EncDNS: A Lightweight Privacy-Preserving Name Resolution Service

Dominik Herrmann, Karl-Peter Fuchs, Jens Lindemann, and Hannes Federrath

University of Hamburg, Computer Science Department, Germany

Abstract. Users are increasingly switching to third party DNS resolvers (e. g., Google Public DNS and OpenDNS). The resulting monitoring capabilities constitute an emerging threat to online privacy. In this paper we present EncDNS, a novel lightweight privacy-preserving name resolution service as a replacement for conventional third-party resolvers. The EncDNS protocol, which is based on DNSCurve, encapsulates encrypted messages in standards-compliant DNS messages. User privacy is protected by exploiting the fact that a conventional DNS resolver provides sender anonymity against the EncDNS server. Unlike traditional privacy-preserving techniques like mixes or onion routing, which introduce considerable delays due to routing messages over multiple hops, the EncDNS architecture introduces only one additional server in order to achieve a sufficient level of protection against realistic adversaries. EncDNS is open source software. An initial test deployment is available for public use.

Keywords: anonymity, obfuscation, confidentiality, encapsulation, DNSCurve, nameserver, DNS proxy, encryption, third-party DNS, open source

1 Introduction

The Domain Name System (DNS) is a globally distributed name resolution service that is used to translate domain names like www.google.com to IP addresses. Clients offload most of the work to so-called "DNS resolvers" that query the authoritative name servers, which store the mapping information, on behalf of users. Due to their central role, DNS resolvers are a preeminent entity for behavioral monitoring as well as for access control. Numerous nations and regimes have made efforts to prevent access to websites that they deem inappropriate, among them the United States (cf. the SOPA and PIPA bills [27]), Germany [26], Pakistan [35], Turkey [45] and China [52].

In some cases users can circumvent the filtering by switching to a different resolver [52]. Apart from well-known offers like Google Public DNS and OpenDNS, there is a huge number of name servers operated by NGOs and individuals (cf. http://public-dns.tk), some of them claiming to offer high availability and confidentiality as well as low latencies. Unfortunately, switching to a freely available resolver inevitably discloses one's online activities to the DNS provider. This gives rise to privacy concerns [20]. Neither the DNS protocol nor the DNSSEC

security extensions account for privacy [3]. Therefore, the resolver can log the IP addresses of its users and the domain names they are interested in. Some experts believe that the discussions about limiting traditional tracking via cookies will result in DNS queries becoming the next target for tracking and profiling [13].

Previous work on improving confidentiality of DNS, namely DNSCurve and DNSCrypt (cf. Sect. 3), only provide link encryption, i. e., these proposals focus on protecting messages while in transit. However, link encryption is not sufficient for users who want to issue DNS queries without disclosing the desired domain names to the DNS provider. If the DNS provider learns the desired domains, privacy may be at risk even when the provider has good intentions and makes sincere commitments. This is exemplified by the case of "Lavabit", an e-mail service that has been legally obliged to disclose personal information to the authorities without being allowed to announce that breach in public [39].

This paper introduces a solution to protect confidentiality against attacks perpetrated by both eavesdropping outsiders *as well as* the DNS provider. Previous research efforts on such a privacy-enhanced DNS have not resulted in readily available systems so far. We believe that this is due to compatibility issues, high complexity as well as the penalty on latency (cf. Sect. 2). In contrast, we aim for a lightweight solution that is compatible with existing infrastructure and can be set up by a single party. Our approach is in line with a recent avenue of research, studying privacy solutions that sacrifice the objective of providing anonymity from strong adversaries in favor of low overhead and latencies [21,22].

The **contribution of this paper** is to propose EncDNS, a novel approach to provide a low-latency, privacy-preserving DNS resolution service. We describe the EncDNS architecture, the corresponding protocol as well as the message format. We have implemented a prototype of EncDNS and demonstrate via empirical evaluation that EncDNS offers low-latency name resolution. Initial tests also indicate that EncDNS is compatible with the majority of the name server implementations currently deployed on the Internet.

The paper is structured as follows. In Sect. 2 we review the Domain Name System, related work and outline the general requirements of a privacy-preserving name resolution service. After that we describe the design of EncDNS, its architecture, the name resolution process and the message format in Sect. 3. In Sect. 4 we carry out a security analysis of the proposal before we provide results from a performance evaluation in Sect. 5. Further, we assess the compatibility of EncDNS with the existing DNS infrastructure in Sect. 6. Limitations are discussed in Sect. 7 before we conclude the paper in Sect. 8.

2 Fundamentals, Related Work and Requirements

2.1 Domain Name System

The Domain Name System (RFCs 1034 and 1035 [33,34]) is used by clients to resolve human-readable domain names into IP addresses. Applications on a client computer use a *stub resolver* to send DNS *queries* to a *recursive name server*,

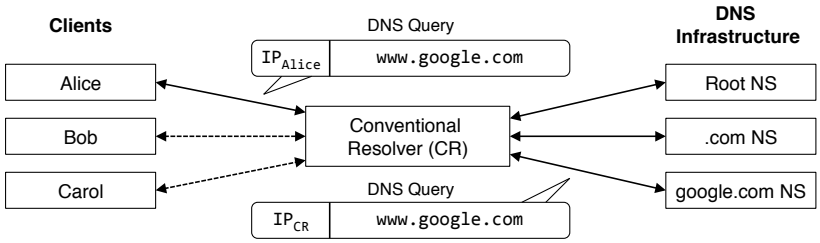

Fig. 1. Architecture of DNS

which is either operated by a user's ISP or by a third party. For reasons of clarity we will refer to the already existing recursive name servers as "Conventional Resolvers" (CRs) in this paper (cf. Fig. 1). The conventional resolver looks up incoming queries in its cache and, in case of a cache miss, retrieves the desired DNS resource record on behalf of the stub resolver from the appropriate *authoritative name servers* (Root NS, .com NS and google.com NS in Fig. 1). Once it has obtained the desired resource record, the conventional resolver will send a DNS *reply* to the stub resolver on the client. DNS messages are delivered via UDP, i. e., each DNS transaction consists of a single query and reply datagram.

The authoritative name servers collectively make up the distributed DNS infrastructure (cf. Fig. 1). An authoritative name server is responsible for a dedicated part of the DNS namespace, which is called a *zone*. The zones form a hierarchy with the so-called root zone at the top and the zones corresponding to so-called top-level domains (e. g., "com", "net", and "org") at the second level. Authoritative name servers can delegate the responsibility for a subtree in the namespace to other servers.

