VPN-Zero: A Privacy-Preserving Decentralized Virtual Private Network

Matteo Varvello
Nokia Bell Labs

Iñigo Querejeta Azurmendi
UC3M Madrid

Antonio Nappa
UC Berkeley/UC3M Madrid

Panagiotis Papadopoulos
Telefonica Research

Goncalo Pestana
Brave Software

Benjamin Livshits
Brave Software

***Abstract*—Distributed Virtual Private Networks (dVPNs) are new solutions aiming to solve the trust-privacy concern of a VPN's central authority by leveraging a distributed architecture. In this paper, we discuss the requirements of a successful dVPN system and we present *VPN-Zero*: a dVPN system with strong privacy guarantees that provides traffic accounting and has minimal performance impact on its users.**

***VPN-Zero* guarantees that a dVPN node only carries traffic it has "allowlisted", without revealing its allowlist or knowing the traffic it tunnels. This is achieved via three main innovations: (a) an attestation mechanism which leverages TLS to certify a user visit to a specific domain, (b) a zero-knowledge proof to certify that some incoming traffic is *authorized* (e.g., falls in a node's allowlist, without disclosing the target domain), and (c) a dynamic *chain* of VPN tunnels to both increase privacy and guarantee service continuation while traffic certification is in place. The paper demonstrates *VPN-Zero* functioning when integrated with two production systems: BitTorrent's Distributed Hash Table and ProtonVPN. Early evaluation results show that the median setup time of *VPN-Zero* is about 10 seconds.**

I. Introduction

A Virtual Private Network (VPN) is a connection method used to add privacy to private and public networks, like WiFi hotspots or the Internet. Traffic between the user (*VPN client*) and a *VPN node* is encrypted so that network elements along the path have no access to this traffic. User's traffic is forwarded with the IP address of the VPN node, a feature many VPN providers offer as a remedy to geo-blocking.

Users have to trust VPN providers not to interfere with or log any of their personal traffic. It is to be noted that VPN providers are commercial entities that might offer their services relying on other commercial entities (e.g., they could use multiple cloud services to obtain a worldwide footprint [32]). It follows that even trusted and respectable vendors might unknowingly incur issues with a specific provider ranging from surveillance [40], misconfiguration [28], and even hacking [11]. Either of these issues can compromise users' privacy. In [27] the authors actively investigate 62 commercial VPN providers and find unclear policies for non logging, some evidence with tampering of their customer traffic, and a mismatch between advertised VPN node locations and actual network location.

Driven by the above issues, decentralized Virtual Private Networks (dVPNs) arose as a fairly new trend with millions of daily users (e.g., Hola [45]). In a dVPN, users are both VPN clients and relay nodes as in a Peer-to-Peer (P2P) network. In spite of the apparent advantages on their privacy, users of dVPNs may need to tunnel through their devices traffic that can be considered harmful or illegal. Indeed, there have been incidents reported [6], [29], where unaware dVPN users have been (ab)used as exit nodes through which DDoS attacks were performed. Similarly, the users have no guarantee on whether a dVPN might inspect, log, and share their traffic.

In this paper, we first investigate the dVPN ecosystem and derive a set of *requirements* from a privacy/performance standpoint. Next, we propose *VPN-Zero*, to the best of our knowledge the first privacy-preserving dVPN with traffic accounting. *VPN-Zero* is founded on the idea that dVPN nodes should be able to decide which traffic they want to carry, e.g., only news websites. At the same time, they should accept such safe traffic in *zero knowledge*, *i.e.,* without being able to tell what this traffic contains. Ultimately, we aim to offer the above features with minimum impact on the user experience.

Note that such strong privacy guarantees are only possible in conjunction with already private traffic, *i.e.,* TLS v1.3 [36] and DoH (DNS over HTTPS) [20] whose adoption is on the rise. *VPN-Zero* further leverages a Distributed Hash Table (DHT) to pair dVPN clients with nodes currently available to serve their traffic. This pairing is realized using privacy preserving *announce* and *lookup* DHT primitives. We further rely on VPN *chains* which help both in preserving user privacy and allowing uninterrupted VPN service. Last but not least, we introduce a zero knowledge traffic attestation mechanism that piggybacks on TLS to enable privacy-preserving allowlists.

We have integrated *VPN-Zero* with BitTorrent's DHT [44] and ProtonVPN [35], a popular VPN provider. We demonstrate the feasibility of *VPN-Zero* while testing a public domain supporting TLS v1.3. We also benchmark *VPN-Zero* performance with respect to DHT lookup, VPN tunnel setup, and zero knowledge proof calculation (about 10 seconds in median values). We identify the current bottleneck in the proof calculation, which we plan to speed up as future work.

