Look-aside At Your Own Risk: Privacy Implications of DNSSEC Look-aside Validation

Aziz Mohaisen, Zhongshu Gu, Kui Ren, Zhenhua Li, Charles Kamhoua, Laurent Njilla, and DaeHun Nyang

Abstract—The Domain Name System Security Extension (DNSSEC) leverages public-key cryptography to provide data integrity, source authentication, and denial of existence for DNS responses. To complement DNSSEC operations, DNSSEC Look-aside Validation (DLV) is designed for alternative off-path validation. Although DNS privacy attracts a lot of attention, the privacy implications of DLV are not fully investigated and understood. In this paper, we take a first in-depth look into DLV, highlighting its lax specifications and privacy implications. By performing extensive experiments over datasets of domain names under comprehensive experimental settings, our findings firmly confirm the privacy leakages caused by DLV. We discover that a large number of domains that should not be sent to DLV servers are being leaked. We explore the root causes, including the lax specifications of DLV. We also propose two approaches to fix the privacy leakages. Our approaches require trivial modifications to the existing DNS standards, and we demonstrate their cost in terms of latency and communication.

1 Introduction

The hierarchical property of Domain Name System (DNS) creates dependencies between multiple administrative domains. The resolution of a domain name requires cooperations among those domains. For example, to resolve `www.example.com`, the collaborations of the DNS root, the authority server for the top level domain (TLD) of `com`, the authority server for the second level domain (SLD) of `example`, and the authority server of the third level domain `www` are required. DNS was originally designed without security in mind; however within today's Internet, DNS hijacking, poisoning, and spoofing have greatly undermined DNS security [27]. To address many of those concerns, Arends *et al.* [4] proposed the Domain Name System Security Extensions (DNSSEC) to ensure authenticity and integrity of DNS responses. The DNSSEC serves as an authenticated directory of the Internet.

DNSSEC leverages cryptographic methods to secure origin authentication, data integrity, and denial of existence of DNS reponses. Digital signatures ensure the authenticity of DNS responses by validating against a public key of the signer. Given the hierarchical nature of DNS, certain trust chains that include involved parties need to be validated up to a trust anchor. Such trust chains end with the root, which has a key pre-distributed and hardcoded in the operating system for validation (see section 2). Accordingly, users only need to trust the root to successfully validate a DNSSEC response: a signature of a record is accepted and consumed by a user if the signature is validated using the signing public key. The authenticity of the signing keys beneath the root is ensured using key signing keys (KSKs). The validation of a record fails when any link in the chain of trust fails.

DNSSEC's deployment is incomplete and only a small proportion of domains have a complete chain of trust up to the root. While 85% of TLDs are signed, only about 3% of SLDs are signed as of early 2016 [24]. Thus, although some domain names have the capability of signing their zones, they cannot be validated up to the root because there is no delegation signer (DS) in the parent zone to validate the authenticity of the signing key. To address this issue, Weiler [49] proposed DNSSEC look-aside validation (DLV) to allow publishing of trust anchors outside of the delegation chain. By publishing DLV records in DLV repositories, domains are validated with the trust anchors in DLV records. Recursive resolvers are configured to use the DLV repositories for validation. The configurations of the DLV options in the recursive resolver vary from one operating system and installation to another.

Both DNS and DNSSEC are designed without any privacy notions in mind. However, DNS privacy has recently emerged as a topic of greater interests [37], [50]. DNS queries provide an advantage to adversaries for profiling, especially under pervasive monitoring [6]. DNS traffic is metadata, which can be easily eavesdropped, analyzed, and used. In addition, such metadata is highly valuable, for its power in characterizing users and understanding their behaviors. DNS queries are used to correctly link users with their browsing history [5], [28]. Various research and development efforts are made to understand DNS query leakage in both academia [43], [21], [42] and industry [17], [16].

Although RFC 5074 specifies DLV's use and the general validator's behavior [49], it leaves out a lot of details to implementations, including guidelines on when a DLV server is queried for the various use cases of DLV (see section 2.3). Whether such lax specifications affect users' privacy and expose unintended queries to DLV servers was not tested before. Indeed, the prior art on DNS privacy treats unsecured DNS. Thus, we analyze the privacy leakage of DLV by highlighting the potential of *unintentional* DLV queries sent to the DLV servers while providing no validation

A. Mohaisen is with the Department of Computer Science at the University of Central Florida, Orlando, FL. K. Ren is with the Department of Computer Science and Engineering at the University at Buffalo, the State University of New York, Buffalo, NY. Z. Gu is with IBM T.J. Watson Research Center, Yorktown Heights, NY. Z. Li is with the School of Software, Tsinghua University. C. Kamhoua is with the Army Research Laboratory, Network Security Branch, Adelphi, MD, and L. Njilla is with the Air Force Research Lab, Rome, NY. D. Nyang (corresponding author) is with Inha University, Incheon, South Korea. E-mail: mohaisen@buffalo.edu, kuiren@buffalo.edu, gzs715@gmail.com, charles.a.kamhoua.civ@mail.mil, laurent.njilla@us.af.mil, and nyang@inha.ac.kr. *An earlier version of this work has appeared in proceedings of IEEE ICDCS 2017 [36]. Approved for public release: distribution unlimited 88ABW-2017-2413, dated 17 May 2017.*

benefits. We perform extensive analyses on 16 configurations of the Berkeley Internet Name Domain (BIND) and Unbound, two popular recursive resolvers, that run on various operating systems. We find the rules of referring to DLV servers for validation are lax, resulting in DNS query leakages. The privacy leakage is highlighted because a third party (DLV server) can observe most queries a user has sent, while providing little validation utility.

Contributions. Our contributions are as follows. 1) We formulate the privacy leakage in DNS caused by unintentional queries. We analyze DNSSEC and DLV in light of this privacy risk in various settings. We anticipate that such partial deployment of DNSSEC and lax rules of DLV would amplify such risks. 2) We validate the privacy risks in various settings (resolvers, operating systems, installation tools, etc.) and use a large number of domains. We find that the amount of unintentional leaked queries is an order of magnitude larger than the number it is supposed to be. We discuss root causes of leakage and provide fixes. To the best of our knowledge, this is the first systematic treatment of privacy leakage in DLV.

Organization. We outline background and preliminaries in section 2, and the threat model in section 3. We discuss experimental setups, datasets, and configurations in section 4. We review the main results in section 5 and provide explanations of root causes and remedies in section 6. The discussion is in section 7, the related work in section 8, and concluding remarks in 9.

2 Background and Preliminaries

In this section we review preliminaries required for understanding the rest of this work. In particular, we provide an overview of DNS in section 2.1, DNSSEC in section 2.2, DLV in section 2.3, and BIND and Unbound in section 2.4. We motivate for DNS privacy in section 2.5.

2.1 An Overview of DNS

DNS is one of the major pillars for Internet operation. It provides translations between names of resources and their IP addresses [35], [34]. DNS is implemented as a hierarchical tree-like structure representing the DNS namespace, where each node stores DNS records of naming information.

Thirteen *root servers* managing the DNS records for TLDs are at the top of the tree, such as `com`, `net`, `edu`, *etc*. Similarly, TLDs manage DNS records for the second-level domains (SLDs), e.g., `com` stores and updates DNS records for `google`, `amazon`, `facebook`, `ibm`, *etc*. DNS namespace is managed in a distributed way by authority delegation. The owner of a subtree in the domain namespace can delegate authority of that subtree. A distinct and contiguous portion of the namespace under the authority of a single manager forms a zone of DNS. In a zone, various types of records could be stored, including 1) the start of authority record (SOA), 2) time-to-live (TTL), 3) A and AAAA records for presenting IPv4 and IPv6 addresses, 4) SMTP mail exchangers (MX), 5) name servers (NS), 6) pointers for reverse DNS lookups (PTR), and 7) domain name aliases (CNAME), among others.

The DNS has multiple entities: authoritative, recursive, and stub resolvers. An authoritative server stores records associated with domain names and serves them to stubs: `ns1.google.com` is an authoritative server providing naming service for the SLD `google`. The response from an authoritative server is either the corresponding DNS records of the queried domain, or a referral

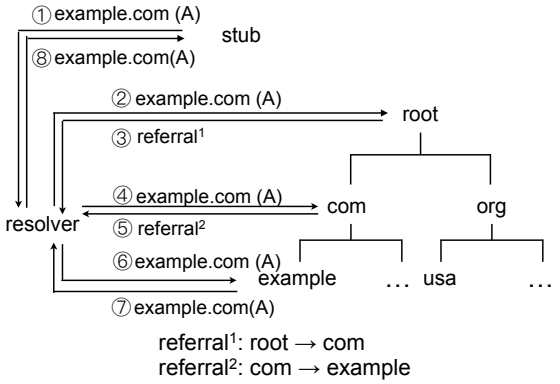


Fig. 1: A workflow of the DNS operation.

to authoritative servers. Recursive resolvers, also acting as DNS servers, resolve a given name by querying authoritative servers recursively, starting from the root to the target servers by using referrals. Caching is usually implemented in resolvers to reduce response time to clients (stubs). The frequency of fetching and time for caching are controlled by the aforementioned TTL in the SOA record. DNS clients (stubs), at the end systems, are commonly configured with a set of addresses of recursive resolvers using DHCP or manually; such a recursive resolver can be a server operated by the Internet Service Provider (ISP), e.g., Time Warner Cable, or public servers such as `8.8.8.8` operated by Google. Figure 1 shows the recursive nature of DNS. When a user types `example.com` in a browser, the stub sends a DNS query to the recursive. The recursive resolver then iteratively queries authoritative servers for the corresponding DNS records until getting the answer or an error (e.g., NXDOMAIN).

