

# PPDNS: Privacy-Preserving Domain Name System

Yanbin Lu

Computer Science Department, University of California, Irvine  
yanbinl@uci.edu

**Abstract**—In current DNS environment, each DNS query generated by a user reveals the origin and target of that query. Over time, a user’s browsing behavior and target domain name’s popularity might be exposed to entities with little or no trust.

This paper presents the design and evaluation of a Privacy-Preserving Domain Name System (PPDNS), which prevents privacy leaks during domain name resolution with bounded communication and computation overhead. PPDNS is based on CoDoNS [3], an overlay DNS system, and the computational private information retrieval (PIR) scheme [2], an advanced cryptographic construct. Our evaluation shows that PPDNS results in significantly improved privacy for DNS queries.

## I. INTRODUCTION

The Domain Name System (DNS) translates human-readable domain names into numerical identifiers associated with networking equipment for the purpose of locating hosts. Given the importance of DNS, much effort has been invested into the next generation of secure DNS, referred to as DNSSEC. The main purpose of DNSSEC is to ensure the authentication and integrity of DNS records. Notably, privacy is not one of DNSSEC’s goals.

We claim that privacy of DNS queries is a very important, and unfortunately largely overlooked problem. In the DNS context, privacy encompasses two issues: (1) relationship between the source (client) and the target of the query, (2) “popularity” of a target domain name, i.e., the number of DNS queries issued against a given target.

Today’s DNS provides no privacy: all DNS messages are transmitted in plaintext and the relationship between a DNS query and its source is very clear. Consequently, users’ web browsing patterns are exposed to possible adversaries. Of course, clients may choose to use an anonymity service, such as TOR [1], to hide DNS queries’ origin. This prevents privacy leaks of source-target relationship and client’s activity. However, target popularity remains exposed and adversaries can take advantage of the popularity information for commercial purposes. For example, if an internet domain registrar, counts the volume to its sub-domain names registered under it and does popularity analysis to them, the registrar can later choose to reserve those very popular sub-domains after their registration expire and sell them at higher prices.

This paper is intended to address the DNS query privacy leak issue by designing a brand new privacy-preserving domain name system (PPDNS). Our solution basically combines CoDoNS [3] with Lipmaa’s computational private information retrieval (PIR) [2] technology. CoDoNS is a peer-to-peer domain name system whose underlying routing structure is Pastry [4]. Due to the prefix-matching identifiers employed by CoDoNS, we are able to launch the same range query for

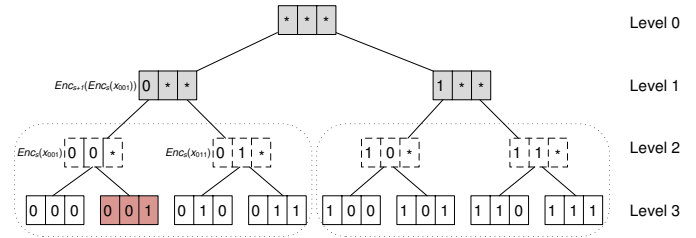


Fig. 1. Identifier space example: Each identifier comprises 3 digits in base 2. Gray boxes mean CoDoNS nodes; solid boxes at level 3 mean resource records; white dashed boxes mean virtual tree nodes

each identifier inside this range, thereby hiding the real query target. Meanwhile, we can also achieve the same range query for each identifier inside the range among all clients, thereby hiding the popularity of domains. What’s more, we find that the prefix-matching identifier can even facilitate the Lipmaa’s computational PIR. Therefore we leverages computational PIR to reduce the traffic caused by our special range query.

The key contributions of this work are: 1) we propose a special range query based on CoDoNS to reduce privacy leaks during DNS resolution; 2) we combine Lipmaa’s computational PIR with prefix-matching DHT identifiers; 3) we formally model two types of privacy leaks during DNS resolution; 4) our scheme significantly outperforms a previous approach in terms of the two security models.

## II. METHODOLOGY

PPDNS takes advantage of prefix-matching DHT identifiers used by CoDoNS to achieve a special range query scheme which has two properties: 1) For one specific domain name, the range query generated at any time by any client is the same. 2) The same range query can be posed for each domain name inside this range.

To illustrate the above two properties possessed by PPDNS, we first briefly introduce CoDoNS [3]. CoDoNS is a peer-to-peer domain name system in which each node and each object has an identifier. Given an identifier of a target domain name, the CoDoNS uses its underlying prefix-matching DHT routing structure [4] to find a CoDoNS node having the resource records for the domain name and returns the resource records to the client. In a prefix-matching DHT, the whole identifier space can be seen as a tree and the routing can be thought of as finding a path from the root to one leaf. For example, Fig. 1 shows an identifier space where each leaf corresponds to one resource record. To retrieve a resource record with identifier 001, the client sends the request to an arbitrary CoDoNS node who can be thought of as the root of the identifier tree. The root then forwards the request to another level-1 node whose

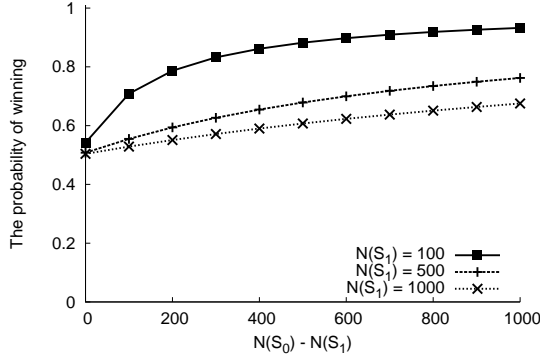


Fig. 2. The adversary’s winning probability in the second model for PPDNS

identifier starts with 0. In this example, the level-1 node (0\*\*) holds the target record, so it can return the result to the client. Inside this level-1 CoDoNS node, we can still think there is a virtual level-2 node (00\*) who forwards the request to the target (001) at level-3.