ISBN 978-3-903176-39-3

II. Motivation

VPN services enable users to bypass geo-blocking and enhance their privacy against snooping ISPs and malicious access points. To avoid the necessity of blindly trusting a centralized service provider that could harm their privacy (e.g., by logging their network connections), dVPNs came to the rescue. DVPNs are P2P VPNs where users forward their traffic through other users and vice-versa. This architecture allows malicious users to abuse the network and perform malicious transactions via unaware users positioned as exit nodes.

In this paper, we assume malicious dVPN users, hidden behind benign users' IPs, abuse the dVPN by: (i) accessing illegal content (e.g., child pornography, darknet markets [14]), or (ii) launching distributed attacks against selected targets [29]. In addition, we assume a snooping ISP that logs the network traffic uploaded by the user's device. These logs can be used later for purposes beyond the control of the user (sold to advertisers [26] or handed over to agencies [23]) that may result in tarnishing the user's reputation or even falsely accusing them of illegal transactions.

A. Requirement Analysis

IP Banning: To be usable, a VPN (both centralized and distributed) needs to publish at least a portion of its vantage point list. It follows that it is relatively easy for a censorship entity or a geo-blocking content provider to access such list and simply ban all the vantage points of a VPN. For centralized VPNs, this is an issue they constantly face and they can hardly solve. For dVPNs, such banning is harder due to the dynamic set of users/IPs involved. DVPN nodes are regular Internet users who frequently change network locations and connect from behind Network Address Translators (NATs). In this case, blocking a NATed VPN node implies blocking the whole subnet with a potentially massive service disruption.

No-Logging: Privacy is one of the main services offered by a VPN. This implies that, at no time, a VPN node should be able to log user traffic. In [27], authors investigate the usage policy offered by several commercial VPNs on their website. They find that when a privacy policy was available (75% of the cases), very few VPN services explicitly claimed a *no-logs* policy. This analysis suggests that VPN providers today should do a better job in terms of transparency of their actions. In a dVPN, logging is sometimes required to offer, for example, protection against IP banning: in VPN Gate [30], each VPN node keeps connection logs (and shares them with a central repository) to inform other VPN servers of a potential censorship authority attempting to discover (and block) the current dVPN footprint.

Traffic Accounting: The founding idea of a dVPN is that users share their resources in exchange of some form of payment; so there must be a system to account for such traffic and grant payments accordingly. *Crypto* dVPNs [39], [42] tackle this issue by leveraging the blockchain to keep track of proof of traffic. This can be challenging depending on which network logging level is allowed/required.

Traffic Blame: From a networking perspective, VPN nodes are the entity originating the traffic they carry. This means that serious offenses (e.g., child pornography, hate speech, drug smuggling), when investigated, will point the authorities to the entity running the VPN node. At this point, the above no-logs policy comes into play where the VPN might (or not) offer extra information about who was indeed originating such traffic. In a dVPN context, there is no legal entity the authority can reach to. Instead, they would reach the owner of the dVPN node whose network was used to carry such traffic. It is thus paramount that a dVPN implements a mechanism to avoid this kind of situation. At the same time, this should be achieved in a privacy preserving way, thus respecting the above no logging requirement.

Quality of Experience (QoE): Offering high QoE is a hard task for dVPNs. This is because of client churn and heterogeneous network conditions; this problem is not specific to dVPNs but an overall generic issue in distributed systems. A VPN footprint, *i.e.,* how many unique locations a VPN can offer, is another important QoE metric. VPN providers constantly battle to offer more vantage points, either by deploying new physical nodes or by introducing "virtual locations" based on the information available from geo-IP databases about the physical locations of their vantage points. A limitation of all centralized VPNs is the lack of residential IP addresses, since they mostly rely on data-centers to deploy their nodes. Contrary to that, by definition, dVPNs consist of a large network footprint of residential IP addresses.

Open Source: A dVPN client/server code is a very critical piece of software since it can potentially gain access to very sensitive data. Despite popular VPN tunneling protocols (OpenVPN, PPTP, and WireGuard) are inherently secure, it is important to note that misconfigurations and/or malicious code are still potential threats [21].

III. VPN-Zero: System Overview

This section presents the design of *VPN-Zero*, a privacy preserving dVPN based on *zero knowledge*. A zero knowledge proof is a cryptographic tool that allows a *prover* to prove to a *verifier* that a certain statement is true, without disclosing any information except the fact that the statement validates. In our case, a dVPN user wants to prove to a dVPN node that its current traffic is contained within the node's *allowlist*, *i.e.,* a set of domains the node is willing to carry traffic for.

