Int. J. Inf. Secur. manuscript No.
(will be inserted by the editor)

OnionDNS: A Seizure-Resistant Top-Level Domain

Nolen Scaife · Henry Carter · Lyrissa Lidsky · Rachael L. Jones · Patrick Traynor

Received: date / Accepted: date

Abstract The Domain Name System (DNS) provides the critical service of mapping canonical names to IP addresses. Recognizing this, a number of parties have increasingly attempted to perform "domain seizures" on targets by having them delisted from DNS. Such operations often occur without providing due process to the owners of these domains, a practice made potentially worse by recent legislative proposals. We address this problem by creating OnionDNS, an anonymous top-level domain (TLD) and resolution service for the Internet. Our solution relies on the establishment of a hidden service running DNS within Tor, and uses a variety of mechanisms to ensure a high-performance architecture with strong integrity guarantees for resolved records. We then present our anonymous domain registrar and detail the protocol for securely transferring the service to another party. Finally, we also conduct both performance and legal analyses to further demonstrate the robustness of this approach. In so doing, we show that the delisting of domains from DNS can be mitigated in an efficient and secure manner.

N. Scaife · P. Traynor
Department of Computer & Information Science & Engineering, University of Florida, Gainesville, FL, U.S.A.
E-mail: scaife@ufl.edu E-mail: traynor@cise.ufl.edu

H. Carter
Department of Computing Sciences, Villanova University, Philadelphia, PA, U.S.A.
E-mail: henry.carter@villanova.edu

L. Lidsky
School of Law, University of Missouri, Columbia, MO, U.S.A.
E-mail: lidskyl@missouri.edu

R. L. Jones
School of Law, University of North Carolina, Chapel Hill, NC, U.S.A.
E-mail: rachael_jones@unc.edu

Keywords Tor hidden services · DNS · proof-of-work · censorship resistance

1 Introduction

The Domain Name System (DNS) provides address translation functionality for the majority of applications used on the Internet. From web browsing to BitTorrent, applications rely on this highly-distributed service to provide mappings between human-readable canonical names and IP addresses. Such translation is critical for allowing hosts to easily communicate with devices that may be disparately and dynamically located throughout the address space.

Attacks against the infrastructure supporting DNS have become commonplace in recent years [56,61]. However, of growing concern is the increasing prevalence of attacks against specific domains and hosts enumerated by this service. Recent legislative proposals including the Anti-Counterfeiting Trade Agreement (ACTA) Union [31], the Stop Online Piracy Act (SOPA) [6] and the PROTECT IP Act (PIPA) [7] allow for the potential delisting of such targets from DNS by third parties. Moreover, a number of other nation-states aggressively modify DNS [12,74], further preventing free access to information. Such "domain seizures" generally fail to provide due process for accused targets.

Domain seizure without due process is becoming increasingly common. In November of 2010, the popular hip-hop music blog `dajaz1.com` was seized by US Immigration and Customs Enhancement (ICE) as part of "Operation in Our Sites." The website, which legitimately provides links to promotional pre-releases sent directly by copyright holders, was flagged by the Recording Industry Association of America (RIAA) as a "rogue site" [2,44]. While the case against `dajaz1.com` was eventually dropped and the domain returned, the domain's owner lost all revenue for a period of nearly a year. Such incidents are not isolated, with ICE mistakenly seizing some 84,000 legitimate domains in 2011 [66].

We combat this problem by developing OnionDNS, an anonymous top-level domain (TLD) and resolution service for the Internet. OnionDNS removes the TLD as a single point of failure for domain seizures by anonymizing the root server and providing redundant access to DNS listings. Specifically, we create a hidden service running an unmodified version of BIND [36] within the Tor network, then create a series of public whitelisted mirrors that can perform DNS lookups for domains using the `.o` TLD. This technique resists takedown by providing multiple mirrors as public portals, making it significantly more difficult for an attacking body to completely delist any domain by seizing all mirrors or the Tor hidden service. Our architecture is designed to scale and avoid attempts to locate the hidden service through the use of mandatory caching, and the results provided by OnionDNS are verifiable by the use of our own DNSSEC signing key. Although Tor is not required for users, we also analyze the performance of OnionDNS requests over Tor and demonstrate that OnionDNS lookups require between 1-2 seconds to complete, imposing only small (and front-loaded) overhead.

We make the following contributions in this paper:

- **Develop and implement a domain seizure-resistant architecture:** We present the OnionDNS architecture and argue that its use of a hidden service, caching, and DNSSEC provides strong resistance against domain seizures with minimal work by end users (i.e., no software to install). By hiding the root TLD service and providing redundant provisioning for public access, our solution strikes a balance between sufficient security guarantees and practical, scalable performance. While alternative models of DNS have been proposed, it is our unique combination of technologies, our application of a distributed architecture and our consideration of operational security issues that makes this approach novel. We then discuss our specific implementation of this proposed architecture.
- **Implement and characterize domain registrar:** A practical architecture for resisting domain seizure requires more than simply establishing a hidden service. Accordingly, we also address the challenges of domain registration, renewal, and transfer with our domain registrar, Leek.
- **Develop a Privacy-Preserving Deployment Protocol for the OnionDNS service:** We recognize that a coordinated effort among collaborating parties interested in protecting freedom of speech online creates the potential for malicious insiders to disrupt the operation of OnionDNS. To resolve this, we develop a protocol for transferring ownership of the OnionDNS root server to an anonymous party using secure multiparty computation. This protocol ensures that the individuals running the service are randomly selected and maintain anonymity, preventing national actors from compelling them to delist any domain. To complement the security guarantees of our system, we provide an analysis of the legal difficulty of compelling disclosure in our architecture, protecting the system with international legal boundaries in addition to our technical constructions.

We note that OnionDNS is not intended to be a complete censorship circumvention solution, as traffic to domains resolved by a query to this service will remain observable. Readers should not confuse this solution with Tor. Instead, OnionDNS takes advantage of the fact that in the majority of domain seizures, delisting from DNS is the only step taken against the target, *as physical seizure of IP address space and widespread filtering are often difficult or impossible to enforce* [54]. Moreover, our approach does not require end users to install Tor, instead allowing for any device pointing to our DNS mirror resolvers to learn the mapping between domain name and IP address without the significant performance penalty of sending the entirety of their traffic through this anonymity network.

This work extends the results presented by Scaife et al. [64] by providing the critical implementation details needed for correct and secure implementation. Because we wish for any party to be able to establish their own takedown-resistant tld service, we extend the conference version of this work with a thorough treatment of the design goals of OnionDNS and Leek. This provides would-be service managers with the critical motivation of every component of the system, ensuring that no vulnerabilities are introduced through improper use. We expand upon our protocol for transferring ownership of the service to allow the protocol to be repeatable for initial owner selection or future management handoffs. Finally, we augment the security of our system by analyzing the legal requirements for taking down the system. Based on this analysis, we now encourage an internationally distributed ownership transfer to protect the privacy of the system through both technical constructions and international jurisdiction boundaries.

The remainder of the paper is organized as follows: In Section 2, we present related work in DNS censorship. In Section 3, we present the design details of both OnionDNS and our registration system, Leek. Section 4 provides implementation details for our systems. In Section 5, we present our performance analysis. Section 6 discusses the details of our plan to give away control of the `.o` TLD obliviously, yet securely. In Section 7, we analyze the legal challenges of subverting our system. In Section 8, we discuss practical challenges to deployment. Finally, in Section 9 and the appendices, we offer background, discussion, and conclusions.

2 Related Work

Due to the criticality of Internet name resolution, there has been extensive work over the last few decades to improve or replace the existing DNS system. Cox et al. proposed a domain name system supporting lookups using a distributed hash table (DHT) [24]. This approach has been further studied to improve latency of queries and robustness against denial-of-service (DoS) attacks [60,62,63,65,14]. However, the DHT approach in general has been shown to have lower availability under node failures and poorer cache performance than the existing DNS system [59].

Furthermore, centralized control over the Internet naming infrastructure has been disputed [34,22], in part because centralization provides powerful entities the ability to perform domain seizures. Mestdagh et al. [46] dismiss concerns about the unilateral control by referencing the Open Root Server Network (ORSN). ORSN attempts to solve the centralization problem by offering an additional geo-diverse network of root servers that are daily mirrors of ICANN's zones. However, as long as the physical locations of the servers is known or easily obtainable, governments and other entities can collude to synchronize domain seizures.

