), in addition to the `targethost` and `targetpath` of the original ODoH. For instance, along with an HTTP header specifying the content type as an encrypted query, a user composes a μODoH query in the form of an HTTP message with the following URL, much like (2).

```
https://relay.example/proxy?
    relayhost[1]=1.relay.example
    &relaypath[1]=proxy
        …
    &relayhost[n − 1]=n-1.relay.example
    &relaypath[n − 1]=proxy
    &targethost=target.example
    &targetpath=/dns-query,                    (4)
```

where a tuple (`relayhost[`i`]`, `relaypath[`i`]`), $i \in \{1, \ldots, n\}$ specifies the URL of the ith relay that the query will visit after the host `relay.example`, much like `targethost` and `targetpath` explained in Section 5.1.2.

A relay host that receives an HTTP request with the μODoH request URL determines the next node, i.e., URL host and path, by parsing the query string: It is a relay specified by `relayhost[1]` for URL host and `relaypath[1]` for URL path if these parameters exist; Otherwise, it is the target resolver specified by `targethost` and `targetpath` as with ODoH. Unless the next node is the target resolver, the relay also updates the query string by removing `relayhost[1]` and `relaypath[1]` and decrementing i in `relayhost[`i`]` and `relaypath[`i`]` for all i if they exist. Then it forwards the HTTP request to the next node with the updated request URL via the POST method. This means that the final relay just dispatches an HTTP request to the target resolver with the URL composed of the target resolver's hostname and URL path in exactly the same manner as relays in ODoH, oblivious proxies. Thus the response to the query is the same as that in ODoH, and it is forwarded from the resolver by trailing the path back to the user via relays.

5.3. Implementations of μADNSCrypt and μODoH

Here we explain our PoC implementations for μADNSCrypt and μODoH. For both protocols, we implemented client modules converting Do53 plaintext queries to μADNSCrypt and μODoH ones, and relay modules forwarding encrypted DNS messages in the network. We note that unlike protocol specifications given in Section 5.2, the fundamental features of their implementations are identical for both protocols. Thus we explain our implementations by focusing on their features regardless of the protocol specification.

5.3.1. Client implementation

Our client implementation for μADNSCrypt [24] is based on *dnscrypt-proxy* [22], and that for μODoH [26] is designed from scratch. In these client implementations, μADNSCrypt and μODoH queries explained in Sections 5.2.1 and 5.2.2 are generated from Do53 queries for some given parameters such as lists for candidates for the user's next-hop, i.e., the user's trusted and dedicated relays, and available intermediate relays subsequent to the next-hop. Specifically, the following configuration options are introduced to realize our PoC protocols.

[List of available targets and relays with flag]: To build the path information specified in (3) and (4) for every query, client modules must have knowledge of locations of relays and target resolvers, i.e., socket addresses in μADNSCrypt and URLs in μODoH. Hence the client configuration of our implementations contains two static lists of node locations: One is about targeted resolvers of the underlying protocol, i.e., DNSCrypt v2 or ODoH; The other is about available relays supporting μODNS relaying protocols described in Sections 5.2.1 and 5.2.2.[11] Here we note that every available relay in the list can have a 'flag' indicating a candidate of the next-hop. Namely, the user's trusted relays are explicitly specified in the list in our client implementation.

[Maximum and minimum number of relays]: For each query message, the number of relays after the next-hop, i.e., $n - 1$ in (3) and (4), are randomly fixed within the range specified by the user. The range is simply given by parameters of the maximum and minimum allowed numbers of relays.

By the above options, our client implementations work as follows: When a Do53 query is given from a user, the client first chooses a *flagged* relay candidate as the user's trusted next-hop from configured lists and simultaneously generates an encrypted query of DNSCrypt v2

[11] For the client of μADNSCrypt [24], lists can be hosted online, e.g., [34], as the vanilla dnscrypt-proxy.

or ODoH from the plaintext query. Then, it constructs the query's path information as (3) and (4) by selecting a target DNSCrypt v2 or ODoH resolver and randomly choosing relays from the rest of the listed relays regardless of the next-hop flag. Finally, it dispatches the generated query to the next-hop flagged relay chosen at first. Here we should note that our client implementations generate queries' path information avoiding any loop path, i.e., duplicated selection of relays.

5.3.2. Relay implementation

The second piece of our implementation is the relay function supporting the relaying protocols described in Sections 5.2.1 and 5.2.2. The μADNSCrypt relay implementation [25] is realized by extending the relay function of ADNSCrypt in *encrypted-dns-server* [23]. On the other hand, the μODoH relay implementation [27] is realized as the one built from scratch as a new module in *doh-server* [28]. To correctly serve μADNSCrypt and μODoH queries, the following configuration option is introduced.

[Overload and loop avoidance]: At a relay, each incoming query message explicitly indicates subsequent relays where the message will visit after the relay. If the number of such relays is unnecessarily large, relays might be severely overloaded, and hence such cases should be suppressed. Towards this end, our relay implementation checks the number of such subsequent relays indicated in each incoming query and drops the one having more hops than the predefined threshold. It also checks duplicated relays, i.e., loop, in the path of relays indicated in each incoming query, and drops it as well when a loop is detected.[12]

Our implementations can serve not only μADNSCrypt and μODoH queries but also those of their underlying anonymized DNS protocols, ADNSCrypt and ODoH. We also note that in our relay implementations, the backward path from the target resolver is adequately managed by the socket connection of UDP/TCP at each relay in a hop-by-hop manner.