Several challenges are involved to realize the above statement in a decentralized system. First, how to distribute such allowlists in a privacy preserving manner. Second, how to build a zero knowledge proof around *traffic*, and implicitly which traffic is suitable for such proof. Last but not least, how to perform the above operations without disrupting the user QoE.

In the remainder of this section, we first set the stage for the traffic type *VPN-Zero* can operate on to guarantee strong privacy requirements. Next, we introduce the cryptographic primitives at the foundation of *VPN-Zero*. We then describe its distributed architecture along with the protocol orchestrating *VPN-Zero*'s operations. Finally, we finish the section with a description of how we construct the zero knowledge proof.

A. Foundations

Traffic Confidentiality: *VPN-Zero*'s ambitious privacy goal translates into strict privacy requirements for the traffic being carried. To this end, *VPN-Zero* must be coupled with TLS v1.3 [36] and DoH (DNS over HTTPS) [20], whose popularity is rapidly growing. However, even TLS v1.3 does not imply mandatory encryption of the Server Name Indication (SNI) field in the `ClientHello`. This would allow a dVPN node to learn which domains a user is visiting, defeating our privacy-preserving goal. To prevent such privacy leak, the TLS Encrypted Client Hello RFC draft [38] (ECH) – formerly known as Encrypted Server Name Indication (ESNI) [37] – proposes a method to encrypt such information using a hybrid public key encryption scheme (HPKE) that involves the public key of the connecting server. ECH's precursor (ESNI) is already supported by all Cloudflare domains using their authoritative name servers [12]. We assume ECH is available which implies a dVPN node only has visibility on the destination IP address, since *VPN-Zero* introduces a mechanism to protect the source IP address, as detailed below.

Cryptographic Primitives: *VPN-Zero* requires each of the nodes to own a public-private key-pair $(\mathsf{pk}, \mathsf{sk})$. We denote the signature on message m with private key sk by $\mathsf{Sign}(m, \mathsf{sk})$. $\mathsf{Verify}(s, \mathsf{pk})$ verifies a signature s with public key pk. If the signature is valid, it outputs $\top$. Finally, we denote the encryption of message m and decryption of ciphertext C with $\mathsf{Enc}(m, \mathsf{pk})$, $\mathsf{Dec}(C, \mathsf{sk})$, respectively.

VPN-Zero relies on ephemeral key-pairs, used each time clients lookup a dVPN node. We use ElGamal key-pair [15], which results in a small overhead for the user (one exponentiation over a finite field for each new key calculation once public parameters are computed for the whole network). We use ephemeral keys so that connection requests cannot be linked to a particular user. We use $\mathsf{ElGamal}$ to denote the operations performed with the ElGamal key-pair $(\mathsf{pk}^{EG}, \mathsf{sk}^{EG})$.

The Encrypted Client Hello RFC draft [38] makes use of Hybrid Public Key Encryption (HPKE). We refer to encryption and decryption of HPKE as $\mathsf{HPK.Enc}(m, \mathsf{pk})$ and $\mathsf{HPK.Dec}(C, \mathsf{sk})$ respectively. Hybrid encryption depends on a Key Encapsulation Mechanism (KEM), which in simple terms, consists in "encapsulating" a symmetric key, such that only the receiver can "open" and use this symmetric key. The construction presented in this work has been verified to work with the RSA-KEM, defined in standards track RFC [5] (for which we have run our experiments).

B. Network Architecture

VPN-Zero is built on top of a Distributed Hash Table (DHT) [25], [47]. The DHT is used to privately identify a set of dVPN nodes willing to act as VPN endpoints for some specific traffic. To this end, dVPN nodes store in the DHT each entry of their allowlists using the hash of the domain name as a key, and as a value the domain's public key along with the current node's public IP address.

Figure 1 shows an example, where node A announces that it accepts traffic towards a destination D_1 (magenta dashed line). The hashing uniform distribution property enforces that

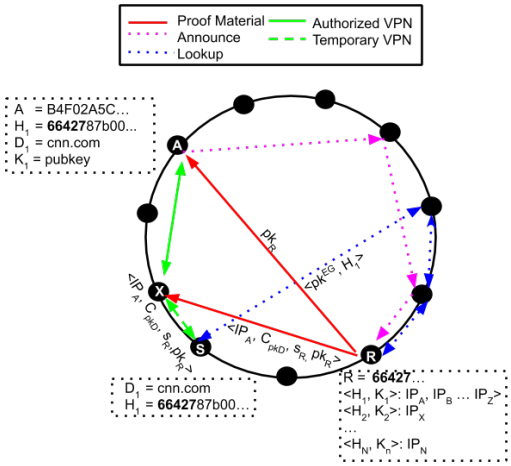

Fig. 1. Flow of a connection using *VPN-Zero*. First, nodes announce their allowlists. Then, when a user wants to connect to a domain, it first uses a temporary VPN, and then performs a lookup to find the node to create an authorized VPN with.