Queries can be issued by DNS clients in two ways, recursively or iteratively, to acquire DNS records. By issuing a recursive query, a DNS client requires the DNS server to respond either the corresponding DNS records or an error message claiming the non-existence of the records. After receiving a recursive query, the DNS server will either respond with the corresponding records, or query other DNS servers until it gets the answer or an error message. In response to an iterative query, the DNS server will either use the corresponding DNS records if it has or a referral which is a pointer to another server which may contain the record at the lower level of the domain namespace. The client can then iteratively query the lower level server followed by referrals until it locates the authoritative server for the queried name, or until an error or time-out condition is met. Stubs usually issue recursive queries to recursive resolvers while recursive resolvers issue iterative queries to authoritative servers to respond to the recursive query from stubs.

2.2 DNS Security Extension

DNS was not built with security in mind originally, so DNSSEC is designed to provide authentication capabilities for DNS records [4]. DNSSEC addresses security issues such as DNS hijacking, poisoning, and spoofing, all using cryptographic signing techniques. All records in DNSSEC protected zones are digitally signed. The signature is provided in the RRSIG record. A security-aware resolver is able to validate the signature by using the signing public key stored in the DNSKEY record. The public key used for validation is also validated to preclude zone poisoning: the

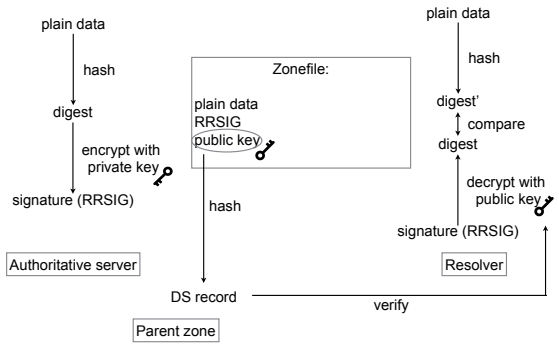


Fig. 2: The procedure of the DNSSEC records validation, highlighting the process of record creation as a digital signature using the private key and the validation, using the public key counterpart.

signing key's signature is stored in a Delegation Signer (DS) record, which is a hash value of the public key in the parent zone. To validate the signing public key for a zone, a chain of trust is needed, starting from the root that is used as a trust anchor to the zone's parent that delegates the authority of signing. Figure 2 shows the workflow of the generation and validation of DNSSEC-related records. As shown in the figure, a DS record created by the private key of the zone owner for the hash of the plain data (i.e., zone contents) is deposited in the zone file, along with a digest calculated using standard hash functions (e.g., SHA-1). At the resolver side, and upon receiving the plain data, a digest is calculated, and compared to the received digest of the plain data. If valid, the resolver proceeds to "decrypt" the signature with the public key of the zone. If the result is valid (the received hash is equal to the "decrypted" hash of the contents), the resolver admits the returned plain data; otherwise it rejects the data. Note that we abuse of the notion, to simplify the explanation: the resolver actually verifies the signature using the public key of the zone, and no decryption is performed.

DNSSEC uses two keys: a zone signing key (ZSK), used to sign DNS records, and a key signing key (KSK), used to sign the ZSK. When a new KSK is created, the DS record must be transferred to the parent zone and published in it. Furthermore, DNS has three DNSSEC-specific flags (bits); DO, AD, and CD [13]. DO (DNSSEC OK) is included in queries to indicate to DNS servers that the resolver is capable of DNSSEC validation. AD (Authenticated Data) in the response indicates the result of validation. CD (Checking Disable) is used in the query to guide the resolver on whether to validate or not.

DNSSEC validation may have one of four status types: `secure`, `insecure`, `bogus`, and `indeterminate`. The recursive resolver will either return the response for the first two status types or return `SERVFAIL` for the last two types to the stub [4]. A `secure` status is given when the resolver can build a chain of signed DNSKEY and DS records from a trust anchor to the authority zone. The `insecure` status means the resolver knows there is no chain of signed DNSKEY and DS records from any trusted anchor pointing to the authority zone, occurring with island of security (section 2.3). A `bogus` status is given when the resolver believes it ought to be able to establish a chain of trust but for some reason it is unable to do so, either due to signature validation error or missing records. The status `indeterminate` indicates that the resolver cannot determine whether the records should be signed, because the resolver is not able to obtain the necessary DNSSEC records.

To highlight the DNSSEC operation, take for example the domain name `example.com`. A security-aware resolver receives the query and sets DO in the DNS query. Upon receiving a response, it begins the validation by verifying the signing public key used in `example` and checks DS records from `root` to `com`, and `com` to `example`. Finally, it verifies the signature in RRSIG record from `example`, with the validated public signing key of `example`. If the DO bit is set in the initial query from a stub, AD will be set by the resolver, indicating the validation result.

2.3 DNSSEC Look-aside Validation

The deployment of DNSSEC to root was completed in July 2010. As of February 2016, more than 85% of delegated TLDs are signed in the root [24]. Nevertheless, the number of secured SLDs that are both signed and have DS registered in the parent zones is quite small compared to the total number of SLDs [47], [38], [46], [15] (see section 6.1).

DNSSEC adds authentication on records in every zone. The partial deployment of DNSSEC results in "islands of security", where all nodes of a *subtree* in the domain namespace implement DNSSEC [31]. For example, having records in `example`, `com`, and `root` signed, while having no DS record in `com` to validate the signing public key used in `example` would result in an island of security: records in `example` cannot be validated, because the signing key is not trusted by the resolver. DLV addresses this problem [49]. With DLV, an owner of a zone can submit the signing public keys as DS records to a DLV registry, delegating a trust anchor. The DLV record is used when the normal operation of DNSSEC fails.

When DNSSEC fails for `example.com`, a security-aware resolver would generate a DLV query by appending the DLV domain after the queried domain. An example of a DLV is run by Internet Systems Consortium (ISC) and the DLV domain is `dlv.isc.org`. The type bit is set to `DLV` as 32769 in the DNS query. The DLV server, `dlv.isc.org`, searches its depository for the corresponding DS records: if there is no deposited DS record, the validator removes the leading label from the query and tries again [49]. This process is repeated until a DLV record is found or it is determined that there are no (enclosing) records applicable to the query in the repository. The enclosing record is helpful for queries containing multiple levels of domains, such as `bbs.sub1.example.com`. If there is no corresponding DS record `No such name` will be returned. If the DLV records are deposited for the queried domain, the resolver can expect the zone to be securely signed and `No error` is returned. Figure 3 shows the workflow of DNSSEC and DLV when `example.com` is queried. While ISC-run DLV server is one of the most popular services online, it is not the only one. For example, other popular public DLV servers include e.g., dlv.secspider.cs.ucla.edu, dlv.trusted-keys.de, dlv.cert.ru, and dlv-test.flame.org.

Aggressive negative caching. For efficiency, aggressive negative caching is implemented at the resolver, where failed queries are also cached [3]. The aggressive negative caching implemented in the validator checks whether any cached and validated NSEC record provides a denial of existence proof for records. In this way, a queried nonexistent domain, which is cached or is proved to be nonexistent by NSEC records, will not be sent to the DLV server for validation [49]. Negative caching, in general, is useful because it reduces the response time for negative answers and limits queries sent to name servers.

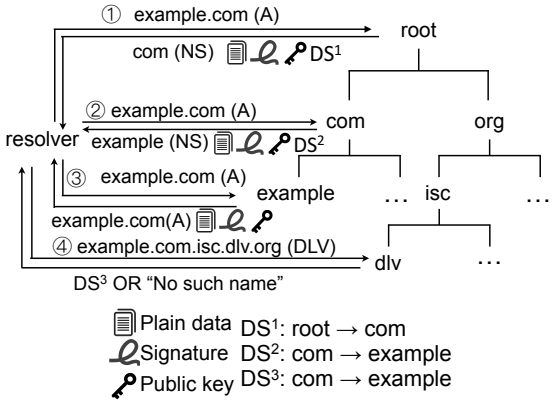


Fig. 3: A workflow of DNSSEC and DLV, which borrows the logic of DNSSEC, although enabling off-the-path validation.

2.4 BIND and Unbound

BIND and Unbound are two popular DNS resolvers that support DNSSEC validation. We examine both of them.

BIND. To support DNSSEC and DLV, three options are configured in BIND [26]. 1) `dnssec-enable`: DNSSEC is supported when this option is `yes` (default). 2) `dnssec-validation`: together with `dnssec-enable`, this option enables DNSSEC validation in BIND when set to `yes` or `auto`. If set to `yes`, a trust anchor must be manually configured for validation. If set to `auto`, a default trust anchor is used. If set to `no`, DNSSEC validation is disabled. The default is `yes`. 3) `dnssec-lookaside`: this option enables DLV when set to `auto`. The built-in DLV trust anchor is used. If set to `no`, the DLV is not enabled and is not used. We note that the default is `no`.

Unbound. Unlike BIND where DNSSEC and DLV are enabled by settings, Unbound takes an implicit configuration: there is no explicit settings enabling DNSSEC and DLV. Furthermore, validation is enabled by including trust anchors in the configuration file. Once the trust anchors are configured, Unbound will automatically do the validation.

2.5 Privacy in DNSSEC and DLV

We note that most privacy concerns associated with DNS discussed in the related work (section 8.1) are also applicable to DNSSEC. Furthermore, given that DNSSEC involves a third party, certain extra privacy risks may arise. To the best of our knowledge, this risk was not previously studied at any level and deserves a treatment of measurement and analysis.

Hypothesis. DLV is used as an alternative off-path validation method. We hypothesize that there is a privacy risk of relying on DLV. We examine this hypothesis in the rest of this paper by measuring DLV in two most adopted DNS resolvers with different installation settings and configurations, running on various operating systems.