In the next section, we will give discussions on μODNS from several viewpoints considering the deployment of μADNSCrypt and μODoH as PoC but practical instances of μODNS. We shall also evaluate the actual performance degradation in μODNS caused by multiple relays by our implementation.

6. Evaluation and discussion

Considering the deployment of realizations of μODNS, e.g., μADNSCrypt and μODoH, we must consider its security and privacy. This section first provides discussions on such considerations. We then introduce a preliminary evaluation of the performance of μODNS using our PoC implementation in terms of the round-trip time (RTT) on the Internet. We moreover discuss some realistic scenarios of the deployment of μODNS.

6.1. Security and privacy

6.1.1. Protocol aspect

The concept of μODNS is simply regarded as an extension of single relay-based schemes, i.e., ADNSCrypt and ODoH, to a multiple relay-based one. Namely, we see that by omitting the assumption of leveraging the user's trusted relays as his next-hop, the μODNS guarantees at least the same security and privacy as ADNSCrypt and ODoH.

To have resistance against the collusion of untrusted relays with a target resolver, as explained in Section 4, we additionally suppose that users leverage their trusted and dedicated next-hop relays. The relays serve not only queries from their directly-connected users but also ones incoming from other relays as shared or public relays. In the following, we will analyze the security and privacy of μODNS in detail, specifically how it protects the user's privacy when dealing with untrusted relays.

First, recall that users are unaware of the locations of colluding relays. In Section 4-(3), we previously mentioned a specific scenario in which a user's next-hop unfortunately and exclusively handles queries originating from or destined for colluding nodes. As we have also explained, the likelihood of such an occurrence is significantly reduced due to the key feature (3), which involves path randomization deployed by both the user and other non-colluding users. This implies that with each subsequent relay, namely the next-hop of other users in the query's path, the probability of encountering such a situation is minimized as well. Therefore, we will consider the scenario in which each relay handles queries forwarded to and from multiple relays, encompassing both non-colluding and potentially colluding ones. We can then categorize this situation into two cases, as illustrated in Fig. 5: (1) a scenario where the relay immediately following the user's next-hop colludes, and (2) a situation in which collusion does not occur at the relay following the next-hop.

Case (1): For a query and its selected path, consider a scenario where a relay in direct communication with the user's next-hop on the path has colluded with a target resolver. Then, we see that as briefly explained in Section 4, there is no guarantee that the received query was sent by the user behind the observed relay. This uncertainty arises because the relay handles queries forwarded from multiple non-colluding relays, making it impossible to uniquely attribute the query to any specific user.

However, this uncertainty becomes compromised when we introduce an additional assumption where the colluding relay attempts to estimate the number of relays chosen by the user for the query. In this stronger situation, the target resolver could potentially ascertain the location of the query's origin, specifically hidden behind the observed next-hop, once it accurately estimates the number of relays. For instance, for a received query, let n be the total number of relays set by a user, and suppose that n is known to the colluding relay. Then, if the remaining path described in the query consists of $n-2$ relays following the colluding relay, it can be precisely identified as originating from a user concealed behind the previous hop, i.e., the user's trusted next-hop. Conversely, for queries sent by other users, the number of remaining relays is always less than or equal to $n-3$.

It is important to note that the key feature (3), as outlined in Section 4, incorporates the randomization of path elements, including relay selection, order, and number. This randomization significantly complicates any attempts at such estimation of the number of relays. Consequently, μODNS establishes collusion resistance in a probabilistic manner. Furthermore, we also see that the randomization feature ensures that the situation of collusion occurring at the relay immediately following the user's next-hop cannot consistently occur.

Case (2): For a query and its selected path, consider a situation where a colluding relay or relays exist along the path, but the next relay following the user's next-hop is not involved in collusion. In this situation, unlike case (1), the colluding relays are unable to uniquely identify the origin of the query, as long as non-colluding relays on the path handle queries received from multiple non-colluding relays and users. This guarantee remains in place even when the colluding relay accurately estimates the total number of relays set by the user. For example, in Fig. 5-(2), let n represent the total number of relays set by a user and known to the colluding relay. Then, even if the remaining path described in the query consists of $n-3$ relays following

[12] Loop avoidance is not necessary as long as only our client implementations in Section 5.3.2 are used.

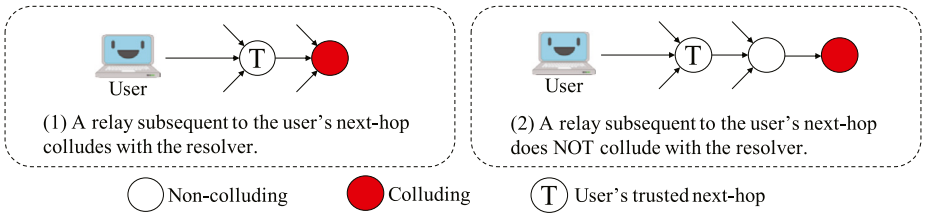


Fig. 5. Two cases where a colluding relay(s) exists on the selected path for a query: (1) the next relay following the user's next-hop on the path collude; (2) the next node following the user's next-hop does not collude.

the colluding relay, it cannot be guaranteed that the query originated from the user depicted in Fig. 5-(2).

