ValidaTor: Domain Validation over Tor

Jens Frieß, *National Research Center for Applied Cybersecurity ATHENE and Technische Universität Darmstadt;* Haya Schulmann, *National Research Center for Applied Cybersecurity ATHENE and Goethe-Universität Frankfurt;* Michael Waidner, *National Research Center for Applied Cybersecurity ATHENE and Technische Universität Darmstadt*

https://www.usenix.org/conference/nsdi25/presentation/friess

This paper is included in the Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation.

April 28–30, 2025 • Philadelphia, PA, USA

978-1-939133-46-5

Open access to the Proceedings of the 22nd USENIX Symposium on Networked Systems Design and Implementation is sponsored by

ValidaTor: Domain Validation over Tor

Jens Frieß[§‡], Haya Schulmann[§†], and Michael Waidner[§‡]

[§]National Research Center for Applied Cybersecurity ATHENE
[‡]Technische Universität Darmstadt [†]Goethe-Universität Frankfurt

Abstract

Domain Validation (DV) is the primary method used by Certificate Authorities (CAs) to confirm administrative control over a domain before issuing digital certificates. Despite its widespread use, DV is vulnerable to various attacks, prompting the adoption of multiple vantage points to enhance security, such as the state of the art DV mechanism supported by Let's Encrypt. However, even distributed static vantage points remain susceptible to targeted attacks. In this paper we introduce ValidaTor, an HTTP-based domain validation system that leverages the Tor network to create a distributed and *unpredictable* set of validators. By utilizing Tor's exit nodes, ValidaTor significantly increases the pool of available validators, providing high path diversity and resilience against strong adversaries.

Our empirical evaluations demonstrate that ValidaTor can achieve the validation throughput of a commercial CA and has the potential to scale to a validation volume comparable to Let's Encrypt, while using minimal dedicated infrastructure and only a small fraction (~0.1%) of Tor's remaining available bandwidth. While unpredictable selection of validators makes ValidaTor fully resistant to targeted attacks on validators, we also show the use of Tor nodes improves path diversity and thereby the resilience of DV to subversion by well-positioned ASes, reducing the number of Autonomous Systems (ASes) capable of issuing fraudulent certificates by up to 27% compared to Let's Encrypt. Lastly, we show that the chance of subversion by malicious, colluding exit nodes is negligible ($\leq 1\%$ even with a quarter of existing exit nodes).

We make the code of ValidaTor as well as the datasets and measurements publicly available for use, reproduction, and future research.[1]

1 Introduction

Digital public key certificates associated with Internet domains are fundamental to Internet security. These certificates include crucial information such as the domain name and the public validation key corresponding to the domain owner's private key, ensuring the authenticity of communicating parties and enabling secure connections. The correct issuance of certificates is critical, as any compromise could allow an adversary to impersonate a legitimate service and intercept sensitive communications.

Certificate Authorities (CAs) use Domain Validation (DV) to verify domain ownership before issuing a certificate. This process typically involves generating a random challenge-response pair, and having the domain owner place the response at the challenged location under the target domain, e.g., in a DNS TXT record or an HTTP-hosted `.txt` file. The CA validates the domain by querying for the challenge and checking that the obtained response matches the challenge. If the CA receives the correct response, the domain owner has successfully demonstrated control over the domain, and the CA issues a signed certificate for the domain with the public key provided by the domain owner.

Although the Public Key Infrastructure (PKI) security relies on the correctness of DV for checking control over domains, the DV procedure itself remains vulnerable. The challenge-response process of DV is conducted over unencrypted and unauthenticated channels and heavily relies on the Domain Name System (DNS) RFC1035 and the Border Gateway Protocol (BGP) RFC4271. Both are largely unprotected and vulnerable to hijacking attacks [1, 2], and adversaries have launched such attacks, e.g., to steal cryptocurrency [3–5].

The attacks and the importance of certificate security triggered research efforts to identify vulnerabilities in the DV procedure and to develop a more secure design. Initially, vanilla DV was performed with a single request to the target domain. However, it was shown that an adversary could manipulate the challenge request of a CA, causing it to be executed against servers that the adversary controls [6]. As a result, the largest CA, Let's Encrypt, developed MultiVA, which uses additional vantage points [7]. During the validation process a number of vantage points independently send a challenge to the target domain. MultiVA became an official market standard with Automated Certificate Management Environment (ACME) in

[1] https://github.com/jenfrie/tova

RFC8555. The use of multiple vantage points aims to increase security despite being conducted over unprotected channels since it is assumed that a realistic adversary cannot control multiple vantage points and it is more difficult to attack them concurrently than a single vantage point.

The MultiVA was also extensively validated in the research community [7, 8]. Simulations have shown that the average resilience of a customer domain against issuance of fraudulent certificates is 88.6% [9], reflecting the percentage of adversarial ASes that cannot intercept communication via same-prefix BGP hiijacks to a sufficient number of vantage points to complete DV successfully. However, the analysis of [9] excluded a number of attack vectors, such as sub-prefix hijacks, as well as attacks against the DNS resolvers. Another important aspect that was not considered in these simulations is the dependency between DNS and the inter-domain routing infrastructures, which further increases the attack surface: DNS cache poisoning can be leveraged to facilitate BGP hijacks [10]. Indeed, [11] exploited a property in the software of many popular DNS resolver implementations to force all the vantage points to query one specific nameserver of the adversary's choice. This reduced the distributed defense to just one weak link, the nameserver. By hijacking the prefix of the nameserver, the adversary could intercept communication with all the vantage points. This shows that fixed vantage points, i.e., always at the same IP address or on the same IP prefix, allow the adversary to launch targeted attack against them in advance, before initiating the domain validation process. Both these issues were exploited in [11] to subvert DV from multiple vantage points, indicating that merely increasing the number of vantage points does not suffice to fully prevent targeted attacks.

A resilience of 88.6% offered by MultiVA [9] against interceptions for a critical building block like PKI is arguably not sufficient. Ideally we should achieve a 100% resilient PKI. Furthermore, 88.6% resilience considers only BGP hijacks, but does not account for other common attack vectors targeting the validators and connected infrastructure, e.g., DNS cache poisoning or infiltration of on-path routers.

The main problem, as pointed out by [11], is that current vantage points are static. In addition to being distributed, the vantage points' selection needs to be *unpredictable*. If the adversary does not know the addresses of the vantage points involved in a given validation, it cannot attack them.

Therefore, to improve security, the validators should be selected at random from the set of vantage points. The pool of vantage points should be large enough that it is infeasible to hijack any significant fraction. With a dedicated deployment, such as that of Let's Encrypt, generating such a large pool is not practical since it would require setting up a significant amount of geographically distributed infrastructure, prohibitively increasing the costs. In fact, scalability and cost are a major inherent limitation of any dedicated infrastructure.

Contributions. Motivated by the limitations in the existing domain validation mechanisms, we aim to design a system that resolves them. We develop a novel approach for distributed domain validation using Tor [12], dubbed ValidaTor. With over 2,200 Tor exit nodes, located across 280 unique BGP origins and 1,221 unique IP addresses, our design substantially increases the scale of the distributed infrastructure available to DV. This is a huge improvement compared to the 7 BGP origins currently used by Let's Encrypt. It allows a randomized selection of a configurable number of validators, effectively preventing BGP hijack attacks for subverting validation. We perform extensive evaluations to demonstrate the security and performance of our proposal based on real-world measurements and simulations. Our contributions:

- Conceptually, we offer a novel use of the Tor network to address practical limitations in DV. Our design and implementation of ValidaTor resolves the scalability of massively distributed DV and enhances its resilience.
- We propose a path overlap metric to capture the diversity of paths taken from validators to target domains through the BGP network, allowing for easy comparison of DV deployments with regard to threats from well-positioned ASes.
- Our experimental evaluations show that ValidaTor decreases average path overlap by ~50% compared to Let's Encrypt's MultiVA, translating to a reduction of 21% with 3 validators (up to 27% with 7 validators) in potential fraudulent certificates from well-positioned ASes. Combined with the resistance to BGP hijacks through unpredictable selection of validators, this significantly hardens DV against subversion.
- We show that with our selection method the chance of an attacker controlling validation with malicious Tor nodes is negligible, amounting to less than 1% even when controlling a quarter of all existing Tor nodes.
- We demonstrate a validation rate of up to 11.9 validations per second with 5 validators and estimate that, even at the scale of the entire Web PKI, Tor's remaining bandwidth would only be taxed by 0.11%. The median duration for a single validation in this case is ~2 seconds, with at least 95% of all validations under 6 seconds. This is comparable with the use of the certbot ACME client.
- We design ValidaTor so that it enables easy adoption and requires no changes to the existing infrastructure of the CAs, which is critical to motivate deployment and avoid misconfiguration. We make our prototype and the collected data available for public use.