Namecoin [5] also attempts to remove central control of the DNS system by creating the `.bit` TLD. It is based on a proof-of-work blockchain that ensures any new domains must have some amount of computational investment involved in their registration. This system is uniquely both unable to scale and yet requires scaling for security because of its reliance on the Bitcoin protocol [55]. Security is provided by the difficulty for an adversary to recreate the blockchain, which is only the case when the blockchain is massive and constantly growing. This restriction causes Namecoin to be unable to scale for clients performing domain lookups. In addition, serious security problems were found in Namecoin's protocol allowing any user to steal any domain [1].

Broader approaches to censorship resistance include early work on Publius [70], Tangler [69], and Free Haven [27], which are all based on the theoretical motivations from Anderson's Eternity Service [10]. These approaches have worked to provide a means to publish content in a manner resistant to censorship. Important progress has been made in this area [23,32,71,26], and several systems, such as Tor [28] have seen widespread adoption. However, each implemented system is challenging to use and integration with existing software requires significant modifications. By tackling a very specific form of censorship, OnionDNS is able to provide a seizure-resistant name system without requiring custom software, architectural changes, or the performance degradation associated with these systems.

None of the mentioned name resolution systems have been successful in providing an alternative name system that has seen widespread adoption, largely because their solutions cannot scale sufficiently. OnionDNS provides a DNS service built on existing, deployed technologies that are widely implemented and proven to scale. This service is designed to be seizure-resistant in that external pressure cannot be placed on the operators. This allows OnionDNS to be easily used within the current Internet infrastructure while preventing the global censorship that occurs at the TLD level.

3 Goals and Design

The primary goal of OnionDNS is to resist global domain seizures. We accomplish this by hiding a trusted top-level domain authoritative nameserver via an anonymity network. This prevents any single entity from pressuring the trusted server to change specific DNS records. The resource records from this top-level domain's zone are mirrored through public mirrors, allowing seizure-resistant domains to scale similar to the existing DNS system.

There are many challenges in designing such a system: it must be able to scale to the size of the current public DNS without making it easy to identify the hidden service; it must perform well at a similar scale; its mirrors cannot be trusted to not modify records; registering a domain cannot be so easy that domain squatting is rampant; the software must be trusted and well-verified; and it must be easy to install and use for a non-technical user. This section presents the goals of OnionDNS and the complementary domain registration (a.k.a. Leek) infrastructure, outlines their design, and describes our implementation.

3.1 OnionDNS Goals

OnionDNS was designed to balance security and usability needs with the following requirements:

Goal 1. ***Seizure-Resistance****: Assuming the system is operated by a trusted actor, it must be resistant to global domain seizures.*

Goal 2. ***Usability****: Users must not have to significantly modify their systems in order to use the OnionDNS domain.*

Goal 3. ***Integrity****: The service must provide users a facility to verify the integrity of received answers.*

Goal 4. ***Attribution****: Domain registrations must necessarily associate to a particular user. Only the domain's owner and the Root Administrator shall be able to make changes to an existing registration (for example, to update a domain's nameservers).*

Goal 5. ***Accessibility****: The service must be easily accessible, but the root must not be easily geo-locatable.*

3.2 OnionDNS Design

3.2.1 Adversaries

The owner of a top-level domain is a high-value target for performing domain seizures since it provides centralized control over all its subdomains. Adversaries that are able to subvert the system's seizure-resistance or integrity goals could exert subtle control without detection. In designing the system, we have cataloged various attacks against OnionDNS, including:

- **Domain seizure**: An adversary may seize a domain (i.e., to have the domain delisted from DNS or have its traffic redirected without permission from said domain) through a centralized system by exerting control over the operator of the DNS servers or system. In particular, entities that may wish to conduct seizures include governments, groups representing copyright or trademark holders and potentially extremist activist groups [58]. Seizures can be executed via direct (e.g., court orders and political pressure) and indirect (e.g., preventing payment for services) influence.
- **Malicious or subverted mirror**: An adversary may attempt to compromise an existing mirror or build a mirror in order to prevent clients from resolving a domain. In effect, this creates the same outcome as a seized domain.
- **Denial-of-registration**: An adversary may register attractive domains in a land rush, exhausting the domain space to lock out legitimate users.

3.2.2 Roles and Responsibilities

There are two roles required for OnionDNS, each maintained by an independent, anonymous actor:

- **Root Administrator**: This actor manages the OnionDNS hidden service and only possesses a temporary DNSSEC ZSK. The Root Administrator is responsible for generating its public/private key pair and sending the public key to the Key Owner for signature.
- **Key Owner**: This actor manages the DNSSEC trust anchor and signs the Root Administrator's ZSK. The Key Owner is responsible for ensuring that the Root Administrator is managing the hidden services correctly. In a scenario where the Root Administrator is no longer trustworthy, this actor must identify a new Root Administrator and sign the new ZSK.

Both parties must maintain their anonymity for the security of the system to hold. This dual-control model was selected because it implements DNSSEC's native key management. We leave discussion of other models (i.e., distributed "hot standby" key management) to future work.

3.2.3 Core Infrastructure

Because DNS is a well-established and proven service, attempting to modify it or replace it would conflict with Goal 2. We instead rely on the use of an anonymizing overlay network to protect the OnionDNS root server from external pressure. This approach comes with many advantages: DNS is proven to scale, popular DNS server software is well-maintained, and support for using the existing DNS protocol is built into nearly every networked computing device. This ensures usability for OnionDNS users. DNSSEC provides necessary integrity guarantees for DNS records provided via OnionDNS, but is not sufficient for security against the adversaries described above as the root DNS servers are still public and visible.

There have been many proposed attacks against hidden services in anonymization systems [37,17,52,57]. However, Tor has had several high-profile hidden services running within its network [17] that the authors do not believe have been fully deanonymized as a result of breaking the Tor network. Due to this, we separate the trust root for the system from the administrator of the hidden service, creating a dual control system to ensure that any misbehaving or compromised Root Administrator will only have a valid trusted key for a short period of time.

We introduce `.o` as the OnionDNS top-level domain.[1] This domain was chosen because it does not conflict with another Internet top-level domain; in addition, introducing a new root domain (`.`) would break

[1] Our solution neither requires nor requests endorsement or support from ICANN.

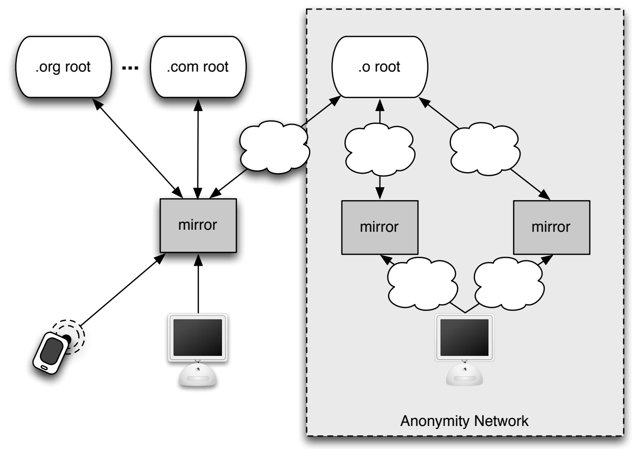

Fig. 1: The OnionDNS architecture. A connection with a cloud inside of it indicates that it is through an anonymous network and the communicating parties are anonymous to each other. Clients make requests to OnionDNS mirrors through the open Internet or through Tor. The mirrors retrieve the `.o` zone from the hidden root server. A client request never directly results in traffic to or from the hidden root server, and mirrors do not need to be trusted due to the use of DNSSEC.

interoperability with other DNS services. The choice of name is ultimately arbitrary, and a number of different seizure-resistant top-level domains could be run in parallel.

Figure 1 shows the OnionDNS architecture. Mirrors are used to serve requests directly to users through the open Internet in order to satisfy Goal 5. The root server accepts queries and zone transfer requests only from a set of whitelisted mirrors who connect to the root using anonymizing Tor circuits. No other system may query or request transfers from the root server; the root will actively reject these requests. The mirrors are authoritative for the `.o` zone, so no client query will directly result in a secondary communication between a mirror and the root. In addition, these mirrors may also provide domain resolution for other TLDs through the existing DNS request forwarding mechanism. However, only DNS responses for the `.o` zone will be guaranteed to have DNSSEC authentication and takedown resistance. A misbehaving mirror is prevented from manipulating `.o` zone records as it does not have any DNSSEC private keys.

These techniques assist in preventing "pumping" queries that generate traffic over Tor in an attempt to deanonymize the root, and also assist with meeting Goal 5. Mirrors are chosen by the Root Administrator, likely through some public channel such as a mailing list with informal proposals by volunteer mirror operators. To ensure scalability and redundancy, these operators should be chosen from a range of volunteers, from smaller private advocates of Internet freedom to large organizations dedicated to freedom of speech (such as the EFF and ACLU). The anonymity of the root server is critical to meeting Goal 1, and while the architecture in Figure 1 meets this goal, it does not necessarily satisfy all of the OnionDNS goals. In particular, simply placing a DNS server within Tor does not make it easily usable.

