Evaluation of Two Privacy-Preserving Protocols for the DNS

Sergio Castillo-Perez[†]
[†]Universitat Autònoma de Barcelona,
Dept. of Inf. and Comm. Engineering,
08193 Bellaterra, Spain

Joaquin Garcia-Alfaro[†,‡]
[‡]Open University of Catalonia,
Computer Science and Multimedia Studies,
08018 Barcelona, Spain

Abstract. *The rise of new Internet services, especially those related to the integration of people and physical objects to the net, makes visible the limitations of the DNS protocol. The exchange of data through DNS procedures flows today into hostile networks as clear text. Packets within this exchange can easily be captured by intermediary nodes in the resolution path and eventually disclosed. Privacy issues may thus arise if sensitive data is captured and sold with malicious purposes. We evaluate in this paper two DNS privacy-preserving approaches recently presented in the literature. We discuss some benefits and limitations of these proposals, and we point out the necessity of additional measures to enhance their security.*

Keywords: IT Security, Privacy, Countermeasures, Domain Name System, Privacy Information Retrieval.

1 Introduction

When the Domain Name System (DNS) was designed in the early eighties, it was not intended to guarantee the privacy of people's queries. It was simply conceived as a federated database with information that needed to remain publicly accessible. However, this design is becoming insufficient to face the changes and innovations of today's Internet. A proper example is the use of the DNS protocol as the underlying mechanism of new lookup services for the Internet, such as the use of DNS procedures on VoIP services for the translation of traditional telephone numbers into Internet URLs [6], and the use of the DNS for the resolution of information linked to items of value. Analyses of critical threats to these services can be found in [13, 14].

Threats and vulnerabilities reported in these works are indeed an heritage of the vulnerabilities existing in the DNS mechanisms. We can find in [3] a complete analysis of threats to DNS technologies. The most important threats to DNS technologies can be grouped as follows: (1) authenticity and integrity threats to the trustworthy communication between resolvers and servers; (2) availability threats by means of already existing denial of service attacks; (3) escalation of privilege due to software vulnerabilities in server implementations. Moreover, the DNS protocol uses clear text operations, which means that either a passive attack, such as eavesdropping, or an active attack, such as man-in-the-middle, can be carried out by unauthorized users to capture queries and responses. Although this can be considered as acceptable for the resolution of host names on Web services, an associated loss of privacy when using DNS for the resolution of new lookup services is reported in [13, 14] as a critical threat.

Although there is intensive research on privacy issues in the Internet community, only few approaches seem to deal with the DNS privacy case scenario. Indeed, beyond limiting and granting access to store people's information, no specific mechanisms have been yet proposed by the Internet community to preserve the invasion of privacy that future lookup services may expose. The use of anonymity-based infrastructures and anonymizers (e.g., the use of the Tor infrastructure [11], based on *Onion Routing* cryptography) is often seen as a silver bullet solution to mitigate privacy problems on the Internet. However, these infrastructures might not be useful for anonymizing the queries themselves against, for example, insecure channels or dishonest servers [8]. The use of the security extensions for DNS (known as DNSSEC), proposed by the IETF in the late nineties, only addresses authentication and integrity problems in the DNS. Although it must certainly be seen as an important asset to enhance the security of DNS applications, it requires to be combined with additional measures to cope the kind of violations discussed in this section.

The use of random noise and Privacy Information Retrieval (PIR) [12] mechanisms have recently been proposed by Zhao et al. in [16, 17]. However, no specific evaluations or practical results were presented in these works. Motivated by the benefits and limitations that these two proposals may suppose, we present and evaluate in this paper their implementation on a research prototype tested upon GNU/Linux setups. We discuss some of the benefits and limitations we observed during a set of evaluations, and we point out the necessity of additional measures to enhance their security or proof their validity. Indeed, our implementation combines the development of these mechanisms together with the use of the DNSSEC extension to preserve authentication and integrity of queries. Although our experimentations reveal that the high bandwidth consumption is the main drawback, we consider these results as a proof of the validity of our enhanced approaches.

Section 2 gives the details of a first proposal based on random noise and presents its evaluation. Section 3 presents an extension of this first proposal based on a PIR approach and also the deficiencies detected in the protocol. Section 4 proposes a new protocol that overcomes the problems detected in these proposals. A performance analysis is also included in this section. Section 5 closes the paper with some conclusions.