Ethical considerations. We ensure that the design of ValidaTor is ethical, does not burden the Tor network and does not interfere with the connections of other Tor clients. As detailed in Section 6.4, the expected network load on Tor from a full switch to ValidaTor by the entire Web, is 635 MBit/s, around 0.1% of the remaining available bandwidth. We also take extensive measures to ensure that our experiments follow the ethical guidelines for network measurements [13, 14].

Organization. We review Related Work in Section 2 and provide an overview of Tor network in Section 4. The design

and implementation of ValidaTor are given in Section 5. We provide experimental evaluations in Section 6 and security analysis against different attack vectors is in Section 7. We conclude our paper in Section 9.

2 Related Work

Security of DV mechanisms in CAs have been extensively studied. We review existing literature related to DV:

Vulnerabilities in DV. Domain validation mechanisms are susceptible to various attacks that can undermine the security of digital certificates. In [15], the authors outline general security considerations for CAs, highlighting potential weaknesses in various validation methods. Compromised or maliciously configured DNS servers can be exploited to redirect validation requests to attacker-controlled servers [16]. Hijacking the DV process is easy against a vanilla DV, where a single validator communicates with a single server in the target domain.

Such attacks can also be extended to MultiVA, where the validation nodes are either fixed, predictable or the set of possible validators is small, allowing the adversary to attack them all in advance. [11] showed that in those cases adversaries can force all the DV nodes to run the validation against a server of the attacker's choice.

These methods allow attackers to pass validation checks and obtain certificates for domains they do not own, posing a serious risk to web security. Furthermore, [17] demonstrated specific attacks on DV mechanisms, including DNS hijacking and email-based validation exploits, which can be used to fraudulently obtain certificates for domains not owned by the attacker. [18] described man-in-the-middle (MitM) attacks and cache poisoning as significant threats to the integrity of domain validation and [6] demonstrated vulnerability of popular CAs to cache poisoning, allowing adversaries to issue fraudulent certificates. These studies underline the necessity for rigorous validation protocols to mitigate the risks of domain spoofing and unauthorized certificate issuance.

Measurements of DV. Empirical evaluations of CA practices have been crucial in assessing the real-world effectiveness of DV and impact of attacks on DV mechanisms. [19] performed a large-scale analysis of HTTPS traffic to assess the security of DV implementations across various CAs. Their findings revealed inconsistencies in how CAs implement and enforce DV protocols, highlighting potential security gaps.

[20] measured the adoption of security enhancements such as DNS Security Extensions (DNSSEC) and found that despite increased awareness, adoption rates remain low. This indicates a persistent vulnerability in the DV ecosystem due to the reliance on insecure DNS infrastructure. Similarly, [21] evaluated the adoption of DNSSEC by CAs, finding that while adoption rates were increasing, significant gaps remained in ensuring consistent and secure validation processes across the board. Additionally, [22] evaluated the security of email-based DV methods, showing that attackers can exploit weak email configurations to intercept validation emails, further emphasizing the need for robust security measures.

Improvements to DV. In response to identified vulnerabilities and attacks, various improvements to DV protocols have been proposed. [23] presented enhanced validation methods leveraging DNSSEC to provide cryptographic guarantees of domain ownership, although its adoption and security still remain a challenge [24]. [25] suggested a multi-factor validation approach that combines DNS-based validation with additional verification steps such as HTTP-based challenges and validation token usage.

Recently, [26] proposed ADDVent: the idea is to inflate the infrastructure available to DV by crowdsourcing web clients over an advertisement network. This approach is expensive due to the involved costs of advertisement network. Another issue is the trust required in the advertisement network - a malicious network can subvert the security of DV.

Despite significant research efforts in the area of DV, the limitations to deployment of distributed DV introduced by the need for dedicated infrastructure has so far not been addressed. Nevertheless, the limited number of vantage points performing validation and their fixed location played a major role in attacks against DV, exposing the PKI to fraudulent certificates. In this work we aim to resolve these limitations by designing ValidaTor: DV based on Tor.

3 Tor vs. Other Proxies

Since ValidaTor leverages the Tor network first and foremost as a massively distributed set of proxies, it is worth discussing alternative solutions that offer the same capabilities for randomized, distributed DV, without some of the constraints incurred by Tor.

Dedicated volunteer networks. RIPE Atlas [27] and NLNOG RING [28] are examples of volunteer-run multi-perspective systems. The former is the RIPE NCC's global Internet connectivity measurement network, consisting of volunteer-hosted probes that can be triggered to ping an arbitrary destination. The latter is a collective of machines exclusively hosted by network operators, made available to participants for testing network connectivity.

In a similar manner, a distributed network of nodes used exclusively for DV could be created. However, whereas in the above use cases there is little incentive for malicious behavior, with DV there is a strong incentive specifically for adversaries to volunteer and control the validation nodes, acting benignly until an opportunity for a high-value certificate presents itself. It is thus risky to allow such a volunteer network to be open to public participation.

A dedicated network would likely only be feasible if **(a)** participation is restricted to CAs (similar to NLNOG's focus on network operators) or **(b)** the network implements sufficient monitoring and maintenance to guard against malicious participants; something Tor already does. In any case, ValidaTor could be readily deployed now and incrementally moved to a dedicated network, when such efforts get off the ground.

Proxy providers. Rather than using Tor circuits, requiring a minimum of two hops to reach a proxy node, we could halve the performance costs[2] by using a simple proxy provider. However, this introduces two problems: **(1)** all proxies are under the control of a single entity, thus requiring trust in a third party. This could potentially be offset by diversifying proxy providers. **(2)** We find that the costs are prohibitive and threaten the economic viability of such a system. Based on a cost comparison of popular providers, Brightdata [29] offers the lowest rate at $0.42/GB. This covers only datacenter IPs, but still provides over 770K IPs across 98 countries. Based on our projected traffic volume of 317 MBit/s for exit nodes (see Section 6.4), this would amount to $4.2M/year.

Public VPN providers. The same considerations apply to VPN providers. These typically offer fewer, but still sufficient and globally distributed, endpoints. ExpressVPN [30], for example, offers 160 endpoints at $100/year, covering 5 devices. Connecting to all 160 endpoints at once to avoid switching overhead would thus result in just $3200/year. While this supposedly covers unlimited bandwidth, with our estimated traffic of ~1.1 TB/hour, this would likely violate terms of service. It is questionable whether this traffic load could be spread among a sufficient number of VPN providers to fall within a reasonable bandwidth use per provider.

Other anonymity networks. Other solutions, like Nym, require cryptocurrency payments to route traffic based on volume. It is also a smaller network than Tor with exit nodes on the order of a few hundred. I2P's focus is on services internal to its overlay network, rather than proxying connections; for DV we require proxies into the clearnet. The selection of outproxies is comparatively limited, on the order of 10 to 100 and not intrinsically mapped out like in Tor.

To conclude, the alternatives to DV over Tor are either limited in their vantage points and capacity, expensive to use, or pose higher engineering hurdles, which hinders their adoption in the real world. As each alternative comes with its own unique implementation challenges, we choose Tor as the most promising candidate to explore in-depth. Tor removes the financial cost as a barrier to the adoption of massively distributed domain validation and allows an easy-to-deploy solution with sufficient capacity.

4 Overview of Tor

Tor (The Onion Router) is a privacy-focused network of volunteer-hosted "relays" designed to enable anonymous, private communication over the Internet by routing users' requests through multiple encrypted hops, preventing traffic analysis and censorship [12]. **Onion proxies (OPs)** are the entrypoint for client software using Tor, responsible for managing circuits and connections to websites. **Onion routers (ORs)** are the relay-components which form the circuits. Each OR decrypts one layer of encryption to determine the next OR to forwards the packets to [31]. The **exit node** is the final node in a circuit, decrypting the user's original packet for its final hop to the destination server. From the server's perspective, the traffic appears to originate from the exit node.

Crucially for ValidaTor, all traffic until the exit node is protected. All unprotected traffic originates at the exit node, including DNS resolution. ValidaTor can therefore leverage a large variety of perspectives (~2.2K) using only a single server. Because the exit node sees the original unencrypted packet, it also presents the most critical threat. Node selection is therefore designed to minimize the risk of malicious, colluding exit nodes (see Section 7.5). Lastly, the list of active ORs is maintained by **directory servers**. These servers provide the necessary information for OPs to build circuits, which is periodically updated and signed to ensure integrity [12].