It is worth noting that when colluding relays accurately estimate n, the pool of potential query origins may be limited. However, we see that employing the key feature of path randomization can make such estimation infeasible to a great extent, as discussed in case (1).

As the above analyses on both cases, we see that the user cannot be identified by the target resolver unless specific conditions are simultaneously met: For a query, the subsequent relay selected after the next-hop being colluding, and the correct estimation of the number of relays. Consider that many entities make relays publicly available in the system. We claim that in such a situation, the condition is quite rarely satisfied if subsequent relays, following the next-hop, are randomly selected from those operated by distinct entities, like the premise of Tor [13].

As a conclusion of this subsection, we mention the security of our PoC implementations. Although we have assumed no wiretappers on transit channels in Section 3, an extra layer of security might be required. For instance, there may be a case where a certain monitoring authority employs a middlebox that drops any transit messages conveyed to specific relays/resolvers. The μODoH messages are exchanged over HTTPS, and hence the messages could be mixed with and might not be differentiated from standard HTTPS messages, as with DoH and ODoH. Moreover, the paths of queries and target resolvers in the URL cannot be revealed thanks to the TLS even if a wiretapper overhears messages on a channel between a user and his next-hop relay. On the other hand, in μADNSCrypt, all the connections among relays are established over a non-secure channel, i.e., no channel encryption like TLS, and query messages can be identified from their header much like the underlying DNSCrypt and ADNSCrypt. Thus, considering channel wiretappers in μADNSCrypt, the target resolver and path of relays are explicitly revealed from an observed query message while the content of the query itself is encrypted. Therefore, if channel wiretappers might exist, we should employ an instance of μODNS with the channel encryption for relaying, i.e., μODoH, even if we need to pay an extra cost to establish secure channels.

6.1.2. Deployment aspect

Considering the deployment of instances of μODNS, there exist several considerations to employ the system securely other than the design of their protocols. Especially, attacks of distributed denial of service (DDoS) should be one of the most serious problems from the viewpoint of relay and target resolver operators. For example, we can consider a case where malicious users issue huge numbers of fake messages to victim relays or resolvers without revealing their origins by exploiting the nature of μODNS. Thus, we should introduce a certain extra mechanism against such an attack to protect the system of μODNS for its deployment.

The obvious way to resolve the above concern is to introduce an authentication/authorization mechanism for message acceptance. That is, every relay should (1) communicate only with authenticated users as their next-hop relay and (2) accept messages sent from pre-authorized external relays as a shared or public relay, in the deployment of μODNS. Similarly to (2), (3) target resolvers should accept messages dispatched only from pre-authorized relays. To this end, we can consider adopting several existing authentication and authorization approaches. For instance for (1) user authentication, we can use the mutual TLS, i.e., authentication using client certificates, or utilization of an ID token of OpenID Connect as a bearer token of OAuth 2.0 in the HTTP header [35]. Here we should note that by the nature of μODNS, anonymity is still guaranteed even if such user authentication is introduced at the user's next-hop relay. On the other hand for (2) and (3), i.e., pre-authorization of relays, we can use a simple allowlist-based approach much like signed lists of (A)DNSCrypt resolvers and relays [21]. Users can refer to such lists to build appropriate paths for their queries as with (A)DNSCrypt. But in our context, relays and resolvers also refer to the lists and use them as allowlists for incoming connections to deny relaying potentially harmful traffic from unauthorized nodes.

Indeed our PoC implementations have already introduced the above authentication and authorization mechanisms to avoid attacks causing DDoS. Specifically, we can use the user authentication based on the ID token between our client [26] and relay implementations [27] of μODoH. Also, our μODoH relay implementation supports the allowlist of connection acceptance based on source IP addresses.

6.2. Performance evaluation

We evaluate the performance of our PoC implementations from the viewpoint of round-trip time (RTT), i.e., the elapsed time to receive a response after a query issuance under several relay settings. The architectural concepts of ADNSCrypt and ODoH are the same as mentioned in Sections 5.1.1 and 5.1.2, and so for μADNSCrypt and μODoH. Thus, we use ADNSCrypt and μADNSCrypt for the performance evaluation since we can expect that the result is the same for ODoH and μODoH. Also, we measure the performance of DNSCrypt over Tor as an instance of Tor-based anonymization schemes for the performance comparison with μADNSCrypt.

6.2.1. Experimental environment

Table 3 summarizes the environment of our experimentation, where all the nodes, i.e., a μADNSCrypt user, μADNSCrypt relays, and a target DNSCrypt v2 resolver, are deployed on virtual private servers (VPS) [36] at Tokyo or Singapore. On the other hand, we carefully chose locations/areas of Tor public relay nodes in order to make our evaluation fair.

