Enhancing the Tor Anonymity Network with K-Anonymous Flashmobs

(extended abstract of the MSc dissertation)

José Brás
Departamento de Engenharia Informática
Instituto Superior Técnico

Advisor: Professor Nuno Santos
Advisor: Professor Diogo Barradas

***Abstract*—Anonymity networks such as Tor are powerful tools to increase the anonymity and security of user communications. The popularity of Tor leads to its usage by whistleblowers or informants, willing to reveal confidential information or activities often involving private, public or governmental organizations. However, by exposing classified material, these people may face prosecution from powerful actors by acting for the sake of liberty and freedom. In this work we propose a privacy-enhancing technology that provides a new primitive for securing Tor users against attacks mounted by global passive adversaries. In particular, we introduce the idea of allowing users to convene a *k-anonymous flashmob*, i.e., to leverage the cooperation of $k-1$ freedom-fighting volunteers to connect simultaneously to the Internet through Tor and generate covert traffic that will help the flashmob organizer blend into the crowd. To materialize this idea, we propose to build a new Tor pluggable transport named Moby. Using Moby, a global passive adversary with the ability to intercept all the communications across the globe will not be able to uniquely deanonymize the communications of flashmob participants even when using state-of-the-art traffic correlation techniques.**

I. Introduction

Whistleblowers or informants are people who reveal privileged information or activities within a private, public, or government organization that are deemed illegal or fraudulent. To merely illustrate the core scenario of our work picture Alice, a notorious whistleblower. By openly denouncing wrongdoing through the publication of information on dedicated websites like WikiLeaks or Football Leaks, Alice might expose herself to prosecution or harassment. To protect her identity, she can resort to state-of-the-art anonymous communication networks such as Tor [1]. Tor's onion routing scheme [2] allows Alice to upload sensitive content on whistleblowing websites while hiding her true IP address [3].

However, over the years, the ever-present looming threat of deanonymization attacks launched at a large scale by global state-level adversaries has been greatly increasing [4]. Such attacks constitute a great risk to Tor users, like Alice, and emerge as a combination of four main different factors. Firstly, a growing consolidation of the Internet infrastructure in the hands of a few global players like Google and other large ISPs make it increasingly easier to collect large portions of Tor traffic from few vantage points [5]. Second, Tor guard nodes – which act as entry-points to the Tor network – are skewed toward a relatively small number of ISPs, making it even more practical to probe into the Tor network and intercept Tor users' communications [6]. Third, the sophistication of traffic analysis techniques is developing at a fast pace due to the application of machine learning algorithms that concur to the building of accurate traffic classifiers that can be used for launching correlation attacks [7, 8]. Finally, with the advent of mass surveillance efforts partially enabled and justified by anti-terrorism or the pandemic, states might use their law enforcement agencies in upholding mandates to monitor anonymous communication networks, therefore being able to deanonymize potential whistleblowers [9, 10, 11].

Over the last few years, there have been many proposals of anonymous communication systems [12, 13, 14, 15, 16, 17] that allow Alice to go incognito and hide her identity from the aforementioned major players. Although providing strong anonymity properties, these systems are often vulnerable to correlation attacks [18]. Due to Tor being the most widely used anonymous communication network, recent works have proposed systems to mitigate this family of attacks against onion routing, such as TorK [19], a pluggable transport leveraging the notion of indistinguishable flows to ensure k-anonymity amongst multiple sources of Tor connections. Still, resisting a global passive adversary – one that can see and correlate all traffic in the network, but cannot modify or interact with it – able to perform intersection attacks and statistical disclosure over a long period of time remains an open research challenge [20, 21]. Although there have been some proposals to fix these everlasting issues [22, 23], none have been prototyped over onion routing-based anonymity schemes, and only a few anonymity networks claim sturdiness against such attacks [24, 25, 26, 27, 28, 29, 30, 31].