2.2 Related Work

In the following we will review existing proposals to provide privacy-preserving name resolution. Previous work has followed two different approaches: query obfuscation and sender anonymity.

The concept of "range queries" hides a query within a set of dummy queries. Zhao et al. [50] propose a straightforward solution: $n-1$ randomly generated dummy queries q_i are submitted together with the desired query q_{desired} to a single conventional resolver. Depending on the choice of the security parameter n, this scheme may significantly increase the load of the resolver. Zhao et al. also present a more efficient scheme [51], which is inspired by private information retrieval [12]. The principal idea consists in sending two sets of queries Q_1 and Q_2 to two different servers with $Q_1 = q_1, q_2, \ldots, q_n$ and $Q_2 = Q_1 \cup q_{\text{desired}}$. Each of the two servers j collects all IP addresses, combines them using the XOR operation and sends the result as a reply r_j to the client. The client can then obtain the desired IP address: $r = r_1 \oplus r_2$. However, this scheme requires two special resolvers, which must not collaborate. Moreover, a passive observer can trivially determine q_{desired}, because the ranges are not encrypted.

Castillo-Perez et al. [10,11] present a variation of the single-server scheme. They propose clients should construct a single range consisting of the desired query as well as $(m \cdot n) - 1$ dummy queries, then split the range into m shares and send each share to a different DNS resolver in parallel. In contrast to the two-server approach this scheme works with conventional resolvers. Moreover, query privacy is preserved even if all resolvers collude. However, the general limitations of range query schemes apply: the dummy queries increase the load on the name servers and the client has to maintain a database of plausible dummy domains.

Lu and Tsudik propose PPDNS [31], a privacy-preserving DNS system, which is built on top of CoDoNS [41], a next-generation DNS infrastructure based on distributed hash tables (DHT) and peer-to-peer technologies. In PPDNS clients issue a range query by retrieving all records whose hash value matches a *hash prefix*, which is obtained by truncating the hash value of the desired domain. While PPDNS is a promising approach, we do not expect that it will be widely adopted in the near future due to the need for a completely different DNS infrastructure and its high computational complexity, which requires special hardware.

More relevant for our work are proposals that aim for *sender anonymity*. General-purpose anonymizers like Tor could be used to hide DNS queries, but they introduce significant delays. Response times are reported to be 45 times higher, if queries are resolved via Tor, with delays reaching up to 15 s [17].

In an earlier work we suggested to implement a special purpose mix cascade that provides unlinkability between queried domain names and the identity of the sender [18]. Although [18] is specifically tailored for DNS messages, relaying messages over multiple mixes has a significant impact on performance. The median response time was 171 ms when three mixes were used; name resolution via mixes takes more than twice as long as without mixes. In order to reduce the effect of high latencies we proposed to push the resource records of popular domain names to clients. This allows clients to resolve queries for popular names with zero latency. Keeping the records of the 10,000 most popular domain names up-to-date on the client requires a bandwidth of 1.5 MB/h. A fundamental limitation of [18], which may hinder deployment, is the fact that query privacy relies on a number of additional servers that have to be run by non-colluding providers.

2.3 Requirements

In the following we briefly outline the properties a privacy-preserving name resolution service should exhibit. First of all, such a service has to ensure that the conventional resolver cannot observe the domains contained in the queries of a user. More generally, no single entity in the final design should be able to link the identity of the sender with the contents of the queries or replies.

Secondly, the design of the service must not introduce significant delays into the resolution process. Currently, DNS queries are resolved within 10–100 ms [1]. A privacy-preserving resolution service will have to achieve a comparable performance in order to be accepted by users.

Thirdly, the name resolution service has to be compatible with the existing DNS infrastructure. Fundamental changes to the DNS are deployed only very

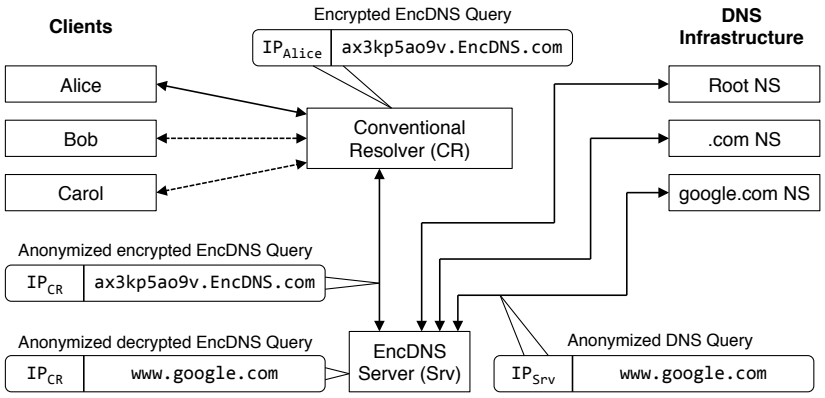

Fig. 2. Architecture of EncDNS

slowly, as exemplified by DNSSEC, which has been standardized more than ten years ago, but is still not widely available [47]. If the privacy-preserving name resolution service required changes to the DNS, it would not see widespread adoption in the near future. Moreover, the design should ensure that barriers for providers that want to offer the service as well as for users who want to use it are low. As a consequence, the service should offer a standards-compliant interface that can be accessed transparently by existing applications.

Fourth, as name resolution is a commodity service on the Internet, relaying and processing queries has to be efficient and scalable. Therefore, computational complexity on servers should be low, the protocols should be stateless and message sizes should be small.

None of the previous proposals meets all of these requirements. In the following we will outline the design of EncDNS, which aims to fulfill these requirements.

3 The EncDNS Design

We propose EncDNS (short for *Encapsulated DNS*) as a novel lightweight approach to enable anonymous usage of the DNS. The main idea of the EncDNS design is depicted in Fig. 2. Instead of using a *conventional resolver* (CR) for name resolution directly, the CR is utilized as a simple proxy that forwards DNS queries in encrypted form to an additional node, the *EncDNS server*. The encryption of queries is performed by clients to prevent the CR from learning the desired domain names.

