OnionDNS: A Seizure-Resistant Top-Level Domain

Nolen Scaife
University of Florida
scaife@ufl.edu

Henry Carter
Georgia Institute of Technology
carterh@gatech.edu

Patrick Traynor
University of Florida
traynor@cise.ufl.edu

***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 enable a high-performance architecture with strong integrity guarantees for resolved records. After discussing the details of our DNS architecture, we present our anonymous domain registrar and detail the protocol for securely transferring the service to another party. We also conduct a performance analysis demonstrating the service is fast with an average request latency between 1 and 2 seconds over Tor. In doing so, we demonstrate that the delisting of domains from DNS can be mitigated in an efficient and secure manner.**

I. 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 [43], [48]. 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 [26], the Stop Online Piracy Act (SOPA) [4] and the PROTECT IP Act (PIPA) [5] allow for the potential delisting of such targets from DNS by third parties. Moreover, a number of other nation-states aggressively modify DNS [9], [57], 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" [3], [37]. 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 [52].

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 [30] 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:

- **Design 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 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 implementation of OnionDNS.
- **Implement and characterize domain registrar:** A practical architecture for resisting domain seizure requires more than establishing a hidden service. Accordingly, we also address the challenges of domain registration, renewal, and transfer with our domain registrar, Leek.
- **Secure transfer of OnionDNS service:** We discuss the difficulty of obliviously transferring our hidden service and present a protocol for handing off control to another party, which we plan to do after publication.

We note that OnionDNS is not intended as 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* [41]. Moreover, our approach does not require end users to install Tor, instead allowing 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 all of their traffic through an anonymity network.

The remainder of the paper is organized as follows: In Section II, we present related work in DNS censorship. Section III describes the design and details the implementation of our system. In Section IV, we present our performance analysis. Section V discusses our protocol to securely hand off control of the `.o` TLD. Finally, in Section VI, we conclude.

II. 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) [19]. This approach has been further studied to improve latency of queries and robustness against denial-of-service (DoS) attacks [47], [49], [50], [51], [10]. 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 [46].

Furthermore, centralized control over the Internet naming infrastructure has been disputed [29], [17], in part because centralization provides powerful entities the ability to perform domain seizures. Mestdagh et al. [38] 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 [1] 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 [42]. 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 [2].

Broader approaches to censorship resistance include early work on Publius [54], Tangler [53], and Free Haven [22], which are all based on the theoretical motivations from Anderson's Eternity Service [7]. 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 [18], [27], [55], [21], and several systems, such as Tor [23] have seen widespread adoption. However, these systems are challenging to use and integration with existing software requires significant modifications. By tackling a specific form of censorship, OnionDNS provides 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.

III. Design and Implementation

The main goal of OnionDNS is to resist global domain seizures. We accomplish this by hiding a trusted TLD authoritative nameserver via an anonymity network. This prevents any single entity from pressuring the trusted server to change DNS records. The resource records from this top-level domain's zone are mirrored through public mirrors, allowing seizure-resistant domains to scale like 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 describes OnionDNS and our complementary domain registrar, Leek.

A. Adversaries

The owner of a TLD 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. These adversaries include:

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 [45]. 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 malicious mirror in order to prevent clients from resolving a domain.

Denial-of-registration: An adversary may register domains in a land rush to block legitimate users.

B. Design Goals

The design of a large, available, and hidden DNS service is not straightforward. We have identified five goals to ensure that OnionDNS provides comprehensive, secure functionality:

- **Seizure-Resistance**: When operated by a trusted actor, the system must be resistant to global domain seizures.
- **Usability**: Users must not have to significantly modify their systems in order to use the OnionDNS domain.
- **Integrity**: The service must provide users a facility to verify the integrity of received answers.
- **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.
- **Accessibility**: The service must be easily accessible, but the root must not be easily geo-locatable.

Furthermore, to secure the process of administering OnionDNS domain records, our registrar must adhere to the following:

- **Land Rush Prevention:** Prevent a registrant from registering all popular domains.
- **Denial of Registration Prevention:** Prevent a registrant from registering as many available domains as possible, exhausting the domain space.
- **Contention Resolution:** Provide an auction-like environment for the (rare) case where multiple registrants request the same domain within a waiting period. This deviates from the first-come, first-served model of domain registration, but is required here 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.
- **Blind Bidding:** Prevent a registrant from discerning if another player is also bidding on a particular domain.

C. Roles and Responsibilities

To implement these goals, two independent, anonymous actors operate OnionDNS:

- **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.