In order to understand the prevalence of these attacks on Tor, imagine that Alice wants to publish information about the violation of civil liberties by her employer, a state-level security agency. Alice can connect to WikiLeaks using TorK and never reveal her personal information when exposing incriminating data. However Mallory, the director of the security agency Alice is employed in, can order network administrators to monitor WikiLeaks and Alice's posting patterns over a long period of time. By observing the user churn in the variable k-anonymity sets and performing the intersection of these, Mallory is able to narrow sets down to one user, Alice. Furthermore, if Mallory focus her efforts in analysing the accesses to WikiLeaks over this long-term passive analysis she will also be able to determine with some probability p the chances for Alice to be the poster of incriminating data against her agency.

To mitigate these risks, we propose Moby, a new privacy-enhancing tool that provides resistance to the pitfalls of TorK [19] such as client churn. Moby will also be the first available tool able to provide resistance against intersection attacks and statistical disclosure on the Tor network. These issues are solved through an abstraction that we have properly named $flashmobs$, building on the concept of a spontaneous group of individuals bound by time performing a specific set of joint actions. By enforcing multiple users to perform web browsing actions on a scheduled time period we allow whistleblowers to go incognito, blending in with a group of individuals with a predetermined communication pattern. If these flashmobs are scheduled to be performed in rounds with the same fixed set of k users accessing the same websites, no global passive adversary can perform an intersection or statistical disclosure attack against our system. The reason for this being that the intersection of flashmob user sets over multiple rounds is the same as the original membership set since churn is not tolerated. Furthermore, we also allow a whistleblower to summon flashmob users – mobbers – to participate in the system, Finally, we have implemented a real-world scenario of the usability of Moby using 9 Raspberry Pi 4 (RPi) devices with flashmobs scheduled within 2h intervals over the course of one week. In all this time, all the participating devices were available, enabling users to access the Tor network with performances varying between 1Mbps and 3Mbps.

II. Related Work

In this chapter, we start by providing an overview of types of attacks that one might utilize to deanonymize TorK and that we plan on mitigating with Moby.

A. Attacks Leveraging Anonymity Sets

This category of attacks focus solely on the analysis of anonymity sets of users connected to the Tor network and the usage of passive analysis methods to correlate traffic between the users of an entry and exit relay.

If at a given point in time, a passive adversary can enumerate the IPs of every user accessing the Tor network

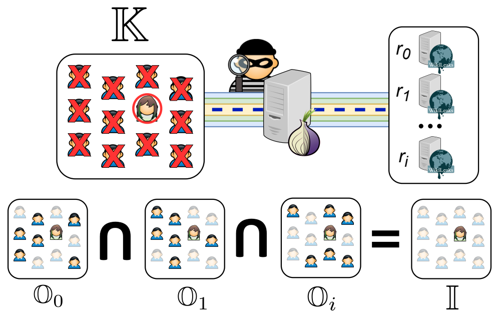


Figure 1. Example of an intersection attack to anonymity set $\mathbb{K}$. The attacker observes the online user sets $\mathbb{O}_i$ and the traffic to WikiLeaks over multiple rounds r_i. The intersection of all sets $\mathbb{I}$ allows to infer the real identity of Alice.

(say, k users in total) and of all destinations being accessed from every Tor relay, then the attacker can guess, with a probability of at most $1/k$, that a given user has visited a particular destination. The higher the size k of a user's anonymity set $\mathbb{K}$, the stronger the anonymity will be. However, it is possible to attack Tor (and other existing anonymity systems) as a result of a continuous erosion of users' anonymity sets. Next, we cover two main attacks:

1) Intersection attacks:: Throughout the passive observation of communications, an attacker can determine which users are online and the destination of their messages. Since typically the pool of destinations (e.g., websites) one accesses when online is limited, through repeated observations an attacker can start differentiating the traffic pattern of each user. For instance, a typical user tends to visit the same destinations over different sessions. Intersection attacks leverage such patterns by intersecting different sets of users active at any given moment in order to gain information and differentiate what websites a user is accessing. Although modern anonymous communications networks offer security against powerful adversaries, most systems of this type falter against a global passive adversary capable of pervasive network traffic analysis in the presence of churn. With numerous users logging in and out and through the sending of linkable messages users can quickly lose their anonymity. Intersection attacks are a well known open problem and very difficult to solve in an efficient manner [4, 20].

