A Comprehensive and Long-term Evaluation of Tor V3 Onion Services

Chunmian Wang[†], Junzhou Luo[†], Zhen Ling[†*], Lan Luo[†], Xinwen Fu[‡]
[†]School of Computer Science and Engineering, Southeast University
Email: {chunmianwang, jluo, zhenling}@seu.edu.cn, lanluo448@gmail.com
[‡]Department of Computer Science, University of Massachusetts Lowell, Lowell, MA, USA
Email: xinwen_fu@uml.edu

***Abstract*—The version 3 (V3) Tor onion service protocol deeply hides onion service domain names to improve anonymity. Existing onion service analysis methods cannot be used any more to understand V3 onion services and the ecosystem such as benign services, abuses and black markets. To understand the scale of V3 onion services, we theoretically analyze the V3 onion service mechanism and propose an accurate onion service size estimation method, which is able to achieve an estimation deviation of 2.43% on a large-scale emulated Tor network. To understand onion website popularity, we build a system and collect more than two years of data of public onion websites. We develop an onion service popularity estimation algorithm using online rate and access rate to rank the onion services. To reduce the noise from the phishing websites, we cluster onion websites into groups based on the content and structure. To our surprise, we only find 487 core websites out of the collected 45,889 public onion websites. We further analyze the weighted popularity of each group using yellow page data and discover that 35,331 phishing onion websites spoof the 487 core websites.**

I. Introduction

Tor is the most popular anonymous communication network. In order to protect the anonymity of onion service providers, researchers introduced an onion service mechanism [1] in 2003 and deployed version 2 (V2) of the onion service on the Tor network in 2004. When an onion service is first started, it generates an onion address, i.e., a domain name of an onion service, which clients can use to access the service. A 6-hop circuit established over 6 Tor nodes is used to achieve two-way anonymous communication between the client and the onion service. However, Version 2 of the onion service protocol has many issues. For example, adversaries can harvest onion addresses on the Tor network by deploying hidden service directories (HSDirs) that are organized on a distributed hash table (DHT). Therefore, Tor deprecates the V2 onion service in 2021 [2] and completely migrates to version 3 (V3) of the onion service. Although a new blinding protocol [3] applied to the V3 onion services effectively keeps the onion service addresses from being directly exposed to the HSDirs, it causes the abuse of the onion services (e.g., botnet servers [4], phishing websites [5] and illegal black markets [6]) to be even hard to be discovered.

To understand the abuse issue, we scrutinize the designs and implementation details of the V3 onion service protocol and perform long-term and comprehensive analysis in an attempt to shed a light on the new V3 onion service on the Tor network. First, we propose an accurate estimation method to estimate the number of V3 onion services on the Tor network. In each period, an onion service generates a specific blinded public key, which is encapsulated in a descriptor and uploaded to several HSDirs. Since a blinded public key can only be associated with an onion service in one day, the recorded number of blinded public keys in such short period of time can be used to estimate the number of V3 onion services. On the basis of this observation, we deploy a HSDir on the Tor network to collect descriptors uploaded by onion services and parse the blinded public keys from the descriptors. After deriving the number of blinded public keys collected by a HSDir, we leverage the position of the HSDir on the DHT to compute a capture probability so as to estimate the number of blinded public keys and thus learn the total number of onion services on the Tor network.

We classify onion services into public and private ones, and analyze the public onion services that can be retrieved via search engines. We design a system to collect public V3 onion addresses and inspect their aliveness and access frequency. We develop a novel popularity estimation algorithm of onion services using their online time and access frequency in an attempt to rank the onion services. Since most of the onion services host Web services, we focus on the rank of onion websites in this paper. We find that a large number of phishing websites significantly affect the ranking results. To mitigate this issue, we leverage a tree edit distance to measure the similarity of Document Object Model (DOM) trees between onion websites so as to cluster the websites into groups. We select the most popular websites, defined as core websites, within the website groups using weighted group popularity and then identify the phishing sites in each group.

Our major contributions are summarized as follows.

- We are the first to propose an accurate Tor V3 onion service size estimation method. We verify our method using a large-scale emulated Tor network that includes 300,000 onion services and achieve an average estimation deviation rate of 2.43% within one day. By deploying 10 HSDirs on the real Tor network, we discover that the current onion service size is around 900,000.
- We design a system to collect public V3 onion services and leverage our HSDirs to observe the behavior of the onion services for more than two years. We analyze the long-term data and develop an onion service popularity

* Corresponding author: Prof. Zhen Ling of Southeast University, China.

algorithm based on online rate and access rate to estimate the onion website rank. Surprisingly, we discover that only 487 out of 45,889 public onion websites are selected as core websites via our approach. 5 out of the first 6 most popular onion websites are black markets.

- We further explore the reason why we have such a small size of core websites. We leverage the structure and content of the onion websites to cluster these sites and discover that 35,331 phishing onion websites impersonating the 487 core websites. Most of the spoofed core websites fall into the *Finance* category. Attackers may try to make profits via so many phishing websites.

II. Onion Service Mechanism

In this section, we first present the latest version 3 of Tor onion service mechanism and then introduce the key building blocks of the mechanism in detail, including onion service address generation, onion service publication, and onion service connection establishment.

A. Overview of Tor onion service mechanism

Tor onion service mechanism refers to enabling users to publish and access internet services anonymously, such as Web and SSH. The onion service mechanism involves six types of Tor nodes as follows:

- **Tor Client.** A Tor client is a Tor program installed on the client side to provide a local Tor proxy that only supports TCP applications. A client application (e.g., a web browser) accesses Internet through the Tor client.
- **Onion Service.** An onion service supports various TCP services such as Web service. The TCP services can be deployed to provide their services via a Tor client.
- **Introduction Point (IPO).** An IPO is selected by an onion service to act as a front-end reverse proxy of the onion service so as to protect the real IP address of the onion service from being exposed. Moreover, the IPO receives the connection requests from the Tor client and forwards them to the onion service.
- **Rendezvous Point (RPO).** A RPO is a Tor relay node selected by the Tor client to concatenate the connections from the onion service and the Tor client.
- **Hidden Service Directory (HSDir).** A HSDir is a special type of Tor relay node on the Tor network. It not only works as a normal Tor relay node to transmit Tor data, but also receives and stores the information of onion services (e.g., IPOs and public keys) and responses the queries from Tor clients to allow them to download the onion service information. In order to locate a specific HSDir with an onion service address, all HSDirs are organized via a distributed hash table (DHT).
- **Authority.** An Authority is an authoritative node officially deployed by Tor. IP addresses of 9 Authorities are hard-coded in the Tor program to allow users to access the Tor network for the first time. They are responsible for maintaining and storing all legitimate Tor node information into a consensus file. Each Tor client and Tor node periodically downloads the consensus file to derive all of the latest Tor node information.