Encrypted queries are standards-compliant DNS messages for a specially crafted domain name that consists of two parts: prefix and suffix. The prefix of this domain name contains the original query of the client, which is encrypted and integrity-protected. The suffix of the domain name (*EncDNS.com* in Fig. 2) is the domain name for which the EncDNS server is authoritative, i. e., it contains

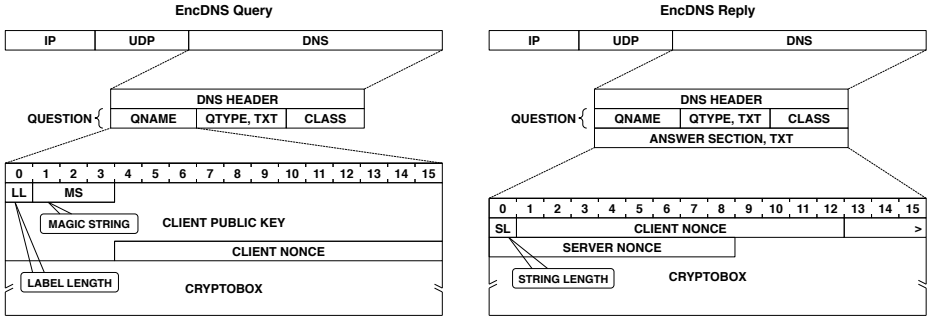

Fig. 3. The EncDNS message format

the *routing information* for the CR. Replies from the EncDNS server are delivered within standards-compliant DNS messages that contain a TXT resource record with the original DNS reply in encrypted and integrity-protected form.

The EncDNS server takes over the tasks carried out by CRs in conventional DNS, i. e., the EncDNS server performs the actual name resolution. While it is able to decrypt queries and thus learns the desired domain names, the EncDNS server cannot learn the client's IP address, since the queries are coming from the IP address of the CR.

In essence, the EncDNS design avoids a single point of trust (i. e., CRs in conventional DNS) by establishing a two-hop sender anonymous tunnel. However, only the second hop is introduced by EncDNS itself; existing conventional DNS resolvers are utilized as first hops. Thus, in EncDNS we have to address two challenges: Firstly, we have to design a message format that is **compatible with DNS**, i. e., it must be encapsulated within standard DNS messages. Secondly, EncDNS **must not introduce significant overhead** in terms of message sizes, reply times, and computational complexity. In the remainder of this section, we will describe how these challenges are addressed in EncDNS.

3.1 Encapsulation

Encapsulation of EncDNS messages in standard DNS messages is required for compatibility with CRs, which are not aware of the EncDNS protocol, but are supposed to forward EncDNS messages.

EncDNS encapsulates **encrypted queries** within the *question name* field of a standard DNS query in binary form. The question name field is the only suitable part of a DNS query for transmitting data to the EncDNS server. Binary query names comply with the DNS protocol specification: While [33, pp. 7–8] notes that "domain name comparisons [...] are done in a case-insensitive manner, assuming an ASCII character set, and a high order zero bit", it also states that implementations "should preserve [the] case" of a received domain name, because "full binary domain names for new services" may someday be needed. More concretely, RFC 2181 specifies "any binary string whatever can be used

as the label of any resource record" [16, p. 13]. A limitation of using the query name field is its maximum length of 255 octets [34, p. 10]. This restriction has implications for the choice of the cryptographic primitives (cf. Sect. 3.2).

Encrypted replies are encapsulated within the data section of a TXT resource record. Although TXT records are designed to hold so-called "character-strings", their contents are not limited to the ASCII set of characters. According to RFC 1035 character-strings are treated as binary information [34, p. 13]. While the query name can only carry a limited amount of information, there are no specific length restrictions for TXT records, apart from the general constraints of DNS or EDNS(0) messages [46].

Although our encapsulation is standards-compliant, we cannot assume that all implementations of CRs will be able to forward EncDNS messages. We evaluate its compatibility with common implementations in Sect. 6.

3.2 Cryptography

As shown in the previous section EncDNS messages, especially queries, are subject to space restrictions due to encapsulation in standard DNS messages. Thus, we have to design a message format that introduces low overhead in terms of message sizes, while still providing confidentiality and integrity of messages. Moreover, computational overhead for message processing should be kept to a minimum, so that a single server can handle a sufficiently large number of clients.

These requirements are addressed by using a message format inspired by DNSCurve [15]. DNSCurve employs a hybrid cryptosystem based on elliptic curve cryptography (Curve25519), the Salsa20 stream cipher and the Poly1305 message authentication code [5,6,7]. The encrypted output of these three cryptographic primitives is referred to as the *cryptobox*.

In the following we outline the construction of queries and replies in the EncDNS protocol. A detailed overview of the cryptographic operations involved in obtaining the cryptobox is given in [8,9].

Each EncDNS client and EncDNS server has a Curve25519 key pair. Whenever an EncDNS client is about to send an **encrypted query**, it uses its *private key a* and the *public key of the EncDNS server B* (which has been obtained out-of-band) in order to calculate a message-independent shared secret key k_{aB}. The secret key is used in conjunction with a *client nonce* in order to create the cryptobox from the original DNS query. The format of EncDNS queries is depicted in Fig. 3. We focus on the part that is encapsulated in the question name of a DNS query. An EncDNS query starts with a magic string (a constant protocol identifier), the (current) public key of the client and the client nonce. The next block is the cryptobox containing the original query. Finally, the client appends the domain name of the EncDNS server, in order to allow the CR to forward the message to its destination. Note that the EncDNS client includes its public key in every query, i. e., the EncDNS server can process messages in a stateless fashion, which is one of the central properties of DNS. This is affordable because Curve25519 public keys consume only 32 octets.

When the EncDNS server receives an encrypted query, it decrypts the cryptobox. To this end, the server derives the secret key k_{Ab} from *its private key b* and the *public key of the client A*, which is equivalent to the secret key obtained by the client, i. e., $k = k_{aB} = k_{Ab}$. Using k and the *client nonce* the EncDNS server decrypts the cryptobox, obtains the plaintext DNS query of the client, and resolves the desired domain name.

Once the EncDNS server has obtained the resource record from the authoritative servers, it will construct an **encrypted reply** as follows: The EncDNS server chooses a *server nonce*, which is used in conjunction with the *client nonce* and the shared secret key k to create a cryptobox from the plaintext DNS reply. The format of EncDNS replies is depicted in Fig. 3. The reply is encapsulated in a TXT record and contains the client nonce, the server nonce and the cryptobox.