OnionDNS mirrors may be configured to not only accept queries over the open Internet but additionally accept queries through Tor. A mirror may wish to provide this service for clients wishing to maintain the anonymity of their DNS requests. Additionally, mirrors concerned with being taken down can hide their own locations by servicing clients exclusively via Tor. This configuration comes with significant performance setbacks for clients, shown in Section 5, but does not affect the security of OnionDNS. We also note here that mirrors could be configured to provide extended DNS services such as client geolocation-based redirecting *if the requests arrive over the open Internet.* No such added service would be possible if the client's IP is anonymized through Tor.

OnionDNS does not provide nameserver hosting for individual domains. Instead, OnionDNS is structured like existing ICANN top-level domain servers and will only provide `NS` and `DS` records along with corresponding `A` and `AAAA` glue records. All other zone records must be returned by the domain's authoritative nameserver, operated and maintained by the domain owner.

3.2.4 Integrity and Keys

OnionDNS leverages DNSSEC [30] to allow users to verify the integrity of answers and meet Goal 3. The `.o` trust anchor and zone-signing key (ZSK) must be maintained independently by separate actors, described previously in Section 3.2.2. In the event of a suspected root server compromise, the holder of the trust anchor can assert a new ZSK owner by providing a signature for the new owner. Once the mirrors have migrated to the new root and the old ZSK has expired, the migration process is complete. The trust anchor must always be widely published and trusted by OnionDNS users.

Each record retrieved from an OnionDNS mirror can be checked for integrity using DNSSEC validation. A malicious or compromised mirror cannot seize a domain because it does not have either the trust anchor or the ZSK private keys. DNSSEC responses also prevent mirrors from claiming existing domains do not exist. This allows the mirrors to be similar to the existing DNS infrastructure today: well-known, easy to find, and fast. However, it is up to each client to perform DNSSEC verification on results.[2]

As is standard with DNSSEC, no integrity is provided for client requests or dropped packets.

3.2.5 Mirror Responses

While our solution has been focused on the integrity guarantees required to thwart many attacks, we must also consider attacks where malicious or subverted mirrors simply refuse to respond to queries for particular domains. For fully transparent operation (i.e., minimizing the impact of our system on usability), OnionDNS mirrors must be configured as a computer's DNS resolvers. In the case of a mirror that does not respond to a query, the user's computer should try any fallback resolvers that are configured in sequence until an honest mirror is found that will return the correct DNS record. Users should configure their fallback resolvers to include a geographically-diverse set of mirrors to avoid this type of attack. Note that while refusing records is a possible attack by a subverted mirror, modifying or delivering false records is not possible based on the authenticity guaranteed by DNSSEC. These two protections ensure that all requests for `.o` domains are handled correctly, while requests for other TLDs can be resolved by the appropriate root server using existing DNS forwarding mechanisms.

Clients and third-party services have the capability to perform monitoring of our system to ensure that the selected operators of the system act honestly. Besides regular monitoring via queries of the mirrors, notary services similar to Perspectives [73] could be implemented to perform ongoing historical analysis and assurance that records are handled correctly. Much like the community whistleblows on seizures on the open Internet, we expect that the community will also notice even subtle changes to domain records inside OnionDNS.

We note that the service, as designed, requires mirrors to function. Furthermore, to maintain availability, this set of mirrors must be large enough that A) all mirrors cannot be subverted or compelled to cooperate with a national entity, and B) the set of mirrors can handle the volume of DNS request traffic that could be placed on the entire `.o` TLD. If no mirror is available or has enough capacity to handle requests, the service has failed and clients will be unable to use it.

[2] OS support for end-to-end DNSSEC validation is growing and many public resolvers, such as Google and Comcast, currently perform DNSSEC validation.

3.3 Leek Goals

OnionDNS requires some registration process in order to populate its listing of domains and satisfy Goal 4. Accordingly, we designed the complementary "Leek" domain registration service with following goals in mind:

Goal 6. Land Rush Prevention: *Prevent a registrant from registering popular domains that should reasonably be expected to belong to the same owners as in other TLDs (such as `google.o`).*

Goal 7. Denial of Registration Prevention: *Prevent a registrant from registering as many available domains as possible, exhausting the domain space and denying other registrants from being able to register a domain.*

Goal 8. Contention Resolution: *Provide an auction-like environment for the (rare) case where multiple registrants are interested in the same domain within a waiting period. This deviates from the de facto first-come, first-served model of domain registration, but is required in OnionDNS for the following reasons:*

– *Network connectivity to hidden services is far from guaranteed, so it is difficult to identify the first to attempt to register the domain.*
– *This system permits multiple registrants to bid simultaneously, preventing a scenario where one registrant denies all other registrants the ability to register a domain by keeping the domain "tied up" in an auction.*

Goal 9. Blind Bidding: *Prevent a registrant from discerning if another player is also bidding on a particular domain.*

Similar to other registrars, Leek is *not* designed to enforce fairness or equality among registrants. It is designed to require interested parties to "bid" on domains, enforcing a real-world cost for OnionDNS domains. The player who is able to offer the greatest quantity of a selected resource has the highest probability of successfully registering the domain. We discuss potential options and our selection of mechanisms for our proof-of-concept implementation below.

3.4 Leek Design

While designing Leek, we surveyed multiple methods for performing domain registration. We note that the registration system is merely a tool for populating the data that OnionDNS serves, though it is critical to the usability of the service (Goal 2). *This is not an exhaustive or perfect list; individual deployments of OnionDNS may choose any combination of methods or design their own registration system.*

3.4.1 Monetary Compensation

This method is currently used by the registrars of public ICANN domains. Registrants must pay a recurring fee to the registrars in order to keep their domain registered.

Registering domains for OnionDNS via this method is troublesome. Money transfers with traditional currency are subject to government and industry regulation, and preventing transfers (and thus denying the registration) could be considered a form of domain seizure. Although digital currency like Bitcoin [55] may allow for privacy-preserving transactions [51], the authors believe that this security assertion is too new to be relied upon [11,40,15]. However, this method can make the process for registration simple and provides a financial incentive for the implementers to maintain the system.

3.4.2 Bootstrapping

An implementer may also choose to gather an existing list of Internet domains and pre-register those domains on behalf of their owners.

Bootstrapping prevents users from registering popular public Internet destinations (such as `google.o`) by making them unavailable for anyone else to register. Without this method, a land rush scenario for these domains is likely. If these new domains are unrelated to the official public sites (e.g., by phishing sites impersonating legitimate sites), user experience is likely to wane, preventing the successful deployment of the system.

However, this method has a number of disadvantages when implemented alone. Without any direct domain owner interaction, the registration must be maintained by the OnionDNS maintainers. Periodically scraping the Internet records would certainly propagate any domain seizure into OnionDNS.

3.4.3 Proof-of-work Submission

Rather than sending money for registrations, this method requires potential registrants to "spend" computation power and time. Proof-of-work puzzles have been actively studied and used in a variety of situations including preventing spam [29,25] and denial-of-service attacks [38,72]. While proof-of-work has significant problems in some systems [43], it has proven effective in other systems such as Bitcoin [55]. Unlike when proof-of-work is used to prevent spam, we believe participants in a domain registration proof-of-work system will be willing to dedicate computational resources for a significant amount of time in order to register a domain. A puzzle is a cryptographic challenge provided by the registration server, and its correct answer is necessary to complete the registration. Proof-of-work systems easily scale because they are difficult to solve but easy to verify.

Similar to spending money, requiring proof-of-work creates an opportunity cost for domains. With bounded computing power, a registrant must choose between potential registrations. Much like an auction, however, the submitter spending the most computation time on the puzzle has the highest odds of successfully completing the puzzle first.

If the difficulty is fixed over the domain space, an adversary with a tremendous amount of computing power (for example, with a custom ASIC) may choose to use the entirety of its power to move quickly from one domain to another. We described this attack earlier as "denial-of-registration." Fixed difficulty does not consider the varying capabilities of registrants; therefore, what is difficult and time-consuming for one registrant may not be for another. An adaptive scheme remedies this issue by attempting to force all registrants to use as much of their computing power as possible, requiring a registrant to focus on a much smaller number of domains at a time.