2 Use of Random Ranges

The approach presented by Zhao et al. in [16] aims at preserving the anonymity of DNS queries from the point of view of the channel and/or the service providers. The authors propose devising the communication protocol involved between DNS clients and servers by considering queries as secrets. Instead of querying the server by a specific host name h, for example, Zhao et al. propose the construction and accomplishment of random sets of host names $[h_1, h_2, \ldots, h_n]$. The resulting protocol aims at avoiding that by eavesdropping the channel, or by controlling the destination service, an attacker learns nothing about the specific host name h from the random list of names. Indeed, a user U, instead of launching just a single query to a given DNS server $\mathcal{NS}$, constructs a set of queries $Q\{H_i\}_{i=1}^{n}$. If we assume DNS queries of type A, the previous range of queries will include up to n different domain names to be resolved. The query $Q\{H_i\}$ will be the only one that includes the domain name desired by U. All the other queries in $Q\{H_1\} \ldots Q\{H_{i-1}\}$ and $Q\{H_{i+1}\} \ldots Q\{H_n\}$ are chosen at random from a database $\mathcal{DB}$. We refer the reader to [16] for a more accurate description of the whole proposal.

2.1 Protocol Analysis

Zao et al. claim in [16] that besides the simplicity of their approach, it may considerably increase the privacy of user when performing DNS queries. Indeed, the only information disclosed by a user U to third parties (e.g., DNS server $\mathcal{NS}$ and possible attackers with either active or passive access to the channel between U and $\mathcal{NS}$) is that the real query $Q\{H_i\}$ is within the interval $[1, n]$. Zhao et al. presume that the probability to successfully predict query $Q\{H_i\}$ requested by user U can be expressed as follows: $P_i = \frac{1}{n}$.

However, we consider that the probability model presented in [16] is very optimistic. We believe that the degree of privacy offered by the model can clearly be degraded if we consider active attacks, in which an adversary is capable of interacting with the channel. Indeed, the approach does not address possible cases in which the resolution of query $Q\{H_i\}$ fails. In case of active attackers that can manipulate network traffic (e.g., by means of RST attacks [2] or sending suitable ICMP traffic [15]), they could launch a blind attack against the resolution protocol. This attack is based on dropping the query $Q\{H_i\}$ — or its associated response. Since attackers do not know which is the query-response pair desired by the client, they will try to force a fail resolution of every query $Q\{H_i\}_{i=1}^{n}$ and theirs associated responses. If so, user U will be forced to restart the process and generate a new range of queries — i.e., requesting once again $Q\{H_i\}$. Depending on how this new range is managed, the degree of privacy estimated by the probabilistic model in [16] clearly decreases. Let $Q_j\{H_i\}_{i=1}^{n}$ be the $j\text{-}th$ consecutive range exchanged for the resolution of the query $Q\{H_i\}$, the probability of success for an attacker trying to guess $Q\{H_i\}$ must then be defined as follows:

$$P_{ij} = \frac{1}{|Q_1\{H_i\}_{i=1}^{n} \cap Q_2\{H_i\}_{i=1}^{n} \cap \ldots \cap Q_j\{H_i\}_{i=1}^{n}|}$$

Let us exemplify this privacy level reduction attack by using the following ideal scenario. We assume a query range size of $n = 3$, a database of queries $\mathcal{DB}= \{H_1, H_2, H_3, H_4, H_5, H_6\}$, a DNS server $\mathcal{NS}$, and a client desired query resolution $Q\{H_1\}$. In the first stage of the protocol (cf. Table 1, Step 1), the client constructs a range query by choosing H_2 and H_3 from $\mathcal{DB}$ at random, resulting on $Q_1 = \{H_1, H_2, H_3\}$. Then, this range is sent to $\mathcal{NS}$ and intercepted by the attacker. In this step, from the point of view of the attacker, we can consider that the guess probability is $P_{i1} = 1/n = 1/3$. At this moment, we suppose that the attacker is able to lead a failed resolution of $Q\{H_1\}$ by manipulating the network traffic. Thus, the client is forced to construct (cf. Step 2) a new range $Q_2 = \{H_1, H_2, H_5\}$ which includes again H_1, and H_2 and H_5 are chosen randomly from $\mathcal{DB}$. When this new range is sent, the attacker can intercept it and calculate the intersection between the previous range and the current one, resulting on a privacy reduction, since $Q_1 \cap Q_2 = \{H_1, H_2\}$ and, consequently, $P_{i2} = 1/2$. Finally, we can see how, if the attacker successfully forces again an incomplete resolution of $Q\{H_1\}$ in Step 2, and intercepts the range $Q_3 = \{H_1, H_6, H_4\}$ built and sent by the client in Step 3, the attacker can deduce the desired query by simply applying the same intersection strategy among Q_2 and Q_3.