To give a better intuition of how intersection attacks work, consider Figure 1. Let us assume Alice wants to post meaningful information about the corruption in the national security firm she is working for on the WikiLeaks website using an anonymity system. Alice will place her posts over multiple rounds r_0 to r_i. In each iteration, a global passive adversary takes note of the website accesses and its respective online user anonymity sets $\mathbb{O}_0$ to $\mathbb{O}_i$. The group of online clients varies in each iteration due to user churn. Assuming the adversary monitors all accesses to WikiLeaks over a long period of time, the intersection of all user sets $\mathbb{I}$, allows for the singling out of Alice progressively reducing the anonymity set size k to 1, i.e., Alice herself.

Most of the works providing intersection attack defenses agree on the usage of a fixed size anonymity set whilst sensitive website are accessed. Wolinsky *et. al.* [22] provide the seminal work on intersection attack defenses. The said defense consists in the creation of a fixed anonymity set to perform round-based posts on specific websites. A similar approach is also used by Hayes *et. al.* [23], the main difference residing on the threshold used to contain user churn. Whilst the first uses an oracle to control the threshold related to the user churn in order to prevent an intersection attack, the latter does not set such threshold allowing for users to be deanonymized more easily. Despite this flaw, the authors argue that whilst not providing strong anonymity their system provides less latency. One of our concerns is that both of these approaches, in the presence of churn, after a few iterations require users to change their identifiers. Such switch of user identifiers effectively deprecates the older instance prohibiting further website accesses with the same identifier. We elaborate on both of these works further down in the next section analysing their algorithms and system architecture.

2) Statistical disclosure attacks: : As explained above, intersection attacks work in a deterministic fashion allowing for the intersection of different user anonymity sets to link a client to a destination. A statistical disclosure attack, on the other hand, is constructed probabilistically. Assuming that, after conducting multiple observations, an attacker focuses on a specific destination, then it is possible to estimate the probability of those messages pertaining to a specific sender using different strategies. Oya et al. [21] propose a compelling analysis regarding the uniformization and comparison of the different categories of statistical disclosure attacks. We base our mathematical analysis on the aforementioned work for easier reader comprehension and will analyse three main statistical disclosure variants: (i) the original statistical disclosure attack, (ii) the generalized statistical disclosure attack and (iii) the least squares statistical disclosure attack.

To present the mathematical models employed in each variant, we shortly review the notation used by Oya *et. al.* relevant for our problem. Firstly, let j be a destination, i the sender, and b background users. $p_{j,i}$ and $p_{j,b}$ represent the probabilities of traffic reaching j being from sender i and background users b respectively. Messages sent and received respectively by users i and j in round r are represented as u_i^r and y_j^r. Furthermore, lowercase letters represent vectors. Let the superscript T represent the algebraic transposing operation, the column vector containing all messages sent by user i from round 1 up to round ρ is represented by $\boldsymbol{u}_i = [u_i^1, u_i^2, ..., u_i^\rho]^T$. Reciprocally, the number of messages received by destination j is defined as $\boldsymbol{y}_j = [y_j^1, y_j^2, ..., y_j^\rho]^T$. Finally, the tilde mark "~" placed on top of $\tilde{\boldsymbol{u}}_i^r$ denotes a binary representation of $\boldsymbol{u}_i^r$ disclosing whether there is at least one message sent by user i in round r ($\tilde{\boldsymbol{u}}_i^r = 1$) or not ($\tilde{\boldsymbol{u}}_i^r = 0$). The vector notation of the aforementioned symbol is $\tilde{\boldsymbol{u}}_i = [\tilde{u}_i^1, \tilde{u}_i^2, ..., \tilde{u}_i^\rho]^T$

1) *Original statistical disclosure attack*: Having analysed the required notation to understand the probabilistic aspect of the attack, let us now analyse the original statistical disclosure proposed by Danezis [32]. The author makes the assumption in his original paper that sender i does not send more than one message in each communication round and other background traffic reaching j is uniform, i.e. $p_{j,b} = \frac{1}{N}$ for $j = 1, 2, ..., N$. The attack is thus modeled as:

$$\tilde{\boldsymbol{u}}_i^T \boldsymbol{y}_j \approx \tilde{\boldsymbol{u}}_i^T \mathbf{1}_\rho \cdot p_{j,i} + \tilde{\boldsymbol{u}}_i^T (\mathbf{1}_\rho \cdot t - \mathbf{1}_\rho) \cdot p_{j,b} \qquad (1)$$