3 Threat Model

We now define our threat model, excluding cases that are out of its scope. For any given query, any entity in DNS is considered either an involved or an uninvolved party.

First, we exclude all directly involved name servers, namely authoritative name servers that are involved in the given DNS queries. Such exclusion relaxes the threat model, given the q-name minimization [10], [11], [9], which makes some of the authoritative servers part of the threat model. Second, we expect a DLV server, as a secondary validation server, to *be queried only for domain names that have records deposited in such server*. Moreover, we exclude the *intentional leakage* of domain names to DLV servers for domain names that are verifiable through it. Moreover, we consider such leakage to be no worse than the leakage of DNS queries to root or TLD name servers (or `com` name servers such as for the case of a query to the SLD of `example.com`) as highlighted above. At the zone level (e.g., SLD), we expect that query may result in an NXD response in the conventional way.

Defining Leakage. Informally, we state that a DNS query is leaked if an uninvolved party can *observe* that query during the DNS resolution [37]. This party observes DNS queries without the corresponding record configured by the registrar or the consent of the user. As such, a DLV server is treated as an uninvolved party when the queried domains do not have a DLV record deposited in the DLV server.

Under this model, we expect a DNS query not to be sent to an uninvolved party; e.g., it is unexpected that `nxdomain.com` be sent to `.net` if `com` responds with an NXDOMAIN response. Conversely, if an authority server *A* signals that the corresponding record of interest is deposited in another server *B* through a referral, the resolver should query *B*. The actual implementations of DNS resolution comply with this rule. However, whether DLV does that or not is untested.

DLV's main design goal is to serve off-path validation only if DNSSEC fails. However, it is unclear if there is any additional signaling for whether the corresponding DLV record is deposited in the DLV server. To this end, we define two cases of leakage with DLV:

- **Case-1** happens when a domain has a DLV record deposited and the recursive resolver queries the DLV server for validation. As such, the DLV server may know what domain is queried, and may also serve them.
- **Case-2** happens when a domain does not have DLV record, while the recursive resolver still queries the DLV server, thus allowing the DLV server to know about the queried domains without providing any validation benefit.

We notice that only the second case qualifies as a privacy issue in our model, since the first case of leakage is no worse than today's primary DNS resolution.

Adversary Formulation. We formulate the adversary as an entity in the DNS infrastructure that is unintentionally involved during the DNS resolution while able to learn information from DNS queries. The DLV server was run by ISC, which can utilize the DLV queries to learn queried domains is therefore under our focus, and is an example. Further use of observed DNS queries taken by the adversary is out of the scope of this work, although it is discussed briefly in section 8.1.

Adversary "Maliciousness". We note that our study and analysis do not require that the DLV server be a malicious adversary (e.g., placing itself on the path of the resolution of domains that are not supposed to use it). Consistent with the original design of DLV, we define the scope of observations that are allowed to the DLV server, as a secondary off-path validation mechanism, to only be domain names that are supposedly deposited in the DLV server and their associated queries. As such, the DLV server should not

observe any traffic for domain names that: 1) do not have DNSSEC enabled in the first place, or 2) domain names that have DNSSEC enabled, but their DS records are not deposited in the DLV server. We notice that is not the case, due to the flaws in the configurations of BIND?s DLV options. The privacy of queries is breached by observing such traffic.

4 Settings and Configurations

In order to measure the privacy leakage due to DLV, we use various configurations, settings (default and manual), and queried domains. We then perform an extensive study under various settings to understand privacy issues with DLV. In the following, we introduce the experiment setting, the dataset of sample domains, and the configurations of BIND and Unbound. We use "resolver" in the rest of this paper to refer to both BIND and Unbound.

4.1 Experiment Setting

We configure BIND and Unbound as recursive resolvers with DNSSEC and DLV enabled. Sample domains are queried from *16 different hosts*. We capture packets of DNS queries and analyze them to profile the involved parties, as shown in Figure 3. For each queried domain, our goal is to analyze: 1) whether the DNSSEC query was successful, 2) the actual rule used to refer to the DLV server, and 3) whether the DLV server can provide validation utility. We use the outcomes of this analysis to understand the privacy risks of DLV.

Network Diversity. We mitigate network locality using a variety of query-initiation points; local (on campus) and virtual private servers (DigitalOcean and Amazon EC2) are used.

Resolvers, Operating Systems, and Versions. For representative findings, we rely on multiple settings and operation contexts. For that, we install BIND and Unbound on CentOS (6.7 and 7.1), Debian (7 and 8), Fedora (21 and 22), and Ubuntu (12.04 and 14.04) with both manual installation (by downloading the latest version of source code) and package installer (`apt-get` or `yum` in the corresponding operating system). We notice that the default configurations of BIND vary depending on the installation methods and operating systems: when using the package installer `apt-get` to install BIND, we obtain certain settings that are different from those provided when using (`yum`). In total, we had 16 different environments (as shown in Table 1). We used the two installation methods to install resolvers in each stub.

TABLE 1: Resolver Versions and Settings Used for the Experiments and Analyses in this Study

Operating System	BIND		Unbound	
	P	M	P	M
CentOS 6.7	9.9.4	9.10.3	1.4.20	1.5.7
CentOS 7.1	9.9.4	9.10.3	1.4.29	1.5.7
Debian 7	9.8.4	9.10.3	1.4.17	1.5.7
Debian 8	9.9.5	9.10.3	1.4.22	1.5.7
Fedora 21	9.9.6	9.10.3	1.5.7	1.5.7
Fedora 22	9.10.2	9.10.3	1.5.7	1.5.7
Ubuntu 12.04	9.9.5	9.10.3	1.4.16	1.5.7
Ubuntu 14.04	9.9.5	9.10.3	1.4.22	1.5.7

P: package installer M: manual

4.2 Dataset

To understand DLV's privacy risk, we use two datasets: Alexa's top 1 million domain names [1], and a list of 45 DNSSEC secured domains by Huque [23]. We use the latter dataset to: 1) test with confidence whether DLV servers are involved unintentionally, and 2) understand how various configurations affect the behavior of the resolver.

```
./named.conf.options
options{
        ...
        dnssec-validation auto;
};
```

Fig. 4: Default configuration by apt-get.

```
./named.conf.options
options{
        ...
        dnssec-enable yes;
        dnssec-validation yes;
        dnssec-lookaside auto;
};
include "/etc/bind.keys";
```

Fig. 5: Default configuration by yum.

4.3 BIND Configuration

In the following section we review the BIND configuration in the various settings described above.

• **apt-get intallation.** Debian-based systems including Ubuntu use `apt-get` as a package installer. Figure 4 shows the default configuration when using `apt-get` for installing BIND. We note that this default configuration does not comply with the manual of BIND [26], which suggests the default of `dnssec-validation` to be `yes`. However, with `apt-get` installation, the DLV needs to be manually enabled and the corresponding trust anchors need to be included in the configuration file. The trust anchor for DNSSEC is generated with the installation, although it is not included by default.

• **yum intallation.** Fedora-based systems, e.g., CentOS, use `yum` as an installer. The default configuration by `yum` installation is shown in Figure 5. We notice that BIND is already configured to support DLV using the trust anchors in the configuration file. This contradicts the manual of BIND, which suggests that DLV is not enabled by default.

• **Manual installation.** When BIND is installed manually, there is no configuration file and users need to create one and enable DLV. With manual installation, DNSSEC is enabled by default, although the trust anchor needs to be included. A correct configuration for DNSSEC and DLV is shown in Figure 6.

• **Comparison.** A comparison is shown in Table 2. Columns 2 to 4 stand for `dnssec-enable`, `dnssec-validation`, and `dnssec-lookaside` in the figures above. The last column indicates whether the required trust anchor is included or not. The rows indicate methods of installation, where N/A means the corresponding attribute is not configured by default. The values in red do not comply with the administrator manual of BIND whereas the values in blue means that a manual configuration is required after installation.

• **Summary of expected behavior.** BIND loads the trust anchor automatically when `dnssec-validation` is set to `auto`, whereas the trust anchor needs to be manually included when

```
./named.conf.options
options{
        dnssec-lookaside auto;
};
include "/etc/bind.keys";
```

Fig. 6: Manual configuration.

```
./unbound.conf
...
auto-trust-anchor-file:
    "/usr/local/etc/unbound/root.key"
dlv-anchor-file: "dlv.isc.org.key"
```

Fig. 7: Unbound configuration.

`dnssec-validation` is set to `yes`, although the trust anchor is the same, which is generated after BIND installation. We summarize the default configuration variations of BIND under different installation methods as follows. 1) When installed using `apt-get`, the user only needs to enable DLV, and the default does not comply with the administrator manual of BIND. 2) When installed using `yum`, the default does not comply with the administrator manual of BIND. 3) When installed manually, the user needs to include the trust anchor and enable DLV. For completeness, we considered Debian/Ubuntu where a user changed to `yes` the `dnssec-validation` after reading the administrator manual. In this case, the DNSSEC is enabled whereas the trust anchor for DNSSEC validation is missing.

4.4 Unbound Configuration

When using a package installer, the DNSSEC is enabled by default. However, DLV must be enabled by including the corresponding trust anchor. On the other hand, when installed manually the default configuration file contains commented statements to enable DNSSEC and DLV. The user needs to uncomment the corresponding statements for DNSSEC and DLV to be enabled. A correct configuration is shown as Figure 7.

The configurations in Unbound are not as explicit as in BIND because there are no intuitive attributes or naming conventions. However, Unbound addresses a special privacy risk that is associated with BIND: the implementations of DNSSEC and DLV are simply enabled by configuring the corresponding files containing the trust anchors. In this way, unintentional leakage due to misconfiguration will not happen.