a allowlist is effectively scattered among multiple peers. This is important since allowlists contain privacy sensitive information. Nodes frequently re-publish their allowlist to account for fresh information and to update reachability information, as commonly done in any DHT [25], [47]. A TTL approach is used to handle deletion of entries from the DHT.

When a user S starts a VPN session, it first opens a *temporary* VPN tunnel to some node X (green dashed line); this can be, for instance, a recently used node. Next, S naturally originates some TLS (v1.3) traffic, e.g., a visit to a secure domain D_1. This temporarily *unauthorized* traffic flows through X for a duration T. Within T, X will be able to locate a dVPN node for which this traffic is authorized, if this exists. Meanwhile, S performs a DHT lookup using the hash of D_1 (h_1). We assume an iterative lookup where each step routes the request to a node whose DHT identifier is closer (bitwise) to h_1 (Figure 1, blue dashed line). Differently from a regular DHT lookup, S does not include its IP address to the request but rather X's IP so that the DHT network does not learn which domain S wants to visit. Note that X cannot perform the DHT lookup directly since this would imply knowing D_1. S also appends its ephemeral public key pk_S^{EG} to the lookup.

The lookup converges to a node R which returns to X the IP address of a dVPN node (A) accepting traffic to D_1 — selection strategies can and should be investigated as a future work. Additionally, node R includes in the message to X an encryption of the domain's public key, $C_{\mathsf{pk}_D} = \mathsf{ElGamal.Enc}(\mathsf{pk}_D, \mathsf{pk}_S^{EG})$, together with a signature of the latter, $s_R = \mathsf{Sign}(C_{\mathsf{pk}_D}, \mathsf{sk}_R)$ and its own public key pk_R. Finally R connects with A to send pk_R, which allows A to verify that the accessed domain is among its accepted domains, without learning the domain. Next, X shares A's IP, together with C_{pk_D}, s_R and pk_R with S. Meanwhile, X opens a temporary VPN tunnel to A, the node selected to carry authorized traffic to D_1. When the tunnel is ready, X creates simple `iptables` rules to connect the two VPN tunnels, effectively realizing a VPN *chain* $(S \rightarrow X \rightarrow A)$.

The above VPN chain has an important effect on the existing TLS connection $(S \rightarrow D)$. From D's perspective,

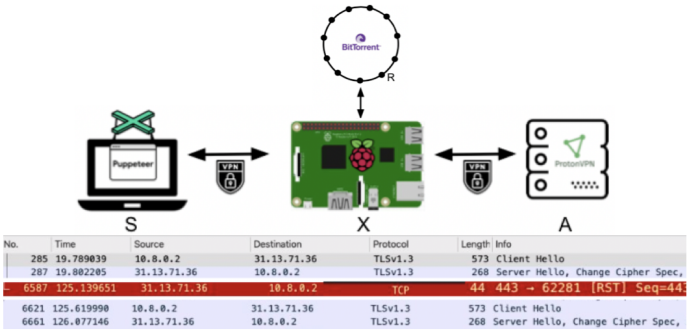

Fig. 2. Overview of *VPN-Zero*'s prototype

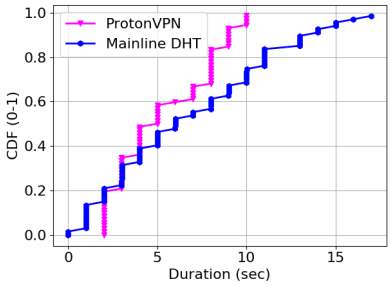

Fig. 3. CDF of lookup duration; ProtonVPN and Mainline DHT.

the endpoint was a socket at X ($IP_X : PORT$). Now, a new socket ($IP_A : PORT$) is introduced. This is similar to TCP hijacking [19] and will be detected by the server, e.g., wrong sequence number, causing a TCP reset connection request (RST). This will travel back to S forcing a new TCP and TLS handshake. The latter detail is very important since it implies that A will observe a new TLS handshake. S uses the Encrypted Client Hello (ECH) of this handshake to prove that the key used in ECH is the one encrypted by R in C_{pk_D}. If the ZKP verification fails, the tunnel is interrupted. Otherwise, this traffic is authorized without A learning the domain's SNI or its public key. A potential avenue of attack here is a collusion between S and R. This translates into a *sybil attack*, a popular attack on the DHT for which many countermeasures exist [9].