When the EncDNS client receives an EncDNS reply, it determines the corresponding query based on the value of the client nonce field, which serves as transaction identifier. If there is no unanswered query, the reply will be dropped. In order to decrypt the contents of the cryptobox the client uses the shared secret key k, the client nonce and the server nonce.

Key Pair Re-use and Secret Key Caching An EncDNS client can re-use its key pair and the derived shared secret key k for multiple queries (see Sect. 4 for security implications of key re-use). Thus, the EncDNS server will obtain the same shared secret key every time it receives a query from this specific client. A straightforward optimization consists in caching k in the EncDNS client as well as in the EncDNS server. This spares client and server from repeatedly performing the same asymmetric cryptographic operations, reducing the overall computational effort. The practice of caching k effectively creates an anonymous tunnel between EncDNS client and server.

3.3 Open Source Prototype and Test Installation

We have implemented an EncDNS client as well as an EncDNS server in the gMix Framework [19] with the Java programming language. For cryptographic operations the implementation uses a Java Native Interface (JNI) binding to "libsodium", which is a platform independent port of the NaCL library, providing fast implementations of the Curve25519, Salsa20, and Poly1305 algorithms.[1]

We have released the EncDNS server and client implementation as open source. Source code as well as pre-compiled binaries for Linux and Windows systems are available at https://svs.informatik.uni-hamburg.de/gmix/. Moreover, for further field tests we have set up a publicly available open resolver, which runs an EncDNS server. We have made this server authoritative for the domain name *enc1.us.to*. It can be accessed by EncDNS clients to test compatibility with various CRs on the Internet. Setup instructions and the public key of *enc1.us.to* can be obtained from the mentioned website. We encourage readers to try out the EncDNS client and server and report any issues observed.

[1] Homepages: https://github.com/jedisct1/libsodium and http://nacl.cr.yp.to

4 Security Analysis

In the following we will analyze the security properties of EncDNS in terms of query privacy, query integrity and availability.

4.1 Query Privacy and Attacker Model

Query privacy is protected if an adversary does not learn the IP address *and* the desired domain name for a given query. EncDNS provides query privacy against the CR as well as against the EncDNS server. It also provides query privacy against observers eavesdropping either on the link between the EncDNS client and the CR or on the link between the CR and the EncDNS server.

EncDNS does *not* offer query privacy if the provider of the CR colludes with the provider of the EncDNS server. If these two servers share their knowledge they can correlate the individual queries along the route, thus linking sender IP addresses with the desired domain names. We point out that other low-latency anonymization services, like Tor and AN.ON are subject to this limitation as well: Entry and exit nodes can correlate incoming and outgoing messages by timing and tagging attacks [28,40].

After a client has obtained an address record via EncDNS, it will usually establish a TCP connection to the target host, exchanging packets that contain the source IP address of the user as well as the destination IP address of the target host. A quite obvious limitation of EncDNS is that the desired domain names cannot be disguised from an adversary that operates the target host or is able to observe the network link between the user and the target host, i. e., a user cannot disguise the websites he visits from the operator of the web servers.

Furthermore, query privacy is not protected, if the CR is also authoritative for a specific DNS zone. If the user issues a query for a domain name in that zone, the provider of the CR will be able to link the plaintext DNS query of the EncDNS server to the corresponding encrypted query of the EncDNS client based on the timing of these two queries. In other words: if a user relays EncDNS queries using the CR of his own organization, he should not expect to be able to resolve domain names of his own organization privately using EncDNS.

Another privacy issue stems from the fact that encrypted messages can be linked as long as the EncDNS client uses the same key pair. Therefore, the EncDNS server can track the activities of a (anonymous) user, even if the user is issuing queries using different IPs and different CRs. To achieve query unlinkability EncDNS clients can be configured to use **ephemeral keypairs**, i. e., at runtime they create new key pairs in regular intervals. For maximum protection clients can be configured to use each key pair for a single query only. However, query unlinkability comes at a cost: it conflicts with key caching (cf. Sect. 3.2).

4.2 Message Integrity

The EncDNS protocol provides message integrity protection between the EncDNS client and the EncDNS server, i. e., manipulation of messages by CRs can be detected. EncDNS does not offer end-to-end integrity, i. e., a malicious EncDNS

server could forge the IP addresses in its replies (DNS spoofing). This is not a specific limitation of EncDNS; users of existing third-party DNS resolvers have to trust the operators as well.

However, while "professional" operators may refrain from tampering with responses facing loss of reputation, the risk of poisonous replies may be higher for voluntarily provided EncDNS servers (cf. the issues with malicious exit nodes in Tor [32,49]). Once DNSSEC is deployed on a large scale, end-to-end integrity protection will be available. A temporary solution would consist in extending the implementation of the EncDNS client, so that it issues queries to multiple EncDNS servers in parallel in order to detect forged replies. This approach resembles CoDNS [38] and is also being investigated to detect faked web server certificates used in man-in-the-middle attacks [48]. However, asking multiple servers introduces new challenges: Content delivery networks reply with various IP addresses, the choice of which depends on the location of the EncDNS server.

4.3 Availability

Finally, we analyze availability aspects. On the one hand there is the risk of a denial of service attack against an EncDNS server. On the other hand, an EncDNS server could be used to leverage an amplification attack.

We start out by considering *denial-of-service attacks* against EncDNS servers. In contrast to DNSSEC, which uses offline signing of messages, EncDNS uses online encryption, which means that an adversary may be able to induce a significant load on the EncDNS server by sending EncDNS messages to it. In the following we analyze the effectiveness of potential mitigation techniques.

First of all, EncDNS messages contain a *magic string*, which allows the EncDNS server to identify EncDNS messages immediately upon receipt. Messages without the magic string, e. g., standard DNS queries, which are seen on EncDNS servers due to bots that probe the Internet for open resolvers, are dropped immediately. However, the magic string cannot protect against denial-of-service attacks by dedicated adversaries, who create a valid EncDNS query once and repeatedly send it to the EncDNS server. Identically replayed messages could be detected by the EncDNS server due to the fact that they contain identical client nonces. However, the adversary could easily vary the client nonce (as well as any other part of the query).