Step	Range	Intersection	Guess prob.
1	$Q_1 = \{H_1, H_2, H_3\}$	—	$P_{i1} = 1/3$
2	$Q_2 = \{H_1, H_2, H_5\}$	$Q_1 \cap Q_2 = \{H_1, H_2\}$	$P_{i2} = 1/2$
3	$Q_3 = \{H_1, H_6, H_4\}$	$Q_2 \cap Q_3 = \{H_1\}$	$P_{i3} = 1$

Table 1. Intersection attack to the protocol.

Moreover, the lack of authenticity and integrity mechanisms on DNS procedures may lead to forgery attacks against the proposed protocol. Thus, attackers can send false responses associated to the queries launched by the client $Q\{H_i\}_{i=1}^{n}$, impersonating the $\mathcal{NS}$ server. This becomes an important threat if a DNS protocol based on UDP is preferred over TCP. If that happens, i.e., the protocol does not have a connection establishment, a forgery attack can be performed more easily.

3 Two Server PIR

Zhao et al. present in [17] a second approach intended to reduce the bandwidth consumption imposed by the previous model. The new approach gets inspiration from Privacy Information Retrieval (PIR) field [4]. It relies indeed on the construction of two ranges $Q_1\{H_i\}_{i=1}^{n}$ and $Q_2\{H_i\}_{i=1}^{n+1}$, where $H_{n+1} \in Q_2$ is the desired query defined by user U. Once defined Q_1 and Q_2, such ranges are sent towards two independent servers: $\mathcal{NS}_1$ and $\mathcal{NS}_2$. Assuming the resolution of DNS queries of type A, each server resolves every query linked with its range, obtaining all the associated IP addresses (defined in [17] as X_i) related to the query H_i. $\mathcal{NS}_1$ computes $R_1 = \sum_{i=1}^{n} \otimes X_i$ and $\mathcal{NS}_2$ computes $R_2 = \sum_{i=1}^{n+1} \otimes X_i$. Both R_1 and R_2 are sent to user U, who obtains the resolution associated to H_{n+1} using the expression $X_{n+1} = R_1 \otimes R_2$. As we can observe, the bandwidth consumption of this new approach is considerably smaller than the one in [16], since only two responses (instead of n) are exchanged.

3.1 Protocol Analysis and Evaluation

The main benefit of this last proposal, beyond the reduction of bandwidth consumption, is its achievement on preserving the privacy of the queries from attacks at the server side. However, it presents an important drawback due to the necessity of modifying the DNS protocol and associated tools. Let us note that the proposal modifies the mechanisms for both querying the servers and responding to the clients. Moreover, it still presents security deficiencies that can be violated by means of active attacks against the communication channel between resolvers and servers. Indeed, attackers controlling the channel can still intercept both ranges Q_1 and Q_2. If so, they can easily obtain the true query established by user U by simply applying $Q_1 \setminus Q_2 = H_{n+1}$.

Similarly, if attackers successfully intercept both R_1 and R_2 coming from servers $\mathcal{NS}_1$ and $\mathcal{NS}_2$, they can obtain the corresponding mapping address by performing the same computation expected to be used by user U, i.e., by computing $X_{n+1} = R_1 \otimes R_2$. Once obtain such a value, they can simply infer the original query defined by user U by requesting a reverse DNS mapping of X_{n+1}. Analogously, an active control of the channel can lead attackers to forge resolutions. Indeed, without any additional measures, a legitimate user does not have non-existence proofs to corroborate query failures. This is especially relevant on UDP-based lookup services, like the DNS, where delivery of messages is not guaranteed. Attacker can satisfactorily apply these kind of attacks by intercepting, at least, one of the server responses. An attacker can for example intercept R_1, compute $R_2^* = R_1 \otimes R_3$ (where R_3 is a malicious resolution), and finally send R_2^* as a resulting response coming from server $\mathcal{NS}_2$. Then, the resolver associated to user U will resolve the mapping address as follows: $R_1 \otimes R_2^* = R_1 \otimes R_1 \otimes R_3 = R_3$.