C. Zero Knowledge Proof

During the forced re-negotiation of the TLS handshake, S leverages the Encrypted Client Hello mechanism as defined in [38]. The Client Hello (CH) is encrypted using a HPKE algorithm under the domain's public key $C_{CH} = \mathsf{HPK.Enc}(CH, \mathsf{pk}_D)$. This, together with the information received from X : C_{pk_D}, s_R and pk_R are the key components of the ZKP used in *VPN-Zero*.

In a nutshell, the user proves that the key used to encrypt CH is the same as the one encrypted in C_{pk_D}, and that $\mathsf{Verify}(C_{\mathsf{pk}_D}, \mathsf{pk}_R)$ validates. We adopt the Camenisch-Stadler notation [8] to denote such proofs and write:

$$\Pi = SPK\{(\mathsf{pk}_D, CH, \mathsf{sk}_S^{EG}) : C_{CH} = \mathsf{HPK.Enc}(CH, \mathsf{pk}_D) \ \wedge \ \mathsf{Verify}(s_R, \mathsf{pk}_R) = \top \wedge \mathsf{ElGamal.Dec}(C_{\mathsf{pk}_D}, \mathsf{sk}_S^{EG}) = \mathsf{pk}_D\}$$

to denote the non-interactive signature proof of knowledge that the prover knows the public key pk_D used to encrypt CH in C_{CH}, and encrypted in C_{pk_D}, where the latter is signed by pk_R. The values between the parenthesis, $(\mathsf{pk}_D, CH, \mathsf{sk}_S^{EG})$, are kept private while the other values used in Π are public.

Such proof is not straightforward. We firstly prove that a ciphertext, C_{CH}, is the result of an encryption without disclosing neither public key nor plaintext. We use the construction presented in [7] for this purpose. Then we link the public key encrypted in clause two, with the one used in clause one. For this we use a proof that two commitments hide the same secret [4]. Finally the third clause can be openly computed by A given that it received the public key from R. Using this, S can convince A that the tunnel created is to a domain that the latter considers valid, without disclosing which one.

IV. Preliminary Evaluation

This section preliminary evaluates *VPN-Zero*. Apart from the zero knowledge calculation, *VPN-Zero* consists of well known components for which large scale production systems already exist. Instead of building a small scale testbed or some form of emulator/simulator, we have integrated *VPN-Zero*— to the extent that is possible without third party cooperation — with Mainline [44] (Bittorent's DHT based on Kademlia with tens of millions of users), and ProtonVPN [35], a popular VPN provider. We further use OpenVPN [31] to run the first VPN path on a Rasberry Pi we control.

Figure 2 shows a graphical representation of our setup. Please note that we kept the node labeling from Figure 1 as a direct reference. The figure is further enhanced with wireshark data from the ongoing traffic captured at S. A headless browser [17] runs on a laptop (node A) and it is instrumented to visit a TLSv1.3-enabled website. For this test we used `facebook.com` (`31.13.71.36` in the figure).

The laptop connection is tunneled through a Raspberry Pi (node X) located in the same LAN as S; note that this is a worst case scenario for *VPN-Zero* since the extra network latency would further hide our traffic verification process. Whenever a new visit is started, node X performs a DHT query in Mainline. Note that in *VPN-Zero* design this operation is accomplished by A while spoofing X's IP address. Since this would require a modification of `transmission` [43], the BitTorrent Linux client we instrumented for our tests, we opted for a simplification which does not impact the performance evaluation. As input for the DHT lookups, we use the top 100 magnet links (DHT hashes) as indicated by ThePirateBay [1]. As soon as the DHT response is received, our script opens a new tunnel to a random ProtonVPN server (node A) from the list provided with a basic subscription. Meanwhile, the ZKP is calculated at S to be then sent to A for tunnel validation. This latter step was not implemented since it would require collaboration with ProtonVPN.

Figure 3 shows the Cumulative Distribution Function (CDF) of DHT lookup duration in the Mainline DHT for its top 100 hashes. Overall, we observe a uniform distribution between 1 and 20 seconds. Contributions to this delay are: diverse network paths taken by DHT lookups, failures along the path, lookup replication factor, etc. It is worth noting that a

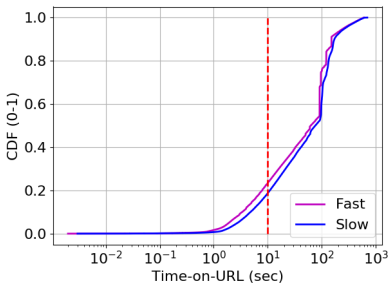

Fig. 4. CDF of time spent on URL ; 1M browser sessions.

similar result was measured for Mainline in [44], and [41] for KAD, the DHT used by eMule [16]. Further, lookup optimizations are possible to speedup these operations, e.g., Steiner et al. [41] show how KAD latency can be halved with no extra load on the DHT.