In this work we leverage the Tor nodes as an overlay to gain two critical benefits: first, we obtain a distributed infrastructure for DV with a large number of potential validator nodes which supports unpredictable selection. Second, the DV communication protocol benefits from security on the network layer making it impossible for any adversary before the exit node to observe the validation requests to the domains.

5 ValidaTor: Design & Implementation

We explain the limitations in existing DV mechanisms, and discuss how we resolve them with our proposal, ValidaTor, which uses Tor as an overlay to validate control over domains. We present the design and the implementation of ValidaTor, and provide motivation for different architectural choices.

5.1 Resolving Limitations in DV

The central problem with DV is that it is, by necessity, unauthenticated and thus vulnerable to manipulation by a Man-in-the-Middle (MitM) adversary. Successive improvements to DV, as laid out in Sections 1 and 2, have therefore focused on increasing the difficulty and resources required to mount a successful attack, and limiting an attacker's degree of influence on the process as a whole.

With vanilla DV, an attacker needs to target only a single server or intercept only a single communication path. With MultiVA this is increased to 3 or 4 servers and previous research has worked towards optimizing the distribution of these vantage points. Nonetheless, there are several ways in which the remaining attack surface can be further reduced and the use of Tor can address all of these by routing DV requests through a random set of exit nodes:

Dynamic validators. A static set of validators is vulnerable to targeted attacks. Adversaries can plan and prepare ahead of time, knowing exactly where their targets are. The same applies for a random but *small* set of validators, which could still be exhaustively targeted. Tor provides access to a large number of endpoints, allowing a vast number of possible combinations to be chosen from for each validation at random. This unpredictability means an attacker needs to be prepared to attack a significant fraction of these endpoints to have a

[2] We would still need to use HTTPS proxies to protect data in transit.

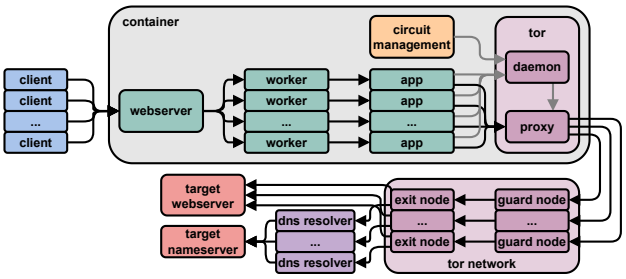

Figure 1: Container design and domain validation request flow.

realistic chance of hijacking DV. The size and distribution of the Tor network makes this prohibitively difficult.

Path diversity. A lack of diversity in the paths from validators to target domains results in more choke points in the network, where an attacker has a larger degree of influence. Whether an adversary is already on-path or uses routing hijacks to gain an on-path position, an attacker's influence in the BGP network is topologically constrained. Thus, reducing the overlap of validation paths reduces the set of remaining potential adversaries. The diverse set of endpoints offered by Tor allows validators to be selected with a significantly higher degree of distribution.

Denial-of-Service. Blocking DV by on-path adversaries can lead to denial-of-service for certificate issuance. As Tor is already set up to circumvent censorship through its distributed nature, Tor-based DV benefits from this as well.

5.2 Design

The core idea behind our design of ValidaTor is to select a random set of Tor exit nodes for each validation to act as validators. The nodes involved in any given validation are selected such that they are highly distributed in the BGP network to limit the degree of path overlap.

Component interactions. To implement a proof-of-concept for the ValidaTor server we construct a containerized environment using Docker, containing the following components: the Tor service, a custom circuit management service, a webserver and Python `Flask` application for handling the validation logic, i.e., triggering requests to target domains and aggregating the results.

As outlined in Figure 1, a validation applicant first sends a validation request to the webserver (see below for further details). A worker process then runs `Flask` application's validation logic, which triggers multiple web requests to the target domain through Tor's SOCKS5 proxy. The application also sends control instructions to the Tor daemon to assign the resulting streams to a random set of circuits, whose exits are ensured to be sufficiently distributed. The circuits are built and managed in parallel by the circuit management service (discussed in detail in Section 5.3), which likewise communicates with the Tor daemon through control instructions. Once streams are assigned to circuits, the Tor proxy forwards each request to its guard node, which then forwards it to the exit node. Each exit node resolves the target domain through its DNS resolver and subsequently queries the challenge resource at the resolved IP address. The responses from the target webserver are then returned via the Tor circuit to the application, which aggregates them and produces the reply that the container's webserver sends back to the client.

Validation. We implement our proof-of-concept ValidaTor as a web service that CAs and other clients can use to validate a domain by specifying the `.txt` resource to be challenged (as used by the ACME standard RFC8555 [32]), the target domain, and the protocol with which to send the request (either HTTP or HTTPS). The service then attempts to retrieve the resource via an initial k exit nodes. If the content of the resource (e.g., in ACME, a 43 character random alphanumeric string) does not match across all k requests, the system attempts to tie-break with additional exit nodes up to a maximum of n. If at least k responses contain the same content, this content is returned, otherwise an error.

This mimics the k-out-of-n validation scheme used by Let's Encrypt, where initially 3 out of 4 validation servers attempt to retrieve a specific `.txt` resource, and only if these do not receive the same result the 4th server is used. Then, if at least 3 out of the 4 results match the expected response, the validation is completed successfully.

5.3 Constructing Circuits

This section details the challenges in managing DV over Tor and how we construct and assign circuits to ensure both randomization and distribution for each validation.

Manual stream assignment. During regular client operation, the Tor service maintains a handful of circuits, constructs new circuits based on demand, i.e., the amount of requests sent through its SOCKS5 proxy, and rebuilds circuits every 10 minutes. A key feature of Tor to protect from profiling is that requests to domains are sent through different circuits. For our purposes such protection does not play a role. It *is*, however, critical that all n requests to the *same* domain are sent through different circuits (specifically, different exit nodes) so that we validate a domain from at least k perspectives.

We therefore configure Tor to allow *us* to choose which circuits to attach which requests (so-called "streams") to. This can be done in Python through Tor's official `stem` library, which provides an interface to send control messages to the Tor daemon. This allows us to ensure that requests for the same domain, i.e., that are part of the same validation, are sent through different circuits.

Choice of exit nodes. In establishing the circuits, we ensure **(a)** sufficient distribution of the exit nodes of the circuits used for a given validation, and **(b)** that exit nodes are chosen uniformly at random. As such, we cannot rely on Tor's relay selection algorithm, which is focused on anonymity and effective load balancing. Multiple requests to the same domain would be routed through the same circuit and there is no guarantee that a new circuit will use an exit node with a different

IP address, let alone an exit node in a different network. Exit nodes are also selected based on available bandwidth, which would reduce the unpredictability of validator selection. We therefore build circuits manually using `stem`. We begin by selecting randomly among all nodes marked with the `EXIT` flag. This flag is assigned to relays that allow their use as an exit node and have been monitored by Tor for some time. We exclude nodes that have been flagged as `BADEXIT`, which is assigned to exit nodes that behave suspiciously (~2.2K out of the ~7K total nodes), but additionally impose the requirement that a selected node's IP address does not share the same /8 prefix with any of the other already selected nodes, thereby ensuring distribution in the BGP network. The prefix length can be adapted to the desired granularity.

Cycling circuits. Constructing a new dedicated circuit for every request produces unnecessary latency and overhead. By default, Tor maintains a handful of open circuits and assigns requests to these based on the target domain. That is, requests to different domains are routed through different circuits. If the current pool of circuits does not accommodate this, new circuits are built on-demand. Circuits are then used for up to 10 minutes, by default, and only cycled after.

Thus, similar to Tor's regular operation, we maintain a set of circuits and periodically rebuild these to avoid too many validations using the same endpoints. However, we maintain a larger set of concurrent circuits than Tor normally would, to increase the unpredictability of the endpoints chosen for a given validation.

A larger pool to select from means a larger number of possible configurations, as given by the binomial function, and thus, greater uncertainty for the selection of any particular configuration. In the opposite extreme, where the number of concurrent circuits is equal to the number of validators, there is exactly one configuration. If this "pool" of circuits is maintained for any significant period of time, an attacker could identify the validators with a single validation request, and use this information to potentially manipulate subsequent validation requests.