Under the above setting, we measured the RTT from the user's query issuance to response retrieval with several choices of μADNSCrypt relays or Tor nodes. In each measurement, the user generates a query for a domain with a random string, `<random_string>.example.com`, to disable caching at the target resolver and measure the RTT. We measured RTTs of ten thousand measurements for each setting of μADNSCrypt relays and Tor nodes and evaluated their average, median, and several metrics. For each setting, we also measured the empirical cumulative distribution of the difference of RTTs from the median RTT of the case of the vanilla DNSCrypt, i.e., no relays/nodes between the user and the resolver.

Table 3
Environments of our experimentation on (A)DNSCrypt, μADNSCrypt and DNSCrypt over Tor.

# of measurements	10,000 times	
Target resolver [23]	Hosted on ConoHa VPS Tokyo	
User [22,24]	Hosted on ConoHa VPS Tokyo	
(A)DNSCrypt and μADNSCrypt	Relay setting [25]	3 nodes: ConoHa VPS Tokyo 2 nodes: ConoHa VPS Singapore
	Relay choices	(1) Random Tokyo t relays ($t = 0, \ldots, 3$) (2) Random Singapore s relays ($s = 1, 2$) (3) Random Tokyo 1 and then Singapore 1 relays (4) Random Singapore 1 and then Tokyo 1 relays
DNSCrypt over Tor	Tor node setting	Entry nodes: Japan Refresh Tor circuit every 100 queries[a]
	Tor node choices	(5) Global middle and exit nodes (No exclusion) (6) Middle and exit nodes in East Asia and East South Asia[b]

[a] The Tor circuit used for the new connection is refreshed every 100 secs in its setting, and we dispatch a DNS query every second.
[b] This setting is done by excluding countries other than cn, hk, jp, kr, tw, kh, tp, id, my, ph, sg, th, and vn in country codes. In other words, Tor circuits were established with relays located in these countries.

Table 4
Results of the experimental evaluation: Average, median, first and 99th percentiles, and standard deviation on RTT for DNS query/response under several conditions of intermediate relays between the user in Tokyo and the resolver in Tokyo, expressed in milliseconds. For the setting 6), we additionally present 0.1st and 0.5th percentiles.

		Relays	Ave	Med	1st%	99th%	Std
(A)DNSCrypt and μADNSCrypt	1)	Direct (0 relay=DNSCrypt)	126.3	97.6	9.0	507.9	112.7
		Tokyo 1 relay (=ADNSCrypt)	137.0	111.8	11.7	502.7	110.3
		Tokyo 2 relays	137.5	113.3	12.6	534.2	109.6
		Tokyo 3 relays	136.1	114.2	13.2	519.8	108.7
	2)	Singapore 1 relay (=ADNSCrypt)	289.8	262.2	161.0	675.7	112.3
		Singapore 2 relays	293.9	265.4	161.9	733.7	121.1
	3)	Singapore relay → Tokyo relay	288.3	258.0	162.2	669.2	113.1
	4)	Tokyo relay → Singapore relay	215.8	202.5	13.0	644.1	136.4
DNSCrypt over Tor	5)	Global middle and exit nodes	1,251.2	1,228.8	786.7	2,008.0	235.7
	6)	Middle and exit nodes in East/East South Asia	473.2	461.8	92.1 (0.5th%: 63.1) (0.1st%: 38.2)	1,096.1	203.7

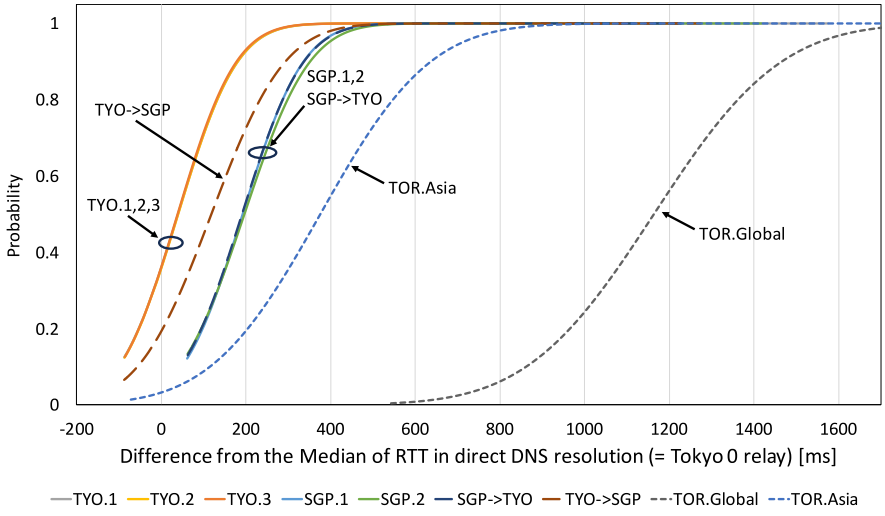


Fig. 6. Empirical CDF of the difference of RTT from the median value of the case of direct DNSCrypt query/response (no relay), where TYO and SGP respectively mean Tokyo and Singapore, N of TYO.N or SGP.N indicates the number of relays, and X of TOR.X describes the location of intermediate and exit nodes in Tor.