3.4.4 Other Methods

There are numerous other approaches or combinations of the above that could be applied to domain registration. In one example, Bogetaft et al. demonstrate a system using secure multiparty computation in a double auction [18]. We chose to implement a combination of bootstrapping and proof-of-work, which we explain below.

4 Implementation

We now discuss the specific tools, protocols and mechanisms used to realize our proof-of-concept versions of OnionDNS and Leek.

4.1 OnionDNS

Our implementation of OnionDNS is built upon the Tor anonymity network. We chose this network because it is widely used, supports hidden services, and has been heavily scrutinized by the security community.

Tor provides anonymous low-latency TCP streams called circuits, and is currently not capable of transmitting UDP flows, which are required for the DNS protocol suite. In order to use established DNS software without modifications, OnionDNS uses OnionCat [33]. Figure 2 shows how OnionDNS mirrors communicate

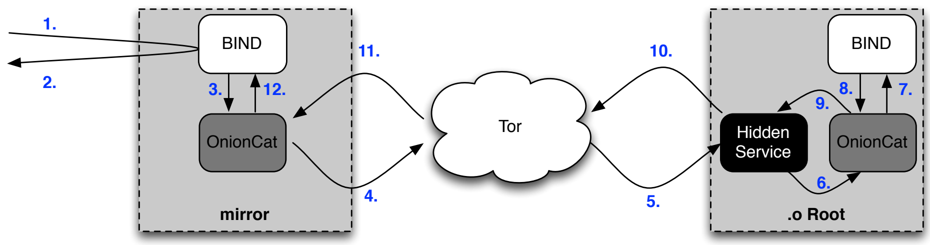

Fig. 2: The flow of a DNS request through the major components of our implementation of OnionDNS. An incoming client request (1) is received by the mirror and handled by BIND. (2) The request is satisfied from the contents of the zone file held in the mirror. Cache misses at the mirror receive an NXDOMAIN (non-existent domain) message or stale records. Client requests do not ever cause mirrors to query the root, as this would allow an adversary to pump huge amounts of traffic into Tor and potentially deanonymize the service. Periodically, (3) BIND requests an update via zone transfer from the root by sending a message to OnionCat, which encapsulates the IPv6 UDP request in TCP and (4) sends the message to the `.o` hidden service. (5) Tor delivers this message to the hidden service, (6) which then passes the request to OnionCat to remove the TCP headers. (7) The UDP DNS zone transfer request is then delivered to BIND running on the `.o` root, (8) which provides the appropriate response. (9) OnionCat and (10) the hidden service then appropriately package the response and return it to the mirror, (11) which subsequently unpacks the zone transfer and then (12) returns the response to the original BIND process.

with the root server and how clients communicate with mirrors. OnionCat is necessary because of its ability to both provide a standard addressing scheme and tunnel UDP traffic in Tor. Additionally, OnionCat's IPv6 addressing provides the IP addresses that are configured as approved zone transfer clients. Since these addresses are generated from Tor hidden service keys, an adversary will need the corresponding Tor key to impersonate an OnionCat address.

Unmodified BIND servers are used to host the `.o` zone on the root server and on the mirrors. The root server is configured to listen only on its OnionCat IPv6 address to prevent correlating its location by responding on the open Internet. BIND supports providing DNS service over TCP, however the server is configured to only provide service over UDP. Sending TCP traffic through OnionCat (which uses a TCP-based Tor hidden service) can cause TCP-over-TCP performance problems, and is easily avoided by using UDP.

For integrity verification, the `dnssec-enable` and `dnssec-validation` options are both enabled on all OnionDNS servers and clients.

4.1.1 Root Nameserver

The root nameserver is responsible for hosting the master copy of the `.o` zone file and for domain registration. The Root Administrator manages this server.

As long as anonymity holds for Tor's hidden services and the Root Administrator maintains the hidden service appropriately, the OnionDNS root server will be difficult to locate and subsequently censor.

4.1.2 Mirror Nameservers

The mirror nameservers host copies of the `.o` zone file and can be placed either inside or outside the Tor network. This enables requests to be directly fulfilled without the need for the request to enter the Tor network. The placement of the mirrors can be seen in Figure 1. Non-mirrors are forbidden from making requests from the root nameserver. Incremental, regular zone transfers from the root nameserver are performed over Tor.

Mirrors must be explicitly whitelisted to perform zone transfers with the root server, as described in Section 3. The root server uses the `allow-query` and `allow-transfer` BIND options to limit queries and zone transfers to the set of whitelisted mirrors.

4.2 Leek

We chose to implement a combination of bootstrapping and proof-of-work submission for domain registration. We present Leek, our implementation and deployment of a registration system for OnionDNS, in this section. We again stress that other deployments can select alternative mechanisms. Our implementation is tightly coupled to OnionDNS, however existing registrars (e.g., GoDaddy) could be outfitted to perform `.o` registrations with future work. Leek provides a core set of functions including domain registration, renewal, nameserver changes, and ownership transfer using a novel proof-of-work scheme.

As we will discuss in Section 8, the Leek server and the root hidden nameserver should be on separate physical servers accessible through separate hidden services. This prevents a vulnerability in Leek from having a direct impact on the root hidden service.

Definitions: The following definitions are used to describe our proof-of-work system:

– **Game:** the process of registering a domain
– **Player:** someone who wishes to register a domain
– **Difficulty:** number of preceding zero bits in a hash
– **Puzzle:** a per-round, per-game nonce
– **Active:** a property that requires a player to interact, in the form of network traffic and computational power, over the course of the game
– **Minimum difficulty:** a configurable difficulty set such that the computational cost is a sufficient burden when aggregated over the rounds
– **Waiting period:** a mandatory period of time before the game begins after which no new players may join
– **Round:** a segment of the game with a separate minimum difficulty, puzzle, and current score

4.3 Domain Bootstrapping

We implemented a simple bootstrapping method that retrieves the `NS` and `DS` records for the top one million domains provided by Alexa [9]. The domains in `.com` are converted into `.o`; all other top-level domains are converted into the corresponding `.tld.o` (e.g., `.org.o`, `.biz.o`, etc). For a domain owner to take control of their domain within OnionDNS, they must provide proof of ownership. For domains that are already DNSSEC-enabled, this is performed automatically by obtaining the existing `DS` upstream key record and using that key as the associated Leek public key. Using a standard digital signature with the corresponding private key, the domain owner can simultaneously demonstrate ownership of the domain and have their public key signed into the chain of trust rooted by the `.o` zone signing key. Domains not yet DNSSEC-enabled may demonstrate ownership by creating a new `TXT` record of the form `ODNS-VER=<DS>` that they can then associate with their existing DNS record. This technique is already commonly used in other applications to verify domain ownership, and represents the best possible method if no DNSSEC key pair has been created. In addition, Leek will continue scraping publicly-available domains beyond the top one million until the majority of the current ICANN-rooted domains are bootstrapped.

It is important to recognize that the OnionDNS zone signing key is only used to verify DNS records within the `.o` TLD that were registered using Leek or have been claimed using the technique described above. Our bootstrapping implementation does not provide protection for domain seizures for unclaimed domains. Specifically, bootstrapped domain records are continually synchronized with publicly-available data, including data which was modified as part of a seizure, until the domain owner claims the domain via a method discussed above. After a domain has been claimed, our bootstrapping implementation will stop maintaining the synchronization and the domain must be managed through Leek.

Bootstrapping is critical to OnionDNS as it prevents malicious early adopters from quickly registering popular domains that would otherwise not belong to them in a land rush. In particular, it satisfies Goal 6.

4.4 Domain Registration Game

Every attempt to register a domain proceeds as follows:

1. A user begins a domain registration. A mandatory waiting period is set to end at midnight UTC of the day of the request. This means if the request is made at 23:59 UTC, the game will start one minute later for that user. Any players who have not registered before midnight UTC will be unable to play. Each user generates a public/private keypair that will be used to identify the user through each round.
2. A set of rounds begin. In each round, each player receives an identical puzzle p and minimum difficulty value d_{min}:
   (a) Players locally set $d_{local} \leftarrow d_{min} - 1$.
   (b) The players begin to search for an answer and check the resulting difficulty d. A correct answer a to a round's puzzle p must satisfy the following:

   $$\mathsf{SHA512}(p \| a) \ \& \ \{1\}^d = 0,$$

   where $\{1\}^d$ represents a d bit string of all 1s and $\&$ is a bit-wise AND.
   (c) As soon as a player has computed an answer with $d > d_{local}$, it transmits the answer to the Leek server and sets $d_{local} \leftarrow d$. Each player submits its answer alongside its public key and a signature of the answer using the corresponding private key.
   (d) The Leek server verifies the player's answer and sets their score for the current round to d.
   (e) The players continue this process, attempting to increase d and obtain a higher score until the round is over.
3. After each round, the Leek server does the following:
   (a) Any players who did not submit a valid answer during the round are permanently removed from

the game. If no players submitted a valid answer, the game is terminated with no winner.
   (b) Each player's total score is updated by adding their maximum difficulty for the given round.
4. At the end of the game, the player with the highest total score wins. Ties are handled by randomly selecting among the winners. Upon successful registration, this public key becomes associated with the domain.