5 Measurement Results

First, we measure the leakage of popular domain names (section 5.1), DNSSEC-secured domains (section 5.2); and the validation utility of DLV (section 5.3). We first quantify the DLV queries for both the top 1 million domains and the 45 DNSSEC-secured domains, and then we quantify the unintentional ones by inspecting the payload of the query and response of a DNS resolution (case 2 in section 3). The number of DLV queries where a domain name is not associated with DLV records highlights the privacy leaked. We note that, unless otherwise specified, the measurements, results, and findings are the same for both resolver software packages, BIND and Unbound.

TABLE 2: Configuration Variations

	DNSSEC	validation	DLV	trust anchor
`apt-get`	Yes	Auto	N/A	N/A
`yum`	Yes	Yes	Auto	Yes
`manual`	N/A	N/A	N/A	N/A

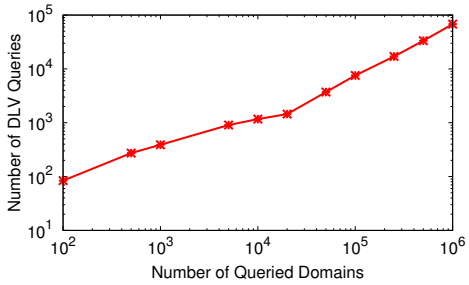


Fig. 8: Number of DLV queries. Notice that the number of leaked domains increases steadily, with the number of leaked domains less than the number of actual queries due to the negative caching effect in place.

5.1 DLV Leakage for Popular Domains

The Alexa's top 100 and 1,000,000 domains are used for testing popular domains. All DLV queries are extracted from the network traffic by filtering the query type. The number of leaked domains, is counted and the proportion of them over the total queried domains is calculated. Results show a large number of DLV queries, e.g., 84% domains are sent to the DLV server when Alexa's top 100 domain names are queried. This corresponds to a representative case in real life, where the number of domains a user is likely to query may not exceed 100. Figure 8 and Figure 9 show the DLV queries when different numbers of sample domains are resolved, accounting for DNS behavior (e.g., aggressive negative caching). In Figure 8, and as expected, the number of DLV queries increases with the number of queried domains: 84 domains are sent to the DLV server when the size of sample domains is 100.

On the other hand, 67,838 domains are queried for DLV when the top million domains are used (a smaller percentage). We notice an decreasing trend on the proportion of the leaked domains as the sample increases. After analyzing the design and implementation of DLV, we found the reason to be the use of aggressive negative caching (section 2.3). The validator collects more NSEC records by sending DLV queries. As such, by utilizing the knowledge of NSEC records, the resolver will not query a domain for which it has a proof of non-existence.

Order Matters. We conduct a measurement in which the same set of queried domain names is shuffled. Shuffling the domain names would result in different outcomes; for the top 100 domains used in the previous measurements we obtain 82%, 84%, and 77%, for three trials. The different results highlight the effect of the aggressive negative caching. If there are two domains that can be proved to be non-existent by the same NSEC record, only the first domain will be queried with DLV.

5.2 DLV Leakage for DNSSEC Domains

To examine the compliance of DLV with the standard operation of DNSSEC, we performed the following. First, 45 DNSSEC-secured domains were queried to determine whether their queries would result in additional DLV queries. When DNSSEC and

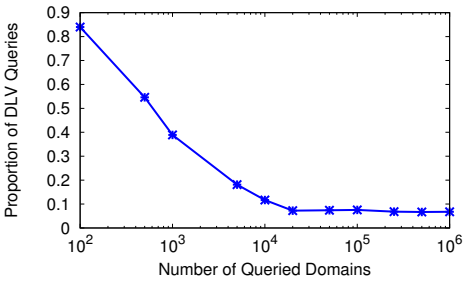


Fig. 9: Proportion of leaked domains. Notice the linear decay (x-axis in log-scale) in the proportion of DLV queries as the number of domains increases, which is understood by the uniformity of the negative caching effect.

TABLE 3: Results Summary of Secured Domains Configuration (*apt-get*† means a user will change the `dnssec-validation` to `yes` in accordance with the manual configuration of BIND.)

	apt-get	apt-get†	yum	manual
DLV	No	Yes	No	Yes

DLV were correctly configured with trust anchor included in the configuration, five of the DNSSEC-secured domains were sent to the DLV server. We found that at the time of the experiment all of those five domain names could not be validated through the on-path DNSSEC validation mechanism: we found that there was no `DS` record for these five domains in their parent zone, thus making them islands of security. The four others were still signed but did not have DS in their parent zones.

DNSSEC-Secured Domains Leaked to DLV. We set `dnssec-validation` to `yes`, while the trust anchor is not included, corresponding to the case where BIND is installed using `apt-get` and `dnssec-validation` is modified to `yes` according to the manual of BIND, or where BIND is installed manually where the trust anchor is not included.

In such cases, the DNSSEC-secured domains cannot be validated since the trust anchor is not included, thus making it impossible to complete the chain of trust. The DNSSEC secured domains are then sent to the DLV server for validation. Table 3 shows whether the secured domains will be queried for DLV when the trust anchor is not manually included. Each column stands for the default configuration of BIND.

• **Unbound.** Since Unbound utilizes a different configuration style where the DNSSEC and DLV are enabled by including the trust anchors, domains do not leak with Unbound.

Practical Implications. We notice that BIND installed manually or by `apt-get` does not contain the statement for including the trust anchor, and only BIND installed by `yum` contains it by default. A user without the practical expertise or careful understanding of the operation of DNSSEC validation, including the knowledge of the used cryptographic schemes and required configuration is unlikely to correctly make the configuration that would not result in leakage.

To understand whether the practical implications highlighted above are real, we performed the following survey. During a DNS-OARC 2015 Workshop, a gathering of DNS operators, administrators, and researchers, we surveyed attendees who use their own recursive for the prevalence of the problem. Of the 56 responses we obtained, 17 respondents (30.35%) indicated that they use defaults with package installer (apt-get or yum), 5 (8.9%) indicated that they use default settings with manual installation, and the rest (60.7%) indicated that they use their own configuration. Of the 56 respondents, 35 (62.5%) indicated that they use ISC's DLV server, while the rest (37.5%) indicated that they use other trust anchors.

5.3 Validation Utility by DLV

The validation utility provided by DLV is measured by inspecting DLV responses, where "No such name" indicates the non-existence of DLV records, and thus the DLV provides no validation utility to the queried domains despite being exposed to the DLV server. According to our threat model, such unintended DLV queries leak privacy of users. DLV responses containing "No error" mean the queried domain is validated by DLV records deposited in the DLV server. By further inspecting the responses (at the packet-level), we conclude that those are the only two types of messages returned by the DLV server. By running this experiment for Alexa's top 10,000 domain names, we found that less than 1.2% of DLV queries received "No error" (1168 domains). As a result, approximately 98.8% of DLV queries are the result of leakage.

6 Root Causes and Remedies

After analyzing the design and implementation of DNSSEC and DLV, we provide possible fixes to the privacy leakage caused by the unintentional DLV queries resulting from the lax rules of DLV design. We note that those fixes are lightweight and require few changes to the existing DNS infrastructure.

6.1 Root Causes of Privacy Leakage

6.1.1 DNSSEC Deployment

DNSSEC is partially deployed despite many years since its creation. Only 0.88% of zones are signed as of November 2015 [38], and the proportion of DNSSEC secured domains (SLD-level) is quite low: 0.43% for `com`, 0.61% for `net`, and 0.89% for `edu` based on a survey we conducted in November 2015. While not particularly a root cause of the leakage of domains to DLV, the current level of deployment of DNSSEC actually contributes to the utility of DLV highlighted in section 5.3. We anticipate such utility to be higher if DNSSEC is more widely deployed.

6.1.2 When to Use DLV

Although intended for the use as an off-path validation mechanism when the main DNSSEC validation fails for one reason or another, the rules used to determine when a DLV server is queried are lax. The configurations associated with various distribution are ambiguous, inconsistent, or poorly documented, depending on the distribution itself and the mechanism being used for the resolver installation. Such lax conditions and rules lack any signaling of when DLV should be used. In general, we believe that not every domain name issued by a stub resolver should be sent to a DLV server, even when the DLV and DNSSEC options are enabled at the recursive resolver, especially because many domains are not DNSSEC-enabled in the first place.

6.2 Possible Remedies of Privacy Leakage

In this section, we propose two possible remedies to leakage, namely DLV-aware DNS and privacy-preserving DLV, which would require minimal changes to the resolver.

6.2.1 DLV-Aware DNS

The key idea is to inform the resolver to only issue DLV queries for those domains that have deposited their DLV records in the DLV server. We assume that the recursive resolver is trusted, and would comply with such signaling on behalf of the stub resolver. We suggest two possible methods to achieve the goal.

Using TXT record. In this method, we add a description (e.g., `DLV=1`) in the DNS TXT record, indicating the existence of DLV records, where the resolver has to query for them in case the main validation method fails. The resolver will be informed by querying and checking the TXT record.

Using Z bit. Another similar way is by setting the spare "Z" bit [14] in the DNS response header to signal the existence of DLV records. Note that using the "Z" bit requires IANA allocating the bit for a special use, although it can easily fit in the current DNS implementation.

6.2.2 Privacy-Preserving DLV

The second remedy involves changing the data format provided for both DLV registration and query. On DLV record registration, instead of depositing `(domain_name, DNSKEY)`, we compute `digest = crypto_hash(domain_name)` and deposit `(domain_name, DNSKEY, digest)` to the DLV server. On DLV query, the resolver only sends the computed hash instead of the `domain_name` to the DLV server. For example, in step 4 of Figure 3, assuming `$hash = crypto_hash("example.com")`, we send `($hash).isc.dlv.org` to the DLV server. At DLV server, it compares `$hash` with the stored `digest`. If they match, it is in the same situation as the Case 1 (section 3), thus leading to no additional leakage. If they do not match, the DLV server is not able to obtain the domain name from the `$hash` (except if it computes exhaustively for all the digests of domain names that are not on its DLV server, which we consider impractical), thus no privacy leakage is compared to Case 2 (section 3).