In order to evaluate the performance of the Two Server PIR proposal, we implemented a custom DNS client based on the *Python* language. The main core of the DNS resolution is based on the *dnspython* module [9]. Such a client incorporates both the TCP and the UDP version of the protocol. On the server side, we modified the source code of the NSD server version 3.1.1 (cf. `http://www.nlnetlabs.nl/projects/nsd/`), adopting its behaviour to the Two Server PIR protocol. The environment used for the tests is the following one. A host R, running on an Intel PIV at 2 GHz and 512MB of memory performs the resolution service G. Such a service is composed of the following two servers: $\mathcal{NS}_1$, running on an Intel PIV 2.6 GHz with 1 GB of memory; and $\mathcal{NS}_2$, running on an Intel Xeon 3.2 GHz with 2 GB of memory. The DNS service configured on each one of these two hosts is based on our modified NSD server. In turn, the configuration of each server in G consists of a database $\mathcal{DB}$ that includes 256 $A - type$ records.

To carry out our evaluation, we determine the latency of the whole process for resolving queries from R to G with differents testbeds, where the size of the query range of each testbed increments from ten to one hundred. Each testbed

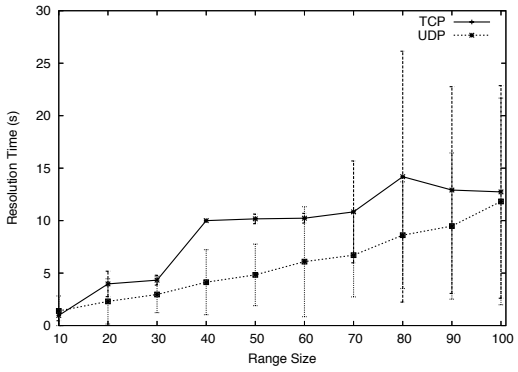


Figure 1. Two Server PIR evaluation results.

consists on the generation of two sets of random queries, one for $\mathcal{NS}_1$ and another (wich includes the query desired by the client) for $\mathcal{NS}_2$. Each testbed is launched multiple times towards cumulative series of $A - type$ queries. Each serie is created at random during the execution of the first testbed, but persistently stored. It is then loaded into the rest of testbeds to allow comparison of results. We split the evaluation in two fases. The first based on TCP transport layer between R and G. The second based on the UDP. Figure 1 depicts the results of our evaluation. We can notice that the latency of the TCP version of the protocol increases linearly to the range size. If we consider, for example, a range of twenty hosts per query, we can see in Figure 1 that the latency for completing the process required up to four seconds. We consider that these results are not acceptable.

Regarding the evaluation of the UDP case, latency performed even worst in some cases. Although one may think that the UDP evaluation should had been better than the TCP case— due to the penalty costs of the connection oriented protocol TCP— the situation is by far different. Indeed, the analysis of our experiments and implementation reveals that Zaho et al. do not address the behaviour of the protocol when a datagram ($Q_1\{H_i\}_{i=1}^{n}$, $Q_2\{H_i\}_{i=1}^{n+1}$, R_1 or R_2) gets lost. Since the UDP is a connectionless protocol, the UDP based client or server, cannot know which packet has not been delivered. To solve this problem, we decided to modify the original proposal by introducing a timeout strategy[1] in both the client and the server. In this manner, when a datagram appears to be missing, both the client and the server related with the missed packet get blocked. Thus, while the client would be waiting for the responses R_1 and R_2, server $\mathcal{NS}_1$ would also be waiting for all the queries $Q_1\{H_i\}_{i=1}^{n}$, and so would be server $\mathcal{NS}_2$ waiting for all the queries $Q_2\{H_i\}_{i=1}^{n+1}$. If the timeout expires, the client is forced to send again the complete range which included the query that is missing, or the range associated to the lost response. At the same time, after the timeout expires for the server, it knows that it must be waiting again for the complete range, and eventually sending its associated response. The cost of sending again each complete range, and waiting their response, results on the large confidence intervals that we can observe in Figure 1.