For μADNSCrypt, we fixed the number of relays chosen, and the user randomly selected relays of the fixed number from the pool of candidate relays. In the choice of μADNSCrypt relays, we executed four types of experimentation: (1) With relays located in Tokyo, (2) With relays located in Singapore, and (3) and (4) With one relay located in Tokyo and another located in Singapore. For DNSCrypt over Tor, we fixed the location of entry nodes to be always in Japan. On the other hand, we examined the following two cases of the selection of middle and exit

nodes: (5) Chosen globally, and (6) Chosen from East and East South Asia including Japan and Singapore, i.e., geographically near Japan and Singapore.[13]

6.2.2. Experimental result

Table 4 summarizes the average, median, first percentile, 99th percentile, and standard deviation of measured RTTs for each type of experimentation. On the other hand, Fig. 6 illustrates the empirical cumulative distribution function (CDF) of differences in measured RTTs from the median RTT of the zero relay case, i.e., the vanilla DNSCrypt v2 with the median 97.6 ms in Table 4. Below we shall explain the results for each setting (1)–(6).

(1) For measurements with relays in Tokyo, we see that from measured metrics in Table 4, the employment of two or more relays does not impair the performance in terms of the RTT by comparing with the case of direct connection (zero relays), i.e., the vanilla DNSCrypt v2, and that of one relay, i.e., ADNSCrypt. The degradation of the RTT by using the multiple relays is within 10–17 ms in the average and median of RTT. We also see that the distribution of difference of RTTs is not affected by the employment of multiple relays as shown in Fig. 6.

(2) Also for measurements with relays in Singapore, the result with one relay is exactly similar to the case of two relays as shown in Table 4 and Fig. 6. The performance degradation in RTTs is about 170 ms in the average and median. Also, we see that the distribution of difference of RTTs is not affected by the employment of multiple relays as (1).

(3) and (4) For measurements with a combination of relays from Tokyo and Singapore, we see that their performance in the RTT is worse than that of (1) but comparable to (2).

(5) For measurements of DNSCrypt over Tor with globally chosen middle and exit nodes, the performance in RTT is significantly degraded from the case of direct communication as shown in Table 4, which is larger than 1,000 ms both in the average and median values. Of course, this is obvious since Tor nodes are widely located on the earth in this case. However, even the first percentile of the RTT is greater than any other case by more than 500 ms.

(6) For measurements of DNSCrypt over Tor with middle and exit nodes located in East and East South Asia, the performance in RTT is much better than (5) on all metrics. We note that there exist cases where Tor circuits are established only with nodes near Japan (or in Japan) sometimes in a probabilistic manner. But all the first, 0.5th, and even 0.1st percentiles of the RTT are still larger than the first percentiles of μADNSCrypt with three Tokyo relays by about 25–60 ms.

6.2.3. Consideration from the experimental result

As easily expected, we can view that from the experimental results, the performance in terms of RTT simply and mostly depends on the geographic locations of relays in our implementation of μADNSCrypt. From the results for cases (1)–(4), it is evident that the relaying operation, as described in Section 5.3.2, excluding transport delays, introduces negligible overhead. Notably, in case (1), it increased the median or average RTT by less than 17 milliseconds even with three relays. Therefore, we assert that, in μODNS, we should primarily focus on the selection and geographic placement of relays to mitigate the degradation of RTTs. In this context, we observe that, besides considering the distance of the chosen relays from the user's next-hop and their number, the distance among these relays also has an impact on the RTT, considering their randomized order. Similarly, the restriction of Tor node locations is important for DNSCrypt over Tor and DoHoT as suggested by the results for cases (5) and (6). Note that Tor-based schemes, however, cannot control the number of Tor nodes on the path to the target resolver, which is always fixed to three, and forces multi-layered encryption that is unnecessary in our expected environment. We expect these overheads might cause the observations on the 0.5th and 0.1st percentiles for (6) described in the previous section.

Here we shall mention the effect of latency on user experiences on the Internet. There exist several studies and reports in this context. For example, Google, Fifty-Five, and Deloitte reported that in mobile environments, the 100 ms improvement of loading speed of retail sites results in an increase in average order value of 9.2% [37,38]. Also, a 100 ms improvement in loading time yields an increase in revenues of 0.6% for Bing [39], 0.7% for Zalando [40] and 1% for Amazon [39]. From these studies, Callejo et al. suggested that migrating to resolvers that increase the RTT by 100 ms or more has a large negative impact on the user experience [41, Section 8].

Based on the above observations of our experimental results and our studies on user experiences, we assert that with an appropriate strategy for selecting intermediate relays, the degradation of RTTs compared to the no relay case can be kept to within 100 ms. Consequently, μODNS can consistently maintain a positive user experience. We can therefore conclude that our implementation achieves reasonable performance. On the other hand, the strategy to restrict the location of Tor nodes is the most critical factor for user experiences in Tor-based schemes as well as μODNS. However, as previously mentioned, unlike μODNS, we cannot control the number of Tor nodes on the path, and the unnecessary multi-layered cryptographic operations are not removable. Therefore, we assert that μODNS is better-designed in the context of mitigating performance degradation compared to Tor-based schemes.