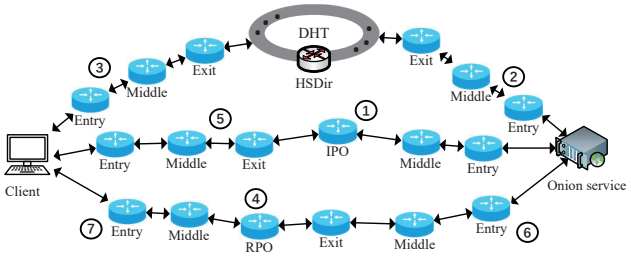

Fig. 1: Onion service communication mechanism

Figure 1 shows the onion service communication process. When a Tor onion service starts for the first time, it generates a master identity public-private key pair and uses the public key to compute the onion service address. ① The onion service then randomly selects multiple relays as IPOs (the default is 3) and establishes a 3-hop circuit to each IPO. ② Once the IPOs are selected, the onion service generates an *onion service descriptor* that contains information of the IPOs. Finally, the onion service uploads the descriptor to 8 responsible HSDirs on the DHT. ③ When a Tor client accesses a target onion service, it calculates the location of the responsible HSDirs on the DHT using information such as onion address, and then retrieves the latest descriptor of the onion service from the responsible HSDirs. ④ The Tor client selects a Tor relay node as the RPO, establishes a circuit to the RPO and generates a rendezvous cookie as the authentication for the onion service. ⑤ After the client decrypts the descriptor to get the IPO, it establishes a 3-hop circuit to the IPO and sends a request, including the RPO information and rendezvous cookie, to the onion service via the IPO. ⑥ When the onion service receives the request, it establishes a 3-hop circuit to the RPO and uses a rendezvous cookie to complete the authentication with the RPO. ⑦ Then the RPO can concatenate the two circuits from the client and the onion service. Finally, the Tor client can communicate with the onion service via the 6-hop circuit.

B. Onion address generation

The V3 onion service address is generated using the master identity public key of the onion service, the key checksum and the version number. The master identity key of the onion service is an asymmetric key generated by the ED25519 algorithm [7]. The V3 onion address is formed using Base32 to encode the concatenated bytes of the public key of the master identity key, the public key checksum and the version of the protocol, and then appending a suffix string of ".onion" to derive a 62 bytes string as shown in Equation (2).

$$onion_{v3} = \text{Base32}(P_k|C_s|V)|\text{“}.onion\text{”} \quad (1)$$

where P_k is the public key of the master identity key, C_s is a checksum of the P_k and V is the version of the protocol (i.e., 3). The checksum is computed as follows:

$$C_s = \text{SHA3-256}(\text{“}.onion\ checksum\text{”}|P_k|V)[0:1] \quad (2)$$

The first two bytes of the SHA3-256 value is used to represent the checksum.

C. Onion service publication

To publish the onion service descriptor, the onion service first uses a consensus file to derive a DHT so as to look up responsible HSDirs. The index of each HSDir on the DHT is calculated by

$$\mathcal{I} = H(\text{“node-idx”}|P_{id}|S|N_p|T), \tag{3}$$

where P_{id} is the identity public key of the HSDir, S is a shared random value published in the consensus file, N_p is the number of time period and T is a time period. The shared random value S is used to prevent attacks from predicting the location of HSDirs on the DHT for publishing an onion service descriptor. The shared random value is negotiated among Authorities and published at 0:00 am in the consensus file. Moreover, the consensus file contains two shared random values of the current time period and the previous time period. The number of time period N_p is calculated using the Unix epoch and 12:00 UTC is the start time of each time period, so that the first time period starts at 12:00 UTC on January 1, 1970. The default time period T is 1440 minutes, i.e., one day. The HSDirs are placed on the DHT in terms of their indices, and the DHT space is 2^{256}. Since the indices of the HSDirs changes in terms of shared random value and the number of time period each day as shown in Equation (3), the DHT should be theoretically regenerated at 0:00 am in each time period as shown in Figure 2.

After deriving the latest DHT, the onion service calculates the indices of its descriptors and maps them to the DHT so as to select the responsible HSDirs for uploading the descriptor. The indices of the descriptors can be calculated by

$$\begin{aligned} &for\ r\ in\ 1 \ldots R: \\ &\quad ID_r = H(\text{“store-at-idx”}|P_b|r|T|N_p), \end{aligned} \tag{4}$$

where $R \in [1, 16]$ (by default R is equal to 2), and P_b is the blinded public key. The blinded public and private keys are periodically generated by using N_p and the onion service identity public and private keys, respectively. When R uses the default value 2, we can have two different indices of the onion service descriptor, i.e., ID_1 and ID_2. Each ID can be mapped on the DHT. Then the onion service selects 4 consecutive HSDirs on the DHT as the responsible HSDirs whose indices immediately follow the ID. If a responsible HSDir is already selected, the onion service can skip it and select a following HSDir as the responsible HSDir. Upon selecting the 8 responsible HSDirs, the onion service uploads the descriptors to them and randomly picks the next uploading time ranging from 60 to 120 minutes.

Once a HSDir receives the uploaded descriptors, it first verifies the format of the descriptors and stores the descriptors. The blinded public key included in the descriptor is used as a hash table index that corresponds to the descriptor, so that the HSDir can quickly look up the stored descriptor given the blinded public key.

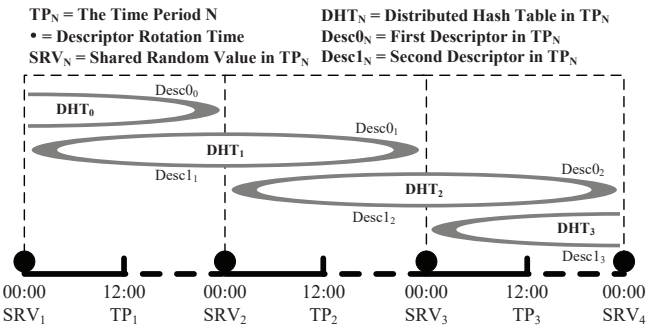

Fig. 2: DHT update process

D. Onion service access

Once a user obtains an onion service address via an out-of-band way (e.g., a public Web forum), she can enter the onion address in the browser that is configured to use the Tor client as the local proxy. Then the Tor client constructs the DHT using the current N_p and the S in the consensus file, and calculates the IDs using Equation (4) to look up the responsible HSDirs of the onion service. The Tor client extracts the master identity public key P_k in the onion address as shown in Equation (1), and generates the blinded public key using P_k and N_p. Afterwards, it randomly selects a responsible HSDir, and sends the blinded public key to the HSDir so as to download the descriptor. Note that the onion service selects the continuous indices of 4 responsible HSDirs after each *ID* on the DHT, while the client only downloads descriptors from the first 3 responsible HSDirs for each *ID*. The rest two HSDirs are backup responsible ones. Next, the client randomly selects a Tor relay node as the RPO and then builds a 3-hop connection to the RPO.

After downloading the descriptor, the Tor client can derive the IPOs of the onion service from the descriptor. Then the Tor client randomly generates a 20-byte $rend\_cookie$ and sends it to the onion service through the IPOs. The onion service establishes a 3-hop connection with the RPO and sends the $rend\_cookie$ to it for authentication purpose. Once the RPO authenticates the connection, it concatenates the two communication connections between the client and the onion service in order to allow the Tor client and onion service to anonymously communicate with each other.

III. Methodology

In this section, we first introduce the overview of our analysis methods on the V3 onion services and then introduce a theoretical method to estimate V3 onion service size by analyzing the V3 onion service mechanism. Further, we present a novel popularity algorithm to rank public V3 onion services and derive the core websites on the Tor network.

A. Overview

Our objective is to achieve a comprehensive analysis of the latest V3 onion services on the Tor network. To this end, we first estimate the number of onion services. Since an onion service generates a unique blinded public key in every day and uploads it to HSDirs, we can deploy a HSDir on the Tor network to receive the descriptors and then count the

number of different blinded public keys. We leverage the capture probability of the HSDir in terms of the location of the HSDir on the DHT to estimate the total number of onion services. Then we further analyze public onion services that can be crawled via search engines. We propose an onion service popularity algorithm to rank the public onion services by their online time and access frequency. Since we discover that a large number of phishing onion services that can affect the popularity results, we leverage a tree edit distance to measure the similarity of the DOM tree of any two websites, and then cluster them into groups based on the similarity so as to find out the phishing sites. Next, we leverage the popularity of onion service addresses on well-known yellow pages to compute the weighted popularity for each group. Finally, we can reveal the core onion website group on the Tor network using an empirical threshold.