2) *Generalized statistical disclosure attack:* The original statistical disclosure attack assumed that the sender i would only send a message per communication round. Matthewson and Dingledine [33] extend on the sender behavior and assume a scenario where i might send j more than one message. This generalized statistical disclosure is more practical and closer to real world applications. On their work the authors theorize their attack against mix networks and other anonymous networks [34]. The generalized statistical disclosure attack is defined as in the equation below. This equation does not include the number of messages sent by user i each round – 1_ρ. Instead, it uses the number of actual messages sent by i, u_i, with $1_\rho \cdot t - u_i = u_b$.

$$\tilde{\boldsymbol{u}}_i^T \boldsymbol{y}_i \approx \tilde{\boldsymbol{u}}_i^T \boldsymbol{u}_i \cdot p_{j,i} + \tilde{\boldsymbol{u}}_i^T \boldsymbol{u}_b \cdot p_{j,b} \qquad (2)$$

3) *The least squares statistical disclosure attack:* Finally, Pérez-Gonzalez [35] proposed a profiling attack based on the maximum likelihood of estimating user profiles by solving the Least Squares problem. Least Squares statistical disclosure ensures that the mean squared error between the real and estimated user profiles is minimized. Let $\hat{p}_{j,k}$ represent the estimator from all outputs when estimating p_{j_k}, the equation of this attack is given as:

$$\boldsymbol{u}_i^T \boldsymbol{y}_j = \boldsymbol{u}_i^T \sum_{k=1}^{N} (\boldsymbol{u}_k \cdot \hat{p}_{j,k}) \ , \ for \ i = 1, ..., N \qquad (3)$$

In summary, intersection and statistical disclosure attacks present a practical issue in existing anonymity networks such as Tor, since a global passive adversary may be able to link the messages received by a destination from a sender. As such, this makes it possible to debunk plausible anonymity of whistleblowers when accessing sensitive websites or posting leaked data.

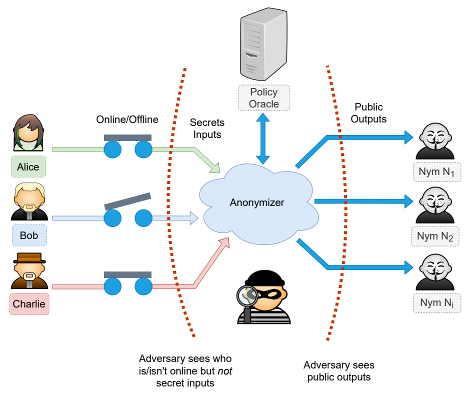


Figure 2. Conceptual Design of Buddies. The adversary observes both the secret inputs and outputs but cannot determine whether users are online or offline. The public output might be either (i) an actual message or (ii) a null message.

B. Avoiding Weaknesses for Anonymity Sets

Despite the anonymity improvements brought about to Tor by TorK, which can thwart traffic analysis attacks whilst being able to preserve k-anonymity, TorK is still vulnerable to intersection and statistical disclosure attacks.

With the possibility of clients connecting in and out of bridges, a global passive adversary is able to keep track of the k-anonymity set and the websites being accessed correlating clients with their destinations. The churn present in TorK, especially the fact that the system allows for recurrent connections facilitates the intersection of i sets throughout a period of time effectively deanonymizing clients and weakening its k-anonymity property. This section delves into general solutions that aim at shielding anonymity sets from global passive adversaries.