We will now discuss the selection of intermediate relays in μODNS, focusing on user experiences. We will begin by considering the optimal locations for these relays. As previously mentioned, a user should select relays after their next-hop from a set of neighboring nodes if they seek improved performance. However, imposing restrictions on the pool of potential subsequent relays could potentially compromise privacy. This is because as follows. Recall that subsequent relays are randomly chosen from this pool of candidate nodes, and the privacy of μODNS is guaranteed under the assumption that only a small part of the pool may collude, as explained in Section 6.1. This implies that, for enhanced privacy, the set of potential subsequent relay candidates should be large. Therefore, as a straightforward operational guideline, we suggest that, as long as the expected RTT is acceptable, the set of subsequent relay candidates should be as extensive as possible. In a practical strategy aimed at improving performance, it is advisable to select relays located in the user's country or neighboring countries and ensure that their mutual distance is relatively close.

Next, we shall discuss the maximum number of subsequent relays following the next-hop relay in μODNS, which is a configurable option as explained in Section 5.3.2. Obviously, the fewer relays, the better the RTT performance. Consider a realistic scenario where only a small number, e.g., 1%, of entities collude with the target resolver. In such cases, it is still unrealistic for colluding relays to trace the previous path of each incoming query, as long as the path and its length are randomized, although this becomes more challenging with a larger maximum path length. Therefore, we suggest that the maximum number of subsequent relays can be kept small. For instance, having at most two relays after the next-hop could be practical, which is similar to Tor's configuration (two nodes after the entry node) but it is randomized from zero to two. Additionally, having at most only one relay after the next-hop may be

[13] Tor circuit could not be established at all when we explicitly fixed the area of middle and exit nodes to be only in Japan, or only in Singapore. We thus simply limited the area to be near them.

acceptable if the next-hop relay serves a number of queries received from other relays, as mentioned in Section 4.

6.3. Expected deployment scenarios

The user anonymity of μODNS relies on the expectation that there exist numbers of participants, i.e., users and relays, composing the network. Thus reality needs to consider how to deploy μODNS on the Internet, and hence here we discuss some deployment scenarios of μODNS, especially that of relays. We can expect several realistic scenarios of deployment: For instance, users in an organization can share one relay as their next-hop relay by deploying the relay at the organization's network edge, e.g., the gateway. In this case, it is also made public for external users and used as a relay after their next-hops. As another example, individual users in a mobile network would instantiate their trusted next-hop relays at their edge-computing nodes. On the other hand, much like Tor [13], some altruistic entities might deploy relays that are transparent and trusted for certain groups of users.

From another perspective of deployment of μODNS, we should consider how to handle the list of relays trusted to connect, which we mentioned in Section 6.1.2. In particular, considerations are on how we compose the list of trusted ones and how relays, resolvers, and users fetch and trust such the list. We can take both centralized and decentralized approaches to generate and host such a relay list: As a centralized approach, the trusted list can be served at a single network repository signed by a certain authority similar to the relay and resolver lists [21] of DNSCrypt community. An immediate approach in a decentralized fashion is that addition, deletion, and modification of relays in the list are done through Blockchain-like distributed ledger technology. In our PoC implementations, we make our node list public with the developer's signature [34] at a repository, and namely, we currently take the former approach.

7. Conclusion

This paper extended the concept of single-relay-based anonymized DNS, i.e., ODoH [7–9] and ADNSCrypt [10,11], and introduced a new concept of a multiple-relay-based DNS for user anonymity in DNS queries, named the μODNS (Mutualized Oblivious DNS). In μODNS, the user just sets his trusted relay as his *next-hop*, i.e., the first relay in the path to the resolver. This next-hop relay conveys not only the user's queries but also queries sent from external relays. Additionally, users randomly select subsequent relays, which may also serve as trusted next-hops for other entities. Under this resource-sharing paradigm, μODNS ensures that the user's identity remains concealed from a target resolver, even if a certain subset of relays have colluded with the target resolver. The concept of the next-hop in μODNS can be likened to the approach used for entry guard nodes in Tor [13], although μODNS is purpose-built and simplified to cater to the specific requirements of DNS. Furthermore, we introduced PoC implementations based on ADNSCrypt and ODoH, referred to as μADNSCrypt and μODoH. We evaluated the performance of our implementation through small-scale experiments in terms of RTTs. Our PoC implementation demonstrated that it can minimize the performance degradation resulting from its privacy enhancements and that it can achieve performance levels that maintain the positive user experiences seen in existing single-relay-based schemes.

In the study on μODNS, there are numerous avenues for future exploration. For instance, given that we have conducted only a small-scale measurement of RTT in this paper, conducting a larger-scale measurement by deploying multiple relays in geographically distributed regions and involving many users should be considered as part of future work.