Ideally, we would maintain a circuit for every available exit node. However, we observe that upwards of around 70 - 80 open circuits become unstable, as the Tor daemon strongly conflicts with our circuit building application, closing circuits that are not in use, and thus limiting the number of circuits we can keep open concurrently. Through experimentation we find that the largest number of circuits that remain stable is at around 50-60 concurrent circuits. We then rebuild circuits every 3 minutes. This can be configured to match the desired trade-off between network overhead and circuit "freshness".

Validation queries for a given domain can then efficiently be assigned to a random subset of circuits (which, as noted above, are already ensured to be distributed).

2-Hop circuits. Another reason to build circuits manually is that Tor by default builds circuits with 3 hops. We do not require anonymity, hence we can increase performance by building our own circuits and reducing the number of hops. Ideally, our Tor client would connect directly to each exit node. However, exit nodes need to be configured explicitly to allow 1-hop circuits; the default configuration is to refuse these. As this would limit our choice of endpoints from which to validate, we opt for building 2-hop circuits.

Choice of entry nodes. Since anonymity is not of vital concern, we optimize our entry nodes for performance. Conveniently, Tor assigns the `GUARD` flag to nodes that have been active for at least 8 days and have been independently measured to confirm their otherwise self-reported available bandwidth. This indicates that these nodes are particularly suitable as entry nodes.

To maintain a degree of load balancing across the network we randomly select an entry node from among the ~2.5K nodes marked `GUARD` and `FAST`. However, we weigh each node by its advertised bandwidth and IP prefix overlap with our own IP address, in order to favor shorter network paths as well as high-throughput nodes.

5.4 Challenge Methods

While both HTTP and HTTPS requests can be sent through the Tor network, thereby allowing for HTTP-based domain validation, Tor currently supports DNS only partially, i.e., for the resolution of domain names via A and AAAA records and reverse-lookup of IP addresses via PTR records. This allows domains to be resolved by the exit node.

Crucially, DNS TXT records, used for DNS-based domain validation, are currently *not* supported. Our proposed system is thus limited to HTTP-based validation until Tor adds support for DNS TXT records. However, since HTTP-based validation is the default for common ACME implementations, we argue that this is not a severe limitation.

5.5 Easy Adoption

Proposals which require substantial changes to the existing infrastructure take decades to deploy, for instance, DNS Security Extensions with DNSSEC RFC9364. Therefore, to facilitate their adoption, new mechanisms should require minimal changes. Our design follows this principle, requiring only to patch ACME's request for the target resource (e.g., `http://example.com/.well-known/acme-challenge/random`) with a request to ValidaTor (e.g., `https://validator.com/http/example.com/.well-known/acme-challenge/random`). If ValidaTor asserts that the responses match, ACME receives the same file content as from the original request and continues operating unchanged. If ValidaTor finds differing responses, it will return an error, causing ACME to fail the validation.

6 Performance Evaluation

In this section we validate the time required for validation and the throughput with ValidaTor. We then show how to use parallelization to further scale the system. Lastly, we discuss

the implications on bandwidth usage and from blocking of Tor traffic.

6.1 Single Container

For our initial evaluation we run our Docker container on a virtual server with 24 cores, allowing a certain degree of parallel processing of validation requests, using an equivalent 24 workers for the webserver in the container. Once the container is running and the 50 Tor circuits have been built, we trigger validation procedures via HTTPS requests to the webserver in the container. We run 80 querying threads in parallel, each continuously submitting a domain to validate and waiting for the response. We select the top 10k Tranco [33] domains that host a webserver and request the `/robots.txt` document to simulate an ACME query.

This initial evaluation provides a baseline performance benchmark for different numbers of validators. We test three different scenarios: 3-out-of-4 (equivalent to Let's Encrypt), 5-out-of-7, and 7-out-of-9, to provide an intuition as to the impact of the number of validators on performance and security. We note here that we observe no disagreement among the validator results across all of our tests. As such, each validation produces exactly k requests.

We note the resulting hardware requirements derived from `docker stats` for interested readers in the Appendix A.

Validation duration. Figure 2 shows the distribution of the duration of each validation. This refers to the time between the server receiving a validation request from an applicant and receiving the last validator response necessary to produce a verdict. While the spread increases as k increases, a client can still expect their validation request to complete within around 20 seconds, even with $k = 7$.

We compare the duration of certificates issuance to certbot [34]. Certbot is a popular open source ACME client by the Electronic Frontier Foundation (EFF) for automatically requesting certifcates from Let's Encrypt via the ACME protocol. When measuring certbot, we start a manual DV procedure, place the created challenge value on our webserver, start our timer, and then trigger certbot to begin the validation. We stop our timer when certbot exits. We find that for $k = 3$ the median duration is comparable to that observed when using certbot, and can be further improved with parallelization, as shown by the multi-container tests.

Since obtaining a certificate is an occasional process, and tools like certbot can often be used to complete the validation process without the user's interactive involvement, we consider the additional latency incurred by the use of Tor circuits to be negligible for the end user.

Validation throughput. Table 1 shows the validations per second we achieve with our prototype implementation. Increasing k leads to an expected decrease in throughput, as more resources are required for each individual validation. However, ValidaTor can also easily be scaled horizontally to accommodate higher validation throughput. By further optimizing the implementation and parallelizing the deployment, it is thus feasible to scale the system to match even the rate of ~65 certificates Let's Encrypt issued on average per second on peak days in late 2024 [35].

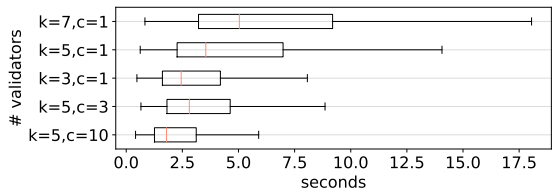

Figure 2: Validation duration with k validators and c containers.

containers	k	validations / s
1	3	2.7
1	5	2.1
1	7	1.6
3	5	6.5
10	5	11.9

Table 1: Validation throughput.

6.2 Scaling

We extend the Tor proxy and circuit management system in ValidaTor to handle multiple validation requests in parallel, assigning each request to a different circuit. As we next show, this parallelization improves the throughput and efficiency of the validation process.

To further scale the throughput of ValidaTor we conduct two additional experiments. For the first we deploy $c = 3$ instances of our container on the same VM, each running 24 workers, building 50 circuits and using $k = 5$ validators per domain. We use 200 client processes to issue the validation requests and load-balance them among the containers.

For the second scaling experiment we increase parallelization even further, deploying $c = 10$ containers, using 20 cores each (on a sufficiently capable server), building 15 circuits each (for the same total of 150 circuits), and also using $k = 5$ validators per domain. We load-balance the same 200 client processes among the containers.

The 3-container deployment achieves the expected, roughly 3x increase over the single-container deployment at 6.5 validations per second. The 10-container deployment does not quite match its expected throughput, but nonetheless increases throughput to 11.9 validations per second.

The increase in performance over the single container is due to parallelization of the Tor daemon (with a separate instance in each container), which mediates the control requests required to manually assign web requests to Tor circuits. This demonstrates encouraging evidence that such a system can be scaled horizontally to production-grade performance through sufficient parallelization.

6.3 Blocking of Tor Traffic

As a well-known anonymity network, Tor may sometimes be discriminated against by networks that seek to prevent anonymous access, potentially limiting its use for DV. We

investigate the significance of this using two validation measurements (2025-02-08), one based on the Tranco top 10K and one sampling 10K domains from Certificate Transparency logs (accessed via certstream), in order to compare potential differences between popular and regular domains, both using a 5-out-of-7 scheme.

We augment our initial implementation to match each exit node and target IP with the request result obtained by that node. If traffic is being blocked we expect to see a `ConnectTimeout` or `ConnectionError`. As shown in Figures 3 and 4, this is the case for around 20% of requests, regardless of the popularity of the domains. CT-sampled domains show no difference to top Tranco domains. In addition, Figure 4 shows that all vantage points tend to agree on the result, i.e. they all either succeed or fail. This strongly suggests that blocking of Tor traffic is done primarily by the domains themselves, rather than transit networks. In the latter case, since the exit nodes are widely distributed, we would expect to see cases where e.g. only one node fails to connect, because its request is routed through a blocking AS, whereas the others are routed differently. This is not the case, however.

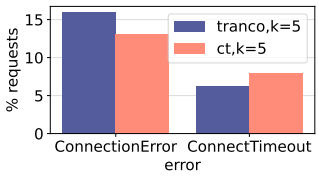

Figure 3: Rates of connection errors.