The next operation post DHT lookup is the setup of the second leg of the VPN chain. Figure 3 also shows the CDF of such VPN setup time computed for 72 VPN nodes — 96 nodes were tested but 24 failed, *i.e.,* negotiation did not succeed within 30 seconds. This high failure rate is potentially due to a ProtonVPN protection for too frequent switches. The figure shows a median reconnect time of 4 seconds and worst case durations of up to 10 seconds.

When putting all together, we estimate that *VPN-Zero* requires a median setup time of 10 seconds, and a worst time of 25-30 seconds. We benchmarked our ZKP calculation using a Python prototype implementation and measured an average of 10 seconds. This indicates that the ZKP calculation can potentially be *VPN-Zero* bottleneck, especially since both DHT lookup and VPN tunnel setup can be optimized, if needed. It is thus clear that our main future work consists in optimizing how the ZKP calculation should be carried to minimize its duration. Our current directions are both an improvement of the protocol such as using Elliptic Curve based (and thus more efficient) KEM [3], and a switch to better performing languages (Rust, C++).

Note that the above latency is currently hidden to the user thanks to *VPN-Zero* design. However, the longer our procedure takes the higher the chance for the user to generate unauthorized traffic. To comment on the latter, we have analyzed one million browsing *session* from Ciao [10], a Chrome plugin which helps discovering and using free HTTP(S) proxies on the Internet. We define a session as the time spent at a specific URL, either manually entered in the browser or opened by clicking on a link. This consists of the page load time plus the actual time the user spends interacting with a page. Note that Ciao only collects page load time, time on site, and bytes transferred. Any other private information like IP address or URL requested are not collected.

Figure 4 shows the CDF of the time spent on a URL for the above browsing sessions. We differentiate between "fast" and "slow" where fast does not account for the PLT which is slower than usual in our dataset since free proxies are used. We further enhance the figure with a vertical line showing *VPN-Zero* median verification time (10 seconds). The figure shows that 20% of the sessions would change *before* traffic verification. Note that this is a worst case analysis since our session definition potentially implies users remaining at a given domain, e.g., by opening a new article on a news site.

TABLE I
COMPARISON OF *VPN-Zero* VERSUS THE EXISTING APPROACHES WITH RESPECT TO THE REQUIREMENTS DESCRIBED IN SECTION II

Requirements	Research	Hola	VPN Gate	Mysterium/ Sentinel	*VPN-Zero*
Open Source	✓	✓	✓	✓	✓
IP Blacklisting	✓	X	✓	✓	✓
QoS Guarantees	X	X	X	X	-
No Logging	X	X	X	X	✓
Traffic Account	X	X	X	✓	✓
Traffic Blame	X	X	X	X	✓

V. RELATED WORK

Zero Knowledge Proofs: In [18], authors introduced a ZKP proof to verify RSA signatures, which could be easily extended to verify RSA encryptions under a certain key. However, for this construction, the knowledge of the public key (modulus and exponent) is necessary. To hide such information, we implemented Camenisch's ZKP of modular exponentiation presented in [7]. However, the range proof included in this paper is not up to date with current state of the art range proofs. Peng and Bao's [33] scheme improves on previous work, which is the construction we have used for *VPN-Zero*.

Distributed Virtual Private Networks: The idea of a dVPN is far from novel, with many designs dating now more than 10 years back, *i.e.,* when P2P research was at its peak. To the best of our knowledge, ELA [2] was one of the first approach to decentralize a VPN. Several other variations then appear, such as SocialVPN [24] a system which drives the peer selection strategy based on social relationships among nodes, or N2N [13] where users share a common encryption key obtained when they join. None of this approach provides strong privacy guarantees as *VPN-Zero*. However, we start noticing a trend towards protecting user identity and traffic, e.g., only routing it through friends and providing strong online identities.

With the recent rise of blockchain, a new form of dVPNs has surfaced. In such, the rationale is to share a user's upload bandwidth in exchange for some crypto tokens such as Filecoin [34]. Another popular example is Mysterium [42], an open source dVPN completely built upon a P2P architecture.