Denial-of-service attacks are a general problem of proxy services that do not require authentication. In future work we plan to adapt the EncDNS protocol so that EncDNS servers can demand that clients include a proof-of-work (client puzzle) in their queries, which can be verified efficiently by the EncDNS server.

In an *amplification attack* the adversary exploits the fact that he can induce a DNS resolver to send out a large reply with a comparatively small query [24]. This traffic amplification effect can be used to carry out a reflected denial-of-service attack against a victim. To this end the adversary will send a large number of small DNS queries containing the IP address of the victim in the source IP address field to (multiple) DNS resolvers, asking for a large resource record, which will effectively result in overloading the victim. EncDNS servers

Table 1. Scalability evaluation of EncDNS server (Experiment 1, key caching enabled)

	EncDNS		Baseline	
Queries/sec	Failures [%]	CPU load [%]	Failures [%]	CPU load [%]
2000	0.00	23.6	0.00	13.9
4000	0.00	47.9	0.00	22.3
6000	0.00	63.9	0.00	30.9
8000	11.82	75.4	0.00	41.6

are of little use for amplification attacks, because the amplification factor is smaller than in conventional DNS due to larger size of encrypted queries.

5 Performance Evaluation

In this section we evaluate the performance of EncDNS. We consider two distinct scenarios. The first scenario (Setup 1: Lab Environment) is used to assess the scalability of the EncDNS components, i. e., we want to determine the number of concurrent users that can be serviced by a single EncDNS server. In the second scenario (Setup 2: Emulated Environment) we want to determine the effective user-perceived latency for name resolution via EncDNS. To this end, we extend Setup 1 by using the network emulator *netem* [30,37] that simulates real-world latencies between the individual components. In both scenarios, the evaluation environment consists of a local network (1 Gbps) with off-the-shelf desktop machines (Intel Core i5-3100 quad cores, 8 GB RAM, CentOS 6).

5.1 Experiment 1: Scalability Assessment

In order to assess the scalability of EncDNS, we start out with experiments in the lab environment (Setup 1). To this end, we deploy load generators on multiple machines. Each load generator sends encrypted EncDNS queries towards a single EncDNS server (bottleneck). The server decrypts the queries and constructs encrypted replies that are sent back to the load generator. The load generator tracks its queries and the corresponding replies. When the EncDNS server becomes overloaded, it will not be able to receive all incoming queries any more, i. e., the load generators will not observe a reply for each query (resolution failure due to overload). In order to determine the maximum achievable throughput, we increased the query rate incrementally from 2000 to 8000 queries/sec.

The results of this set of experiments are shown in Table 1 (EncDNS columns). For up to 6000 queries/sec, all queries are processed. The CPU loads, which are denoted in the table, increase in a linear fashion with the query rate, because due to the EncDNS design every query can be processed independently from all other queries. At 8000 queries/sec, 11.82 % of queries remain unanswered due to overload of the EncDNS server. At this point the average CPU load (averaged over all four cores) is 75.4 %. Further analysis has shown that CPU loads

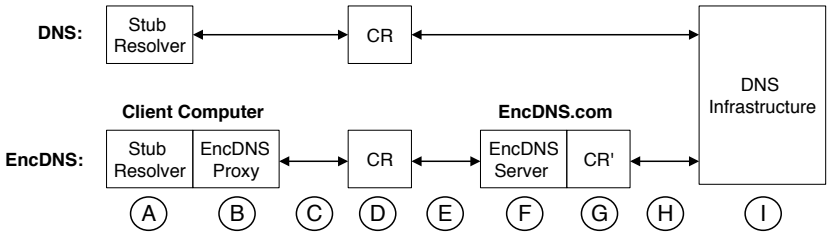

Fig. 4. Components and connections involved in (Enc)DNS name resolution

cannot approach 100 % due to context switches between the EncDNS server and resolver components running on the machine under test. The baseline measurements shown in Table 1 (same experiment without cryptographic operations) indicate that the cryptographic operations account for roughly 50 % of the load.

5.2 Experiment 2: User-perceived Latency

In the following we extend the setup of Experiment 1 to resemble a real-world deployment of EncDNS. The goal is to assess the impact of EncDNS on query latency from a user's perspective. Figure 4 shows the components and connections involved in the resolution process. The effective latency is the sum of individual latencies introduced by network connections (labeled with C, E and H in Fig. 4) and the components A, B, D, F, G and I. We focus on the delay introduced between A and G, because this part of the system replaces the conventional name resolution process. We exclude the latency introduced by H and I, because this part of the resolution process is not affected by the use of EncDNS and lookup latencies vary heavily depending on the performance of the authoritative name servers [29]. To this end CR' (label G in Fig. 4) is configured to be authoritative for all queries issued during the experiment.

We consider three configurations: (1) a baseline measurement, which routes messages through EncDNS client, CR, and EncDNS server *without any encryption*, (2) EncDNS with key caching disabled, and (3) EncDNS with key caching enabled. During all experiments the client uses a single key pair for all queries.

By comparing the results for configurations (1) and (2) we can observe the impact of the asymmetric as well as the symmetric cryptographic operations. The performance of configuration (2) is to be expected when clients use a new ephemeral key pair for every single query (worst case; assuming that key pairs are pre-generated by the client). Comparing configurations (2) and (3) allows us to specifically observe the time needed for the asymmetric operations, i. e., we can evaluate the utility of the key caching mechanism (best case).

In order to observe the computational delay without any bias we start out by excluding network delays (connections C and E) from the measurements. Later on we employ a network emulator to incrementally increase network delays, which allows us to study the performance under more realistic conditions. We measure the **response time**, which equals the duration of a DNS lookup from

Table 2. Response times in Experiment 2 for various emulated delays between CR and EncDNS Server

Measurement	Delay (ms)	P_{25}	P_{50}	P_{75}
(1.1) Baseline	$C + E = 0$	1.36	1.39	1.41
(1.2) EncDNS	0	1.77	1.80	1.84
(1.3) EncDNS + cache	0	1.61	1.65	1.68
(2.1) Baseline	30+10	41.97	42.00	42.02
(2.2) EncDNS	30+10	42.50	42.53	42.56
(2.3) EncDNS + cache	30+10	42.13	42.17	42.19
(3.1) Baseline	30+50	81.98	82.00	82.03
(3.2) EncDNS	30+50	82.48	82.51	82.54
(3.3) EncDNS + cache	30+50	82.16	82.19	82.22
(4.1) Baseline	30+100	132.01	132.04	132.06
(4.2) EncDNS	30+100	132.45	132.49	132.52
(4.3) EncDNS + cache	30+100	132.13	132.16	132.19

a user's perspective: To this end we issue queries via a single EncDNS client (A) at a fixed rate (30 ms interval between queries). The response time consists of the time span between issuing a query and receiving the corresponding reply. To obtain significant results we issued 10,000 queries in each experiment. Based on the observed response times we calculated the 25th (P_{25}), 50th (P_{50}, the median) and 75th percentile (P_{75}). The results are shown in Table 2.