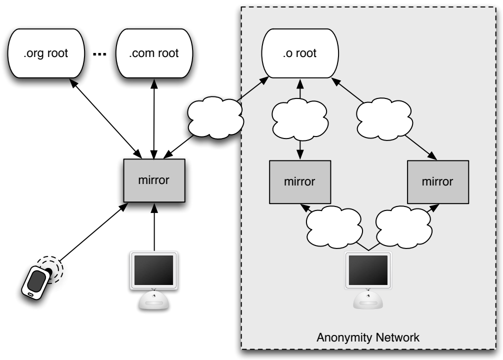

Fig. 1: The OnionDNS architecture. Connections through clouds indicate anonymity networks. Clients make requests to OnionDNS mirrors through the open Internet or through Tor. The mirrors retrieve the `.o` zone from the hidden root server. A client request never directly results in traffic at the root server, and mirrors need not be trusted due to the use of DNSSEC.

D. Core Infrastructure

We introduce `.o` as the OnionDNS TLD.[1] This domain was chosen because it does not conflict with another Internet TLD; in addition, introducing a new root domain (`.`) would break interoperability with other DNS services. The choice of name is ultimately arbitrary, and a number of different OnionDNS implementations could be run in parallel.

Figure 1 shows the high-level OnionDNS architecture. For accessibility, mirrors are used to serve requests directly to users through the open Internet and Tor. The root server is responsible for hosting the master copy of the `.o` zone file. It accepts queries and zone transfer requests only from a set of whitelisted mirrors. 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.

DNS is a well-established and proven service and attempting to modify it or replace it would conflict with our usability goal. We instead rely on the use of Tor to hide the OnionDNS root server's physical location. The anonymity of the service is critical to providing seizure-resistance, and while the architecture in Figure 1 meets this goal, it does not satisfy all of the OnionDNS goals.

Mirrors host copies of the `.o` zone file and perform regular zone transfers from the root nameserver over Tor. These mirrors may choose to serve requests over the open Internet, Tor, or both. A mirror may provide Tor service for clients wishing to anonymize their DNS requests. Additionally, mirrors concerned with being taken down can hide their locations by servicing clients exclusively via Tor.

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

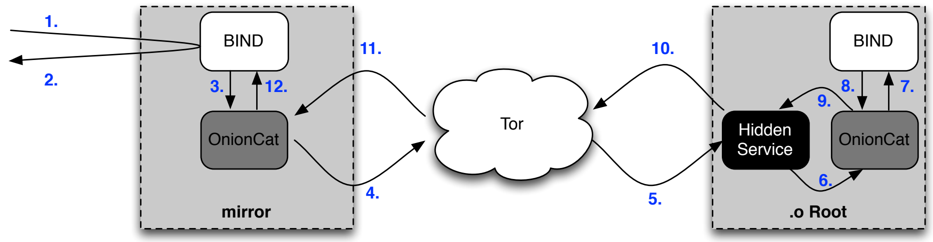

Fig. 2: The flow of a DNS request in OnionDNS. An incoming client request (1) is received by the mirror and handled by BIND. (2) The request is satisfied from the contents of the zone file held in the mirror. Cache misses at the mirror receive an NXDOMAIN (non-existent domain) message or stale records. Client requests never cause mirrors to query the root, as this would allow an adversary to pump 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 package the response and return it to the mirror, (11) which unpacks the zone transfer and (12) returns the response to the original BIND process.

In order to use BIND unmodified, OnionDNS uses OnionCat [28]. Figure 2 shows how OnionDNS mirrors communicate with the root server and how clients communicate with mirrors. OnionCat provides both a standard addressing scheme and the ability to tunnel UDP traffic in Tor. Its IPv6 address generation provides the IP addresses that are configured as approved zone transfer clients. Since these addresses are created from Tor hidden service keys, an adversary will need the corresponding Tor key to impersonate an OnionCat address.

To prevent impersonation of DNS records, OnionDNS leverages DNSSEC [25] to allow users to verify the integrity of answers (including answers claiming a domain does not exist). In our dual-control system, the holder of the trust anchor can assert a new Root Administrator by providing a signature for the new party without user intervention. Each record retrieved from an OnionDNS mirror can be checked for integrity using DNSSEC validation. Clients can easily detect manipulated `.o` zone records since mirrors do not possess the DNSSEC private keys. However, it the responsibility of each client to perform DNSSEC verification on results.[2]

These records are not the complete host records for individual domains. OnionDNS does not provide nameserver hosting. Instead, OnionDNS is structured like existing ICANN TLD 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.