Goal 7 is accomplished by requiring a minimum difficulty, selected such that each domain has a sufficient computing cost to prevent widespread domain registration by a single player. Additionally, because of the active property and because the proofs are not precomputable, the player must return to the Leek server each round to retrieve a new puzzle and start work. All game start times are synchronized, further requiring players to prioritize their registrations.

Goal 8 is accomplished by having an adaptive difficulty where each player attempts to "beat" their current difficulty each round, allowing the player to play exactly to their maximum difficulty. This means that in the event of a game with two or more players, the player with the most computational power *dedicated to a single game* wins.

Goal 9 is accomplished by providing players uniform information about the game. All players receive the same puzzle and minimum difficulty value, and all players participate simultaneously in the current round. No function of the Leek registration game has an output to a player that results from another player's input. This also disincentivizes players from submitting a single answer at the minimum difficulty level each round and spreading their efforts across multiple games. The waiting period is configured so a player has no knowledge about whether another player has joined, because all games start at midnight UTC regardless. *Since each player knows nothing about whether or not there are other players in the game, a rational player is incentivized to spend as much power as possible towards a single game.*

4.5 User Identity

Domain ownership is maintained by a database of domains and associated RSA or DSA public keys. The use of these keys is two-fold. First, the public key is used by Leek to verify the signature of configuration commands submitted by clients (described later). Second, the public key is used in the generation of a `DS` record to verify the domain's ownership via DNSSEC. This requires domain owners to use their domain's `DNSKEY` key as their Leek identification key.

Using the public keys as owner identifiers obscures the physical identity of the domain owner, who may wish to not have his/her identity disclosed.[3] An owner may also register domains with different keys, preventing those domains from being attributed to the same owner. This privacy may be desired to promote fairness in a domain dispute or to resist data trending which may deanonymize the owner.

4.6 Domain Maintenance

Commands for domain maintenance in Leek are submitted as Base64-encoded strings along with a signature of the command string. Only two commands are available to users:
SETNS: This command allows the user to set the nameservers for the domain. For nameservers n_1 and n_2 with IPv4 or IPv6 addresses i_1 and i_2, respectively, the command is:

$$\mathsf{SETNS}\|n_1\|i_1\|n_2\|i_2.$$

SETKEY: Domain owners may choose to rotate their associated key as a best practice [41] using this command. Alternatively, it can be used to transfer ownership of a domain. For domain d, new public key U'_{PK}, and new private key U'_{SK}:

$$\mathsf{SETKEY}\|U'_{PK}\|\mathsf{Sign}(U'_{SK}, d).$$

In our implementation, the domain being configured is provided in the request URL (e.g., `/config/example.o`).

5 Experimental Evaluation

We configured an OnionDNS root and mirror on the Tor network as described in Section 3. The OnionDNS zone is a standard zone file, served using standard DNS software, so mirrors servicing clients over the open Internet have *equivalent performance* to any other DNS server. The work described in this section focuses on these communications occurring over the Tor anonymity network:

– Zone transfers between mirrors and the root
– Client requests to mirrors, for clients and mirrors choosing to communicate over Tor

[3] This type of privacy service is common among domain registrars, where a customer may be charged a service fee to obscure the domain's WHOIS information.

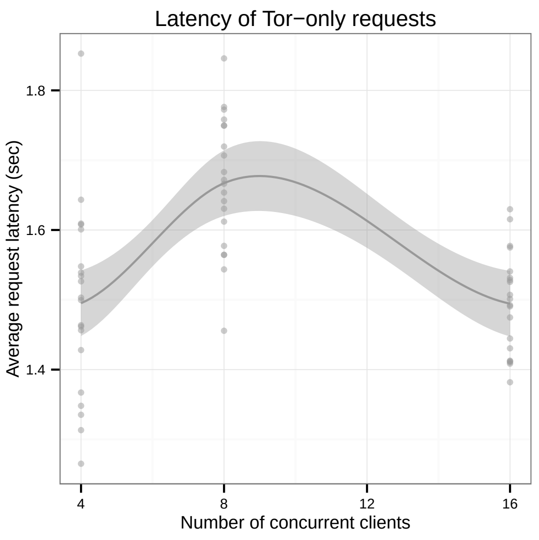

Fig. 3: The average latency of queries through the real-world Tor network. These queries were made to a mirror that was listening on an OnionCat-routable IPv6 address. Twenty trials of 4, 8, and 16 clients were performed.

We note here that while these Tor-based communications represent potential performance bottlenecks for some aspects of our system, we expect the vast majority of OnionDNS requests to occur between clients and the whitelisted mirrors *over the open Internet*. Since the only party required to remain anonymous is the root TLD, average users can enjoy reliable domain resolution without directly using Tor at all. Under these circumstances, the primary performance bottleneck of the system reduces to standard DNS query resolution with DNSSEC verification. Since the performance of DNSSEC is already a well-studied problem in [13,47,68] and others, we omit our own evaluation.

The root server's zone file was created by generating a list of randomly-named domain `NS` and associated glue records with random IPs. The number of domains generated was 10,000, but since the clients were configured to continually request domains, the total number was not important. The zone was signed then presented by BIND. The experimental mirror transferred the zone file from the root over Tor then listened for client connections. Zone transfers were fast over Tor, performing at an average of 482 kbps over 15 trials.

We passed a list to each client containing 10,000 valid domains from the generated zone and 1,000 invalid domains. To perform the queries and collect queries-per-second and query latency data, clients used Nominum DNSPerf. DNSPerf queried `NS` records for the provided domains in 120 second intervals. We collected twenty such intervals for each trial.

This OnionDNS mirror was configured to listen on the Tor network. We ran up to 16 clients against the mirror to acquire latency data. The average latencies from this experiment can be found in Figure 3. The figure shows the performance of OnionDNS queries from Tor clients to a mirror listening on Tor. The latencies are, on average, between 1 and 2 seconds with only 16 clients. These latencies may seem high, but since they are performed end-to-end through Tor, they are expected for a client. Additionally, the mirror supported over 1,000 queries per second during this experiment. Although OnionDNS mirrors must communicate with the root server via Tor, they are not required to service client queries over Tor. In the event that they do, a client should expect a higher front-loaded latency, depending on the client's Tor circuit.

6 Transferring Ownership of .o

One of the critical protections for this system is the inability to identify the administrators of this service. By publishing this paper, we directly invalidate this precondition and make ourselves the target of intimidation. However, before publication, we intend to relinquish control of this hidden service through the use of strong cryptographic constructions that allow us to obliviously delegate ownership.

To achieve such a delegation, Secure Function Evaluation (SFE) protocols allow parties to jointly compute some result while maintaining the privacy of each of their inputs. Moreover, many of these protocols allow for each party to receive a unique output, preserving the privacy of this output from other participants in the computation. While these protocols have been seen as largely impractical constructions, they have notably been used to solve simple real-world problems with strong cryptographic guarantees of privacy [18]. For our application, we would first turn the DNSSEC signing key over to a reputable foreign organization to delegate the key. We recommend this based on the legal analysis in Section 7, which showed that by adding an additional layer of protection by crossing legal jurisdiction boundaries, the ability of a national actor to compel an organization to reveal details about their takedown resistant TLD is severely restricted. In turn, the selected organization would input the signing key into an SFE computation, which would randomly select and output this key to a single recipient among a large set of possible recipients. After the protocol is run and the signing key securely deleted, the privacy guarantees of the computation would prevent us or the