VI. DISCUSSION

In its current design, *VPN-Zero* suffers from a limitation when operating on traffic towards a CDN. The `ClientHello` (and thus the SNI) is encrypted using the CDN's public key, so that the CDN can decrypt, and decide where to send the connection – for which it needs to extract the SNI. In presence of a CDN, the current design of `ClientHello`/SNI encryption limits *VPN-Zero* visibility to which CDN serves some traffic rather than which websites, which reduces the granularity of the allowlists. We plan to address this in future work exploring solutions like `DECO` [46], where using an interactive protocol, a prover can convince a

verifier of the origin (the website) of some data. In our case, the requirement of interactiveness is implicit in the protocol, and hence, using such a solution would require no added rounds of communication.

The above issue disappears in the context of IPFS and libP2P [22], a new generation of overlay network which may allow to remove CDNs. Libp2p allows to create an application endpoint over different transports and uses a peer identifier to advertise the service in the network. Any connected computer can host a website together with x509 certificate, thus enabling our proving scheme. To scale, libp2p allows to replicate services among trusted peers, which hold a copy of the original x509 certificate, thus allowing *VPN-Zero* to properly work.

REFERENCES

[1] Anonymous. The pirate bay. https://www.thepiratebay.org, 2019.

[2] S. Aoyagi, M. Takizawa, M. Saito, H. Aida, and H. Tokuda. Ela: a fully distributed vpn system over peer-to-peer network. In *The 2005 Symposium on Applications and the Internet*, Feb 2005.

[3] Richard Barnes, Karthikeyan Bhargavan, and Christopher A. Wood. Hybrid Public Key Encryption. Internet-Draft draft-irtf-cfrg-hpke-04, Internet Engineering Task Force, May 2020. Work in Progress.

[4] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology — EUROCRYPT 2000*, pages 431–444, Berlin, Heidelberg, 2000. Springer.

[5] John Brainard, Burt Kaliski, Sean Turner, and James Randall. Use of the RSA-KEM Key Transport Algorithm in the Cryptographic Message Syntax (CMS). RFC 5990, September 2010.

[6] Martin Brinkmann. Beware: Hola vpn turns your pc into an exit node and sells your traffic. https://www.ghacks.net/2015/05/28/\\beware-hola-vpn-turns-your-pc-into-an-exit-node-and-sells-your-traffic/, 2015.

[7] Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *EUROCRYPT '99*, pages 107–122, Berlin, Heidelberg, 1999. Springer.

[8] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In *Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '97, pages 410–424, 1997.

[9] Thibault Cholez, Isabelle Chrisment, and Olivier Festor. Evaluation of sybil attacks protection schemes in kad. In *IFIP International Conference on Autonomous Infrastructure, Management and Security*, pages 70–82. Springer, 2009.

[10] CIAO Team. Automated free proxies discovery/usage. https://chrome.google.com/webstore/detail/automated-free-proxies-di/ojjklffhhhfpeaelghfocilljceokage?hl=en.

[11] Catalin Cimpanu. Hacking vpn servers to plant backdoors in companies. https://www.zdnet.com/article/iranian-hackers-have-been-hacking-vpn-servers-to-plant-backdoors-in-companies-around-the-world/, 2020.

[12] Cloudlfare. Browsing experience security check. https://www.cloudflare.com/ssl/encrypted-sni/, 03/2021.

[13] Luca Deri and Richard Andrews. N2n: A layer two peer-to-peer vpn. In *IFIP International Conference on Autonomous Infrastructure, Management and Security*, pages 53–64. Springer, 2008.

[14] Martin Dittus, Joss Wright, and Mark Graham. Platform criminalism: The 'last-mile' geography of the darknet market supply chain. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 277–286, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee.

[15] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in Cryptology*, pages 10–18, 1985.

[16] Merkur et al. Emule. https://www.emule-project.net/home/perl/general.cgi?l=1, 2019.

[17] Google. Puppeteer. https://github.com/GoogleChrome/puppeteer, 2019.

[18] Louis C. Guillou and Jean-Jacques Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In *EUROCRYPT'88*, 1988.

[19] Brendon Harris and Ray Hunt. Tcp/ip security threats and attack methods. *Computer communications*, 22(10):885–897, 1999.

[20] P. Hoffman. Dns queries over https (doh). https://tools.ietf.org/html/rfc8484, 2018.

[21] Muhammad Ikram, Narseo Vallina-Rodriguez, Suranga Seneviratne, Mohamed Ali Kaafar, and Vern Paxson. An analysis of the privacy and security risks of android vpn permission-enabled apps. In *Proc. ACM IMC*, pages 349–364. ACM, 2016.