In the baseline configuration (1.1), we observe a median latency of 1.39 ms, which is caused by forwarding the query and reply between A and G. Enabling cryptographic operations (1.2) increases median latency by 29 % to 1.80 ms. When the key caching mechanism (1.3) is enabled, median latency decreases by 8 % to 1.65 ms, i. e., the median latency is only 19 % higher than in the baseline measurement. The values for P_{25} and P_{75} are very close to the median value.

The remaining experiments take network latencies into account. For connection C we set the round-trip time to a fixed value of 30 ms (latency of 15 ms in each direction), because we assume that the CR is operated by the ISP of the user or the user has selected a CR, which is geographically close. As the effective latency between the CR and the EncDNS server depends on the distance between them in practice, we simulate various typical conditions found on the Internet by varying round-trip times between 10 and 100 ms (cf. Table 2). The results indicate that the overhead introduced by EncDNS is constant and independent of the network delay. In practical deployments we expect that network delays and lookup latency (labels H and I in Fig. 4) will dominate the user-perceived latency. As anonymization services that rely on distributing trust need to forward traffic over at least two hops, some additional network delay is inevitable. The delay caused by message encryption is much smaller and is expected to be negligible in practice.

Table 3. Results of compatibility tests for popular CR implementations

Software	Version	Binary labels	Binary TXT records
BIND	9.7.3	✓	✓
MaraDNS	1.4.03	✓	✓
Unbound	1.4.6	✓	✓
PowerDNS	3.2	✓	✓
dnscache	1.05	✓	✓
Windows Server	2012 R2	×	✓

6 Compatibility Assessment

As explained in Sect. 3.1, encrypted EncDNS queries are encapsulated within the query name field of standard DNS queries in binary form using octets in the range from 0x00 to 0xff. Some CRs and intermediate DNS forwarders may not expect binary domain names. They might mangle the query name or discard the encrypted queries altogether. In order to assess the practicability of our encapsulation scheme, we have relayed EncDNS queries over commonly used recursive name server implementations in their default configuration.

The results of our compatibility tests are depicted in Table 3. The table indicates whether the respective implementation can handle binary labels (needed for transportation of encrypted queries) and binary data in TXT records (needed for transportation of encrypted replies). According to the results almost all popular implementations forward EncDNS queries and replies without interference. Only the name server of Windows Server 2012 R2 fails to relay EncDNS traffic: While it forwards our encrypted queries, it fails to link the encrypted replies to them; it reports a SERVFAIL error to the EncDNS client instead.

While this result look promising, we point out that all CRs that use the "0x20 encoding" security mechanism [14] *will* interfere with the encrypted query names. When a CR is configured to employ the 0x20 encoding scheme, it performs a random ad-hoc modification of the capitalization of each letter in the query name before forwarding the query to the authoritative name server (i. e., the EncDNS server). This measure is supposed to increase the entropy within DNS queries to foil cache poisoning attacks. In a 2013 survey only 0.3 % of the evaluated DNS resolvers used 0x20 encoding [44]. If its use becomes more widespread in the future, the EncDNS message format will have to be adapted (cf. Sect. 7).

7 Discussion and Future Work

In the following we will discuss limitations and open questions regarding the security analysis (Sect. 4), the performance evaluation (Sect. 5) and the compatibility assessment (Sect. 6). Additionally, we will point out possible deployment issues and how they can be overcome in future work.

We outlined the privacy properties of EncDNS in the **security analysis**. EncDNS prevents the CR from learning the queried domains. However, the current implementation does not incorporate *message padding*. This may allow the CR to infer the queried domains from the size of the encrypted payload. As the majority of the domain names is short, we conjecture that inference attacks have only limited effectiveness. In future work we will study the utility of various padding schemes and their impact on performance. Our preliminary tests indicate that message sizes have only a negligible effect on user-perceived latencies.

The results from the **performance evaluation** indicate that EncDNS is sufficiently scalable. Given the result of [18], who found that a user issues 0.05 DNS queries/second on average, the results of our scalability assessment (Experiment 1) suggest that a single EncDNS server, which can handle 6000 queries/second according to our scalability assessment, would be able to serve up to 120,000 concurrent users. However, this extrapolation is inadmissible, because DNS queries are neither evenly distributed among users, nor are they sent at a constant rate. DNS traffic typically contains bursts of queries, which occur when a browser retrieves a website that contains content from multiple web servers. In future work, we plan to assess the scalability of the EncDNS server with trace-driven simulations in order to provide a more realistic account of its scalability. The initial results presented in this paper indicate that a single EncDNS server will be able to create a sufficiently large anonymity set.

In Experiment 2 we measured the user-perceived latency and found that the overhead of the EncDNS components is almost constant and independent of network latency. This is mainly due to our design decision to relay EncDNS messages statelessly with UDP. Therefore, EncDNS is not subject to issues found in TCP-based overlay networks such as *head-of-line blocking* [25,36,42] or *cross-circuit interference* [2,42]. These effects result from the combination of TCP congestion control with multiplexing and have been shown to have a significant impact on the performance of systems like Tor.

According to the results of the **compatibility assessment** EncDNS queries and replies are forwarded by common name server implementations. If it turns out that implementations have difficulty with the binary message format, we could switch to a more conservative encoding such as Base32 [23]. Additionally, the message format could be extended to support message fragmentation in order to handle large queries and replies.

Future work will also have to consider some **deployment issues**: Firstly, the EncDNS design does not contain a directory service, i. e., techniques for server discovery are out of the scope of our proposal. Initially, a central bulletin board on a website may be sufficient for this purpose. Closely related to server discovery is the matter of key distribution: EncDNS clients need an authenticated copy of the public key of their EncDNS server. In the long run, DANE [4] could be used for authenticated key storage and distribution.