This performance drawback is directly proportional to the range size. Indeed, a bigger range increases the odds of losing either datagrams or responses. Similarly, the distance between the client and the servers, i.e., the number of hops between the client and the servers, also increases such a probability. We therefore conclude that the proposal presented by Zhao et al. in [17] may only work properly when it is deployed under reliable environments — resulting on a much slower protocol, for an acceptable anonymity level, if the environment does not include this premise.

[1] An alternative could be the use of an acknowledgment-based strategy.

4 Enhanced proposal

To overcome the security problems and performance deficiencies detected in the previous proposals, we present in this section an enhanced protocol inspired in Zaho et al. works.

In general terms, our implementation aims at constructing different ranges of queries for various servers $\mathcal{NS}_1 \ldots \mathcal{NS}_m$. The ranges are distributed among the different servers. When the responses associated to these queries are obtained from the set of servers, our proposed protocol verifies that the anomymized query — hidden within the range of queries — has been successfully processed. If so, the rest of information is simply discarded. On the other hand, and in order to guarantee integrity of queries, authenticity of queries, and non-existence proofs, our proposal relies on the use of the DNS security extensions [5].

The formal description of our proposed protocol is the following one:

- Let U be a user who wishes to perform anonymous resolution of a query $Q^*\{H\}$, and $\mathcal{DB}$ a database of queries.
- User U builds up a table Q of ranges, with every range $Q_j \in Q$ on the interval $j \in [1, m]$, and where the following properties apply:
  - $|Q_j| = n$ (the size of every range is n)
  - $\exists!\ v \in [1, m]$ such as $Q^*\{H\} \in Q_v$
  - $Q_j\{H_{ji}\}_{i=1}^{n} \neq Q^*\{H\}$ are selected at random from $\mathcal{DB}$, such that $\bigcap_{i=1}^{n} Q_j\{H_{ji}\} = \varnothing$
  - $\bigcap_{j=1}^{n} Q_j = \varnothing$
- User U concurrently and randomly sends each range Q_j to a different server $\mathcal{NS}_w \quad \forall w \in [1, m]$ with DNSSEC extensions enabled.
- User U verifies that all the responses have been properly received and their DNSSEC signatures are correct. Otherwise, the failed queries are retried until the responses are received and their signatures are correct, or until a certain number of retries R are achieved. In that case, the user is warned and the whole protocol is aborted.
- User U discards all those resolutions that are not associated to $Q^*\{H\}$.

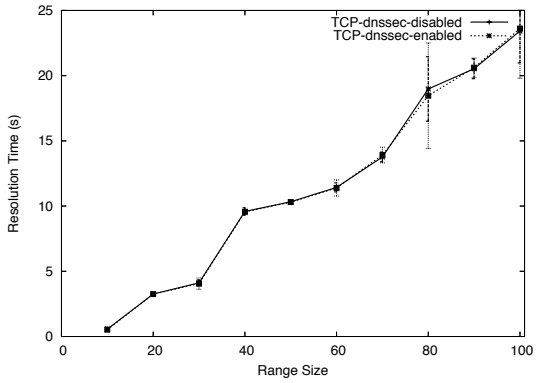
We evaluated our implementation by measuring the latency penalty on a real network scenario. We measured the resolution of both DNS and DNSSEC queries. The hardware setup used for our experimental scenario is the following. A host R, running on an Intel PIV 3.2 GHz and 4 GB

of memory, performs DNS queries to a global resolution service G. The implementation and deployment of our proposal in R is based on the *Python* language. More specifically, we base our implementation on the module *dnspython* [9] for the construction and resolution of DNS queries; and the module *m2crypto* [10] to access the *OpenSSL* library for the verification of digital signatures defined by DNSSEC.