B. V3 onion service size estimation

We deploy several HSDirs and then leverage the blinded public keys collected by our controlled HSDirs in each time period, i.e., one day, to evaluate the total number of V3 onion services. Recall that a new blinded public key is generated by an onion service in each time period and enveloped into two descriptors that are uploaded to 8 HSDirs by default. Therefore, we only have one day to use our deployed HSDirs to collect the blinded public keys in the descriptors so as to evaluate the total number of onion services. Since the DHTs and the IDs of onion service descriptors can be changed periodically, we should choose an appropriate *evaluation time period (ETP)* in which the mapping relationship between the descriptors and their responsible HSDirs cannot be changed so as to estimate the number of V3 onion service in the ETP. According to Equation (3) and Equation (4), we choose the ETP ranging from 0:00 UTC to the next 0:00 UTC.

Without loss of generality, we first estimate the total number of onion services using one controlled HSDir that resides on one of two DHTs. Note that there are always two DHTs during an ETP as shown in Figure 2. For simplicity, we illustrate how to evaluate the onion service size using the old DHT in an *ETP*. It is assumed that there are N HSDirs on the Tor network. We denote the index of the i^{th} HSDir on the DHT by $\mathcal{I}_i$ ($i \in [0, N-1]$) that is calculated in terms of Equation (3). The interval between the i^{th} and $i-1^{th}$ HSDir is denoted by L_i, where $L_i = \mathcal{I}_i - \mathcal{I}_{i-1}$. Recall that, if a descriptor ID falls in the interval L_i, the i^{th} HSDir can receive the descriptor. Then the probability that a descriptor ID falls in the interval L_i can be computed by

$$p_i = \frac{L_i}{\sum_{j=0}^{N-1} L_j} \tag{5}$$

We define a *capture probability* that a HSDir receives a new blinded public key in an ETP. Note that an onion service generates two descriptors that contain the same blinded public key and each descriptor is uploaded to 4 consecutive HSDirs in terms of ID as shown in Figure 3. Let B_i be the event that the i^{th} HSDir collects an onion service descriptor. Discrete random variables A_1 and A_2 indicate that the descriptor ID_1 and ID_2 of the onion service fall in a range of the DHT, respectively. $a1$ and $a2$ are the ranges of $(\mathcal{I}_{i-4}, \mathcal{I}_i]$ and $(\mathcal{I}_{i-8}, \mathcal{I}_{i-4}]$ in which a descriptor ID falls, respectively, and $a3$ indicates that the descriptor ID falls in a location other than $a1$ and $a2$ (i.e., $ID \notin (\mathcal{I}_{i-8}, \mathcal{I}_i]$) as shown in Figure 4. According to law of total probability, the *capture probability* P_i^1 of the i^{th} HSDir on the old DHT becomes $P_i^1 = P(B_i) = P(A_1 = a1)P(B_i|A_1 = a1) + P(A_1 = a2)P(B_i|A_1 = a2) + P(A_1 = a3)P(B_i|A_1 = a3)$. Law of total probability is then used to compute three conditional probabilities, i.e., $P(B_i|A_1 = a1)$, $P(B_i|A_1 = a2)$, and $P(B_i|A_1 = a3)$. The *capture probability* becomes

$$\begin{aligned} P_i^1 = &\sum_{j=i-3}^{i} p_j + \sum_{j=i-7}^{i-4} p_j \sum_{j=i-7}^{i} p_j \\ &+(1 - \sum_{j=i-7}^{i} p_j) \sum_{j=i-3}^{i} p_j \end{aligned} \tag{6}$$

Please refer to Appendix VII for detailed derivation of Equation (6).

Then, the total estimated number of onion services on the old DHT can be evaluated by

$$\hat{E_i^1} = \frac{X_i^1}{P_i^1}, \tag{7}$$

where X_i^1 is the number of distinct blinded public keys received by the i^{th} HSDir in an ETP.

Since an onion service maintains two different DHTs during an ETP, we discuss how to evaluate the onion service size using two DHTs. Denote distinct blinded public keys collected in the latest DHT in an ETP as X_i^2. X_i is the number of distinct blinded public keys received by the i^{th} HSDir on both the old and the latest DHT. We have

$$X_i = X_i^1 + X_i^2 = \hat{E_i^1} * P_i^1 + \hat{E_i^2} * P_i^2, \tag{8}$$

where $\hat{E_i^2}$ is the estimated onion service size using the latest DHT and P_i^2 is the capture probability that the i^{th} HSDir obtains a blinded public key of an onion service on the latest DHT. Since HSDirs organized on the two DHTs receive the same number of distinct blinded public keys, the estimated onion service size is theoretically the same, i.e., $\hat{E_i^1} = \hat{E_i^2}$. The total estimated onion service size becomes

$$\hat{E_i} = \frac{X_i}{P_i^1 + P_i^2}, \tag{9}$$

where $\hat{E_i} = \hat{E_i^1} = \hat{E_i^2}$.

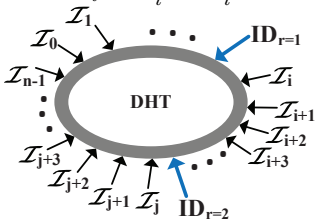

Fig. 3: DHT structure

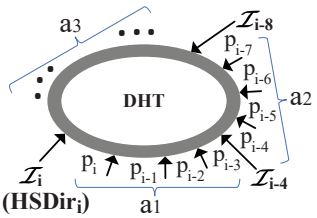

Fig. 4: The 3 cases of descriptor ID falling on a DHT

C. V3 onion service popularity and core websites

Once we derive the estimated size of onion services, we build a collection system to collect V3 onion addresses, and

then try to comprehensively analyze the popularity of V3 onion services so as to discover the core websites on the Tor network. We first classify the onion services into two groups: *public onion services* and *private onion services*. Onion services whose addresses can be searched through public search engines, e.g., Google and Bing, are referred to as public onion services in this paper. The owners of these onion services intend to publish their service to users around the world by posting their onion addresses to various public forums. The rest onion services, referred to as private onion services, are only available to users who derive the onion addresses from the owners of onion services via out-of-band channels. In contrast to the V2 onion service mechanism, the enhanced V3 onion service mechanism can keep the HSDirs from collecting the onion addresses. Consequently, we can only collect the public onion addresses. We build a V3 onion address collection system to derive the public V3 onion addresses as depicted in Section IV-A.