Wolinsky *et. al.* proposed Buddies[22], the seminal work about intersection attack resistance. This system allows for strong anonymity properties [36] bulking anonymity sets in the face of a global passive adversary. Figure 2 describes the proposed conceptual model of the Buddies system. Let us assume Alice wants to access the WikiLeaks website and post information about her corrupt employer without having her identity disclosed. Buddies will group Alice with a fixed set $\mathbb{K}$ of other $k-1$ users with each user accessing a predetermined sensitive content website in round robin fashion. Instead of using her unique identifier, Alice will rely on a pseudonym dubbed *Nym* [37] essentially working as an anonymous online handle. Buddies will attribute to Alice Nym N_1 when she joins the system and further WikiLeaks posts will be under her new handle instead of her real identity. Everytime Alice wants to post information on WikiLeaks with her Nym N_1 the users on her anonymity set – Bob and Charlie – will generate cover traffic to be sent to the Anonymizer. The Anonymizer is simply a black box anonymous communication network in the Buddies conceptual model focused on the secure and untraceable routing of traffic – the authors based their prototype on Dissent [13, 38, 39]. However, before allowing the message to be sent, the Anonymizer must first check whether it is still safe to use the same Nym. To this end, it queries the Policy Oracle which computes how much group-intersection information would be leaked allowing an adversary to mount an attack. If this information reaches a certain threshold, then that Nym is no longer considered safe and the user is informed to adopt another Nym.

Claiming that Buddies induces excessive communication latency and may prevent users from posting messages, Hayes *et al.* developed TASP [23], a system with the goal of protecting anonymity networks from intersection attacks while providing strong anonymity and lower latency. Similar to Buddies, TASP preserves fixed anonymity sets. Differently from Buddies, however, it groups users into anonymity sets based on their traffic patterns allowing the generation of more effective cover traffic. To group clients into such sets it uses machine learning-based traffic analysis algorithms working in two phases of operation, a learning phase and a working phase. Imagine that Alice has an interest in whistleblower websites such as WikiLeaks. During the learning phase she will be grouped with users with similar fingerprints which have either accessed Wiki-Leaks or exhibited a related traffic pattern. During the working phase the subset of online users $\mathbb{O}$ will provide cover traffic routed to an Anonymizer that will send Alice's posts to the WikiLeaks web servers. In contrast to Buddies, TASP has no threshold for user churn effectively allowing for the deanonymization of clients thus dropping its strong anonymity requirement. Whilst TASP's authors argue that the performance of their technique can outperform Buddies, incurring into a reduced latency and achieving higher throughput for online posts, the reality is that it does not offer long term protection against a state-level adversary nor it offers an elegant solution for the usage of user pseudonyms.

III. Design & Implementation

In this Section we address the our implemented work for Moby. Mainly, we will focus on how this system was designed to defeat intersection attacks and statistical disclosure, how Moby is implemented on top of TorK as an add-on, and on the real-world implementation of Moby using microcomputers.

A. System Overview

To offer Tor users stronger anonymity guarantees, we introduce the concept of *k-anonymous flashmobs* (henceforth simply referred as *flashmobs*). Drawing this analogy from real world flashmobs, the idea is to allow a Tor user to schedule an event where a community of other $k-1$ users (designated *mobbers*) is summoned to connect to the Internet, around the same time, and generate covert Tor traffic that will allow the flashmob organizer to "blend in with the

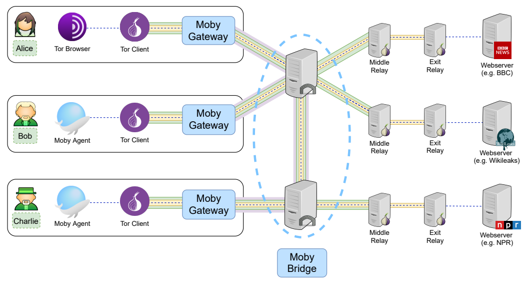


Figure 3. Moby architecture. Alice, Bob, and Charlie form a flashmob that can access BBC, NPR, or WikiLeaks websites. A global adversary can observe the network traffic exchanged by every node.

crowd" and access an intended website through Tor with k anonymous protection. Similar to TorK, flashmobs will also ensure traffic pattern indistinguishably, thus protecting against traffic correlation attacks. However, differently from TorK, flashmobs will also prevent intersection and statistical disclosure attacks by forcing the set $\mathbb{K}$ of k participants and the set $\mathbb{M}$ of m accessible websites to be locked, preventing these sets from changing for each specific flashmob. If a flashmob is instantiated multiple times in the future, this invariant ensures that the anonymity sets will always remain constant. As a result, a global passive adversary will not be able to erode k-anonymity by taking advantage of anonymity set fluctuations.