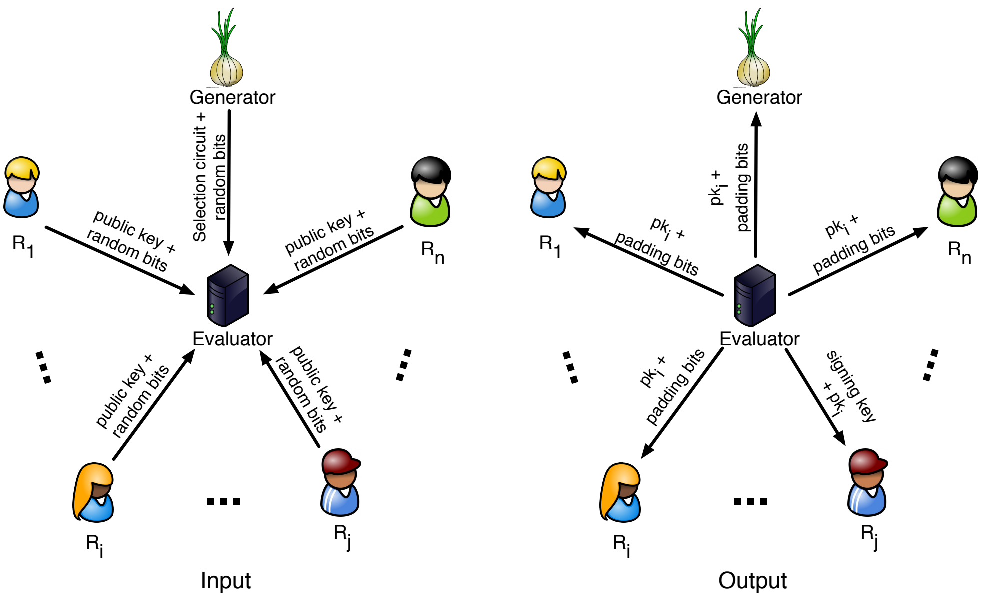

Fig. 4: The signing key delegation protocol. Essentially, all parties input their public keys and some random bits, which the evaluator uses to randomly and obliviously select both the Root Administrator (R_i) and the Key Owner (R_j). All outputs must be padded to the same length to prevent a passive adversary from observing which party receives the signing key. We select a foreign organization to play the role of the generator, inputting the selection circuit that will be evaluated.

foreign delegating organization from ever learning who actually received the signing key. The distributed nature of the protocol is key: not only does it guarantee that transcripts of the protocol will not reveal information about the administrators, but also obscures traffic patterns, removing the side channel attack of monitoring network accesses. This would remove any incentive for a national player to intimidate us into bringing down OnionDNS, and as we discuss later in Section 7, make it difficult for the foreign organization to be legally compelled to give up the list of possible recipients.

6.1 Delegation Protocol

The protocol execution involves n candidates, one of whom will receive the OnionDNS signing key while a second will be chosen as the Root Administrator. The computation also requires one party who will generate the garbled circuits used in the transaction, and one party who will evaluate the garbled circuit. For the remainder of this section, we refer to the candidates to receive the key as "recipients", the delegating body as the "generator", and the evaluating party as the "evaluator" (see Figure 4 for a high-level protocol diagram). The recipients should be composed of a number of large organizations that actively promote freedom of speech, like (but not necessarily including) the EFF and the ACLU, as well as other small organizations from within the privacy research community. To minimize any one government's ability to track down and intimidate individual members of these organizations, we will also choose groups that are based internationally. Each organization contacted will be responsible for selecting a set of members who will participate in the computation but whose identities will be unknown to us or the delegating organization. Having each organization select internal members will provide some level of vetting to ensure that the participants in the computation are reliable enough to manage the system correctly and maintain anonymity. This is critical since there is no SFE protocol that can identify a government-placed honeypot during the computation. If a government actor were to participate in the transfer protocol, there would be some probability that the system could be compromised from the inside. Thus, we must rely on internal vetting and the probabilistic nature of the selection to keep the keys from a malicious party.

Once the n recipients are selected, we prepare to run the SFE protocol. For this application, we choose to run a protocol that is secure in the semi-honest model. This threat model provides the necessary security guarantees for our application and runs in a practical amount of time. When considering malicious model attacks on garbled circuit protocols run for this particular application, the only party that can corrupt the computation without being caught is the generator. Thus, since the generator *must* behave honestly, we will carefully select the foreign organization to play this role in the computation. Since our goal is to maintain the anonymity of the party managing OnionDNS, we have incentive to choose a reliable organization and would ensure they securely delete any protocol transcripts that could later be used to reconstruct the list of possible recipients. We note that it is possible to run this computation using a maliciously secure protocol similar to the outsourcing protocols developed by Kamara et al. [39] and Carter et al. [20], but these provide unnecessary security guarantees at the cost of significantly longer execution times.

We run the garbled circuit SFE protocol as follows:

1. The generator will produce a garbled version of the "selection circuit." This circuit will take the DNSSEC signing key from the generator, and a concatenation of public keys and random strings from the possible recipients as inputs. The circuit will output the signing key to exactly one recipient (the Key Owner). All of the recipients will receive a copy of one recipient's public key, which is to be the Root Administrator's public key. This key can then be signed by the Key Owner for verifying DNSSEC requests to OnionDNS.
2. The generator will generate a random string c_0 that will be used in the selection process. We then concatenate this value with the OnionDNS signing key sk, producing our input value $s = c_0||sk$. For each $i \in \{1, \dots, n\}$, the recipient R_i will generate a public key pair sk_i, pk_i; a random string c_i as input to the selection circuit; and a one-time pad p_i. The one-time pad will be XORed with recipient R_i's output within the circuit so that the evaluating party cannot see what each party's output is. The input for R_i is the concatenation $r_i = pk_i||c_i||p_i$.
3. The generator will then execute oblivious transfers with each of the recipients to convert their real inputs to the circuit into garbled inputs. By the guarantees of oblivious transfers, the generator can learn nothing about any party's input from this operation, and each party will receive a garbled version of their input that is indistinguishable from randomness to any other party.
4. The generator will send the garbled version of the selection circuit to the evaluator, and each recipient will send their garbled input to the evaluator.
5. The evaluator then evaluates the garbled circuit, which will produce n output values. Recall that these output values are masked with the one-time pads provided by each recipient as a part of their input.
6. The evaluator delivers the blinded outputs to the recipients, who can recover their respective output by XORing their one-time pad p_i with the received output. Based on the circuit construction, all parties will receive the Root Administrator's public key, while one of these outputs will also contain the signing key.
7. After the Key Owner and Root Administrator have been chosen, the delegating organization will sign the Key Owner's public key and post this signed key as the trusted root for the OnionDNS.

6.2 Selection Circuit

The goal of the selection circuit is to choose exactly one recipient in a uniformly random fashion to receive the OnionDNS signing key. To accomplish this, the circuit should take two steps. First, the circuit should generate a uniformly random number based on the inputs of all parties. Ideally, to accomplish this step, all of the participants would provide input to a coin-flipping protocol. Such a protocol would produce a uniformly distributed random value that can only be biased a negligible amount by any of the players' input strings. However, since these protocols are interactive, they cannot be evaluated in a garbled circuit. Because we are assuming that all parties are behaving semi-honestly, we can fall back to a simpler solution. Informally, we allow each party to input a random bit string of length $2n-3$ and take the XOR of all of these values. Essentially, if any one player inputs a string that is uniformly distributed, then this string $c_i \oplus m$, where $m = \bigoplus_{j=0, j\neq i}^{n} c_j$, is statistically indistinguishable from a uniform distribution, following from the security proof of the one-time pad.

The second step is to interpret the resulting string as selection bits for the output wire identifiers id_O, id_A for which recipient will be the Key Owner and which will be the Root Administrator. To do this, we execute two tournaments, which can be easily represented in boolean circuits as a tree structure, with the leaves representing each of the n players. We then use the first $n-1$ bits to determine the "winner" of each round at every parent node in the tree, such that the left child wins for bit "0" and the right child for bit "1". After $n-1$ rounds, the remaining player will be the winner and will be designated id_O, or Key Owner. Using the remaining $n-2$ bits and excluding id_O, we repeat the tournament to select id_A, or Root Administrator. We note here that it could be possible to perform the actual signing of public key pk_j with sk inside of the garbled circuit. However, garbled circuits evaluating functions like modular exponentiation with 1024-bit numbers are only now becoming feasible in practice. For this application, we get more security from having the Key Owner sign pk_j outside of the circuit with a stronger signing key than 1024-bits.

7 Legal Analysis

"If the king's writ reaches only so far as the king's sword, then much of the content on the Internet might be presumed to be free from the regulation of any particular sovereign."

- James Boyle, *Foucalt in Cyberspace: Surveillance, Sovereignty, and Hardwired Censors [19]*

The U.S. government has asserted broad authority to seize domain names. The government claims the right to seize any domain name managed by a U.S. company, regardless of where the domain name owner is located [4,42]. Thus, any domain name with `.com` or `.net` is subject to seizure by the U.S. government because VeriSign, a U.S. corporation, manages them. The U.S. Congress has also passed a law that prohibits the distribution of technologies designed to circumvent copyright law [67], and, as legal scholar Derek Bambauer has explained [16], Congress could pass a similar ban to prohibit the distribution of technologies to circumvent domain name seizures, though such a ban would not be perfectly enforceable.