Intuitively, since the online time of a public V3 onion service can reflect how popular an onion service is, we propose an onion service aliveness detection approach without compromising the privacy of an onion service. We leverage the onion service publication mechanism to detect the aliveness of an onion service so that we can tackle the risk of endangering the privacy of the onion service. Recall that onion services periodically upload descriptors to the responsible HSDirs, enabling the clients to obtain the IPO information to access the onion service. Therefore, we can check the online status of the onion service by inspecting the existence of the descriptors of the target onion service on the Tor network. To determine the existence of descriptors, we first extract the public key of the public onion address in terms of Equation (1) so as to derive the blinded public key of the onion service. Then indices of all the HSDir on the current DHT are calculated via Equations (3). Next, we can obtain the indices of responsible HSDirs of the target onion service using the blinded public key of the target onion service in terms of Equation (4). Then we emulate a Tor client to create a 3-hop circuit to the responsible HSDirs of each public onion service and send requests with the blinded public keys to download the descriptors. If the responsible HSDirs respond the requests with a descriptor, we can infer that the target onion service is online. Otherwise, it is offline. Since our approach is a non-intrusive one that does not initiate a connection to a target onion service, our detectors cannot compromise the privacy of the onion service. Moreover, the accuracy of the detection is 100% in the light of the onion service publication mechanism. In practice, our approach is used to inspect all onion addresses periodically, i.e., one day, and thus obtain the online time of all public onion services. To more objectively represent the online status of an onion service, we define the *online rate* of an onion service as the ratio of the online time of the onion service to the total monitored time. Let o_i and O_i be the number of online days and the total monitored days of the i^{th} onion service after we discover it, respectively, then the online rate of the target onion service is

$$\mathcal{R}_o^i = \frac{o_i}{O_i} \tag{10}$$

In addition, we leverage the access frequency to evaluate the popularity of onion services. Recall that, when a client accesses the target onion service, it calculates the blinded public key using the onion address and requests a descriptor from a responsible HSDir with the blinded public key. Therefore, the number of descriptor requests received by the responsible HSDirs in one day can represent the access frequency of the onion service. We deploy HSDirs on the Tor network to passively collect descriptor requests from clients. After collecting the blinded public keys, we correlate the blinded public keys with the public onion addresses, since the blinded public key is generated using a one-way hash function, i.e., the blinded public key can be derived from the onion address. By counting the number of requests for a blinded public key, we can derive the frequency of the corresponding onion service accessed by clients.

We propose a novel onion service popularity evaluation approach to ranking public onion services based on the online rate and access frequency, rather than based on the content [8]–[10] that cannot precisely reflect the real rank of onion services. Suppose that there are n public V3 onion services, and the access frequency of the i^{th} public V3 onion service is K_i. Then we define an *access rate* of each onion service as a normalized access frequency, i.e.,

$$\mathcal{R}_k^i = \frac{\ln(K_i + 1)}{max\{\ln(K_1 + 1), \ldots, \ln(K_n + 1)\}} \tag{11}$$

We use logarithm on the access frequency so as to reduce the large differences between the access frequencies of different onion services. Then the popularity $\mathcal{U}_i$ of i^{th} public V3 onion service becomes

$$\mathcal{U}_i = (1 - \alpha)\mathcal{R}_o^i + \alpha\mathcal{R}_k^i \tag{12}$$

where α is the parameter used to adjust the weight of the online time and access frequency. Since most of the public onion services are Web services, we can find the most popular core websites from the public onion services via our method. However, we find the mirror and phishing websites [5] on the Tor network that may affect our evaluation results.

To identify the phishing and mirror websites on the public onion services, we evaluate the edit distance between Document Object Model (DOM) trees of any two websites so as to cluster the near-identical websites into a group. We extract the homepage of each onion service and derive the DOM trees by analyzing the HTML code. We only keep the tag name, class attribute and content of each node in the DOM tree so as to preserve the structure, style and visual content features of homepages. Then we leverage a tree edit distance method to compute the minimum costs of a sequence of node edit operations, e.g., insertion, deletion or substitution, so as to transform one tree into another. We define the cost of both the insertion and deletion operations as 3. For the substitution operation, if the tag names are different, the substitution cost is 3. Otherwise, we compare the style attribute and text

content respectively, each of which has a substitution cost of 1. Afterwards, we accumulate all of the operation costs to derive the tree edit distance. The smaller the tree edit distance, the more similar the two homepages are. Let $\mathcal{C}(T_a, T_b)$ be the costs between the website homepage T_a and T_b. Denote the number of HTML tags of T_a and T_b by $|T_a|$ and $|T_b|$, respectively. We then normalize the similarity by

$$\mathcal{S}(T_a, T_b) = 1 - \frac{\mathcal{C}(T_a, T_b)}{3 * (|T_a| + |T_b|)} \tag{13}$$

After deriving the normalized similarity, we use a density-based clustering algorithm, i.e., *DBSCAN* [11], to cluster website homepages of the public onion services so as to discover near-identical websites.

We compute the popularity of each group and derive the final rank of the groups. We accumulate the access frequency of each onion service for each group, and use the largest online rate of onion services for each group. Then we can obtain the raw popularity of the i^{th} group, i.e., $\mathcal{G}_i$, using Equation (12). Afterwards, we further leverage onion addresses recorded in various yellow websites from Internet and onion services to objectively evaluate the popularity of onion services. The yellow websites maintain a series of well-known public onion addresses to allow users to easily discover these onion services. Then we define the *appearance rate* as $\frac{m_i}{\mathcal{M}}$ to show how popular the onion services in the i^{th} group in these yellow pages, where m_i is the number of yellow pages that include any onion address in the i^{th} group and $\mathcal{M}$ is the total number of yellow pages. Then the weighted group popularity becomes

$$\mathcal{W}_i = (1 - \beta)\mathcal{G}_i + \beta\frac{m_i}{\mathcal{M}} \tag{14}$$

where β is used to adjust the weight of the group popularity and the appearance rate. After deriving the group popularity, we can use an empirical threshold to obtain the top $\mathcal{K}$ popular groups. If there are multiple onion services in a group, we choose the onion service with the largest popularity $\mathcal{U}$ to represent this group. Finally, we can derive the top $\mathcal{K}$ core websites.

IV. Evaluation

In this section, we first introduce the implementation of the onion service address collection and detection system. Next, we verify the effectiveness of V3 onion service size estimation method, and evaluate the popularity of collected public onion services. Finally, we cluster near-identical onion websites to derive the core websites and discover the phishing websites.

A. Implementation

As shown in Figure 5, our *public V3 onion address collection* module crawls an initial set of onion addresses from some yellow pages such as DARKWEBLINKS [12] and hidden wiki [13]. It then searches the collected initial onion addresses via two public search engines, i.e., Google and Bing, and two onion service search engines deployed in a Tor onion service, i.e., Ahmia [14] and TorDex [15], so as to crawl new onion addresses from the searched pages. We also crawl web pages from the collected onion services to further derive new onion addresses. We store the new extracted onion addresses in a database and ensure they are all crawled.

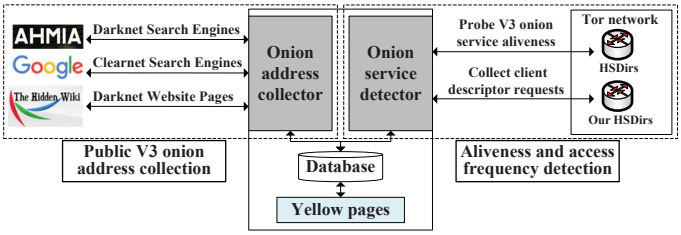

Fig. 5: Onion service address collection and detection system

The *aliveness and access frequency detection* module detects the online time and access frequency of V3 onion services discovered via the onion address collection module. We leverage the Tor control protocol [16] to control 20 Tor clients to download the descriptors of collected onion services every day to detect their online time. We deploy 10 HSDirs for around two years to collect the blinded public key of the descriptors of the onion services to estimate access frequency.

We also implement a Tor onion service emulator to upload its blinded public key to HSDirs to verify the effectiveness of V3 onion service size estimation method. Specifically, we modify version 0.4.2.7 of the source code of vanilla Tor and revise its code of time management module to speed up the emulation. We collect the information of all the real world Tor nodes, a total of 6,786 Tor nodes including 4,000 HSDirs, and store it in a consensus file. The emulator is configured as an onion service to periodically generate a new blinded public key and upload it to 8 HSDirs selected from 4000 HSDirs. We can launch any number of emulators within a virtual same time period and use the same consensus file to emulate a large number of onion services on an emulated Tor network so that we can know the real number of onion services and evaluate the effectiveness of our onion service size estimation method.