6.2.3 Analysis of the DLV-Aware DNS

In the following, we analyze the DLV-aware DNS resolution above in two ways: against attacks and in terms of overhead.

Attacks. While not requiring modifications to DNS, the proposed fixes using the DLV-aware DNS are vulnerable to zone poisoning and man-in-the-middle. An attacker can also mislead the recursive resolver by modifying the TXT record, or by flipping the "Z" bit. A potential remedy to such attack is to sign the response, where the recursive resolver validates the response to check whether to contact the DLV server.

Overhead Measurement. The effectiveness of our remedies is not determined by preventing forwarding queries to the DLV server through signaling only, but also by the resulting overhead. For the overhead evaluation, we consider each of TXT and Z bit.

Evaluation Metrics. We measure the overhead when using `TXT` queries with three evaluation metrics: the *response time* (in seconds), *traffic volume* (in MB) and the *number of issued queries*.

Datasets. The overhead is measured using four datasets to account for the different caching behaviors. In particular, we use 100, 1K, 10K, and 100K domains, respectively. Note that we take the record `TXT` as a "spare" record, where a domain registrant configures the information indicating the existence of the corresponding DLV records. Alternatively, the domain registrant can configure such information in other unused DNS records. For signaling, we set the TXT record to include either `dlv=1` or `dlv=0`.

TABLE 4: Number of Different Types of DNS Queries

# Domains	A	AAAA	DNSKEY	DS	NS	PTR
100	467	243	32	221	36	2
1k	4032	1881	96	1963	285	13
10k	30972	10566	390	18582	2701	43
100k	283949	66498	3264	203683	33402	331

Results Using TXT Record. In each of the datasets highlighted above, we insert the `TXT` queries after each original query to measure the overhead. We compare the overhead with DLV alone. Statistics are extracted under the original traffic without `TXT` queries and responses, the actual overhead, and the total. We provide both numerical and visual representation of the overhead (in figures) to highlight the relative order compared to the DLV operation without remedies. We visualize the overhead as the number of queries increases.

Six types of DNS queries are issued besides the `DLV` and `TXT`, namely `A`, `AAAA`, `DS`, `DNSKEY`, `NS`, and `PTR`. The number of issued queries of each type is given in Table 4. The overhead of the remedies, using the measures highlighted above, is shown in Table 5. When 100 domains are queried, the `TXT` option as a remedy increases the response time by 18.68%, the traffic volume by 6.67%, and the number of issued queries by 10.79%, which are relatively small. The overhead as a percent, however, grows to 23%–29% (latency), 8.46%–9.97% (traffic volume) and 13.5%–19.66% (issued queries) as shown Table 5 for 1k–100k datasets. While still less than 30%, 10%, and 20% for latency, traffic volume and queries, respectively, the overhead itself grows significantly as the number of queries increases. The increasing overhead is in part due to the caching dynamics as the number of queries grows within a short period of time. Figure 10 shows those results visually.

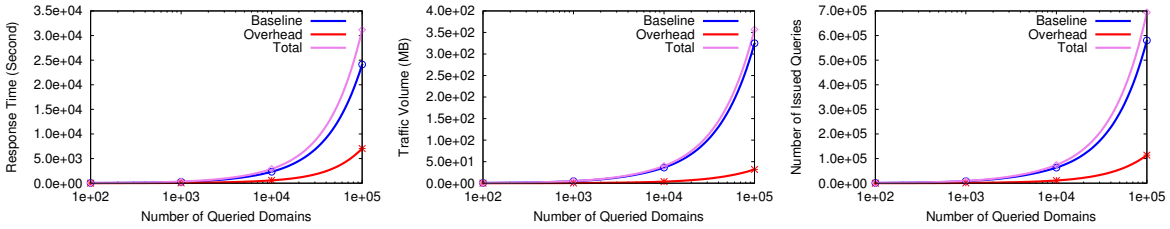
In Figure 10 we observe that the *response time* (or latency; Figure 10a) represents the largest overhead component with respect to the correspond baseline. By a further inspection, we found that the large overhead is because not all domains are configured with the `TXT` record. To this end, we conclude that the actual overhead will be reduced with a wide deployment. Finally, the ratio of the issued queries as a measure of overhead increases from approximately 10% to 20% when 100K domains are queried, which implies that there are more queries on average for the referrals of SLDs.

Results Using Z bit. Similar to the experiment above using the `TXT` for a remedy, we now use the Z bit. Overall, the Z bit option incurs a minimal overhead. For the lack of space, however, we skip the individual results of Z, and highlight the overall relative order of the overhead (as response time, traffic volume, and query number) across three different options: TXT, Z bit, and DLV as shown in Figure 11 (Subfigures a, b and c compare the response time, traffic volume, and number of queries) . We notice that while the TXT option incurs the highest overhead, as highlighted above, the Z bit incurs a very small overhead (minimal) in terms of response time, traffic volume, and queries. This is in particular explained by the fact that enabling the Z bit in a response would not necessitate sending additional packets, queries, etc. as the bit can be masked in the same response as the original response.

Large-scale Experiment. Overhead-wise, we notice that the TXT option provides an upper-bound on the cost of our solutions. Thus, we examine the cost of this approach when evaluated on a large-scale dataset. Since the TXT record with the signaling information to the recursive name server has to be sent back by the authoritative name-server, the cost is amortized (given that

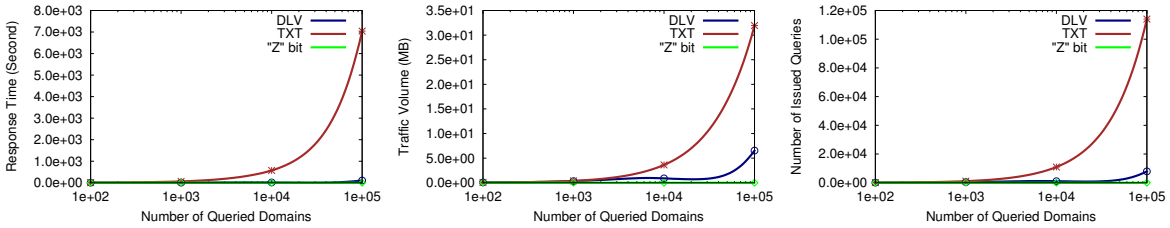
TABLE 5: Overhead of the Original and Increased in Three Metrics

	Response Time (Seconds)			Traffic Volume (MB)			# Issued Queries		
#Domains	Baseline	Overhead	Ratio	Baseline	Overhead	Ratio	Baseline	Overhead	Ratio
100	38.16	7.13	18.68%	0.60	0.04	6.67%	1001	108	10.79%
1K	270.278	63.28	23.41%	4.61	0.39	8.46%	8270	1120	13.54%
10K	2324.45	571.69	24.59%	36.31	3.62	9.97%	63254	10960	17.33%
100K	24119.23	7043.17	29.20%	324.90	31.95	9.83%	580127	114043	19.66%



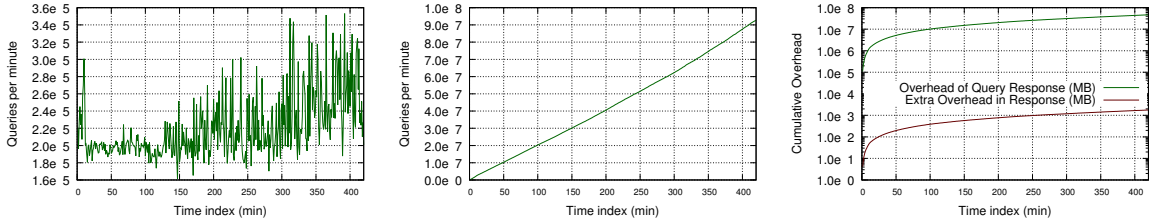
(a) Response Time (b) Traffic Volume (c) Number of Queries

Fig. 10: Baseline, overhead, and total performance in terms of response time (measured in seconds), the total traffic volume (measured in megabytes), and the number of queries. The baseline corresponds to the case when not using the remedies, whereas the overhead is the actual cost of the remedy.



(a) Response Time (b) Traffic Volume (c) Number of Queries

Fig. 11: A comparison between the standard DLV and the remedies when using the TXT and "Z" bit performance, respectively, across the response time, traffic volume, and number of queries.



(a) Query volume over time (b) Cumulative query over time (c) Overhead vs. extra overhead

Fig. 12: An evaluation of the overhead (in MB) when using the TXT option with a large-scale dataset in a trace-driven experiment.

there are a large number of authoritative name servers issuing the response). On the other hand, the cost of receiving the response, parsing it, and taking further actions upon receiving it are aggregated at the recursive resolver. Thus, we consider the cost at the recursive resolver with further investigation. For this evaluation, we use a large-scale recursive traces of DNS resolution from the Day In The Life (DITL) of the Internet data collection effort (https://www.dns-oarc.net/oarc/data/ditl).

Figure 12 shows the main characterization of the dataset and the overhead of running the remedy. As for the size of the workload used in this experiment, we provide two plots: a per-minute volume (over the period of the data collection) and a cumulative total number of queries up to the point of measurement. The first plot, shown in Figure 12a, highlights the total number of queries issued by the recursive over time (per minute). Those are actual observations corresponding to workload of the recursive resolver. In this figure, we notice that the query rate fluctuates over time, and stays always between 160,000 and 360,000 queries per minute (2667–6000 queries per second). As shown, the workload corresponds to a large recursive. Second, Figure 12b shows the cumulative query rate over time, with a total of 92,705,013 queries after 7 hours period of monitoring; for any time index t, the cumulative queries count in Figure 12b is calculated as $\sum_{i=1}^{t} q_i$, where q_i is the number of queries in time index i. This cumulative

query count is more appropriate for evaluating the cumulative overhead due to our approach, which we pursue.