Finally, we remark on a practical privacy issue: If users configure their operating system to use the EncDNS client as DNS resolver, this will relay *all DNS queries* to the EncDNS server. This is undesirable in scenarios where users are

in a local network with a *split-horizon* or hybrid DNS server that functions as recursive name server, but is also authoritative for some internal domain names (e. g., *database-server.corp.local*). Queries for internal domain names will be forwarded to the EncDNS server, which will be unable to resolve them, because they are not part of the public DNS namespace. As a result users may be unable to reach internal services. Moreover, private information (internal hostnames, for instance) may be disclosed to the EncDNS server. A straightforward countermeasure consists in a blacklist within the EncDNS client that explicitly denotes the domain names that should not be forwarded to the EncDNS server. Note that other drop-in anonymization services that use a proxy on the user's machine are subject to this privacy issue as well.

8 Conclusion

In this paper we have presented EncDNS, a lightweight open source name resolution service that leverages existing DNS resolvers to protect the privacy of its users. We have described the EncDNS architecture, the protocol and the message format. DNS queries are encrypted by the EncDNS client software, which runs on the machine of a user, and forwarded to the EncDNS server via a conventional DNS resolver. The EncDNS server decrypts incoming queries, obtains the desired resource records and responds with an encrypted reply. As EncDNS relies on conventional resolvers, encrypted messages are encapsulated in standards-compliant DNS queries and replies. According to our experiments EncDNS provides low-latency DNS resolution and is compatible with almost all popular DNS resolvers. We encourage researchers and users to evaluate the EncDNS prototype and to report on any issues found in practice.

References

1. Ager, B., Mühlbauer, W., Smaragdakis, G., Uhlig, S.: Comparing DNS Resolvers in the Wild. In: Allman, M. (ed.) SIGCOMM Conference on Internet Measurement 2010 (IMC 2010). pp. 15–21. ACM (2010)
2. AlSabah, M., Goldberg, I.: PCTCP: per-circuit TCP-over-IPsec transport for anonymous communication overlay networks. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) Conference on Computer and Communications Security (CCS 2013). pp. 349–360. ACM (2013)
3. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: DNS Security Introduction and Requirements. RFC 4033 (2005)
4. Barnes, R.: Use Cases and Requirements for DNS-Based Authentication of Named Entities (DANE). RFC 6394 (2011)
5. Bernstein, D.J.: The Poly1305-AES Message-Authentication Code. In: Gilbert, H., Handschuh, H. (eds.) 12th International Workshop on Fast Software Encryption (FSE 2005), Revised Selected Papers. Lecture Notes in Computer Science, vol. 3557, pp. 32–49. Springer (2005)
6. Bernstein, D.J.: Curve25519: New Diffie-Hellman Speed Records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) 9th International Conference on Theory

and Practice of Public-Key Cryptography (PKC 2006). Lecture Notes in Computer Science, vol. 3958, pp. 207–228. Springer (2006)
7. Bernstein, D.J.: The Salsa20 Family of Stream Ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs, Lecture Notes in Computer Science, vol. 4986, pp. 84–97. Springer (2008)
8. Bernstein, D.J., Lange, T., Schwabe, P.: The Security Impact of a New Cryptographic Library. In: Hevia, A., Neven, G. (eds.) 2nd International Conference on Cryptology and Information Security in Latin America (LATINCRYPT 2012). Lecture Notes in Computer Science, vol. 7533, pp. 159–176. Springer (2012)
9. Bernstein, D.J.: Cryptography in NaCl. Tech. rep., Department of Computer Science (MC 152), The University of Illinois, Chicago, IL (Mar 2009), http://cr.yp.to/highspeed/naclcrypto-20090310.pdf
10. Castillo-Perez, S., García-Alfaro, J.: Anonymous Resolution of DNS Queries. In: On the Move to Meaningful Internet Systems (OTM 2008). Lecture Notes in Computer Science, vol. 5332, pp. 987–1000. Springer (2008)
11. Castillo-Perez, S., García-Alfaro, J.: Evaluation of Two Privacy–Preserving Protocols for the DNS. In: 6th International Conference on Information Technology: New Generations (ITNG 2009). pp. 411–416. IEEE (2009)
12. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private Information Retrieval. In: Proceedings of the 36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin. pp. 41–50. IEEE (1995)
13. Conrad, D.: Towards Improving DNS Security, Stability, and Resiliency (2012), http://www.internetsociety.org/sites/default/files/bp-dnsresiliency-201201-en_0.pdf
14. Dagon, D., Antonakakis, M., Vixie, P., Jinmei, T., Lee, W.: Increased DNS Forgery Resistance Through 0x20-bit Encoding: Security via Leet Queries. In: Ning, P., Syverson, P.F., Jha, S. (eds.) Conference on Computer and Communications Security (CCS 2008). pp. 211–222. ACM (2008)
15. Dempsky, M.: DNSCurve: Link-Level Security for the Domain Name System. Internet Draft draft-dempsky-dnscurve-01, RFC Editor (2010)
16. Elz, R., Bush, R.: Clarifications to the DNS Specification. RFC 2181 (1997)
17. Fabian, B., Goertz, F., Kunz, S., Müller, S., Nitzsche, M.: Privately Waiting – A Usability Analysis of the Tor Anonymity Network. In: Santana, M., Luftman, J.N., Vinze, A.S. (eds.) 16th Americas Conference on Information Systems (AMCIS 2010). p. 258. Association for Information Systems (2010)
18. Federrath, H., Fuchs, K.P., Herrmann, D., Piosecny, C.: Privacy-Preserving DNS: Analysis of Broadcast, Range Queries and Mix-Based Protection Methods. In: Atluri, V., Díaz, C. (eds.) ESORICS. Lecture Notes in Computer Science, vol. 6879, pp. 665–683. Springer (2011)
19. Fuchs, K.P., Herrmann, D., Federrath, H.: Introducing the gMix Open Source Framework for Mix Implementations. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS. Lecture Notes in Computer Science, vol. 7459, pp. 487–504. Springer (2012)
20. Goodson, S.: If You're Not Paying For It, You Become The Product. Forbes.com (2012), http://onforb.es/wVrU4G
21. Hsiao, H.C., Kim, T.H.J., Perrig, A., Yamada, A., Nelson, S.C., Gruteser, M., Meng, W.: LAP: Lightweight Anonymity and Privacy. In: IEEE Symposium on Security and Privacy (S&P 2012). pp. 506–520. IEEE (2012)
22. Jansen, R., Johnson, A., Syverson, P.F.: LIRA: Lightweight Incentivized Routing for Anonymity. In: 20th Annual Network and Distributed System Security Symposium (NDSS 2013). The Internet Society (2013)