B. V3 onion service size estimation

We leverage our Tor onion service emulators to generate a total of 300,000 onion services and set the time period as 60 days and then define a metric to measure the effectiveness of our estimated method. Figure 6 shows the estimated number of onion services within the emulated 60 days using 5 randomly selected HSDirs. We can see that the estimated number using each HSDir fluctuates around 300,000, similar to the real number of onion services. Moreover, the average estimated size is even closer to the real number of the emulated onion services. We further define a *deviation rate* as $D = \frac{|\hat{E} - E_t|}{E_t}$ to represent the bias of the estimated size, where $\hat{E}$ is the estimated number of onion services and E_t is the real number of onion services. A smaller deviation rate D indicates that the estimated number is closer to the real one.

Figure 7 illustrates the deviation rate of 4,000 HSDirs respectively used to estimate the size within one day using our emulated Tor network. We can see that more than the deviation rate of 99.6% of the HSDirs is less than 10%, and more than 90% of the HSDirs are evaluated with deviations of less than 5%. In addition, the average deviation rate of 4,000 HSDirs is 2.43%. Therefore, our onion service size estimation method is remarkably effective by using a single or multiple HSDirs

within one day. Moreover, Figure 8 shows the average deviation rate within one day by selecting the worst top k-percent HSDirs. As we can see from the figure, the worst top 0.2% HSDirs, i.e., 8 HSDirs, can achieve less than 10% deviation rate.

After verifying the effectiveness of our method on an emulated Tor network, we estimate the number of onion services on the real Tor network and continuously monitor it for up to 2 years. According to our emulation results, we can use a few HSDirs to derive an accurate estimation result, therefore, we deploy 10 HSDirs on the real Tor network. Figure 12 depicts the estimated number of V3 onion services on the real Tor network from July 1, 2020 to July 2, 2022. We can see that the number of V3 onion services gradually and steadily increases from 70,000 on July 1, 2020 to 1,200,000 on May 28, 2022. We also compare our estimated size with that using the existing estimation methods [17], [18]. However, the method proposed by Hoeller *et al.* [17] is inaccurate. Recall that, since a HSDir can receive blinded public keys from both two DHTs in one day as shown in Figure 2, two capture probabilities of the HSDir are different in these two DHTs. They use the total number of blinded public keys and only one of the capture probability rather than both capture probabilities to estimate the onion service size. Alternatively, the official estimated data [18] is also online starting at Sep. 18, 2021 without publishing the detailed estimation method. The estimated data is less than that of our method as they may remove some temporarily appeared blinded public keys. Since our method is effectively verified on a large-scale emulated Tor network, our estimated V3 onion service size is very reliable.

C. V3 onion service usage status

We measure V3 onion service usage by analyzing the upload and download of descriptors from the server-side and the client-side, respectively. Table I shows the number of onion service descriptors and blinded public keys received by our deployed HSDirs and the number of descriptors and blinded public keys downloaded by Tor clients ranging from July 1, 2020 to July 2, 2022. As shown in the table, we receive a large number of descriptor and blinded public key download and upload requests, respectively. The successful onion service upload rates of the descriptors and blinded public keys are 99.26% and 99.99%, respectively, since few descriptors and blinded public keys are not correctly parsed due to the format issues, timestamp issues, etc. However, Only 13.01% and 8.42% of uploaded descriptors and blinded public keys are downloaded by clients, respectively. Note that the descriptors are uploaded to the 8 responsible HSDirs and only downloaded randomly from 6 HSDirs, i.e., 75% uploaded descriptors can be used. Suppose that the descriptor is downloaded by a client only once within a day. Then the probability that one of our deployed HSDirs is selected is 12.5%, i.e., $\frac{6}{8}\times\frac{1}{6}$, which is higher than 8.42%, indicating that some onion services may be rarely or not accessed.

TABLE I: Descriptor statistics from server-side and client-side

	Description	Descriptor		Blinded public key	
		Number	Rate	Number	Rate
Server-side	Total upload	87,200,319	-	4,265,280	-
	Successful upload	86,556,546	99.26%	4,265,274	99.99%
	Usage	11,261,361	13.01%	359,513	8.42%
Client-side	Total download	246,931,239	-	1,659,174	-
	Successful download	30,571,721	12.38%	348,007	20.97%

D. V3 onion service popularity and core websites

We leverage our onion service address collection and detection system to collect public V3 onion addresses and detect the aliveness and access frequency of the onion services. Particularly, we collect a total of 57,531 valid public V3 onion addresses from July 1, 2020 to July 20, 2022. Then we collect descriptor download requests for these addresses during this period to calculate the access frequency of the onion services. We perform the online status detection from January 1, 2021 to July 20, 2022 so as to calculate the online rate of the onion service. Figure 9 and Figure 10 illustrate the CDF of online rate and access rate for the public V3 onion services, respectively. As we can see from Figure 9, the online rate of 80% of onion services is over 80%. Thus, most of public onion services are very stable. The access rate of less than 20% of public onion services is greater than 20% as shown in Figure 10. Accordingly, only a few of the onion services are frequently accessed by users. Finally, according to Equation (12), we set parameter α to 0.6 to derive the popularity of all public onion services as the popular onion services are always accessed by users. The CDF of the popularity of the public V3 onion services is shown in Figure 11. The popularity of around 50% of the onion services is between 0.4 and 0.5, while the popularity of only 10% of the onion services is above 0.5.

We cluster near-identical onion websites from public onion services into groups and then compute the group popularity so as to obtain the core websites on the Tor network. We discover 45,889 onion websites out of 57,531 onion services and crawl the homepages of the websites. Then, we compute the similarity between the homepages of each website and derive 6,177 website groups by clustering with the similarity greater than 0.9. Afterwards, we collect a total of 112 yellow pages from Internet and onion services that contains more than 50 distinct onion addresses. Then we calculate the appearance rate in terms of Equation (14) to obtain the popularity of the website groups, where the parameter β is empirically set to 0.1. Figure 9, 10 and 11 show the CDF of online rate, access rate and popularity for website groups, respectively. According to the popularity distribution, we set the threshold to 0.55 and obtained 487 core website groups, accounting for 8% of the total. Finally, we select the most popular onion service in each website group as a representative and derive the top 487 core websites on the Tor network.