In PPDNS, when a client wants to resolve a domain name, the client will launch a range query, with size  $m$ , which is defined to be the leaves of a subtree that can cover the target identifier. For example, in Fig. 1, if  $m = 4$ , the client, in order to retrieve 001, will launch a query of range from 000 to 011 which makes up the leaves of a subtree whose root resides at level 1 in the identifier space. It is easy to check that the range query for 001 generated at any time by any client is the same. It is also clear that the same subtree range query can be posed for any identifier from 000 to 011. Therefore, PPDNS holds the two properties mentioned at the beginning of this section.

Purely doing range query could cause a severe network jam. To reduce the traffic, PPDNS employs Lipmaa’s computational PIR [2]. Note a  $d$ -digit CoDoNS identifier can be seen as a  $d$ -dimensional hypercube in Lipmaa’s PIR, which means we can directly apply Lipmaa’s PIR without any modification to the identifier structure. Take Fig. 1 as an example. Let  $x_{i_1 i_2 i_3}$  be the record attached to identifier ( $i_1 i_2 i_3$ ). If the target record’s identifier is 001, the client will compose  $\{Enc_{s+1}(1), Enc_{s+1}(0)\}$  for the second digit,  $\{Enc_s(0), Enc_s(1)\}$  for the third digit and sends them to the node 0\*\*. The  $Enc$  used here is an additive homomorphic encryption with the client knowing both its public and private key, and  $s$  is a length parameter. For identifier 000 and 001, the target node will compute  $Enc_s(0)^{x_{000}} \cdot Enc_s(1)^{x_{001}} = Enc_s(x_{001})$  and put it at the parent of 000 and 001. Similarly,  $Enc_s(x_{011})$  is computed for 010 and 011. Finally, the target node computes  $Enc_{s+1}(1)^{Enc_s(x_{001})} \cdot Enc_{s+1}(0)^{Enc_s(x_{011})} = Enc_{s+1}(Enc_s(x_{001}))$  and sends it back to the client. The client gets  $x_{001}$  by decrypting the response by 2 times. Here we can see that, instead of replying 4 messages to the client, the 0\*\* node only need to reply 1 message.

### III. EVALUATION RESULTS

In this section, we evaluate the security of PPDNS according to two adversary models and compares it to a previous random-range query approach [5].

In the first adversary model, the adversary, given a range query, tries to guess the target domain name a client wants to resolve. We endow the adversary with the power of launching denial-of-service attack to make the client re-issue a range query for the same target domain name. For a random range query, as long as the adversary makes the client re-issue the random range query for the same domain name for enough times, the intersection over all these range queries approaches the target domain name with probability 1. For PPDNS, the probability of successfully guessing the target domain name is always  $\frac{1}{m}$ , where  $m$  is the range size, since the range query for the same identifier keeps to be the same.

In the second adversary model, the goal of the adversary is to guess the popularity difference between two domain names. We use  $N(d)$  to denote the number of real resolution attempts at domain name  $d$  and use  $N'(d)$  to denote the resulting number of queries generated to  $d$  in a fixed interval. The adversary is assumed to be able to record  $N'(d)$  for each  $d$  in the domain name space. The adversary’s task is to pick two domain names  $d_0$  and  $d_1$  at its discretion and outputs a bit  $b$ . We say the adversary succeeds if  $N'(d_b) > N'(d_{1-b})$ . For the random range query, we claim the adversary succeeds with probability 1. This is because, for any two domain names, the expected number of times of being chosen to be inside other domain names’ random query range is the same. So, on average,  $N'(d_0) - N'(d_1) = N(d_0) - N(d_1)$ . Therefore, the adversary can use  $N'(d_0)$  and  $N'(d_1)$  to tell which domain name is more popular.

For PPDNS, the best the adversary in the second model can do is to choose  $d_0$  and  $d_1$  in two separate ranges  $S_0$  and  $S_1$ , and count the total number of hits,  $N(S_0)$  and  $N(S_1)$  to these two ranges. However, the value of  $N(S_0)$  and  $N(S_1)$  depends on the sum of the real resolution attempts at all identifiers in the range. Therefore  $N(S_0) > N(S_1)$  does not necessarily mean  $N(d_0) > N(d_1)$ . Fig. 2 shows the adversary’s winning probability by guessing  $N(d_0) > N(d_1)$  if  $N(S_0) > N(S_1)$ . As we can see, the adversary’s winning probability increases sublinearly with the gap between  $N(S_0)$  and  $N(S_1)$ . And even in the case  $N(S_0) = 2N(S_1)$ , the winning probability of the adversary is smaller than 80% in our settings.

### REFERENCES

- [1] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *SSYM’04: Proceedings of the 13th conference on USENIX Security Symposium*, Berkeley, CA, USA, 2004.
- [2] H. Lipmaa. An oblivious transfer protocol with log-squared communication. In *Information Security*, pages 314–328. 2005.
- [3] V. Ramasubramanian and E. G. Sirer. The design and implementation of a next generation name service for the internet. In *SIGCOMM’04*, New York, NY, USA, 2004.
- [4] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware’01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg*, pages 329–350. Springer-Verlag, 2001.
- [5] F. Zhao, Y. Hori, and K. Sakurai. Analysis of privacy disclosure in dns query. In *Proceedings of the International Conference on Multimedia and Ubiquitous Engineering*, Washington, DC, USA, 2007.