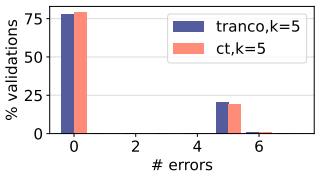

Figure 4: Errors per validation.

We further investigate this by mapping the IP addresses of exit nodes and target servers to their respective AS and simulating the AS paths taken. We then count how often a given AS is on-path for requests that ended in a connection error versus requests that succeeded. If an AS blocks Tor traffic, we would expect to see all requests routed through it to produce connection errors. The impact of such an AS on ValidaTor is significant, if it is implicated in a large number of validation attempts. However, as we see from the data, the impact is not significant. For the CT-based measurement, we see only 10 out of 816 ASes that consistently feature on the paths of failing requests, with only 4 seen in more than 1 validation round and none of them transit ASes. For the Tranco-based measurement it is 35 out of 2531 ASes that consistently produce failures, with each only appearing in 1 validation round and only 8 of them being transit ASes.

These results further solidify the case that blocking of Tor traffic by transit ASes is a rare occurrence without significant impact on the performance of ValidaTor. The key source of validation failures are the destinations of the validation requests. This means it is within the power of domains to allow DV traffic from Tor nodes.

6.4 Bandwidth Impact

In January 2025, the Tor network measured an average 347 GBit/s of consumed bandwidth across all relays (98 GBit/s at exit nodes), a little over a third of the total available advertised bandwidth of 917 GBit/s (less than a third of the 314 GBit/s advertised by exit nodes) [36].

The maximum total traffic produced by our container (as measured by `docker stats`) was 482MB for 10K validations and $k = 7$ (including circuit creation). The average size of the HTTP response bodies observed in our measurements is 528 bytes, whereas an ACME challenge text file is 43 bytes in length. Thus, substituting the proper content for the DV requests, while maintaining the same HTTP headers and Tor circuit building, results in $482MB - 10,000 * 528B + 10,000 * 43B = 477MB$.

As indicated by the steady increase in total traffic and runtime difference between the $k = 3$, $k = 5$ and $k = 7$ conditions (shown in Figure 11), the network load scales linearly with the number of validations and validators. Thus we can fairly reliably extrapolate to the scale of the entire Web PKI. As an estimate of the current certificate volume we track certificate issuances via Certificate Transparency logs (using certstream) for a period of 15 hours, during which we observe an average rate of 104 new certificates per second. Since this accounts only for successfuly validations, we can safely assume at least twice the number of validation attempts. Thus, we can expect:

$$\frac{208}{s} * \frac{477MB}{10,000} * 8 = 79.4MBit/s$$

that is, ~$11.3MBit/s$ per validator. For $k = 5$ and $k = 3$ we get similar rates of $11.7MBit/s$ and $12.7MBit/s$, respectively, matching the expectation that the traffic volume is roughly linear in the number of validators.

In addition to unsuccessful validation attempts, we need to account for chasing of redirects, which is often done during DV, but is not included in our bandwidth data. Redirects are processed by the web client and thus result in two or more distinct requests. Assuming for simplicity an average of 1 redirect per validation, we can expect a rate of $2 * 79.4MBit/s = 158.8MBit/s$. This is a conservative estimate, since we observe redirects on only ~$\frac{1}{3}$ of all domains, with popular Tranco domains using redirects twice as often as the broader Internet, sampled from CT logs.

Our bandwidth measurement also only covers the data flowing in and out of our server. This includes both request and response data, but only for the link between our server and the entry node of our Tor circuit, thus considering only the ingress load at the entry node. This same data is also transmitted between entry and exit node, adding the same load at both the entry node egress and the exit node ingress, as well as between exit node and target server, producing another egress load at the exit node. Thus, the overall load on the Tor network is $4 * 158.8MBit/s = 635.2MBit/s$, with $317.6MBit/s$ hitting the exit node.

Thus, we conclude that even at the current volume of the entire Web PKI and using 7 validators, routing DV traffic through Tor consumes a negligible amount of the network's bandwidth: just 0.11% of Tor's *remaining* total bandwidth and 0.15% of the exit nodes' *remaining* bandwidth. Furthermore, ValidaTor containers could be horizontally scaled to the point where each container has a handful of dedicated Tor exits, avoiding the need for periodic recreation of Tor circuits and thus reducing bandwidth usage.

7 Security Analysis

ValidaTor ensures high path diversity by selecting a random set of exit nodes for each validation request. This randomization makes it difficult for attackers to predict and intercept the validation traffic. We demonstrate experimentally on heuristically derived datasets that path diversity attained with Tor nodes is significantly higher than that obtained with the dedicated infrastructure of Let's Encrypt. Additionally, ValidaTor configures the Tor client to avoid using exit nodes with the same /8 IP prefix, ensuring that the exit nodes are sufficiently distributed across the network. This configuration enhances security by reducing the likelihood that an attacker can control multiple exit nodes in the same circuit.

7.1 Perspective Diversity

Routing domain validation traffic through the Tor network allows us to access many more validation perspectives than a dedicated infrastructure, such as that of Let's Encrypt.

We create a new domain, trigger validation requests for this domain using certbot, and observe the IP addresses logged on our webserver. We repeat this 5 times[3], collecting a total of 11 unique IP addresses. Using Tor's stem library, we obtain all known relays and filter for those marked with the EXIT flag and *not* marked BADEXIT. At the time of our experiments this list contained 2254 nodes with 1221 unique IP addresses.

To consider the diversity in perspectives with regards to BGP routing, we need to map IP addresses to ASes. CAIDA [37] provides a prefix-to-ASN mapping based on BGP origin announcements collected by the RouteViews project. However, as pointed out in [9], this underestimates routing diversity when different prefixes are announced through different paths using the same ASN, as is the case for e.g. AWS datacenters, which host Let's Encrypt validators. Multiple unique origins are thus aggregated under the same ASN. We "de-aggregate" ASNs by considering each prefix as a unique origin, if it is announced through a different path, in line with [9]. This is done based on the most recent available route collector data from RIPE RIS [38] and RouteViews [39].

For the 11 Let's Encrypt IP addresses we obtain 7 unique BGP origins (instead of just 2 when considering aggregated ASNs), whereas for the 1221 Tor exit IP addresses we obtain 280 unique BGP origins (210 with aggregated ASNs).

[3] Certbot allows a maximum of 5 validation attempts in short succession. We see no new BGP origins after the first 3 attempts.

7.2 DNS Perspective Diversity

Tor is implemented such that, when proxying an HTTP request to a given domain through a Tor circuit, the exit node will take care of resolving that domain to an IP address. As pointed out in [9], the DNS resolution path is part of the attack surface for hijacking domain validation. As such, diversifying validators is only useful if the DNS resolvers used by those validators are also diversified. It could be possible that many of the exit nodes use the same few DNS resolvers. While Tor explicitly encourages the use of default resolvers and resolvers within an exit node's network, and discourages the use of public DNS resolvers to avoid centralization, we aim to confirm this by measuring the IP addresses of the resolvers used by exit nodes.

To investigate this we set up a fresh domain with an NS record pointing to a server under our control and log incoming DNS requests on port 53 using `tcpdump`. For Let's Encrypt we again trigger 5 validation requests through certbot, for Tor we send a web request through each of the 2254 exit nodes. In each case this triggers the attempt to resolve our domain using our server, allowing us to track which IP addresses the DNS requests are coming from. By correlating the timeframe of each request as recorded client-side with the timestamps of DNS requests received server-side, we match each Tor exit node IP address with the most likely resolver IP address. Note that, across all certbot measurements, we never actually complete the validation process. Therefore, no certificate is created to be logged in and scraped from Certificate Transparency logs. Thus, our webserver logs are not contaminated by various Internet scans (see for example [40]).

For Let's Encrypt we observe 28 unique resolver IP addresses, corresponding to 9 unique de-aggregated BGP origins. For Tor we matched 1118 unique resolver IP addresses to the 1221 unique exit node IP addresses, corresponding to 174 unique deaggregated BGP origins. While we observed additional IP addresses from another 100 BGP origins, we could not clearly match these to exit nodes and thus use 174 as a lower bound.

7.3 Path Diversity

To test how well diversity of perspectives translates to diversity of BGP paths taken from validators to a target domain we reconstruct the BGP network at the prefix level from route collector data and apply the most recently available CAIDA AS relationships dataset [41]. We map each IP, as observed in our experiments, to its de-aggregated ASN and calculate the paths from each source ASN to the observed destination ASN using the `pybgpsim` library from [42]. We also consider the return paths from destination ASN to source ASN, since these might be different.