To figure out the types of the core websites, we manually categorize and compute the popularity of each category, as depicted in Figure 13. Most of core websites fall into the category of *Pornography*, followed by the *Finance*. We use the average popularity of core websites in each category to represent the category popularity. We discover that the *Finance* is the most popular category, offering diverse services such as counterfeit money, cryptocurrency, money transfer and credit cards. The *Service* are some general services deployed by content providers, such as website hosting, search engine, escrow,

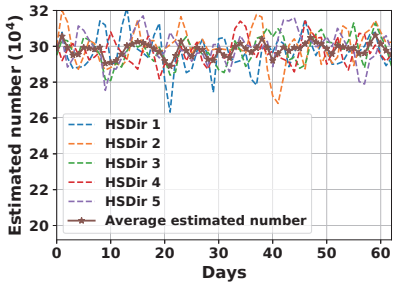

Fig. 6: Estimated numbers via 5 HSDirs

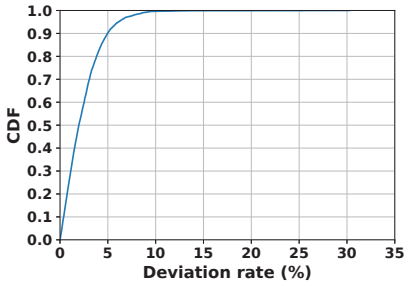

Fig. 7: Deviation rate

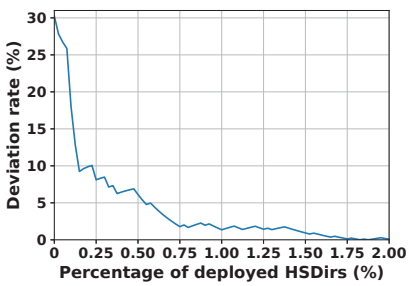

Fig. 8: Average deviation rate

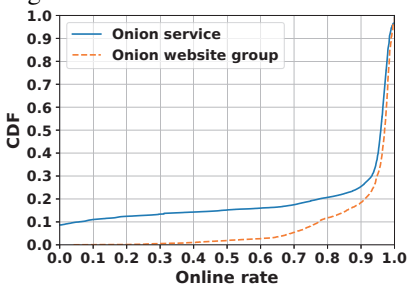

Fig. 9: CDF of online rate

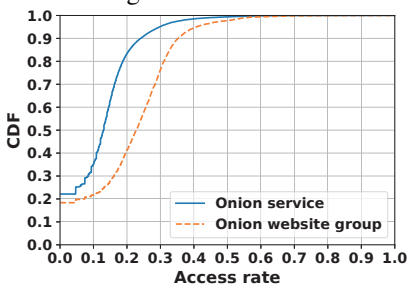

Fig. 10: CDF of access rate

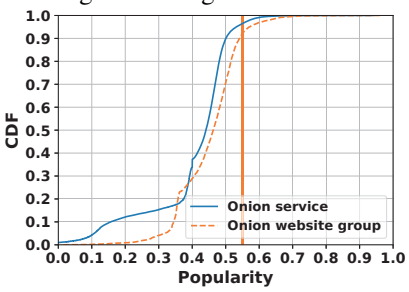

Fig. 11: CDF of popularity

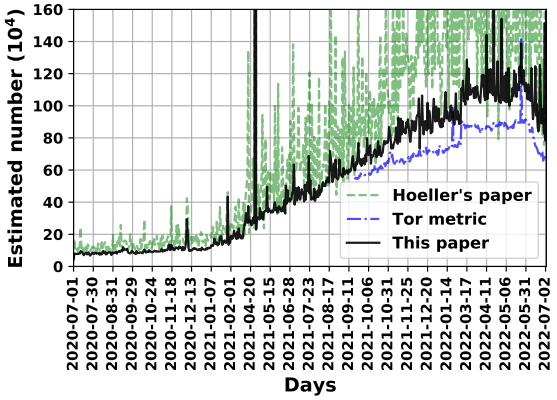

Fig. 12: Estimated number of V3 onion services

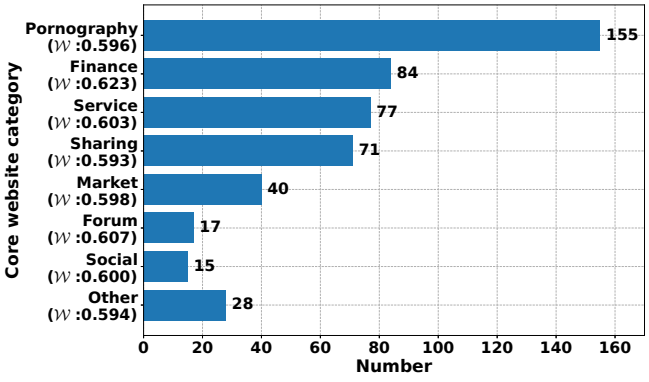

Fig. 13: Core website category

email and imageboard. The *Sharing* includes websites that share resources, such as wiki, library, yellow page and leaked data. The *Market* and the *Forum* are places for trading and information exchange and attract the attention of many illegal users, such as Archetyp market [19] and Helium Forum [20]. The *Social* includes some social networking platforms or chat rooms on the Tor network. We put the rest of the unidentifiable websites into the *Other*. Finally, the top 6 most popular core websites are Archetyp market [19], ROYALMARKET [21], DARKMONEY [22], VClub [23], dark.fail [24] and ASAP market [25]. The fifth website is a well-known yellow page and the rest are black markets. It shows that black markets are very popular on the Tor network.

E. Phishing website identification

We leverage the popularity of the onion services to identify the official onion websites within each website group. To obtain the core websites, we cluster the websites with near-identical content into groups that may contain both mirror sites and phishing sites. Table II illustrates the distribution of the number of websites within a group. There are 6 groups that contain total 8,234 onion services and each of which includes more than 1,000 onion services. All the 6 groups belong to the core website groups. 63 groups that contain 25,270 have the number of onion services ranging from 100 to 1,000, 60 of which belong to the core websites. We find that 149 website groups have the number of onion services between 10 and 100. The corresponding core website groups have 2,190 onion services although there are only 69 of them. The number of the rest website groups is 5,959, each of which includes less than 10 onion services. There are 352 core website groups in these groups. We find that a large number of near-identical websites belong to the core websites. Commonly, websites are not open a large number of mirror sites because of the overhead of the hosting cost. Consequently, most of the near-identical websites are phishing websites. However, due to the anonymity of the onion services, it is nontrivial to identify phishing websites without any clues. Since an official website should be the most popular one within each website group and is the least likely to be a phishing website, we take core websites in each website group as the default official one.

TABLE II: Clustering results for websites

Range of each group	All website group (#)		Core website group (#)	
	Group	Onion service	Group	Onion service
≥1000	6	8,234	6	8,234
[100,1000)	63	25,270	60	24,616
[10,100)	149	3,873	69	2,190
<10	5,959	8,512	352	913

TABLE III: Identified phishing websites

Category	Core website groups (#)	Official websites (#)	Phishing websites (#)
Pornography	156	160	3,053
Finance	85	85	23,875
Service	75	111	2,449
Sharing	74	94	2,122
Market	40	96	3,731
Forum	17	22	24
Social	15	16	17
Other	25	38	60
Total	**487**	**622**	**35,331**

We leverage the official onion website to find mirror sites in two ways. One is to access the official website and search the mirror onion addresses. Alternatively, we can search the mirror addresses of the target website on well-known yellow pages, such as dark.fail [24], Onion.Live [26] and Darknetlive [27]. We can find all mirror addresses of the core websites using the two ways. In addition to the core website and the mirror websites, we consider the rest of onion services within a website group as phishing websites. Table III shows the number of identified phishing websites and official websites including the mirror websites from the core website groups. Within the 487 core website groups, we identify a total of 622 official websites and 35,331 phishing websites. The *Finance* is the most popular category of onion services. However, the number of phishing websites is also the highest with 23,875. The purpose of these phishing websites is commonly to scam users out of their money. The *Market* is harder to spoof due to the usage of complicated login schemes and PGP signatures for mirror addresses, resulting in fewer market phishing websites. Other categories, such as the *Pornography*, the *Service* and the *Sharing*, can be directly cloned by attackers to attract user visits to earn advertising fees or for other purposes. The *Forum*, the *Social* and the *Other* are not profitable, resulting in few phishing websites. Finally, we discover 35,331 phishing websites within the core website groups.