The main results of overhead evaluation of our approach are shown in Figure 12c. The figure shows two quantities: the actual cumulative overhead at the recursive resolver due to answering those queries (baseline), and the additional cumulative overhead due to incorporating the TXT record for signaling whether the recursive resolver needs to contact the DLV server or not. While the cumulative overhead due to employing our approach is non-negligible in the absolute sense (i.e., about 1.2GB over 7 hours, which corresponds to 0.38Mbps additional bandwidth at the recursive for signaling), we notice that such an overhead is quite small compared to the overhead of serving the queries.

In conclusion, this experiment confirms that the overhead is small, relative to the overall in bandwidth required at the recursive to answer this large number of queries. Furthermore, in this experiment we consider signaling for all domains, and not only those with DNSSEC, which are likely to be deposited in the DLV server (constitute a smaller fraction of domains). This, in turn, signifies that the overhead in reality is way smaller as a fraction of the overall bandwidth required at the recursive resolver. In conclusion, this trace-driven experiment highlights the feasibility of our approach.

6.2.4 Analysis of the Privacy-Preserving DLV

Dictionary Attacks. Notice that this solution would provide certain privacy guarantees only if the number of possible domains in general, potentially stored at the DLV server, is large enough that the DLV server cannot precompute all possible hashes of domains to find out which domain is being queried by the user at a point in time. This happens to be the case with respect to the domain names on the Internet today: the total number of domains is over 350 million domains, which are large enough to prevent a dictionary attack.

The careful reader may note that not all domain names are potentially going to use DLV. Rather, domain names that have DS records (DNSSEC-enabled) are the potential domains to use DLV, and the attacker would only need to consider those domains in his dictionary. We note, however, and despite that only 6% of all domains use DNSSEC, the capabilities of DNSSEC are not limited to second-level domains (which are limited), but may also be used for other subdomains. The number of such subdomains, for only 6% of the total number of second-level domains which enable DNSSEC, is exponentially large, making the dictionary attack difficult to launch. Nonetheless, a determined adversary might be able, with some effort, to launch an offline dictionary attach on this solution. To address this problem, however, one may use this approach in combination with the two other approaches above. By launching the dictionary attack, the adversary would be able to know which domain that has deposited its record in the DLV server is being queried, thus limiting the scope of the dictionary attack's implications.

6.3 Addressing Configurations Issues

As mentioned earlier, there are many configuration variations of BIND. Furthermore, various distributions of BIND are inconsistent in producing the default configurations, and some of which are not even compliant with the standard configuration document (stated in the administrator manual of BIND and the RFC documents). Without a correct configuration, even DNSSEC-secured domains will be sent to the DLV servers. An easy but essential fix is needed to create a consistent set of configurations across all software releases in different Linux distributions. Such an idea, in spirit, is followed in Unbound.

In addition to the configuration recommendations, we suggest a careful review of the lax guidelines and rules utilized for when a DLV server is queried. In particular, such rules should outline a transparent process that is clear to the end-points (stub or domain) for decisions a recursive resolver is taking on behalf of them.

7 Discussion

7.1 Experiment Generality

For the generality of this study and findings, our experiments, measurements, and analyses are performed on various popular Linux distributions and different versions of two representative resolvers. Both local machines using campus network and VPS provided by DigitalOcean and Amazon EC2 are used. Results among different platforms remain the same, with negligible differences (stated when they occurred). The dataset standing for popular domains contains the top 1 million domains ranked by Alexa, which is large enough to draw conclusions. In order to cover as many domains as possible, we only use SLDs without any subdomains, e.g., `images.google.com`. We believe that although the percentage of leaked domains may be affected by use of third level domains, not using such domains would not affect the general conclusions concerning the role that configurations play in the leakage and the leakage itself is a phenomenon. Furthermore, we believe that the additional leakage one would realize from third-level domains is perhaps negligible, given that the DLV server uses *enclosing records* for validation: the validator will remove the leading label from the query to find the DLV records until the apex of the domain [49].

Impact of DLV Increased Deployment. We note that the findings in this paper, although alarming, could become less significant if more domain names are populated in the DLV server for off-path validation. Furthermore, the same effect could be achieved if more on-path validation is performed. However, we note that DNSSEC is still not widely deployed, and it would take years before it is widely adopted to prevent cases that necessitated DLV in the first place. Moreover, the way that DLV servers are phased out (e.g., by ISC) signifies the problem: the option in BIND and UNBOUND are not muted in future releases, an empty zone of DLV at ISC will be still used, and the queries directed due to misconfiguration are going to be intercepted by the third-party servers.

Trust Assumptions. In our solutions to address the privacy leakage by the DLV-enabled recursive resolver, we do not require any additional trust that is not already in the DNS resolution ecosystem. In particular, the Z bit is adjusted by the authoritative name server of the domain, which is trusted. An additional trust relationship would actually need to exist between the authoritative name server (of the domain) and the recursive resolver, which performs the look up and the subsequent DLV queries. However, we note that this form of trust is not a new assumption: the same level of trust needs to exist between the recursive resolver and the authoritative name server, particularly with respect to the DNSSEC validation.

7.2 Configuration of the Resolver

As mentioned in section 4.3, when BIND, for example, is installed via *apt-get*, the default configuration file *named.conf* does not

comply with the manual document of BIND. BIND is also not completely configured by default for DNSSEC validation because the trust anchor is not included. As a consequence, DNSSEC validation will always fail and BIND will send DLV queries for all domains, no matter if they are DNSSEC secured or not. *apt-get* is used in Debian- and Ubuntu-based Linux systems, which are both popular Linux distributions. In addition, the same issue occurs when BIND is manually installed without any configuration files. This case is also likely to happen since the user may want to download the latest source code of BIND.

The configuration of BIND is very involved, and a user may not always configure BIND with the trust anchor included properly, since a correct configuration would require a deep understanding of the internals of DNSSEC and DLV. Improvements on the implementation of BIND need to be made. One idea is either including the trust anchor by default or making BIND use the hardcoded trust anchor for DNSSEC validation whenever there is no trust anchor included. In this way, advanced users can change the trust anchor to an anchor of their choice while the ordinary users will not worry about properly including the trust anchor for DNSSEC validation. Note that the domains that are not DNSSEC secured will still be leaked and such domains are taking the majority.

7.3 NSEC, NSEC3, and NSEC5

Performance benefits of DLV are seen in the aggressive negative caching, which utilizes NSEC records. However, NSEC is vulnerable to enumeration attacks. An attacker can gain knowledge of all domains in the zone by sending DNSSEC validation queries of random domains. After a sufficient number of queries, the attacker will potentially know all domains in the DLV zone. Our inspection of the DLV communication demonstrates the use of aggressive negative caching. NSEC3 is proposed to use the hash value of domains instead of the plaintext in NSEC [30]. NSEC5 is introduced to prevent zone enumeration as well, while it does not require keeping private keys as in NSEC3. However, NSEC3 and NSEC5 do not allow aggressive negative caching [49] by design, leading to a tradeoff between performance and privacy introduced by DLV. If DLV is to use NSEC3 or NSEC5 [18], the information leakage would be even greater, since aggressive negative caching is not fully utilized by the resolver. Every query to the resolver would trigger a query to the DLV server.

7.3.1 Other Privacy Mitigations

One may argue that if queries are sent by a public recursive resolver on behalf of multiple stubs, the DLV server will not be able to map the query to the actual querying stub. Although the granularity of leaked privacy is beyond the scope of this paper, we claim that an interested party may utilize other network traffic profiling and correlation techniques to link DNS queries to stubs and even users [45]. Similarly, other arguments for a remedy exist using anonymous communication systems, such as Tor. Similarly, an interested party can utilize de-anonymization techniques to infer privacy information from the DLV queries [8]. Furthermore, such techniques, even when valid, do not change the status quo that DLV provides a risk to privacy for its lax rules of design and implementation.

7.3.2 Recent Developments

DLV servers are required to provide continuous and stable service. There have been several DLV outages causing validation to break down. In a recent development the operator of a major DLV server, ISC, has announced plans to discontinue its DLV server in 2017. In explaining the motive of this termination, it has been announced that DNSSEC is growing to a point where there is no need to run DLV at ISC. Although no other explicit reasons are given, the privacy risk highlighted in this paper might be a crucial reason why such termination (or an improvement of the rules upon which the DLV server is contacted) should have happened long ago. The contribution of this paper would not be degraded due to the planned turndown of this DLV server, since ISC is only one of many used in the wild. By analyzing the privacy leakage of the DLV implementations, we want to guide future design and implementation of privacy-friendly DNS infrastructure.

At the time of writing this paper, the intent of ISC was to phase out DLV, without further details of how this phase out would be (e.g., entire take down of the DLV servers, removal of the delegated zones in DLV, etc.). As of October 2017, the approach used by ISC to phase out DLV is to remove all delegated zones and keep the empty service running (https://www.isc.org/downloads/bind/dlv/). As a result, queries would still be unintentionally forwarded/leaked to the DLV server. In fact, the problem highlighted in the paper has become more severe due to the phasing out approach of the DLV server taken by ISC: all queries sent to the DLV server would belong to case 2, which constitute a privacy leakage

8 Related Work

In this section, we review the related work on DNS privacy and neighboring topics, including the theoretical and experimental literature quantifying privacy risks, prior designs for DNS privacy, measurements, and evaluations of DNSSEC as well as prior work and discussions on DLV.