Our comprehensive approach comes with many advantages: DNS is proven to scale, popular DNS server software is well-maintained, and support for using DNS is built into nearly every networked computing device. This ensures OnionDNS users can quickly and easily perform lookups while maintaining the anonymity of the service's operators.

Security of Hidden Services. There have been many proposed attacks against hidden services in anonymization systems [31], [12], [40], [44]. However, Tor has had several high-profile hidden services running within its network [12] that the authors do not believe have been fully deanonymized. In particular, it is widely believed that both of these takedowns deanonymized the hidden service without breaking the anonymity guarantees provided by Tor. Our dual-control policy ensures that any misbehaving or compromised Root Administrator will only have a valid trusted key for a short period of time.

[2]OS support for end-to-end DNSSEC validation is growing and many public resolvers, such as Google and Comcast, currently perform DNSSEC validation. As is standard with DNSSEC, no integrity is provided for client requests or dropped packets.

E. *Leek, the Anonymous Domain Registrar*

A domain name server without content to deliver provides no value to its users, so our solution is incomplete without a domain registration service. In this section, we introduce Leek, our anonymous domain registrar. We previously outlined the qualities that such a registrar must possess in Section III-B.

1) Enforcing Cost: Without a cost for domain registration, it is conceivable that the space of domains would be quickly exhausted. While designing Leek, we surveyed multiple methods for acquiring a domain registration:

Monetary Compensation. This method is used by the registrars of public ICANN domains. Registrants must pay a recurring fee 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 [42] may allow for privacy-preserving transactions [39], the authors believe that this security assertion is too new to be relied upon [8], [34], [11]. However, this method can make the process for registration simple and provides a financial incentive for the implementers to maintain the system.

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 Internet destinations (such as `google.o`) by making them unavailable for anyone else to register. Without this method, a land rush 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.

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 [24], [20] and denial-of-service attacks [32], [56]. While proof-of-work has significant problems in some systems [36], it has proven effective in other systems such as Bitcoin [42]. 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. Our auction scheme remedies this by attempting to force all registrants to use as much of their computing power as possible, requiring registrants to focus on a much smaller number of domains at a time.

Selection. Our above list is not comprehensive, and 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 [13]. For Leek, we implemented a combination of bootstrapping and proof-of-work, which we explain below.

2) Bootstrapping Leek: We implemented a simple bootstrapping method that retrieves the `NS` and `DS` records for the top one million domains provided by Alexa [6]. 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. Domains not yet DNSSEC-enabled may create a new `TXT` record of the form `ODNS-VER=<DS>`. 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.

Our bootstrapping implementation does not provide protection for domain seizures for unclaimed domains. Specifically, it will synchronize records 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 a land rush whereby malicious early adopters quickly register popular domains (e.g., `google.o`) and creating confusion among users of the system.

3) Domain Auction and Registration Game: Leek provides core functions including domain registration, renewal, nameserver changes, and ownership transfer using a novel proof-of-work scheme. This section describes our auction protocol.

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:** the 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

Protocol. An attempt to register a domain proceeds as follows:

1) A user begins a domain registration. A mandatory waiting period is set to 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:
   $$\mathsf{SHA512}(p \| a) \ \& \ \{1\}^d = 0, \qquad (1)$$
   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.

Similar to other registrars, Leek is *not* designed to enforce fairness or equality among registrants. Our auction system requires interested parties to "bid" on domains, enforcing a real-world cost (e.g., power, time, heat) for OnionDNS domains. The player who is able to offer the greatest amount of resources has the highest probability of successfully registering the domain.

We prevent denial-of-registration attacks 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 we require players to remain active and 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.

The auction protocol alleviates domain contention 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.

Leek enforces blind bidding 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) Ownership and Maintenance: 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.

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:
$$\mathsf{SETNS}\|n_1\|i_1\|n_2\|i_2. \tag{2}$$
- **SETKEY**: Domain owners may choose to rotate their associated key as a best practice [35] 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}:
$$\mathsf{SETKEY}\|U'_{PK}\|\mathsf{Sign}(U'_{SK}, d). \tag{3}$$

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

Domain Expiration and Renewal. ICANN domains may be renewed by again paying the registration fee in advance of an expiration date. This allows unwanted domains to re-enter the pool of available domains. Bootstrapped OnionDNS domains are removed from the `.o` zone when the associated Internet domains are unregistered.