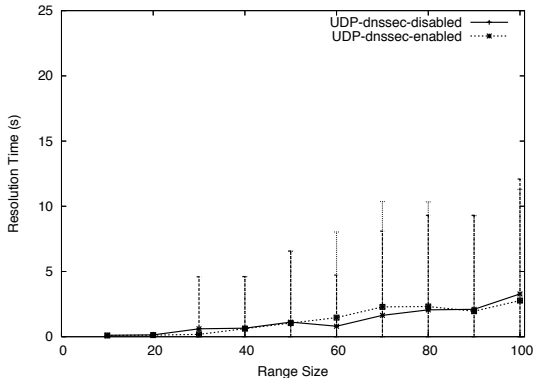
The global resolution service G is in turn implemented by means of three different hosts: $\mathcal{NS}_1$, that runs on an Intel PIV 2.6 GHz with 1 GB of memory; $\mathcal{NS}_2$, that runs on an Intel Xeon 3.2 GHz with 2 GB of memory; and $\mathcal{NS}_3$, that runs on an Intel Core Duo 3 GHz with 1 GB of memory. DNS and DNSSEC services configured on each one of these hosts are based on NSD 3.1.1. The configuration of each server in G consists of a database $\mathcal{DB}$ that includes 256 $A-type$ records. Each one of these DNS records are linked with their appropriate DNSSEC signature. We use for this purpose the *zonec* tool that comes with NSD 3.1.1. The key sizes are of 1024 bits. The generation of keys is based on the RSA implementation of *dnssec-keygen*. Although the use of ECC signatures seems to reduce the storage space of signed zones [1], the algorithm we use is RSA instead of ECC since the latter is not yet implemented in NSD 3.1.1.

During our evaluation, we measured the time required for resolving queries from R to G with different testbeds, where the size of the query range of each testbed increments from ten to one hundred. Each testbed consists indeed on the generation of three sets of random queries, one for each $\mathcal{NS}_i \in G$. Each testbed is launched multiple times towards cumulative series of $A - type$ queries. Each series is created at random during the execution of the first testbed, but persistently stored. It is then loaded into the rest of testbeds to allow comparison of results. We split our whole evaluation in four different stages. During the first two stages, the transport layer utilized between R and G is based on the TCP protocol. First stage is used for the resolution of DNS queries, while stage two is used to resolve DNSSEC queries. Similarly, stage three and four are based on UDP traffic for the resolution of, respectively, DNS and DNSSEC queries. During these two last experiments based on DNSSEC, R verifies the integrity and the authenticity of the queries received from the different servers in G. The verification procedures have been implemented as defined in DNSSEC RFCs [5]. We show in Figure 2 the results that we obtained during the execution of these four experiments.

We can appreciate by looking at Figure 2 that the latency increases linearly with the size of the range of queries. TCP-based experiments show worst performance than UDP-based queries — due to the penalty imposed by the traffic that guarantees the delivery of packets. UDP protocol is clearly the best choice for the deployment of our proposal. Contrary to the UDP Two Server PIR implementation (cf. Section 3), the enhanced proposal presented here does not



(a) TCP tests with/without DNSSEC



(b) UDP tests with/without DNSSEC

Figure 2. Evaluation of our proposal.

suffer from the performance degradation outlined in Section 3.1 to guarantee a proper reception of complete ranges. Indeed, in our enhanced proposal, when a query or its associated response is not delivered, the protocol re-sends this particular query again, instead of a complete range.

From the point of view of the privacy, given an acceptable latency of no more than two seconds, UDP results show that the probability of guessing the true query is $P_i = \frac{1}{3 \cdot 80} = \frac{1}{240} \simeq 0.004167$. We consider this result as satisfactory. In general terms, we should expect that the certainty for obtaining a query i within a range of size n and m different servers is $P_i = \frac{1}{n \cdot m}$.

Besides the difficulties imposed by our model for predicting the original petition, we are conscious of the high bandwidth increase that it represents. This is an important drawback in scenarios where the bandwidth consumption is a critical factor. However, if this is the case, it is possible to reduce the size of the range of queries. Since there is a clear relation between both parameters, i.e., the bandwidth consumption is inversely proportional to the prediction probability, we believe that a proper balance between bandwidth consumption and prediction probability can be enough to

enhance the privacy of the service. Let us recall that reducing the size of each range of queries to a fifty per cent, the prediction probability for the attacker is proportionally increased by two. On the other hand, let us observe how the penalty in the response times introduced by DNSSEC is not specially significant, solving the integrity and authenticity problems that appeared in the other approaches. This is the reason why we consider the activation of DNSSEC as a decisive factor to avoid integrity attacks.