V. Related Work

The Tor Metrics project [28] is the official website for measuring, analyzing, and visualizing the Tor network. After the onion services protocol was upgraded from V2 to V3, the previous method [29] of evaluating the number of onion services is no longer effective. Hoeller *et al.* [17] propose a V3 onion service size estimation method, however they only use one of the capture probability rather than both capture probabilities to estimate the onion service size, resulting in large fluctuations in evaluation results. Tor officially starts to provide the estimated number of V3 onion services from September 18, 2021, however, does not provide a specific evaluation method. Our method is theoretically analyzed and effectively evaluated on a large-scale emulated Tor network.

The popularity and content of onion services attract the interest of many researchers. For example, Biryukov *et al.* [30] [31] estimate the popularity of onion services by inspecting the request rate of clients for onion service descriptors. Al-Nabki *et al.* [8] propose a ToRank algorithm to rank onion services, which constructs a graph by using onion services as nodes and hyperlinks into and out of onion services as edges. The weights of neighboring nodes are accumulated to update the popularity of each onion service. Another study by them [9] uses features extracted from onion service content to investigate a Learning-to-Rank algorithm, automatically learn the ranking function and detect the most influential onion services. Spitters *et al.* [32] apply classification and topic model-based text mining techniques to the content of onion websites to model topic organization and linguistic diversity to enable topic classification. However, the content-based ranking method is inaccurate and fails to reflect the real popularity of onion services. There are other studies on the longevity of V2 onion services [33] [34], which conclude that most onion services are online for relatively short periods of time. However, during one and a half year period, we discover that most of public V3 onion services are stable.

VI. Conclusion

We perform a comprehensive and long-term analysis to shed light on the latest V3 onion services. We propose an onion service size estimation approach and discover around 900,000 onion services on the Tor network. Since the V3 protocol significantly enhances privacy of onion services, we can only focus on the public onion services that can be learned from search engines. Then we leverage a set of two-year onion service behavior data derived by deploying 10 HSDirs on the Tor network, and investigate a novel onion service popularity estimation method to rank the public onion websites. We only discover 487 core websites out of 45,889 onion websites. Furthermore, we analyze the structure and content of the onion websites and discover that 35,331 phishing websites spoof the 487 core websites. We infer that attackers may make a great profit by hosting so many phishing websites.

Acknowledgment

This research was supported in part by National Natural Science Foundation of China Grant Nos. 62022024, 61972088, 62072103, 62102084, 62072102, 62072098, 62232004, and 61972083, by US National Science Foundation (NSF) Awards 1931871, 1915780, and US Department of Energy (DOE) Award DE-EE0009152, Jiangsu Provincial Natural Science Foundation of China Grant No. BK20190340, Jiangsu Provincial Key R&D Program Nos. BE2021729, BE2022680, and BE2022065-4, Jiangsu Provincial Key Laboratory of Network and Information Security Grant No. BM2003201, Key Laboratory of Computer Network and Information Integration of Ministry of Education of China Grant Nos. 93K-9, and Collaborative Innovation Center of Novel Software Technology and Industrialization. Any opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," *Naval Research Lab Washington DC*, 2004.

[2] dgoulet, "Onion service version 2 deprecation timeline | tor blog." https://blog.torproject.org/v2-deprecation-timeline, 2022.

[3] N. Hopper, "Proving security of tor's hidden service identity blinding protocol," *Tor Tech Report*, 2013.

[4] G. Owen and N. Savage, "Empirical analysis of tor hidden services," *IET Information Security*, vol. 10, no. 3, pp. 113–118, 2016.

[5] C. Yoon, K. Kim, Y. Kim, S. Shin, and S. Son, "Doppelgängers on the dark web: A large-scale assessment on phishing hidden web services," in *Proceedings of the 30th World Wide Web Conference (WWW)*, pp. 2225–2235, 2019.

[6] R. Van Wegberg, S. Tajalizadehkhoob, K. Soska, U. Akyazi, C. H. Ganan, B. Klievink, N. Christin, and M. Van Eeten, "Plug and prey? measuring the commoditization of cybercrime via online anonymous markets," in *Proceedings of the 27th USENIX Security Symposium (USENIX security)*, pp. 1009–1026, 2018.

[7] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *Journal of Cryptographic Engineering*, vol. 2, no. 2, pp. 77–89, 2012.

[8] M. W. Al-Nabki, E. Fidalgo, E. Alegre, and L. Fernández-Robles, "Torank: Identifying the most influential suspicious domains in the tor network," *Expert Systems with Applications*, vol. 123, pp. 212–226, 2019.

[9] M. W. Al-Nabki, E. Fidalgo, E. Alegre, and D. Chaves, "Content-based features to rank influential hidden services of the tor darknet," *arXiv preprint arXiv:1910.02332*, 2019.

[10] M. Faizan, R. A. Khan, and A. Agrawal, "Ranking potentially harmful tor hidden services: Illicit drugs perspective," *Applied Computing and Informatics*, 2020.

[11] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, vol. 96, pp. 226–231, 1996.

[12] DARKWEBLINKS.COM, "Dark web links." https://darkweblinks.com/, 2022.

[13] Hidden Wiki, "The hidden wiki." http://zqktlwiuavvvqqt4ybvgvi7tyo4hjl5xgfuvpdf6otjiycgwqbym2qad.onion/wiki/index.php/Main_Page, 2022.

[14] J. Nurmi, "Ahmia - search tor hidden services." https://ahmia.fi/, 2022.

[15] TorDex, "The uncensored index of tor onion sites - tordex." http://tordex u73joywapk2txdr54jed4imqledpcvcuf75qsas2gwdgksvnyd.onion/, 2022.

[16] Tor project, "control-spec.txt - torspec - tor's protocol specifications." https://gitweb.torproject.org/torspec.git/tree/control-spec.txt, 2022.

[17] T. Hoeller, M. Roland, and R. Mayrhofer, "On the state of v3 onion services," in *Proceedings of the ACM SIGCOMM 2021 Workshop on Free and Open Communications on the Internet*, pp. 50–56, 2021.

[18] Tor project, "Onion services - tor metrics." https://metrics.torproject.org /hidserv-dir-v3-onions-seen.html, 2022.

[19] Archetyp Market, "Archetyp market." http://4pt4axjgzmm4ibmxplfiuvo pxzf775e5bqseyllafcecryfthdupjwyd.onion/, 2022.

[20] Helium community, "Helium forum & marketplace." http://fahue6hb7o dzns36vfoi2dqfvqvjq4btt7vo52a67jivmyz6a6h3vzqd.onion/, 2022.

[21] ROYALMARKET, "Royalmarket." http://royalyygxzq5fadtlgzftq3dwp dycq4gvddmvhrk3l2gjzsfqqwpvwad.onion/, 2022.

[22] DARKMONEY, "Darkmoney." http://darkmonn6oy55o7kgmwr4jny2gi2 zj6hyalzcjgl444dvpalannl5jid.onion/, 2022.

[23] Anonymous, "Vclub." http://vclubccvd426icndg43sjpaowcdkiatcmdfj3z w4mc2z6zwbhvecwmyd.onion/, 2022.

[24] Darkdot Research, Inc, "Dark.fail." http://darkfailenbsdla5mal2mxn2uz 66od5vtzd5qozslagrfzachha3f3id.onion/, 2022.

[25] ASAP Market, "Asap market." http://asap2u4pvplnkzl7ecle45wajojnftja 45wvovl3jrvhangeyq67ziid.onion/, 2022.

[26] OnionDotLive, "Onion.live." https://onion.live/, 2022.