OnionDNS domains expire one year after their registration. To renew a domain, a user must complete a single proof-of-work at a *renewal* difficulty level to disincentivize squatting. The proof can be requested during the final 30 days of the domain validity period and once submitted, the domain's expiration is extended. If no valid proof is submitted, the domain is removed from the `.o` zone and becomes available for registration.

IV. Experimental Evaluation

We configured an OnionDNS root and mirror on the Tor network as described in Section III. The OnionDNS zone is a standard zone file, served using standard DNS software, so mirrors servicing clients over the open Internet have *equivalent*

[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.

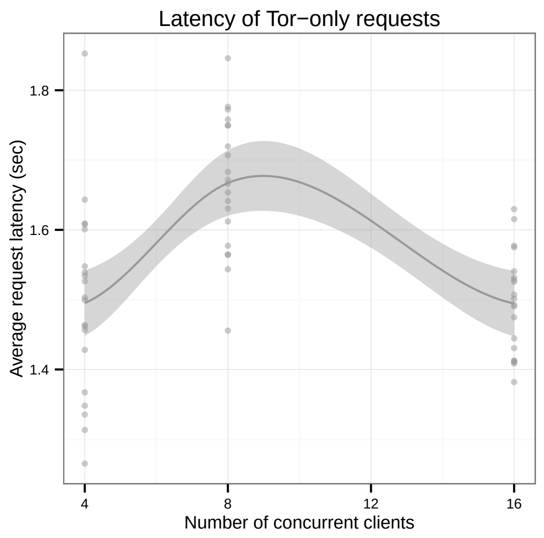

Fig. 3: The average latency of queries through the real-world Tor network. These queries were made to a mirror that was listening on an OnionCat-routable IPv6 address. Twenty trials of 4, 8, and 16 clients were performed.

performance to any other DNS server. The work described in this section focuses on communications occurring over the Tor anonymity network, including *1.* zone transfers between mirrors and the root and *2.* client requests to mirrors, for clients and mirrors choosing to communicate over Tor.

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 continually request domains, the total number was not important. After signing, BIND presented the zone. The experimental mirror transferred the zone file from the root over Tor and 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 listened for incoming connections on the Tor network. We ran up to 16 clients against the mirror to acquire latency data. Figure 3 shows the average latencies from this experiment. The figure quantifies 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. During each trial, we allowed clients to regenerate Tor circuits, leading to the variation between individual trials. 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, our experiments show that a client should expect a higher front-loaded latency, depending on the client's Tor circuit.

V. 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, the use of strong cryptographic constructions will 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 [13]. For our application, we would first turn the DNSSEC signing key over to a reputable foreign organization to delegate the key. 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 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.

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 to generate the garbled circuits used in the transaction, and one party to 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 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. Thus, we must rely on internal vetting and the probabilistic nature of the selection to keep the keys from a malicious party.

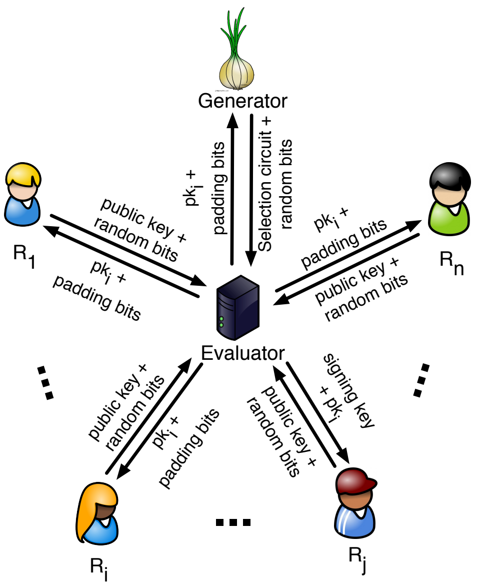

Fig. 4: The ownership delegation protocol. All parties input their public keys and some random bits, which the evaluator uses to randomly and obliviously select both the Root Administrator (R_i) and the Key Owner (R_j). All outputs must be padded to the same length to prevent a passive adversary from 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.

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. [33] and Carter et al. [15], [14], [16], 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 both public keys and random strings from the recipients as inputs. The circuit will output the signing key to 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 will 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 $i \in \{1, \ldots, n\}$, 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 the 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.

VI. 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.

Acknowledgments

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] https://en.bitcoin.it/wiki/Namecoin.

[2] Namecoin was stillborn, I had to switch off life-support. https://bitcointalk.org/index.php?topic=310954 (archived at http://www.webcitation.org/6KXauX8uC).