CRediT authorship contribution statement

Jun Kurihara: Conceptualization, Methodology, Software, Funding acquisition, Investigation, Writing – original draft, Project administration. **Toshiaki Tanaka:** Funding acquisition, Validation, Writing – review & editing, Supervision. **Takeshi Kubo:** Conceptualization, Methodology, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jun Kurihara reports financial support was provided by National Institute of Information and Communications Technology. Toshiaki Tanaka reports financial support was provided by National Institute of Information and Communications Technology. Jun Kurihara reports financial support was provided by Japan Society for the Promotion of Science. Toshiaki Tanaka reports financial support was provided by Japan Society for the Promotion of Science. Jun Kurihara reports financial support was provided by KDDI Foundation. Toshiaki Tanaka reports financial support was provided by KDDI Foundation.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was supported in part by the National Institute of Information and Communications Technology, Japan [Grant Number NICT22401], the Japan Society for the Promotion of Science, Japan [KAKENHI Grant Numbers JP22K11994, JP21H03442], and the KDDI Foundation, Japan Research Grant.

References

[1] J. Kurihara, T. Kubo, Mutualized oblivious DNS (μODNS): Hiding a tree in the wild forest, 2021, arXiv:2104.13785.
[2] D.J. Bernstein, DNSCurve, 2009, https://dnscurve.org/. (Last checked 23 May 2023).
[3] F. Denis, Y. Fu, DNSCrypt (project web page), 2015, https://dnscrypt.info. (Last checked 23 May 2023).
[4] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, P. Hoffman, Specification for DNS over transport layer security (TLS), 2016, RFC7858, [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7858.
[5] P. Hoffman, P. McManus, DNS queries over HTTPS (DoH), 2018, RFC8484, [Online]. Available: https://datatracker.ietf.org/doc/html/rfc8484.
[6] P. Schmitt, A. Edmundson, A. Mankin, N. Feamster, Oblivious DNS: Practical privacy for DNS queries, in: Proc. PETS 2019, Vol. 2019, No. 2, Stockholm, Sweden, 2019, pp. 228–244, [Online]. Available: https://www.petsymposium.org/2019/files/papers/issue2/popets-2019-0028.pdf.
[7] E. Kinnear, P. McManus, T. Pauly, T. Verma, C.A. Wood, Oblivious DNS over HTTPS, 2022, RFC9230, [Online]. Available: https://datatracker.ietf.org/doc/html/rfc9230.
[8] S. Singanamalla, S. Chunhapanya, M. Vavruša, T. Verma, P. Wu, M. Fayed, K. Heimerl, N. Sullivan, C.A. Wood, Oblivious DNS over HTTPS (ODoH): A practical privacy enhancement to DNS, in: Proc. PETS 2021, Vol. 2021, No. 4, 2021, pp. 575–592, Online, [Online]. Available: https://www.petsymposium.org/2021/files/papers/issue4/popets-2021-0085.pdf.
[9] S. Singanamalla, S. Chunhapanya, M. Vavruša, T. Verma, P. Wu, M. Fayed, K. Heimerl, N. Sullivan, C.A. Wood, Oblivious DNS over HTTPS (ODoH): A practical privacy enhancement to DNS, 2020, arXiv:2011.10121.
[10] DNSCrypt Project, Anonymized DNS, 2021, https://github.com/DNSCrypt/dnscrypt-proxy/wiki/Anonymized-DNS, Commit ID: 0b21176.
[11] DNSCrypt Project, Anonymized DNSCrypt specification, 2023, https://github.com/DNSCrypt/dnscrypt-protocol/blob/master/ANONYMIZED-DNSCRYPT.txt, Commit ID: 51f0e52.
[12] C. Deccio, J. Davis, DNS privacy in practice and preparation, in: Proc. ACM CoNEXT 2019, Orlando, FL, USA, pp. 138–143.
[13] Tor Project, https://www.torproject.org/. (Last checked 23 May 2023).
[14] P. Mockapetris, Domain names - Concept and facilities, 1987, RFC1034, [Online]. Available: https://datatracker.ietf.org/doc/html/rfc1034.