Using this method we can recover the transit ASes A for each path and see how many ASes the paths of a given validation have in common. That is, how many ASes can intercept what percentage of validators for how many domains?

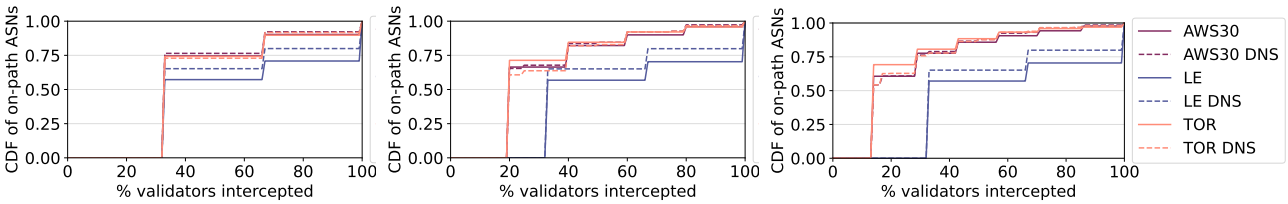

Figure 5: Cumulative distribution of the percentage of validators intercepted by each on-path ASN for each domain ($k = \{3,5,7\}$ from left to right).

In essence, we look at each on-path ASN for each domain validation and determine the percentage of validators that transit through this ASN. We plot the resulting cumulative distribution for each $k \in \{3,5,7\}$ in Figure 5.

For Tor, we use the validator-target pairs as observed in our experiments. For the Let's Encrypt comparison, we simulate the validator-target pairs by truncating to the first 3 pairs (since Let's Encrypt does 3-out-of-4 validation, only using the 4th if the first 3 do not agree) and replacing the first validator with Let's Encrypt's main network (23.178.112.0/24, which is present in every validation attempt triggered with certbot) and the two other validators with a random selection of 2 out of the remaining 6 BGP origins observed in Section 7.1. The same substitution process is also used for the AWS30 comparison, an optimally distributed AWS deployment of 30 locations (discussed in detail in Section 8).

As indicated by Figure 5, using Tor exit nodes as validators significantly decreases the number of ASes in a position to intercept a large fraction of validators for a large number of domains. This is particularly true for ASes that can intercept 100% of validators, and thus forge the overall validation result (reduction of 21%). This number further decreases with additional validators ($k \geq 5$), as the overlap between validation paths decreases (reduction of up to 27%). The comparison with the hypothetical AWS30 deployment shows that such path diversity can also be achieved with a cloud-based deployment, albeit at much greater cost.

7.3.1 Path Overlap Metric

While reducing the number of ASes that can intercept all validators translates directly to fewer threats of fraudulent certificates, it is important to consider the degree of influence of individual ASes on validation more generally. The lower the path overlap between validators, the lower this influence. We therefore propose a metric that summarizes the overall degree of path overlap in a single number, for easier characterization and comparison between deployments.

First, we define the path from a validator i to a domain d as the set of ASes transited both from validator to domain and from domain back to validator, denoted P_{id}. The degree of overlap between the paths can then be calculated as the Jaccard distance between the sets:

$$jaccard(P_{id}, P_{jd}) = \frac{P_{id} \cap P_{jd}}{P_{id} \cup P_{jd}}$$

A value of 0 therefore means the paths of no ASes in common, whereas a value of 1 means both validators transit all the same ASes to reach the target domain. To capture the degree of overlap for a given validation with a group of k validators, we calculate the pairwise Jaccard distance and average over the number of possible pairs:

$$val\_overlap(d) = \frac{\sum_{i=1}^{k} \sum_{j=i+1}^{k} jaccard(P_{id}, P_{jd})}{\frac{k(k-1)}{2}}$$

Finally, we average the overlaps across all D validations for a given deployment: $overlap(D) = \sum_{d=1}^{D} val\_overlap(d)/D$.

We calculate this metric for ValidaTor $\forall k \in \{3,5,7\}$, as well as the the simulated Let's Encrypt and AWS30 deployments for each of our measurements, and compare the results in Figure 6. We see that the switch to more nodes introduces the most dramatic decrease in average path overlap (~50%), with smaller incremental decreases when using additional validators with the larger Tor pool.

7.4 DNS Path Diversity

To evaluate the diversity of the paths from DNS resolvers to nameservers, we repeat the same evaluation, but with the observed resolver and nameserver IP addresses. Specifically, for ValidaTor, we replace each exit node IP with the matched DNS resolver IP. For the Let's Encrypt comparison we again truncate the validator-target pairs to 3, replace the first exit node IP with the IP of the DNS resolver of the main Let's Encrypt server, and replace the remaining 2 exit node IP addresses with a random selection of 2 IP addresses from the observed DNS resolvers (from different BGP origins).

In both scenarios the target IP of each validator-target pair is replaced with the IP of a random nameserver of the target domain. These IP addresses were previously collected by querying the NS records of each domain and then the A records for each NS result. Due to the randomization we conduct 5 rounds for each deployment and average the resulting path overlap metrics.

The results are also shown (dashed) in Figures 5 and 6. While Let's Encrypt's DNS resolvers show higher path diversity than its validation servers, Tor still offers marked improvements.

7.5 Malicious Exit Nodes

We consider attacks by malicious Tor exit nodes and show how to configure the ValidaTor parameters to substantially decrease the risk of such attacks.

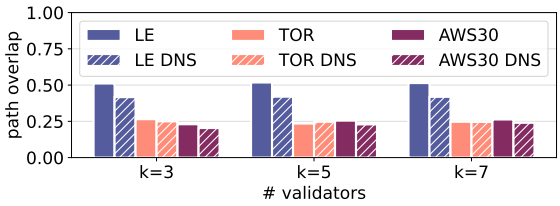

Figure 6: Path overlap for Let's Encrypt, Tor, an optimally distributed AWS deployment of $N = 30$ nodes and respective DNS resolvers.

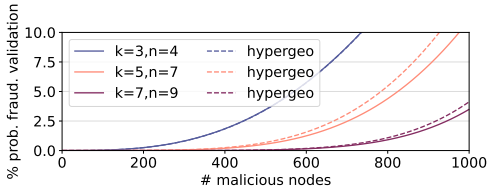

Figure 7: Probability of fraudulent validation using compromised exit nodes.

7.5.1 Probability of Fraudulent Validation

Leveraging Tor exit nodes as vantage points for domain validation introduces the risk of attackers running malicious nodes to intercept and manipulate the validation process. In essence, this is similar to the man-in-the-middle threat posed by BGP hijacks and other routing manipulations. Here, we model the probability of an attacker successfully manipulating validation this way and show that is exceedingly small.

Random selection. Due to the k-out-of-n successful query results required for validation, an attacker would need at least k of his nodes selected for validating the same domain. With a fully random selection, i.e., each node having an equal chance of being selected, the probability of exactly k out of n nodes being malicious is given by the hypergeometric distribution:

$$hg(M,N,n,k) = \frac{\binom{M}{k}\binom{N-M}{n-k}}{\binom{N}{n}}$$

where N denotes the total number of exit nodes in the Tor network (2293 measured at the time of writing) and M denotes the number of nodes under the attacker's control. The probability for *at least* k malicious nodes is thus:

$$p(M,N,n,k) = \sum_{j=k}^{n} hg(M,N,n,j)$$

This is plotted as dashes in Figure 7 for different k and n.

Accounting for prefix-aware selection. However, the node selection process is not fully random, since every time a node is selected, all nodes on the same (configurable) network prefix are excluded as validators for the same validation to ensure distribution. This means an attacker would need to control nodes in at least k different networks to even have the possibility of using them to validate the same domain, significantly raising the attacker's resource requirements.

To simplify the model and give the attacker the highest chance of success, we assume the attacker's M nodes are distributed equally across all networks. Similarly, we assume that each network contains the same total number of nodes. While this is not strictly the case in practice, it allows us to simplify the chance of selecting a malicious node to an average probability of $\frac{M}{N}$, regardless of how many nodes have already been selected. This is because the ratio of malicious to total nodes remains constant if the same number of malicious and legitimate nodes are removed from the pool each round.