Represented in Figure 3, Moby itself consists of two main components: *bridge* and *client*. Taken together, these components implement a new pluggable transport for Tor. Bridges are responsible for managing flashmobs and relaying the Tor traffic generated by the client software running on users' devices. Bridges are interconnected to enable the enrolment in the same flashmob of a large number of clients, possibly connecting from various regions of the world. Clients run a software bundle comprised of two subcomponents: *gateway* and *agent*. The gateway regulates the transmission of Tor traffic between the local client and the bridge. The local source of this traffic can be either a standard Tor browser or an agent: the former is used by the flashmob organizer to access the intended website, and the latter is used by a mobber for generating covert traffic in the background every time the flashmob takes place.

To illustrate these concepts, imagine Alice wants to access WikiLeaks and post incriminating information about her national security agency employer – a notorious global passive adversary – whilst covering her tracks. She will only connect to Moby when wanting to leak information. In other words, Alice will not use Moby for regular everyday website accesses but only sporadic whistleblowing actions. As such, in our design, Alice is able to connect and engage with trustworthy freedom-loving participants to help her process of blending in with a crowd. Figure 3 illustrates how Alice can publish this content while covering her tracks with the help of two mobbers: Bob and Charlie. At a predefined time scheduled by Alice, these three participants must connect to Moby's bridges. This mesh of TorK bridges happens so flashmobs can span different geographical regions which also eases the recruitment of users for our anonymity set. When the flashmob starting time is reached (all users are online), Alice, Bob, and Charlie will access the web servers of BBC, NPR and WikiLeaks. Whilst Bob and Charlie are merely aiding Alice by providing cover traffic (generated by the local agents), Alice will post relevant whistleblowing information instead, using her Tor browser instance.

By requiring our fixed membership set $\mathbb{K}$ of k mobbers, to access the same fixed set $\mathbb{M}$ of m websites we are allowing for two specific defenses. (i) First, by requiring every user to remain online over multiple flashmob rounds r_i we will essentially resist every intersection attack. Let $\mathbb{O}_i$ be the set of online users for some round i, if this set remains immutable over all rounds r_i we resist any intersection attack. In other words if $\mathbb{O}_i = \mathbb{K}$ no global passive adversary can infer a whistleblower's identity because the intersection of all online user sets $\mathbb{I}$ equals $\mathbb{K}$. (ii) Second, by requiring every user to access every website in $\mathbb{M}$ we make it impossible to statistically disclose Alice's identity. Let k_i be a randomly picked user from $\mathbb{K}$ and m_i a randomly chosen site from $\mathbb{M}$. The probability p_{k_i,m_i} of k_i accessing m_i essentially equals $\frac{1}{k}$. In other words, the probability of any of k accessing a random website of m is the same. Next, we discuss some important design challenges we need to overcome when building Moby.

B. Managing Flashmobs

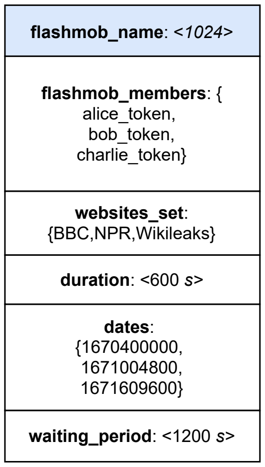
To provide a k-anonymous flashmob service for Tor, we need to characterize the stages that constitute the life cycle of a particular flashmob. We also need to specify how various actors will be engaged at each stage and what their expected behavior will be. In addition, we need to determine the meta-data required to coordinate the flashmob throughout its life cycle. From this analysis, we will draw the necessary information to design the security protocols responsible for managing flashmobs in our system. Based on our preliminary study, we present our first insights on how flashmobs will be managed in Moby.