[15] P. Mockapetris, Domain names - Implementation and specification, 1987, RFC1035, [Online]. Available: https://datatracker.ietf.org/doc/html/rfc1035.
[16] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, DNS security introduction and requirements, 2005, RFC4033, [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4033.
[17] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, Resource records for the DNS security extensions, 2005, RFC4034, [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4034.
[18] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, Protocol modifications for the DNS security extensions, 2005, RFC4035, [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4033.
[19] S. Farrell, H. Tschofenig, Pervasive monitoring is an attack, 2014, RFC7258, [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7258.
[20] L. Zhu, Z. Hu, J. Heidemann, D. Wessels, A. Mankin, N. Somaiya, Connection-oriented DNS to improve privacy and security, in: Proc. IEEE SP 2015, San Jose, CA, USA, 2015, pp. 171–186.
[21] DNSCrypt Project, Lists of public DNSCrypt and DoH servers, 2023, https://github.com/DNSCrypt/dnscrypt-resolvers, Commit ID: fcdcdba.
[22] DNSCrypt Project, dnscrypt-proxy, 2023, https://github.com/DNSCrypt/dnscrypt-proxy, Commit ID: c66023c.
[23] DNSCrypt Project, encrypted-dns-server, 2023, https://github.com/DNSCrypt/encrypted-dns-server, Commit ID: 2068aa4.
[24] https://github.com/junkurihara/dnscrypt-proxy-modns, 2023, Commit ID: ac9f85a.
[25] https://github.com/junkurihara/encrypted-dns-server-modns, 2023, Commit ID: 539c23d.
[26] https://github.com/junkurihara/doh-auth-proxy, 2023, Commit ID: c59dd2e.
[27] https://github.com/junkurihara/doh-server, 2023, Commit ID: 8514495.
[28] DNSCrypt Project, doh-server, 2023, https://github.com/DNSCrypt/doh-server, Commit ID: e5f6f2a.
[29] DNSCrypt Project, DNSCrypt version 2 protocol specification, 2023, https://github.com/DNSCrypt/dnscrypt-protocol/blob/master/DNSCRYPT-V2-PROTOCOL.txt, Commit ID: 611c4f8.
[30] F. Denis, The DNSCrypt protocol, 2023, Internet Draft, [Online]. Available: https://datatracker.ietf.org/doc/html/draft-denis-dprive-dnscrypt.
[31] R. Barnes, K. Bhargavan, B. Lipp, C.A. Wood, Hybrid public key encryption, 2022, RFC9180, [Online]. Available: https://datatracker.ietf.org/doc/html/rfc9180.
[32] J. Alwen, B. Blanchet, E. Hauck, E. Kiltz, B. Lipp, D. Riepel, Analysing the HPKE standard, in: Proc. EUROCRYPT 2021, Zagreb, Croatia, 2021, pp. 87–116, [Online]. Available: https://eprint.iacr.org/2020/1499.
[33] B. Lipp, An analysis of hybrid public key encryption, 2020, Cryptology ePrint Archive, Paper 2020/243, [Online]. Available: https://eprint.iacr.org/2020/243.
[34] https://github.com/junkurihara/experimental-resolvers, 2022, Commit ID: b2e57e4.
[35] M. Jones, D. Hardt, The OAuth 2.0 authorization framework: Bearer token usage, 2012, RFC6750, [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6750.
[36] GMO Internet, ConoHa VPS, 2013, https://www.conoha.jp/vps. (Last checked 23 May 2023).
[37] A. Boulte, How speeding up your mobile site can improve your bottom line, 2020, Think with Google: Marketing Strategtiges, https://www.thinkwithgoogle.com/intl/en-154/marketing-strategies/app-and-mobile/mobile-page-speed-data/. (Last checked 6 Sep. 2023).
[38] Deloitte Ireland LLP, Milliseconds make Millions: A study on how improvements in mobile site speed positively affect a brand's bottom line, 2020, https://www2.deloitte.com/ie/en/pages/consulting/articles/milliseconds-make-millions.html. (Last checked 6 Sep. 2023).
[39] R. Kohavi, A. Deng, B. Frasca, T. Walker, Y. Xu, N. Pohlmann, Online controlled experiments at large scale, in: Proc. ACM KDD 2013, Chicago, IL, USA, 2013, pp. 1168–1176.
[40] C.L. Schelhowe, S. Kagawa, T. Gruda, J. Cybulski, D.M. Jones, Loading time matters, 2018, Zalando Engineering Blog, https://engineering.zalando.com/posts/2018/06/loading-time-matters.html. (Last checked 6 Sep. 2023).
[41] P. Callejo, M. Bagnulo, J.G. Ruiz, A. Lutu, A. García-Martínez, R. Cuevas, Measuring DoH with web ads, Comput. Netw. 212 (2022) 109046.

Jun Kurihara received degrees of the B.E., M.E. and Ph.D. in Engineering from Tokyo Institute of Technology in 2004, 2006 and 2012, respectively. From 2006 to 2017, he was with KDDI Corp. and KDDI R&D Labs., Inc. as a strategic planner and a research engineer. He is currently an associate professor at Graduate School of Information Science, University of Hyogo and also a principal researcher at Zettant Inc. He was a visiting researcher at Palo Alto Research Center (PARC), CA, USA from 2013 to 2014, and Carnegie Mellon University, PA, USA in 2022. His research interests include coding theory, networking architecture and privacy in networking. He received the Best Paper Award from IEICE in 2014.

Toshiaki Tanaka received the B.E. and M.E. degrees in Communication Engineering from Osaka University in 1984 and 1986 respectively, and the Ph.D. degree from Kyushu University in 2007. He had been with KDDI Corp. from 1986 to 2021, and was the Vice President of KDDI Research Inc. and the Fellow of KDDI Corp. since 2017. He is currently a professor at Graduate School of Information Science, University of Hyogo. His research interests include 5G security, network security, data trust and privacy. He received the MEXT Commendation for Science and Technology in 2014, the IEICE Achievement Award in 2015, and the TTC Information and Communication Technology Award in 2019.

Takeshi Kubo received the B.E. and M.E. degrees of Electronic Science and Engineering from Kyoto University in 1999 and 2001, respectively. From 2001 to 2017, he was a researcher and a project leader at KDDI Corp. and KDDI R&D Labs., Inc. He is currently the CEO of Zettant Inc. since 2017. He received the Young Researcher's Award in 2008 from IEICE.