8.1 Privacy Risks with DNS

With the rise of pervasive surveillance as a threat [6], DNS privacy has attracted some attention recently [51], [37], [19], [41], [29], [12], [50]. A monitor performing pervasive surveillance is able to gather artifacts from which he can breach the privacy of users [16]. DNS traffic is highly valuable for two reasons: 1) it is metadata, thus it is easy to analyze and use, and 2) it often includes explicit information about user's behavior. To highlight this risk, the literature provides various studies in various contexts. Banse *et al.* [5] performed a behavior-based tracking system to analyze DNS query logs for a large user group. They showed that more than 2000 users are correctly linked to 88.2% of all sessions with their DNS query log. Konings *et al.* [28] performed device identification based on DNS traffic: using a one-week network traffic dataset from a university public Wi-Fi network, they showed that 59% of the devices names include personal information and 17.6% of the information contains both first and last name.

Privacy risks of DNS prefetching are explored by Krishnan *et al.* [29]. Shulman [41] performed a meta-study of existing proposals that implemented encryption in DNS requests for privacy. She highlighted that a straightforward application of encryption alone may not suffice to protect the privacy in DNS.

8.2 Designs for DNS Privacy

Zhao *et al.* [50] proposed to ensure DNS privacy by concealing actual queries using noisy traffic. Castillo-Perez *et al.* [12]

evaluated this approach and demonstrated that the privacy ensured by added queries is somewhat difficult to analyze, and that the technique introduces additional latency and overhead, making it less practical. Hermann *et al.* [19] proposed EncDNS, a lightweight privacy-preserving implementation that replaces the conventional third-party resolvers, and provides a client software that forwards queries to it through conventional DNS forwarders. EncDNS provides an end-to-end encryption, thus queries are not exposed to the forwarders.

The IETF has recently established a working group for addressing DNS privacy concerns (called DNS PRIVate Exchange, DPRIVE). The group proposed various techniques that are currently being under consideration [37]. Zhu *et al.* [51], [22] proposed a connection-oriented DNS transport over TCP and used TLS for privacy. The authors argue that the overhead of their approach is modest with careful implementations.

Reddy *et al.* [44] proposed to use the Datagram Transport Layer Security (DTLS) for DNS exchange. They add a protection layer for the sensitive information in DNS queries, which would withstand a passive listener and certain active attackers, and argue that their mechanism reduces the round trip time of DTLS and the handshake phase. Side-channel attacks on encrypted DNS try to exploit the difference in the size of requests and responses of various domain names to infer which domain name is being queried. To address this problem, Mayrhofer [33] proposed a padding scheme, in which severs can pad requests and responses by a variable number of octets.

Systems that greatly change the existing DNS have also been studied. Lu *et al.* [32] proposed a Privacy-Preserving DNS (PPDNS). PPDNS introduces an alternative naming infrastructure using distributed hash tables (DHTs) and uses computational private information retrieval (cPIR) to ensure privacy. PPDNS leverages the DHT index structure to provide privacy on query resolution, while using cPIR to reduce communication overhead for bandwidth-limited clients. To defend against a different threat (domain-name takedown), Scaife *et al.* [40] proposed OnionDNS – a Tor hidden-service based system for domain name registry.

8.3 DNSSEC Measurements and Evaluations

There have been several studies on the measurements, analysis, and evaluation of DNSSEC and concepts. Bau *et al.* [7] formally modeled the cryptographic operations in DNSSEC and discovered a vulnerability that allows an attacker to add a forged name into a signed zone. Herzberg *et al.* [20] presented a comprehensive overview of challenges and potential pitfalls of DNSSEC, including vulnerable configurations, increased vulnerabilities due to incremental deployment, and interoperability challenges in large DNSSEC responses. Goldberg *et al.* [18] demonstrated zone-enumeration vulnerabilities on the NSEC and NSEC3. They showed that the security against attackers tampering with DNS messages and protection against zone enumeration cannot be satisfied simultaneously. They also proposed NSEC5, a provably secure DNSSEC denial of existence mechanism.

8.4 DNSSEC Look-aside Validation

DLV and its operation has been a topic of debate in the IETF community [2], [39]. Discussions have been mainly focused on trustworthiness of the third party running the DLV servers. For example, it is argued that a DLV server should be continuously running in order for the DLV to serve its intended purpose. However, this is not always guaranteed, given several reported outages. Finally, Osterweil [39] argued, although did not measure, that DLV presents a privacy risk, by allowing a third party to observe queries initiated by users. In February 2015 [48], a plan was been announced by ISC to discontinue DLV operations, stating that they would remove all zones in 2017 [25]. However, this announcement is not motivated by privacy consideration, but rather by ISC's belief that DNSSEC adoption has reached a point of maturity where DLV is not needed anymore.

DLV is published by ISC without explicit reasons for its termination [25]. However, one could tell that DLV was originally designed as an alternative for DNSSEC validation and DLV would not serve all the time as more domains were secured by DNSSEC. One point of an ongoing discussion is that phasing out DLV will encourage registrars to support DNSSEC validation for their own domains.

9 Conclusion

In this paper, we analyzed privacy leakage in the DLV. Experiments were performed on two popular DNS resolver software under extensive experimental environments and various settings. Results showed that DLV's rules are lax and resulted in privacy leakage of unintended queries to third parties. We also proposed fixes to the problem and evaluated their scalability. While highlighting privacy risks of this particular protocol, this study also aimed at calling for further efforts to understand the privacy risks in the domain name system.

Acknowledgement

This work was supported by NSF grant CNS-1809000, NRF grant NRF-2016K1A1A2912757, and the Air Force Office of Scientific Research (AFOSR) Summer Faculty Fellowship Program (2016).

References

[1] Alexa, "Does Alexa have a list of its top-ranked websites," Online, http://bit.ly/1J5KVXG, 2016.
[2] M. Andrews, "dnssec with freebsd's resolver," Online.
[3] ——, "Negative caching of DNS queries (DNS NCACHE)," IEFT RFC 2308, 1998.
[4] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS security introduction and requirements," IETF RFC 4033, 2005.
[5] C. Banse, D. Herrmann, and H. Federrath, "Tracking users on the internet with behavioral patterns: Evaluation of its practical feasibility," in *Information Security and Privacy Conference, SEC*, 2012, pp. 235–248.
[6] R. Barnes, B. Schneier, C. Jennings, T. Hardie, B. Trammell, C. Huitema, and D. Borkmann, "Confidentiality in the face of pervasive surveillance: A threat model and problem statement," IETF RFC 7624, 2015.
[7] J. Bau and J. C. Mitchell, "A security evaluation of DNSSEC with NSEC3," in *Network and Distributed System Security Symposium, NDSS*, 2010.
[8] A. Biryukov, I. Pustogarov, and R. Weinmann, "Trawling for tor hidden services: Detection, measurement, deanonymization," in *IEEE Symposium on Security and Privacy, SP*, 2013, pp. 80–94.
[9] S. Bortzmeyer, "Possible solutions to DNS privacy issues," IETF Internet Draft, 2013.
[10] ——, "DNS privacy considerations," RFC 7626, 2015.
[11] ——, "DNS query name minimisation to improve privacy," IETF Draft, 2015.
[12] S. Castillo-Perez and J. Garcia-Alfaro, "Evaluation of two privacy-preserving protocols for the dns," in *International Conference on Information Technology*, 2009, pp. 411–416.
[13] D. Conrad, "Indicating resolver support of DNSSEC," IETF RFC 3225, 2001.
[14] D. Eastlake 3rd, "Domain name system (DNS) IANA Considerations," IETF RFC 5395, 2008.

[15] Educause, "Average Counts of .EDU Domains," Online; http://bit.ly/1JNvcMU, 2015.

[16] S. Farrell and H. Tschofenig, "Pervasive monitoring is an attack," IETF RFC 7258, 2014.

[17] O. Garcia-Morchon, S. Keoh, S. Kumar, R. Hummen, and R. Struik, "Security considerations in the IP-based internet of things," IETF Internet Draft, 2012.

[18] S. Goldberg, M. Naor, D. Papadopoulos, L. Reyzin, S. Vasant, and A. Ziv, "NSEC5: provably preventing DNSSEC zone enumeration," in *Network and Distributed System Security Symposium, NDSS*, 2015.

[19] D. Herrmann, K. Fuchs, J. Lindemann, and H. Federrath, "Encdns: A lightweight privacy-preserving name resolution service," in *European Symposium on Research in Computer Security - ESORICS*, 2014, pp. 37–55.

[20] A. Herzberg and H. Shulman, "DNSSEC: security and availability challenges," in *IEEE Conference on Communications and Network Security, CNS*, 2013, pp. 365–366.

[21] ——, "DNS authentication as a service: preventing amplification attacks," in *Annual Computer Security Applications Conference, ACSAC*, 2014, pp. 356–365.

[22] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman, "DNS over TLS: Initiation and performance considerations," IETF Internet Draft, 2015.

[23] S. Huque, "DNSstat-some DNS zone statistics," Online, 2012.

[24] ICANN, "TLD DNSSEC Report," Online; http://stats.research.icann.org/dns/tld_report/, 05 2016.

[25] Internet Systems Consortium, "Decommissioning the DLV," Online; https://www.isc.org/blogs/dlv/, 2015.

[26] ISC, "BIND Administrator Reference Manual (ARM)," Online; http://www.bind9.net/arm910.pdf.

[27] C. Jackson, A. Barth, A. Bortz, W. Shao, and D. Boneh, "Protecting browsers from DNS rebinding attacks," *TWEB*, vol. 3, no. 1, pp. 2:1–2:26, 2009. [Online]. Available: http://doi.acm.org/10.1145/1462148.1462150

[28] B. Könings, C. Bachmaier, F. Schaub, and M. Weber, "Device names in the wild: Investigating privacy risks of zero configuration networking," in *International Conference on Mobile Data Management*, 2013, pp. 51–56.