Accounting for staged selection. The second difference to a simple k-out-of-n selection is that, for efficiency, the selection is done in stages. That is, we begin with a selection of k validators and only incrementally add additional validators (up to a maximum of n) if agreement has not yet been reached with the validators selected thus far. This means, in order to reach the threshold of k by the n^{th} selection, an attacker would need at least $k-(n-k)$ of his nodes to be selected among the first k. Exactly how many malicious nodes are among the first k then determines how many additional selections are made and need to also be malicious. With the simplifying assumptions above, this selection process results in the following probability, that an attacker can control the outcome of a validation using malicious exit nodes:

$$p(M,N,n,k) = \sum_{j=k-(n-k)}^{k} \binom{k}{j} (\frac{M}{N})^k (1-\frac{M}{N})^{k-j}$$

Figure 7 plots this function for different parameters k and n. While very similar to the hypergeometric distribution, the more accurate modeling of the selection process works slightly to the disadvantage of the attacker for $k > 3$. Overall, we see that, given the current number of $N = 2293$ Tor exit nodes, the attacker's probability of success is very small even when controlling several hundred exit nodes. Note that we assume for the calculation that the attacker controls M of the *existing* N nodes, giving him a stronger advantage than adding M *additional* nodes. Second, we see that the risk can easily and significantly be reduced by simply increasing k. For example, at $M = 700$ the attacker's chance of success can be reduced from ~10% with $k = 3$ to ~2.5% with $k = 5$ and to less than 1% with $k = 7$.

Tor has seen malicious actors adding large numbers of nodes in the past (e.g. ~1000 in 2020 [43]). In this example the attacker would have an 8.7% chance of having all $k = 3$ of their nodes selected to validate their request but only a 0.4% chance of having all $k = 7$ of their nodes selected. A possible mitigation for such attacks could be the temporary increase of k when the Tor network appears to be under attack. For example, at $k = 9$ the attacker's chance of success falls to 0.05%, despite controlling 1000 out of a total of 3293 nodes.

7.5.2 Additional Countermeasures

The threat of colluding exit nodes threatens the security of every other application of Tor. The network is therefore continuously monitored for bad actors by the Tor Project [44], researchers [45] and other volunteers. Implicated nodes are either flagged as `BADEXIT` or removed from the network consensus, aiding the selection against malicious nodes.

Such monitoring could be further extended in the context of our validation system. Across our experiments we very rarely see discrepancies between the results reported by different validators, even when $k >= 7$. Multiple discrepancies involving the same exit IP addresses could therefore easily be detected and flagged, avoiding the use of these exits. Similarly on the the client-side, multiple failed validations in a short period of time could lead to blocklisting of a client's account for that domain, as implemented by Let's Encrypt with certbot. As such, we can effectively prevent an attacker from playing the numbers game and simply attempting many validations until one succeeds.

8 Randomized Distributed DV in the Cloud

Following the comparisons between the Tor network and Let's Encrypt's current MultiVA deployment, it begs the question what a cloud deployment would look like that achieves the same security properties. We begin by fixing the desired maximum probability of a fraudulent validation at a generous 1%. We then consider the following two scenarios: **(a)** using 3-out-of-4 validation as is currently done by Let's Encrypt (more beneficial if traffic is more expensive than additional datacenter nodes) and **(b)** using 7-out-of-9 validation (more beneficial if nodes are more expensive than traffic). We then plot the probability of fraudulent validation, as described in Section 7.5.1, for different numbers of total vantage points N and vantage points hijackable by an attacker x. The results for each scenario are shown in Figure 8.

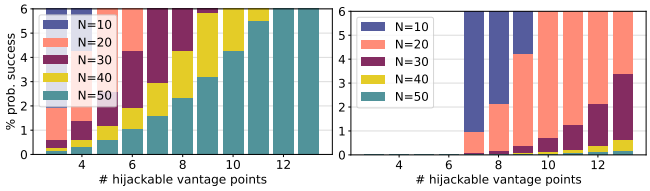

Figure 8: Probability of fraudulent validation by hijacking cloud vantage points, using $k = 3$ (left) and $k = 7$ (right) validators.

For 3-out-of-4 validation, at least $N = 30$ vantage points are required to keep the probability of fraudulent validation under 1%. However, if, for example, the attacker is assumed to be able to hijack up to 6 vantage points, at least $N = 50$ is required. 7-out-of-9 validation naturally remains 100% secure if the attacker cannot hijack at least 7 vantage points. However, even with up to 10 compromised vantage points, the probability of success is still well below 1% at $N = 30$.

To evaluate the path overlap (see Section 7.3.1), for such a deployment, we consult the published IP ranges of AWS EC2 networks and select the 30 vantage points with the least pairwise overlap between network addresses. We then perform the same simulation of AS paths for validation requests as in Section 7.3. For $k = 3$ we see a path overlap of 0.23, comparable to Tor-based validation. For $k = 7$ we see a slightly higher overlap of 0.29 (see Figure 6). It is therefore possible to achieve similar degrees of distribution simply with additional cloud vantage points. However, an increase from 4 to 30 vantage points and the additional traffic of $k = 7$ versus $k = 3$ validators in this scenario would multiply the costs many times. Additionally, a lower risk threshold of under 1% increases the required number of vantage points.

9 Conclusions

Our design of ValidaTor, i.e., DV over Tor, resolves many of the existing limitations of DV and MultiVA. Leveraging Tor as a massively distributed set of vantage points from which to conduct multi-perspective domain validation offers clear advantages over existing systems:

Randomization. Vantage points can be randomly chosen from a large set of available nodes, preventing attackers from knowing in advance which validation nodes to target. This prevents any type of targeted attack, such as prefix BGP hijacks and DNS cache poisoning.

Resilience to interception and malicious Tor exit nodes. Tor exit nodes, as well as the DNS resolvers they use, are more distributed than the vantage points of existing systems. As we show in our experimental evaluations, this results in a significant reduction in the number of ASes that can intercept *all* vantage points used for a validation of a given domain. Such ASes can impersonate validation by intercepting the validation process to obtain fraudulent certificates.

In order to stand a statistically significant chance of fraudulently validating a domain, an attacker needs to control on the order of several hundreds of Tor nodes. Since Tor has a vested interest in avoiding such scenarios to protect from, e.g., traffic correlation attacks, this is unrealistic to persist. Simply increasing the number of validators for a given validation drastically decreases an attacker's chance of success.

DoS resistance. If individual Tor relays, ASes, or routers maliciously drop domain validation traffic, the system can choose new circuits through which to route traffic. Ensuring the same degree of availability with MultiVA requires a much larger deployment.

Lightweight load on Tor network. Since domain validation traffic is lightweight (only querying small text resources, most often via plain HTTP) and circuit management can be implemented without overly frequent rebuilding of circuits, the projected load on the Tor network produced by even a Let's Encrypt-scale volume of validation requests is negligible compared to Tor's available bandwidth.

Open infrastructure. ValidaTor scales with the expansion of the Tor network. The more users operate Tor nodes, the higher the path diversity offered to DV, and the lower load from DV on individual nodes.

Easy adoption. The validation system requires no additional infrastructure to be set up for each vantage point. ValidaTor integrates seamlessly with the CAs and their existing infrastructure, and is easy to adopt into existing DV systems.

Acknowledgements

This work has been co-funded by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and Arts within their joint support of the National Research Center for Applied Cybersecurity ATHENE and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) SFB 1119.

References

[1] H. Birge-Lee, Y. Sun, A. Edmundson, J. Rexford, and P. Mittal, "Using bgp to acquire bogus tls certificates," *HotPETS'17*, 2017.

[2] H. Birge-Lee, L. Wang, J. Rexford, and P. Mittal, "Sico: Surgical interception attacks by manipulating bgp communities," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 431–448.

[3] Sharon Goldberg, "The myetherwallet.com hijack and why it's risky to hold cryptocurrency in a webapp," https://medium.com/@goldbe/the-myetherwallet-com-hijack-and-why-its-risky-to-hold-cryptocurrency-in-a-webapp-261131fad278, 2018.

[4] P. Kacherginsky, "Celer Bridge incident analysis," 2022. [Online]. Available: https://www.coinbase.com/de/blog/celer-bridge-incident-analysis

[5] H. Birge-Lee, L. Wang, G. Cimaszewski, J. Rexford, and P. Mittal, "Attackers exploit fundamental flaw in the web's security to steal 2USD million in cryptocurrency," 2022. [Online]. Available: https://freedom-to-tinker.com/2022/03/09/attackers-exploit-fundamental-flaw-in-the-webs-security-to-steal-2-million-in-cryptocurrency/

[6] M. Brandt, T. Dai, A. Klein, H. Shulman, and M. Waidner, "Domain validation++ for mitm-resilient pki," in *CCS '18: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 2060–2076.