23. Josefsson, S.: The Base16, Base32, and Base64 Data Encodings. RFC 4648 (2006)
24. Kambourakis, G., Moschos, T., Geneiatakis, D., Gritzalis, S.: Detecting DNS Amplification Attacks. In: Lopez, J., Hämmerli, B.M. (eds.) 2nd International Workshop on Critical Information Infrastructures Security (CRITIS 2007), Revised Papers. Lecture Notes in Computer Science, vol. 5141, pp. 185–196. Springer (2007)
25. Karol, M., Hluchyj, M., Morgan, S.: Input versus output queueing on a space-division packet switch. IEEE Trans. on Communications 35(12), 1347–1356 (1987)
26. Kleinschmidt, B.: An International Comparison of ISP's Liabilities for Unlawful Third Party Content. I. J. Law and Information Technology 18(4), 332–355 (2010)
27. Lemley, M., Levine, D.S., Post, D.G.: Don't Break the Internet. 64 Stan. L. Rev. Online 34 (2011), http://www.stanfordlawreview.org/online/dont-break-internet
28. Levine, B.N., Reiter, M.K., Wang, C., Wright, M.K.: Timing attacks in low-latency mix systems (extended abstract). In: Juels, A. (ed.) Financial Cryptography. Lecture Notes in Computer Science, vol. 3110, pp. 251–265. Springer (2004)
29. Liang, J., Jiang, J., Duan, H.X., Li, K., Wu, J.: Measuring Query Latency of Top Level DNS Servers. In: Roughan and Chang [43], pp. 145–154
30. Linux Foundation: Netem (2009), http://www.linuxfoundation.org/collaborate/workgroups/networking/netem
31. Lu, Y., Tsudik, G.: Towards Plugging Privacy Leaks in the Domain Name System. In: 10th International Conference on Peer-to-Peer Computing (P2P 2010). pp. 1–10. IEEE, IEEE (2010)
32. McCoy, D., Bauer, K.S., Grunwald, D., Kohno, T., Sicker, D.C.: Shining Light in Dark Places: Understanding the Tor Network. In: Borisov, N., Goldberg, I. (eds.) Privacy Enhancing Technologies (PETS 2008). Lecture Notes in Computer Science, vol. 5134, pp. 63–76. Springer (2008)
33. Mockapetris, P.: Domain Names: Concepts and Facilities. RFC 1034 (1987)
34. Mockapetris, P.: Domain Names: Implementation and Specification. RFC 1035 (1987)
35. Nabi, Z.: The Anatomy of Web Censorship in Pakistan. CoRR abs/1307.1144 (2013)
36. Nowlan, M.F., Wolinsky, D., Ford, B.: Reducing Latency in Tor Circuits with Unordered Delivery. In: 3rd USENIX Workshop on Free and Open Communications on the Internet. USENIX Association (2013)
37. Nussbaum, L., Richard, O.: A Comparative Study of Network Link Emulators. In: Wainer, G.A., Shaffer, C.A., McGraw, R.M., Chinni, M.J. (eds.) Proceedings of the 2009 Spring Simulation Multiconference. SCS/ACM (2009)
38. Park, K., Pai, V.S., Peterson, L.L., Wang, Z.: CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups. In: 6th Symposium on Operating System Design and Implementation. pp. 199–214. USENIX Association (2004)
39. Poulson, K.: Edward Snowden's E-Mail Provider Defied FBI Demands to Turn Over Crypto Keys, Documents Show. Wired, http://www.wired.com/threatlevel/2013/10/lavabit_unsealed
40. Pries, R., Yu, W., Fu, X., Zhao, W.: A new replay attack against anonymous communication networks. In: International Conference on Communications (ICC 2008). pp. 1578–1582. IEEE (2008)
41. Ramasubramanian, V., Sirer, E.G.: The Design and Implementation of a Next Generation Name Service for the Internet. In: SIGCOMM 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. pp. 331–342. ACM (2004)
42. Reardon, J., Goldberg, I.: Improving Tor using a TCP-over-DTLS Tunnel. In: USENIX Security Symposium. pp. 119–134. USENIX Association (2009)

43. Roughan, M., Chang, R.K.C. (eds.): Passive and Active Measurement (PAM 2013), Lecture Notes in Computer Science, vol. 7799. Springer (2013)
44. Schomp, K., Callahan, T., Rabinovich, M., Allman, M.: Assessing DNS Vulnerability to Record Injection. In: Faloutsos, M., Kuzmanovic, A. (eds.) Passive and Active Measurement (PAM 2014). Lecture Notes in Computer Science, vol. 8362. Springer (2014)
45. Verkamp, J.P., Gupta, M.: Inferring Mechanics of Web Censorship Around the World. In: 2nd USENIX Workshop on Free and Open Communications on the Internet. USENIX Association (2012)
46. Vixie, P.: Extension Mechanisms for DNS (EDNS0). RFC 2671 (1999)
47. Wander, M., Weis, T.: Measuring Occurrence of DNSSEC Validation. In: Roughan and Chang [43], pp. 125–134
48. Wendlandt, D., Andersen, D.G., Perrig, A.: Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing. In: USENIX Annual Technical Conference. pp. 321–334. USENIX (2008)
49. Winter, P., Lindskog, S.: Spoiled Onions: Exposing Malicious Tor Exit Relays. CoRR abs/1401.4917 (2014)
50. Zhao, F., Hori, Y., Sakurai, K.: Analysis of Privacy Disclosure in DNS Query. In: International Conference on Multimedia and Ubiquitous Engineering (MUE 2007). pp. 952–957. IEEE (2007)
51. Zhao, F., Hori, Y., Sakurai, K.: Two–Servers PIR Based DNS Query Scheme with Privacy–Preserving. In: International Conference on Intelligent Pervasive Computing (IPC 2007). pp. 299–302. IEEE (2007)
52. Zittrain, J., Edelman, B.: Internet Filtering in China. IEEE Internet Computing 7(2), 70–77 (2003)