[29] S. Krishnan and F. Monrose, "DNS prefetching and itś privacy implications: When good things go bad," in *USENIX Workshop on Large-Scale Exploits and Emergent Threats, LEET*, 2010, pp. 1–10.

[30] B. Laurie, G. Sisson, R. Arends, and D. Blacka, "Dns security (DNSSEC) hashed authenticated denial of existence," IEFT RFC 5155.

[31] E. Lewis, "DNS security extension clarification on zone status," IETF RFC 3090, 2001.

[32] Y. Lu and G. Tsudik, "Towards plugging privacy leaks in the domain name system," in *IEEE Tenth International Conference on Peer-to-Peer Computing, P2P*, 2010, pp. 1–10.

[33] A. Mayrhofer, "The EDNS(0) padding option," IETF Internet Draft, 2015.

[34] P. Mockapetris, "Domain names-implementation and specification," IETF RFC 1035.

[35] P. V. Mockapetris and K. J. Dunlap, "Development of the domain name system," *Computer Communication Review*, vol. 25, no. 1, pp. 112–122, 1995. [Online]. Available: http://doi.acm.org/10.1145/205447.205459

[36] A. Mohaisen, Z. Gu, and K. Ren, "Privacy implications of dnssec look-aside validation," in *International Conference on Distributed Computing Systems, ICDCS*, 2017, pp. 1–6.

[37] A. Mohaisen and A. Mankin, "Evaluation of privacy for DNS private exchange," IETF Internet Draft, 2015.

[38] nTLDStats, "DNSSEC Breakdown," Online; https://ntldstats.com/dnssec, 2015.

[39] E. Osterweil, "Unplanned DLV zone outage on 2009-Apr-06," Online, 2009.

[40] N. Scaife, H. Carter, and P. Traynor, "Oniondns: A seizure-resistant top-level domain," in *IEEE Conference on Communications and Network Security, CNS*, 2015, pp. 379–387.

[41] H. Shulman, "Pretty bad privacy: Pitfalls of DNS encryption," in *ACM Workshop on Privacy in the Electronic Society, WPES*, 2014, pp. 191–200.

[42] H. Shulman and M. Waidner, "Towards forensic analysis of attacks with DNSSEC," in *IEEE Security and Privacy Workshops SPW*, 2014, pp. 69–76.

[43] ——, "Towards security of internet naming infrastructure," in *European Symposium on Research in Computer Security ESORICS*, 2015, pp. 3–22.

[44] D. W. T. Reddy and P. Patil, "DNS over DTLS (DNSoD)," IETF Internet Draft, 2015.

[45] N. V. Verde, G. Ateniese, E. Gabrielli, L. V. Mancini, and A. Spognardi, "No nat'd user left behind: Fingerprinting users behind NAT from netflow records alone," in *International Conference on Distributed Computing Systems, ICDCS*, 2014, pp. 218–227. [Online]. Available: http://dx.doi.org/10.1109/ICDCS.2014.30

[46] VeriSign, "The domain name industry brief," Online; http://bit.ly/1W5qLPu, 2015.

[47] ——, "Domains Secured with DNSSEC," Online; http://scoreboard.verisignlabs.com/, 2015.

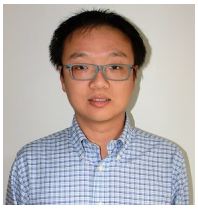
[48] Vicky Risk, "Sunset for the DLV," Online; http://bit.ly/1Pv0FQC, 2015.

[49] S. Weiler, "DNSSEC lookaside validation (DLV)," IETF RFC 5074.

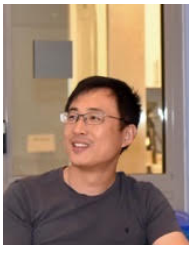
[50] F. Zhao, Y. Hori, and K. Sakurai, "Analysis of privacy disclosure in DNS query," in *International Conference on Multimedia and Ubiquitous Engineering (MUE 2007)*, 2007, pp. 952–957.

[51] L. Zhu, Z. Hu, J. S. Heidemann, D. Wessels, A. Mankin, and N. Somaiya, "Connection-oriented DNS to improve privacy and security," in *IEEE Symposium on Security and Privacy, SP*, 2015, pp. 171–186.

Aziz Mohaisen obtained M.S. and Ph.D. degrees in Computer Science from the University of Minnesota, both in 2012. He is currently an Assistant Professor at the Department of Computer Science and Engineering at the University at Buffalo. Previously, he was a Senior Research Scientist at Verisign Labs and a Member of Engineering Staff at the Electronics and Telecommunication Research Institute, a large research and development institute in South Korea. His research interests are in the areas of networked systems, systems security, data privacy, and measurements. He is a senior member of IEEE and a member of ACM.



Zhongshu Gu received his Ph.D. from Purdue University in 2015 and B.S. from Fudan University in 2007, both in Computer Science. He is currently a Research Staff Member in the Security Research Department of the IBM T.J. Watson Research Center. His research interests are in the areas of systems security, security analytics, and cyber forensics.



Kui Ren is a professor of Computer Science and Engineering and the director of UbiSeC Lab at State University of New York at Buffalo. He received his Ph.D. degree from Worcester Polytechnic Institute. Kui's current research interest spans Cloud & Outsourcing Security, Wireless & Wearable Systems Security, and Mobile Sensing & Crowdsourcing. His research has been supported by NSF, DoE, AFRL, MSR, and Amazon. He was a recipient of SEAS Senior Researcher of the Year in 2015, Sigma Xi/IIT Research Excellence Award in 2012, and NSF CAREER Award in 2011. Kui has published 150 peer-review journal and conference papers and received several Best Paper Awards including IEEE ICNP 2011. He currently serves as an associate editor for IEEE Transactions on Dependable and Secure Computing, IEEE Transactions on Mobile Computing, IEEE Wireless Communications, IEEE Internet of Things Journal, and IEEE Transactions on Smart Grid. Kui is a Fellow of IEEE, a Distinguished Lecturer of IEEE, a member of ACM, and a past board member of Internet Privacy Task Force, State of Illinois.

Zhenhua Li is an assistant professor at the School of Software, Tsinghua University. He received the B.Sc. and M.Sc. degrees from Nanjing University in 2005 and 2008, and the Ph.D. degree from Peking University in 2013, all in computer science and technology. His research areas cover network security, cloud computing/storage/download, big data analysis, and mobile Internet.

Charles A. Kamhoua is a researcher at the Network Security Branch of the U.S. Army Research Laboratory in Adelphi, MD where he is responsible for conducting and directing basic research in the areas of game theory applied to cyber security. Prior to joining the Army Research Laboratory, he was a researcher at the U.S. Air Force Research Laboratory (AFRL), Rome, New York for 6 years and an educator in different academic institutions for more than 10 years. He has held visiting research positions at Oxford and Harvard. He has co-authored more than 100 peer-reviewed journal and conference papers. He has been invited to more than 40 keynote and distinguished speeches and co-organized more than 10 conferences and workshops. He has mentored more than 50 young scholars including students, postdocs, and AFRL Summer Faculty Fellow. He has been recognized for his scholarship and leadership with numerous prestigious awards including the 2017 AFRL's Information Directorate Basic Research Award "for outstanding achievements in basic research", the 2017 Fred I. Diamond Award for the best paper published at AFRL's Information Directorate, 40 Air Force Notable Achievement Awards, the 2016 FIU Charles E. Perry Young Alumni Visionary Award, the 2015 Black Engineer of the Year Award (BEYA), the 2015 NSBE Golden Torch Award—Pioneer of the Year, and a selection to the 2015 Heidelberg Laureate Forum, to name a few. He received B.S. in electronics from the University of Douala (ENSET), Cameroon, in 1999, an M.S. in telecommunication and networking and a Ph.D. in electrical engineering from Florida International University (FIU), in 2008 and 2011, respectively. He is currently an advisor for the National Research Council, a member of the FIU alumni association, the ACM, and a senior member of IEEE.

Laurent L. Njilla received his B.S. in Computer Science from the University of Yaoundé 1 in Cameroon, an M.S. from the University of Central Florida (UCF) in 2005 and a Ph.D. in Electrical Engineering from Florida International University (FIU) in 2015. He joined the Cyber Assurance Branch of the U.S. Air Force Research Laboratory (AFRL), Rome, New York, as a Research Electronics Engineer in 2015. Prior to joining the AFRL, he was a Senior Systems Analyst in the industry sector for more than 10 years. He is responsible for conducting basic research in the areas of hardware design, game theory applied to cyber security and cyber survivability, hardware Trojan, online social network, cyber threat information sharing, and category theory. Dr. Njilla's research has resulted in more than 20 peer-reviewed journal and conference papers and multiple awards including Air Force Notable Achievement Awards, the 2015 FIU World Ahead Graduate award, etc. He is a reviewer of multiple journals and serves on the technical program committees of several international conferences. He is a member of the National Society of Black Engineer (NSBE).

DaeHun Nyang received a B.Eng. degree in electronic engineering from Korea Advanced Institute of Science and Technology, and M.S. and Ph.D. degrees in computer science from Yonsei University, Korea in 1994, 1996, and 2000 respectively. He has been a senior member of the engineering staff at Electronics and Telecommunications Research Institute, Korea, from 2000 to 2003. Since 2003, he has been an associate professor at the Computer Information Engineering Department of Inha University, Korea where he is also the founding director of the Information Security Research Laboratory. He is a member of the board of directors and an editorial board of Korean Institute of Information Security and Cryptology. Dr. Nyang's research interests include cryptography and network security, privacy, usable security, biometrics and their applications to authentication and public key cryptography.