Nonetheless, domain name seizures are difficult to enforce when the corporation that manages the domain is located exclusively outside the U.S., because the U.S. can only enforce its law with the cooperation of the foreign government [35]. For example, the U.S. must obtain the consent of the foreign government to serve a subpoena [45,53]. If the foreign government chooses not to cooperate, the targeted websites remain largely outside the reach of U.S. law enforcement [21].

A recent domain name seizure cases illustrates the difficulties the U.S. faces in compelling foreign companies or organizations to shut down domain names [3]. The case involved the seizure of the domain names `rojadirecta.com` and `rojadirecta.org` on the grounds that they had been used to commit criminal violation of U.S. copyright law. The domain name owner, a Spanish company, contested the seizure in federal district court, arguing that the seizure created a substantial hardship by depriving it of business "in the United States and throughout a substantial part of the world." The U.S. government countered this argument by showing that the practical effects of its domain name seizure were limited: although Rojadirecta initially experienced a 32 percent reduction in traffic, the domain name owner transferred the websites to alternative domains, such as `www.rojadirecta.me`, `www.rojadirecta.es`, and `www.rojadirecta.in`, which the government conceded were beyond its jurisdiction. The federal district judge therefore held that the seizure was not a substantial hardship for the Spanish company, observing that "[t]he United States Government cannot seize these foreign domain names, but United States residents can access them without restriction." The judge further noted that Rojadirecta could redirect users to its new domain names simply by leveraging its "large internet presence" and "distribut[ing] information about the seizure and its new domain name to its customers."

Our solution introduces serious practical obstacles to identifying the operators of the service. First, the OnionDNS root services exists solely inside an anonymity network, which is well-studied and maintained. Second, our secure transfer protocol ensures only one party from a large, global list of invited organizations receives the OnionDNS key. To eliminate the possibility that law enforcement or administrative agencies might seek to compel the authors to provide the list of participants in the secure transfer, we will select a single, reputable, foreign organization to publicly execute the secure transfer protocol. In doing so, the authors will not have information about the potential operators of the system and can divulge nothing. Moreover, even if the government were able to obtain the identity of participants in the transfer, those participants operate completely outside the borders of the United States. Thus, the government would need to first overcome the hurdle of identifying where the system is operated, then cooperate with foreign governments to serve a subpoena. Practically speaking, our solution curtails domain seizure as a tool of censorship.

8 Deployment Challenges

8.1 SSL/TLS Certificates

Domain registrants using a `.o` domain alongside their open Internet domain (as would be the case with bootstrapped domains) may face a certificate name mapping problem. In particular, these sites will need to use one of the following methods to address this issue:

- Use a multiple-domain certificate to list the `.o` address alongside the open Internet addresses.
- Support the Server Name Indication TLS extension [8] to present an alternate certificate in response to client requests containing their intended destination.
- Operate the `.o` address with a separate IP address and present a separate certificate on that address.

8.2 OnionDNS and Web Service Separation

Our implementation of OnionDNS includes the Leek web service to interact with users. Although not technically required, it is advantageous to separate the registration interface from the root server and implement a third actor in the system. We whitelist the mirrors, preventing clients from directly making DNS queries to the OnionDNS root server. This forces adoption of the mirror system and allows the system to scale horizontally without needing the Root Administrator to provide an unreasonable amount of system resources. The registration system, however, needs to provide open access over Tor.

Because the registration directly interfaces with end users, it is important that any vulnerability of the web service be abstracted from the root server to prevent unauthorized changes to the root zone. Future development on OnionDNS may include a third actor, the *registrar operator* to run the registration service. It is possible that the service could support multiple registrar operators in a similar way as Internet domain registrations. Only the approved, whitelisted OnionDNS registrars would be permitted to push changes to the root zone.

8.3 Legitimate Delisting of Domains

Like the majority of information security technologies, there exists the potential to use the OnionDNS architecture to support illegal operations. For instance, the actions taken to legally combat botnets often include domain seizures. The owner of a botnet could potentially use their own implementation of OnionDNS to ensure the robustness of their domain resolution requests.

There is nothing about OnionDNS that prevents the administrator of an implementation from performing as a good actor. For instance, legal botnet takedowns are generally preceded by a court order in the proper jurisdictions [48–50]. The administrator of the `.o` TLD could delist such domains as long as the court order is specific and made public. Likewise, an administrator may choose to provide an arbitration method for trademark/name disputes and squatting.

Making such policy public allows for other organizations (e.g., the EFF, the ACLU, universities, etc.) to justify running of one of the mirror resolvers. An adversary trying to maintain an independent OnionDNS TLD is unlikely to receive support from such institutions, and may still be subject to attack should that implementation's small number of mirror resolvers be directly targeted.

9 Conclusion

The Domain Name System is a critical resource to nearly every networked application. As such, it is also an attractive choke point for those seeking to quickly make such applications unreachable. While such delisting can be the result of legal action, recent legislation including SOPA and PIPA seek to make such delisting possible without due process. We address this problem through the creation of OnionDNS, an anonymous top-level domain and resolution service for the Internet. Our solution establishes a hidden service running within Tor, and uses redundant public mirrors to ensure a high-performance architecture that is resistant to domain takedowns. Moreover, our approach does not require end users to install software, making it easily adoptable by virtually any platform. Finally, we take steps to ensure that the new TLD is safely and obliviously handed to another party. As a result, we demonstrate that the delisting of domains from DNS can be efficiently and securely mitigated.

Acknowledgements The authors would like to thank an anonymous contributor for inspiration and assistance with the development of this system. This work was supported in part by the US National Science Foundation under grant number CNS-1464088. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

1. Namecoin was stillborn, I had to switch off life-support. `https://bitcointalk.org/index.php?topic=310954` (archived at `http://www.webcitation.org/6KXauX8uC`).
2. The List Of Sites Challenging Domain Seizures. `http://www.techdirt.com/articles/20110612/21573514664/list-sites-challenging-domain-seizures.shtml`.
3. Order, puerto 80 projects, s.l.u. v united states. `https://www.eff.org/files/rojadirectaorder.pdf`, 2011.
4. Testimony of John Morton, Director, U.S. Immigration and Customs Enforcement, Before the U.S. House of Representatives Committee on the Judiciary, Subcommittee on Intellectual Property, Competition and the Internet on "Promoting Investment and Protecting Commerce Online: Legitimate Sites v. Parasites, Part II". `http://www.dhs.gov/news/2011/04/05/testimony-john-morton-director-us-immigration-and-customs-enforcement-promoting`, 2011.
5. Namecoin, 2015. `http://namecoin.info/`.
6. 112th Congress of the United States of America. H.R. 3261 - Stop Online Piracy (SOPA) Act. `http://thomas.loc.gov/cgi-bin/query/z?c112:H.R.3261:`, 2011.
7. 112th Congress of the United States of America. Senate Bill 986 - Preventing Real Online Threats to Economic Creativity and Theft of Intellectual Property Act (PIPA). `http://thomas.loc.gov/cgi-bin/query/z?c112:S.968:`, 2011.