[27] Darknetlive, "Darknet market news, links, and guides | darknetlive." ht tps://darknetlive.com/, 2022.

[28] Tor project, "Welcome to tor metrics." https://metrics.torproject.org/, 2022.

[29] G. Kadianakis and K. Loesing, "Extrapolating network totals from hidden-service statistics," *Tor Tech Report*, 2015.

[30] A. Biryukov, I. Pustogarov, and R.-P. Weinmann, "Trawling for tor hidden services: Detection, measurement, deanonymization," in *Proceedings of the 34th IEEE Symposium on Security and Privacy (SP)*, pp. 80–94, 2013.

[31] A. Biryukov, I. Pustogarov, F. Thill, and R.-P. Weinmann, "Content and popularity analysis of tor hidden services," in *Proceedings of the 34th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pp. 188–193, 2014.

[32] M. Spitters, S. Verbruggen, and M. Van Staalduinen, "Towards a comprehensive insight into the thematic organization of the tor hidden services," in *Proceedings of the 2014 IEEE Joint Intelligence and Security Informatics Conference (JISIC)*, pp. 220–223, 2014.

[33] G. Owenson, S. Cortes, and A. Lewman, "The darknet's smaller than we thought: The life cycle of tor hidden services," *Digital Investigation*, vol. 27, pp. 17–22, 2018.

[34] A. Sanatinia, J. Park, E.-O. Blass, A. Mohaisen, and G. Noubir, "A privacy-preserving longevity study of tor's hidden services," *arXiv preprint arXiv:1909.03576*, 2019.

VII. APPENDIX

We define a *capture probability* that a HSDir receives a new blinded public key in an ETP. Note that an onion service generates two descriptors that contain the same blinded public key and each descriptor is uploaded to 4 consecutive HSDirs in terms of ID as shown in Figure 3. Let B_i be the event that the i^{th} HSDir collects an onion service descriptor. Then we denote two discrete random variables by A_1 and A_2 to indicate that the descriptor ID_1 and ID_2 of the onion service fall in a range of the DHT, respectively. Next, let $a1$ and $a2$ be the ranges of $(\mathcal{I}_{i-4}, \mathcal{I}_i]$ and $(\mathcal{I}_{i-8}, \mathcal{I}_{i-4}]$ in which a descriptor ID falls, respectively, and $a3$ indicates that the descriptor ID falls in a location other than $a1$ and $a2$ (i.e., $ID \notin (\mathcal{I}_{i-8}, \mathcal{I}_i]$) as shown in Figure 4. According to law of total probability, the *capture probability* P_i^1 of the i^{th} HSDir on the old DHT becomes

$$\begin{aligned} P_i^1 &= P(B_i) \\ &= P(A_1 = a1)P(B_i|A_1 = a1) + \\ &\quad P(A_1 = a2)P(B_i|A_1 = a2) + \\ &\quad P(A_1 = a3)P(B_i|A_1 = a3) \end{aligned} \tag{15}$$

Then we illustrate how to derive the three condition probabilities one by one, i.e., $P(B_i|A_1 = a1)$, $P(B_i|A_1 = a2)$, and $P(B_i|A_1 = a3)$.

Given an event $A_1 = a1$, i.e., an onion service descriptor $ID_1 \in (\mathcal{I}_{i-4}, \mathcal{I}_i]$, the i^{th} HSDir can definitely collect the descriptor ID, i.e., $P(B_i|A_1 = a1) = 1$, since the descriptor can be uploaded to four consecutive HSDirs whose indices immediately follow the descriptor ID as shown in Figure 3. Then we have $P(A_1 = a1)P(B_i|A_1 = a1) = P(A_1 = a1) = \sum_{j=i-3}^{i} p_j$.

Given an event $A_1 = a2$, i.e., $ID_1 \in (\mathcal{I}_{i-8}, \mathcal{I}_{i-4}]$, according to law of total probability, we can have

$$\begin{aligned} &P(B_i|A_1 = a2) = \\ &P(A_2 = a1)P(B_i|A_1 = a2, A_2 = a1) + \\ &P(A_2 = a2)P(B_i|A_1 = a2, A_2 = a2) + \\ &P(A_2 = a3)P(B_i|A_1 = a2, A_2 = a3) \end{aligned} \tag{16}$$

Since either the descriptor ID_1 or ID_2 falls into the range of $(\mathcal{I}_{i-4}, \mathcal{I}_i]$, the HSDir can receive one of the descriptor. Then we know that $P(B_i|A_1 = a2, A_2 = a1) = 1$. Given events $A_1 = a2, A_2 = a2$, the HSDir can still receive the descriptor. Note that if the descriptor ID_1 and ID_2 fall in the same range on the DHT, the onion service can avoid uploading to the

same HSDirs that receive the descriptor ID_1, and then select four consecutive HSDirs whose indices immediately follow the index of the last HSDir that receives the descriptor ID_1 to upload the descriptor ID_2. In this case, the condition of $A_1 = a2, A_2 = a2$ is the same with that of $A_1 = a2, A_2 = a1$ and then we have $P(B_i|A_1 = a2, A_2 = a2) = 1$ as well. Given events $A_1 = a2, A_2 = a3$, the HSDir cannot collect the descriptors, i.e., $P(B_i|A_1 = a2, A_2 = a3) = 0$. Therefore, we have $P(A_1 = a2)P(B_i|A_1 = a2) = P(A_1 = a2)(P(A_2 = a1)+P(A_2 = a2)) = \sum_{j=i-7}^{i-4} p_j(\sum_{j=i-3}^{i} p_j+\sum_{j=i-7}^{i-4} p_j) = \sum_{j=i-7}^{i-4} p_j \sum_{j=i-7}^{i} p_j$.

Given an event $A_1 = a3$ that is similar to the event $A_1 = a2$, according to law of total probability, we have

$$\begin{aligned} &P(B_i|A_1 = a3) = \\ &P(A_2 = a1)P(B_i|A_1 = a3, A_2 = a1) + \\ &P(A_2 = a2)P(B_i|A_1 = a3, A_2 = a2) + \\ &P(A_2 = a3)P(B_i|A_1 = a3, A_2 = a3) \end{aligned} \quad (17)$$

Obviously, as illustrated in Figure 4, we can derive that $P(B_i|A_1 = a3, A_2 = a1) = 1$, $P(B_i|A_1 = a3, A_2 = a2) = 0$, and $P(B_i|A_1 = a3, A_2 = a3) = 0$. As a result, we can derive $P(A_2 = a3)P(B_i|A_1 = a3) = P(A_2 = a3)P(A_2 = a1) = (1 - \sum_{j=i-7}^{i} p_j) \sum_{j=i-3}^{i} p_j$. Finally, the *capture probability* P_i^1 of the i^{th} HSDir on the old DHT can be derived by

$$\begin{aligned} P_i^1 &= P(B_i) \\ &= P(A_1 = a1)P(B_i|A_1 = a1) + \\ &\quad P(A_1 = a2)P(B_i|A_1 = a2) + \\ &\quad P(A_1 = a3)P(B_i|A_1 = a3) \\ &= P(A_1 = a1) + P(A_1 = a2)(P(A_2 = a1) + \\ &\quad P(A_2 = a2)) + P(A_1 = a3)P(A_2 = a1) \\ &= \sum_{j=i-3}^{i} p_j + \sum_{j=i-7}^{i-4} p_j \sum_{j=i-7}^{i} p_j + \\ &\quad (1 - \sum_{j=i-7}^{i} p_j) \sum_{j=i-3}^{i} p_j \end{aligned} \quad (18)$$