[22] IPFS Team. Inter-Planetary File System. https://ipfs.io/.

[23] Mark Jackson. Ipact - controversial new uk isp internet snooping bill becoming law. https://www.ispreview.co.uk/index.php/2016/11/controversial-new-uk-internet-snooping-bill-approved-mps.html, 2016.

[24] Pierre St Juste, David Wolinsky, P Oscar Boykin, Michael J Covington, and Renato J Figueiredo. Socialvpn: Enabling wide-area collaboration with integrated social and overlay networks. *Computer Networks*, 2010.

[25] M Frans Kaashoek and David R Karger. Koorde: A simple degree-optimal distributed hash table. In *International Workshop on Peer-to-Peer Systems*, pages 98–107. Springer, 2003.

[26] Jacob Kastrenakes. Congress just cleared the way for internet providers to sell your web browsing history. https://www.cnbc.com/2017/03/28/congress-clears-way-for-isps-to-sell-browsing-history.html, 2017.

[27] Mohammad Taha Khan, Joe DeBlasio, Geoffrey M Voelker, Alex C Snoeren, Chris Kanich, and Narseo Vallina-Rodriguez. An empirical analysis of the commercial vpn ecosystem. In *Proc. ACM IMC*, 2018.

[28] Jeremy Kirk. Nordvpn says server compromised due to misconfiguration. https://www.bankinfosecurity.com/nordvpn-says-server-compromised-due-to-misconfiguration-a-13278, 2019.

[29] Alexander J Martin. Do you use hola vpn? you could be part of a ddos, content theft - or worse. https://www.theregister.co.uk/2015/06/10/hola_gets_holes_poked_in_client_lulzsec/, 2015.

[30] Daiyuu Nobori and Yasushi Shinjo. VPN gate: A volunteer-organized public VPN relay system with blocking resistance for bypassing government censorship firewalls. In *Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*, pages 229–241, Seattle, WA, 2014. USENIX.

[31] OpenVPN. Open vpn. https://openvpn.net/, 2019.

[32] ownVPN.io. Create your own vpn. https://proprivacy.com/guides/create-your-own-vpn-server, 2019.

[33] Kun Peng and Feng Bao. An efficient range proof scheme. In *Proceedings of the 2010 IEEE Second International Conference on Social Computing*, SOCIALCOM '10, pages 826–833, Washington, DC, USA, 2010. IEEE Computer Society.

[34] Protocol Labs. Filecoin, a decentralized storage system. https://filecoin.io/, 2021.

[35] ProtonVPN. Proton vpn. https://protonvpn.com/, 2019.

[36] E. Rescorla. Tls 1.3 rfc. https://tools.ietf.org/html/rfc8446, 2018.

[37] Eric Rescorla, Kazuho Oku, Nick Sullivan, and Christopher A. Wood. Encrypted Server Name Indication for TLS 1.3. Internet-Draft draft-ietf-tls-esni-03, Internet Engineering Task Force, 3 2019. Work in Progress.

[38] Eric Rescorla, Kazuho Oku, Nick Sullivan, and Christopher A. Wood. TLS Encrypted Client Hello. Internet-Draft draft-ietf-tls-esni-07, Internet Engineering Task Force, June 2020. Work in Progress.

[39] Sentinel.co. Sentinel: Interoperable network layer for distributed resources. https://sentinel.co/.

[40] Craig Silverman. Popular vpn and ad-blocking apps are secretly harvesting user data. https://www.buzzfeednews.com/article/craigsilverman/vpn-and-ad-blocking-apps-sensor-tower, 2020.

[41] Moritz Steiner, Damiano Carra, and Ernst W Biersack. Evaluating and improving the content access in kad. *Peer-to-peer networking and applications*, 2010.

[42] The Mysterium Network. Mysterium network: Decentralised vpn built on blockchain. https://mysterium.network/.

[43] TransmissionBT. Transmission. https://transmissionbt.com/, 2019.

[44] Matteo Varvello and Moritz Steiner. Traffic localization for dht-based bittorrent networks. In *International Conference on Research in Networking*, pages 40–53. Springer, 2011.

[45] Ofer Vilenski and Derry Shribman. Hola free vpn - unblock any website. https://hola.org/, 2020.

[46] Fan Zhang, Deepak Maram, Harjasleen Malvai, Steven Goldfeder, and Ari Juels. DECO: liberating web data using decentralized oracles for TLS. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 1919–1938. ACM, 2020.

[47] Ben Y Zhao, Ling Huang, Jeremy Stribling, Sean C Rhea, Anthony D Joseph, and John D Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on selected areas in communications*, 22(1):41–53, 2004.