[3] The List Of Sites Challenging Domain Seizures. http://www.techdirt.com/articles/20110612/21573514664/list-sites-challenging-domain-seizures.shtml.

[4] 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.

[5] 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.

[6] Alexa. Top 1,000,000 Sites. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[7] R. Anderson et al. The eternity service. In *Pragocrypt96*, pages 242–252, 1996.

[8] E. Androulaki, G. Karame, M. Roeschlin, T. Scherer, and S. Capkun. Evaluating User Privacy in Bitcoin. *IACR Cryptology ePrint Archive*, 2012:596, 2012.

[9] Anonymous. The collateral damage of internet censorship by DNS injection. *ACM SIGCOMM Computer Communication Review*, 42(3), 2012.

[10] B. Awerbuch and C. Scheideler. Group spreading: A protocol for provably secure distributed name service. *Automata, Languages and Programming*, pages 187–210, 2004.

[11] 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.

[12] A. Biryukov, I. Pustogarov, and R.-P. Weinmann. Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization. 2013.

[13] 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.

[14] H. Carter, C. Lever, and P. Traynor. Whitewash: Outsourcing Garbled Circuit Generation for Mobile Devices. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2014.

[15] 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.

[16] H. Carter, B. Mood, P. Traynor, and K. Butler. Outsourcing secure two-party computation as a black box. Cryptology ePrint Archive, Report 2014/936, 2014. http://eprint.iacr.org/.

[17] 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.

[18] 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.

[19] R. Cox, A. Muthitacharoen, and R. Morris. Serving DNS using a peer-to-peer lookup service. *Peer-to-Peer Systems*, pages 155–165, 2002.

[20] L. F. Cranor and B. A. LaMacchia. Spam! *Communications of the ACM*, 41(8):74–83, 1998.

[21] R. Dingledine. Obfsproxy: the next step in the censorship arms race. *Tor Project official blog*, 2012.

[22] 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.

[23] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[24] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Advances in Cryptology*, pages 139–147. Springer, 1993.

[25] D. E. Eastlake et al. Domain name system security extensions. 1999.

[26] Electronic Fontier Foundation. Anti-Counterfitting Trade Agreement (ACTA). https://www.eff.org/issues/acta, 2012.

[27] 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.

[28] B. R. Fischer. OnionCat: A Tor-based Anonymous VPN. In *Proceedings of the 25th Chaos Communication Congress*, 2008.

[29] A. M. Froomkin. Wrong Turn in Cyberspace: Using ICANN to Route around the APA and the Constitution. *Duke Law Journal*, 50(1), 2000.

[30] Internet Systems Consortium. https://www.isc.org/downloads/bind/.

[31] 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.

[32] A. Juels and J. G. Brainard. Client Puzzles: A Cryptographic Countermeasure Against Connection Depletion Attacks. In *NDSS*, volume 99, pages 151–165, 1999.

[33] 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.

[34] 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.

[35] O. Kolkman and M. Gieben. RFC 4161 DNSSEC Operational Practices, 2006.

[36] B. Laurie and R. Clayton. "Proof-of-Work" proves not to work; version 0.2. In *Workshop on Economics and Information, Security*, 2004.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1:2012, 2008.

[43] R. Naraine. Massive DDoS Attack Hit DNS Root Servers. http://www.internetnews.com/dev-news/article.php/1486981, 2002.

[44] L. Overlier and P. Syverson. Locating hidden servers. In *Security and Privacy, 2006 IEEE Symposium on*, pages 15–pp. IEEE, 2006.

[45] 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.

[46] 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.

[47] K. Park, V. Pai, L. Peterson, and Z. Wang. CoDNS: Improving DNS Performance and Reliability via Cooperative Lookups. *OSDI*, pages 199–214, 2004.

[48] D. Piscitello. Anatomy of a DNS DDoS Amplification Attack. http://www.watchguard.com/infocenter/editorial/41649.asp, 2011.

[49] L. Poole and V. Pai. ConfiDNS: Leveraging scale and history to improve DNS security. *Proceedings of WORLDS*, 2006.

[50] 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.

[51] 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.

[52] TorrentFreak. U.S. Government Shuts Down 84,000 Websites, 'By Mistake'. http://torrentfreak.com/u-s-government-shuts-down-84000-websites-by-mistake-110216/.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] D. Yadron. Syrian Electronic Army's Alleged Attacks Expose Soft Spot. http://online.wsj.com/news/articles/SB10001424127887324009304579040900023429122, 2013.