5 Conclusions

We have presented in this paper the evaluation of two DNS privacy-preserving approaches recently proposed in the literature. The preservation of privacy of these two proposals is achieved by introducing random noise during the execution of DNS queries. We analyzed the benefits and limitations of these two proposals. Since no specific evaluations or practical results about these two proposals have appeared in the previous literature, we implemented and evaluated them on a research prototype tested upon GNU/Linux setups. Our evaluation confirms that the main benefit of the first proposal is its simplicity; and that the main drawback of the first approach is the increase of latency and bandwidth during the execution and resolution of queries. The second approach, which aims at reducing these limitations, gets inspiration from Privacy Information Retrieval (PIR) techniques. We observed that although the second approach successfully reduces bandwidth consumption, it significantly modifies the DNS protocol. It therefore requires the modification of both DNS client and servers. Moreover, we pointed out to serious security flaws on both proposals if active attackers can target those mechanisms. We addressed these flaws on an improved version of the two proposals, and concluded that they still require additional improvements to be effective.

Acknowledgments — This work is partially supported by the Spanish Ministry of Science and Innovation and the FEDER funds under the grants *TSI2007-65406-C03-03 E-AEGIS* and *CONSOLIDER-INGENIO 2010 CSD2007-00004 ARES*.

References

[1] Ager, B., Dreger, H., and Feldmann, A. Predicting the DNSSEC Overhead Using DNS Traces. *40th Annual Conf. on Information Sciences and Systems*, 2006.

[2] Arlitt, M. and Williamson, C. An analysis of TCP reset behaviour on the internet. *ACM SIGCOMM Computer Communication Review*, 35(1):37–44, 2005.

[3] Atkins, D. and Austein, R. Threats Analysis of the Domain Name System (DNS). *RFC 3833*, 2004.

[4] Chor, B., Kushilevitz, E., Goldreich, O., and Sudan, M. Private Information Retrieval *Journal of the ACM*, pp. 965–981, New York, USA, 1998

[5] DNSSEC Deployment Initiative. Available from: http://dnssec-deployment.org/

[6] Faltstrom, P. and Mealling, M. The E.164 to Uniform Resource Identifiers Dynamic Delegation Discovery System Application. *Request for Comments, RFC 3761*, IETF, 2004.

[7] Garcia-Alfaro, J., Barbeau, M., Kranakis, E. Analysis of Threats to the Security of EPC Networks. *6th Annual Conference on Communication Networks and Services Research (CNSR'08)*, IEEE, pages 67–74, Canada, May 2008.

[8] Garcia-Alfaro, J., Barbeau, M., Kranakis, E. Evaluation of Anonymized ONS Queries. *Security of Autonomous and Spontaneous Networks (SETOP 2008)*, pp. 47–60. Loctudy, France, 2008.

[9] Nomium Inc. A DNS Toolkit for Python http://www.dnspython.org/

[10] Siong, et al. Mee Too Crypto. http://chandler-project.org/bin/view/Projects/MeTooCrypto.

[11] Dingledine, R., Mathewson, N., and Syverson, P. F. Tor: The second-generation Onion Router. *13th conference on USENIX Security Symposium*, 2004.

[12] Ostrovsky, R. and Skeith, W.E. A Survey of Single Database PIR: Techniques and Applications. *Proceedings of Public Key Cryptography (PKC-2007)*, 2007.

[13] Rossebø, J., Cadzow, S., and Sijben, P. *e*TVRA, a Threat, Vulnerability and Risk Assessment Tool for *e*Europe. *4th Int'l Conf. on Trust Management (iTrust 2006), Springer, LNCS 3986, pp. 467–471, Pisa, Italy, 2006*.

[14] Rossebø, J., Cadzow, S., and Sijben, P. eTVRA, a Threat, Vulnerability and Risk Assessment Method and Tool for eEurope. *2nd Int'l Conf. on Availability, Reliability and Security, ARES 2007, pp. 925–933, Vienna, Austria, 2007*.

[15] Singh, A., Nordstrom, O., Lu, C., dos Santos, A. Malicious ICMP Tunneling: Defense against the Vulnerability. *8th Australasian Conference on Information Security and Privacy, ACISP 2003*, pp. 226–235, Wollongong, Australia, July, 2003.

[16] Zhao, F., Hori, Y., and Sakurai, K. Analysis of Privacy Disclosure in DNS Query. *IEEE International Conference on Multimedia and Ubiquitous Engineering*, pp. 952–957, 2007.

[17] Zhao, F., Hori, Y., and Sakurai, K. Two-Servers PIR Based DNS Query Scheme with Privacy-Preserving. *IEEE International Conference on Intelligent Pervasive Computing*, pp. 299–302, 2007.