[7] H. Birge-Lee, Y. Sun, A. Edmundson, J. Rexford, and P. Mittal, "Bamboozling certificate authorities with bgp," *SEC'18: Proceedings of the 27th USENIX Conference on Security Symposium*, pp. 833–849, 2018.

[8] K. Bhargavan, A. Delignat-Lavaud, and N. Kobeissi, "Formal modeling and verification for domain validation and acme," in *Financial Cryptography and Data Security: 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers 21*. Springer, 2017, pp. 561–578.

[9] G. H. Cimaszewski, H. Birge-Lee, L. Wang, J. Rexford, and P. Mittal, "How effective is multiple-vantage-point domain control validation?" in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 5701–5718.

[10] T. Hlavacek, P. Jeitner, D. Mirdita, H. Shulman, and M. Waidner, "Behind the scenes of rpki," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 1413–1426.

[11] T. Dai, H. Shulman, and M. Waidner, "Let's downgrade let's encrypt," *CCS '21: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1421–1440, 2021.

[12] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13*. USENIX Association, 2004, pp. 21–21.

[13] Z. Durumeric, E. Wustrow, and J. A. Halderman, "Zmap: Fast internet-wide scanning and its security applications." in *Usenix Security*, vol. 2013, 2013.

[14] C. Partridge and M. Allman, "Ethical considerations in network measurement papers," *Communications of the ACM*, vol. 59, no. 10, pp. 58–64, 2016.

[15] S. Chokhani, W. Ford, R. Sabett, C. Merrill, and S. Wu, "Rfc3647: Internet x. 509 public key infrastructure certificate policy and certification practices framework," 2003.

[16] X. Liu, X. Wang, and W. Chen, "Case study of dns server attacks: Exploits and countermeasures," in *Proceedings of the 2013 ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2013, pp. 79–84.

[17] M. Kranch and J. Bonneau, "Certificate transparency and its implications for the security of domain validation," in *Proceedings of the 2015 ACM Conference on Computer and Communications Security*, 2015, pp. 241–252.

[18] J. Zhang, M. Wu, K. Sun, and X. Du, "Privacy and security risks in domain validation for ssl certificates," *Journal of Network and Computer Applications*, vol. 65, pp. 84–96, 2016.

[19] J. Amann, N. Vallina-Rodriguez, C. Kreibich, and N. Weaver, "Mission accomplished? https security after diginotar," *Proceedings of the 2017 Internet Measurement Conference*, pp. 325–340, 2017.

[20] Q. Scheitle, O. Gasser, and G. Carle, "Measuring the impact of dnssec on dns and http performance," in *Proceedings of the 2018 ACM Conference on Internet Measurement*, 2018, pp. 1–14.

[21] J. Brian, A. Smith, and S. Zander, "An empirical study of the security of dnssec in the wild," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 1, pp. 1–25, 2019.

[22] T. Fiebig, Q. Scheitle, O. Hohlfeld, and G. Carle, "Something from nothing (there): Collecting global ipv6 dns open resolvers," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 963–979.

[23] Q. Scheitle, O. Gasser, T. Fiebig, and G. Carle, "Rise of the botivo: Evaluating the deployment of dane," in *Proceedings of the 2017 Internet Measurement Conference*, 2017, pp. 1–14.

[24] Internet Systems Consortium. (2024) Bind 9 security release and multi-vendor vulnerability handling, cve-2023-50387 and cve-2023-50868. Internet Systems Consortium. [Online]. Available: https://www.isc.org/blogs/2024-bind-security-release/

[25] H. Birge-Lee, J. Kao, A. Mirian, J. Rexford, and P. Mittal, "Towards improved domain validation in public key infrastructures," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 644–660.

[26] J. Frieß, H. Schulmann, and M. Waidner, "Crowdsourced distributed domain validation," in *Proceedings of the 23rd ACM Workshop on Hot Topics in Networks, HOTNETS 2024, Irvine, CA, USA, November 18-19, 2024*. ACM, 2024, pp. 318–325.

[27] RIPE NCC. (2024) Ripe atlas. RIPE Network Coordination Centre. [Online]. Available: https://atlas.ripe.net

[28] J. Snijders. (2024) Nlnog ring. NLNOG. [Online]. Available: https://ring.nlnog.net/

[29] Bright Data. (2024) Proxy services. Bright Data. [Online]. Available: https://brightdata.com/proxy-types

[30] ExpressVPN. (2024) High-speed, secure & anonymous vpn service. ExpressVPN. [Online]. Available: https://www.expressvpn.com/

[31] P. F. Syverson, D. M. Goldschlag, and M. G. Reed, "Towards an analysis of onion routing security," in *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*. Springer, 2001, pp. 96–114.

[32] R. Barnes, J. Hoffman-Andrews, D. McCarney, and J. Kasten, "Automatic Certificate Management Environment (ACME)," RFC 8555, Mar. 2019. [Online]. Available: https://www.rfc-editor.org/info/rfc8555

[33] V. L. Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczyński, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," in *Proceedings of Network and Distributed Systems Security (NDSS) Symposium*, 2019.

[34] Electronic Frontier Foundation. (2024) Certbot. Electronic Frontier Foundation. [Online]. Available: https://certbot.eff.org/

[35] Let's Encrypt. (2024) Let's encrypt stats. Internet Security Research Group. [Online]. Available: https://letsencrypt.org/stats/

[36] The Tor Project. (2024) Traffic. The Tor Project. [Online]. Available: https://metrics.torproject.org/bandwidth-flags.html

[37] CAIDA. Routeviews prefix-to-as mappings (pfx2as) for ipv4 and ipv6. [Online]. Available: https://publicdata.caida.org/datasets/routing/routeviews-prefix2as/

[38] R. NCC. Ris raw data. [Online]. Available: https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/ris-raw-data

[39] University of Oregon, "Route views project," http://bgplay.routeviews.org/, 2012.

[40] S. Pletinckx, T.-D. Nguyen, T. Fiebig, C. Kruegel, and G. Vigna, "Certifiably vulnerable: Using certificate transparency logs for target reconnaissance," in *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2023.

[41] CAIDA. The caida as relationships dataset, 2024-05-01. [Online]. Available: https://www.caida.org/catalog/datasets/as-relationships/

[42] M. Brandt and H. Shulman, "Optimized bgp simulator for evaluation of internet hijacks," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2021.

[43] C. Cimpanu. (2021, May) Thousands of tor exit nodes attacked cryptocurrency users over the past year. [Online]. Available: https://therecord.media/thousands-of-tor-exit-nodes-attacked-cryptocurrency-users-over-the-past-year

[44] G. Koppen. (2021) Recent rejection of relays. [Online]. Available: https://lists.torproject.org/pipermail/tor-relays/2021-December/020048.html

[45] P. Winter, R. Ensafi, K. Loesing, and N. Feamster, "Identifying and characterizing sybils in the tor network," in *Proceedings of the 25th USENIX Security Symposium*, 2016.

A Hardware Requirements

Using `docker stats`, we track the CPU and memory usage of our container over the course of the experiments, in order to give an idea of the hardware required for deployment at scale. As shown in Figures 9 and 10, respectively, the container uses an equivalent of 5 - 8 CPU cores and 800 - 1000MB of memory. There is some dependence on the number of validators used, particularly for CPU usage, but overall the resource usage is fairly stable, both across time and number of validators. This allows fairly accurate extrapolation of the hardware requirements for a large-scale system.

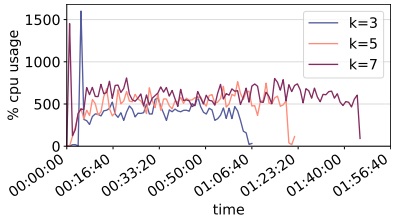

Figure 9: CPU usage per ValidaTor container over time.

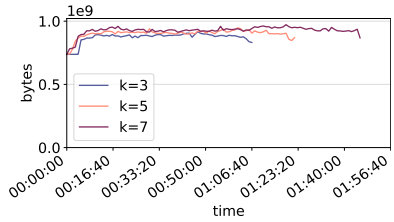

Figure 10: Memory per ValidaTor container over time.

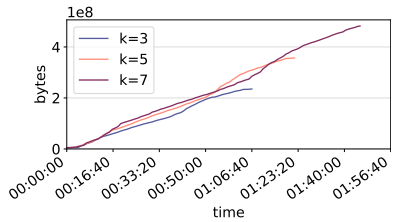

Figure 11: Cumulative network traffic per ValidaTor container over time.