8. D. E. 3rd. Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066 (Proposed Standard), Jan. 2011.
9. Alexa. Top 1,000,000 Sites. `http://s3.amazonaws.com/alexa-static/top-1m.csv.zip`.
10. R. Anderson et al. The eternity service. In *Pragocrypt96*, pages 242–252, 1996.
11. E. Androulaki, G. Karame, M. Roeschlin, T. Scherer, and S. Capkun. Evaluating User Privacy in Bitcoin. *IACR Cryptology ePrint Archive*, 2012:596, 2012.
12. Anonymous. The collateral damage of internet censorship by DNS injection. *ACM SIGCOMM Computer Communication Review*, 42(3), 2012.
13. Asia Pacific Network Information Centre Labs. Measuring dnssec performance. `https://labs.apnic.net/?p=341`, 2013.
14. B. Awerbuch and C. Scheideler. Group spreading: A protocol for provably secure distributed name service. *Automata, Languages and Programming*, pages 187–210, 2004.
15. M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar. On bitcoin and red balloons. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 56–73. ACM, 2012.
16. D. E. Bambauer. Orwell's armchair. *Univ. Chic. Law Rev.*, 79(3):863–944, 2012.
17. A. Biryukov, I. Pustogarov, and R.-P. Weinmann. Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization. 2013.
18. P. Bogetoft, D. L. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. D. Nielsen, J. B. Nielsen, K. Nielsen, J. Pagter, et al. Secure multiparty computation goes live. In *Financial Cryptography and Data Security*, pages 325–343. Springer, 2009.
19. J. Boyle. Foucault in cyberspace: Surveillance, sovereignty, and hardwired censors. *Univ. Cincinnati Law Rev.*, 66:177, 1997.
20. H. Carter, B. Mood, P. Traynor, and K. Butler. Secure Outsourced Garbled Circuit Evaluation for Mobile Devices. In *Proceedings of the USENIX Security Symposium*, 2013.
21. A. Chaitovitz, C. Hampton, K. Rosenbaum, A. Salem, T. Stoll, and A. Tramposch. Responding to online piracy: Mapping the legal and policy boundaries. *CommLaw Conspectus*, 20(1), 2012.
22. D. R. Cheriton and T. P. Mann. Decentralizing a global naming service for improved performance and fault tolerance. *ACM Transactions on Computer Systems (TOCS)*, 7(2):147–183, 1989.
23. I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies*, pages 46–66. Springer, 2001.
24. R. Cox, A. Muthitacharoen, and R. Morris. Serving DNS using a peer-to-peer lookup service. *Peer-to-Peer Systems*, pages 155–165, 2002.
25. L. F. Cranor and B. A. LaMacchia. Spam! *Communications of the ACM*, 41(8):74–83, 1998.
26. R. Dingledine. Obfsproxy: the next step in the censorship arms race. *Tor Project official blog*, 2012.
27. R. Dingledine, M. Freedman, and D. Molnar. The free haven project: Distributed anonymous storage service. In *Designing Privacy Enhancing Technologies*, pages 67–95. Springer, 2001.
28. R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.
29. C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology*, pages 139–147. Springer, 1993.
30. D. E. Eastlake et al. Domain name system security extensions. 1999.
31. Electronic Frontier Foundation. Anti-Counterfeiting Trade Agreement (ACTA). `https://www.eff.org/issues/acta`, 2012.
32. N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger. Infranet: Circumventing web censorship and surveillance. In *Proceedings of the 11th USENIX Security Symposium*, pages 247–262. San Francisco, CA, 2002.
33. B. R. Fischer. OnionCat: A Tor-based Anonymous VPN. In *Proceedings of the 25th Chaos Communication Congress*, 2008.
34. A. M. Froomkin. Wrong Turn in Cyberspace: Using ICANN to Route around the APA and the Constitution. *Duke Law Journal*, 50(1), 2000.
35. L. Henkin. *Restatement of the Law, Third: The Foreign Relations Law of the United States.* American Law Institute-American Bar Association (ALI-ABA), 1987.
36. Internet Systems Consortium. `https://www.isc.org/downloads/bind/`.
37. A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson. Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2013.
38. A. Juels and J. G. Brainard. Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks. In *NDSS*, volume 99, pages 151–165, 1999.
39. S. Kamara, P. Mohassel, and B. Riva. Salus: A system for server-aided secure function evaluation. In *Proceedings of the ACM conference on Computer and communications security (CCS)*, 2012.
40. G. Karame, E. Androulaki, and S. Capkun. Two Bitcoins at the Price of One? Double-Spending Attacks on Fast Payments in Bitcoin. *IACR Cryptology ePrint Archive*, 2012:248, 2012.
41. O. Kolkman and M. Gieben. RFC 4161 DNSSEC Operational Practices, 2006.
42. K. Kopel. Operation seizing our sites: How the federal government is taking domain names without prior notice. *Berkeley Technology Law Journal*, 28, 2013.
43. B. Laurie and R. Clayton. "Proof-of-Work" proves not to work; version 0.2. In *Workshop on Economics and Information, Security*, 2004.
44. T. B. Lee. ICE admits year-long seizure of music blog was a mistake. `http://arstechnica.com/tech-policy/2011/12/ice-admits-months-long-seizure-of-music-blog-was-a-mistake/`, 2011.
45. F. A. Mann. The doctrine of international jurisdiction revisited after twenty years, 1984.
46. C. D. V. Mestdagh and R. W. Rijgersberg. Rethinking accountability in cyberspace: a new perspective on ICANN. *International Review of Law, Computers and Technology*, 21(1):27–38, 2007.
47. Microsoft. Dnssec performance considerations. `https://technet.microsoft.com/en-us/library/dn593667(v=ws.11).aspx`, 2014.
48. Microsoft Corporation. Microsoft Corporation v. Dominique Alexander Piatti; Jone Does 1-22. 2011. Virginia Eastern District Court.
49. Microsoft Corporation. Microsoft Corporation v. Peng Yong et. al. 2012. Virginia Eastern District Court.

50. Microsoft Corporation. Microsoft v. John Does 1-39. 2012. New York Eastern District Court.
51. I. Miers, C. Garman, M. Green, and A. D. Rubin. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. In *IEEE Symposium on Security and Privacy*, 2013.
52. P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov. Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 215–226. ACM, 2011.
53. S. S. Mody. National cyberspace regulation: Unbundling the concept of jurisdiction. *Stan. J. Int'l L.*, 37:365, 2001.
54. Y. Nadji, M. Antonakakis, R. Perdisci, D. Dagon, and W. Lee. Beheading Hydras: Performing Effective Botnet Takedowns. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2013.
55. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1:2012, 2008.
56. R. Naraine. Massive DDoS Attack Hit DNS Root Servers. `http://www.internetnews.com/dev-news/article.php/1486981`, 2002.
57. L. Overlier and P. Syverson. Locating hidden servers. In *Security and Privacy, 2006 IEEE Symposium on*, pages 15–pp. IEEE, 2006.
58. M. Panzarino. Syrian Electronic Army Apparently Hacks DNS Records Of Twitter, NYT Through Registrar Melbourne IT. `http://techcrunch.com/2013/08/27/syrian-electronic-army-apparently-hacks-dns-records-of-twitter-new-york-times-through-registrar-melboune-it/`, 2013.
59. V. Pappas, D. Massey, a. Terzis, and L. Zhang. A Comparative Study of the DNS Design with DHT-Based Alternatives. *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–13, 2006.
60. K. Park, V. Pai, L. Peterson, and Z. Wang. CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups. *OSDI*, pages 199–214, 2004.
61. D. Piscitello. Anatomy of a DNS DDoS Amplification Attack. `http://www.watchguard.com/infocenter/editorial/41649.asp`, 2011.
62. L. Poole and V. Pai. ConfiDNS: Leveraging scale and history to improve DNS security. *Proceedings of WORLDS*, 2006.
63. V. Ramasubramanian and E. Sirer. The design and implementation of a next generation name service for the internet. *ACM SIGCOMM Computer Communication Review*, page 331, 2004.
64. N. Scaife, H. Carter, and P. Traynor. OnionDNS: A Seizure-Resistant Top-Level Domain. In *Proceedings of the IEEE Conference on Communications and Network Security (CNS)*, 2015.
65. Y. Song and K. Koyanagi. Study on a hybrid P2P based DNS. *2011 IEEE International Conference on Computer Science and Automation Engineering*, pages 152–155, June 2011.
66. TorrentFreak. U.S. Government Shuts Down 84,000 Websites, 'By Mistake'. `http://torrentfreak.com/u-s-government-shuts-down-84000-websites-by-mistake-110216/`.
67. U.S. Copyright Office. Circumvention of copyright protection systems. `http://copyright.gov/title17/92chap12.html`, 2015.
68. R. van Rijswijk-Deij, A. Sperotto, and A. Pras. Making the case for elliptic curves in dnssec. *SIGCOMM Computer Communication Review*, 45(5):13–19, Sept. 2015.
69. M. Waldman and D. Mazieres. Tangler: a censorship-resistant publishing system based on document entanglements. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 126–135. ACM, 2001.
70. M. Waldman, A. D. Rubin, and L. F. Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *9th USENIX Security Symposium*, pages 59–72, 2000.
71. Q. Wang, X. Gong, G. T. Nguyen, A. Houmansadr, and N. Borisov. CensorSpoofer: asymmetric communication using IP spoofing for censorship-resistant web browsing. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 121–132. ACM, 2012.
72. X. Wang and M. K. Reiter. Defending against denial-of-service attacks with puzzle auctions. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 78–92. IEEE, 2003.
73. D. Wendlandt, D. G. Andersen, and A. Perrig. Perspectives: Improving SSH-style host authentication with Multi-Path probing. In *USENIX Annual Technical Conference*, pages 321–334. static.usenix.org, 2008.
74. D. Yadron. Syrian Electronic Army's Alleged Attacks Expose Soft Spot. `http://online.wsj.com/news/articles/SB10001424127887324009304579040900023429122`, 2013.