1) Flashmob life cycle:: Taking Alice's scenario as example, our preliminary protocol for bringing a k-anonymous flashmob into existence consists in four stages:

1) *User registration:* Initially, Alice will sign up and log into the system. Signing up is an anonymous action and only Moby will know her IP address. She is attributed a secret and transient token that serves as a user identifier. Alice logs in by establishing a connection with a bridge. Alice can switch the bridge she connects to at any time during her session's lifetime.

2) *Flyer generation:* Next, Alice announces her intention to organize a flashmob and creates a *flyer*. A flyer is a manifest that establishes the size of the flashmob k Alice wants to blend into and the list of websites necessary to provide cover traffic m. During this generation phase, Alice is able to hand pick which users she wants to include in her flashmob. Several recruitment policies are possible, e.g., Alice can invite mobbers from a pool of "freedom fighters" willing to help cover her tracks for free or in exchange for money.
3) *Event scheduling:* During this phase, Alice will schedule the performance of a flashmob for a specific period of time. Let t_i and t_f be the initial and final timestamps of a performance, it is expected for every mobber to be present throughout the performance until its completion. As such, Alice will also establish a waiting room where the system will wait for every mobber to be present before starting the flashmob. Let t_{wr} be the starting time instant for the waiting room, it is expected for $t_{wr} \leq t_i$.
4) *Flashmob gathering:* Lastly, the flashmob must be put in motion. Every mobber is expected to connect to the waiting room between t_{wr} and t_i. If a single one of the k users is not online by t_i, the flashmob will be canceled. At moment t_i every mobber k starts the cover traffic generation. Reciprocally, at t_f every one of the k mobbers will cease traffic generation. During t_i and t_f, Alice will get to access to a sensitive-content website of her choosing without a global passive adversary being able to pinpoint which of the k users is accessing any of the m websites. Assuming there is no churn on the anonymity set over every flashmob round r_i, no global passive adversary can deanonymize Alice. Moreover, if k users keep on accessing m with equal probability, we nullify any statistical disclosure attack to our flashmob.

2) Flashmob flyer:: The flyer for a flashmob is constructed as the four phase flashmob creation protocol is run. Figure 4 depicts the flyer advertising Alice's flashmob. Having broadcast her intention to create a flashmob in phase 2, Moby will define a unique identifier as the flashmob's name. Furthermore, this is the phase where Alice hand picked her colluding mobbers generating her anonymity set. The system registers the mobbers in a set consisting of their user identifiers. Having chosen BBC, NPR and the WikiLeaks websites for set $\mathbb{M}$, these are also represented in the flyer in the next field. During phase 3, Alice scheduled the duration of every flashmob and the dates for their performances. Alice chose three different UNIX time dates and defined the performance and waiting room periods, for every performance, to be 600 and 1200 seconds respectively. Every user in the system possesses an updated version of this data structure. Every update to this data structure is broadcast to users pertaining to flashmob 1024.



flashmob_name: <*1024*>

flashmob_members: {
alice_token,
bob_token,
charlie_token}

websites_set:
{BBC,NPR,Wikileaks}

duration: <600 *s*>

dates:
{1670400000,
1671004800,
1671609600}

waiting_period: <1200 *s*>

Figure 4. Flyer advertising flashmob #1024 proposed by Alice.

C. Managing Users

We need to overcome several challenges involving the management of users, especially dealing with churn and mobber recruitment. We discuss them briefly.

1) Dealing with churn:: One of the main concerns with our system, naturally, is mobber churn. In Moby, differently from other systems [22, 23], churn is by no means tolerated. If a user does not show up for a flashmob gathering at time instant t_i, the flashmob is canceled. However, the incentive and usage model Moby users fall into, differs from those of other systems, such as TorK[40]. Mobbers are not passive but active users willing to help a whistleblower blend in with a crowd. As such, the risk of users connecting and disconnecting as they perform their everyday online routines, enabling an intersection attack, is largely reduced. However, there is still the risk that users do not connect to the waiting room during t_{wr} and t_i or disconnecting between t_i and t_f. Next, we also discuss how we propose to tackle this issue.

2) Mobber recruitment:: Moby allows for the hiring of individuals to increase the size of anonymity sets. Fundamentally, we propose two main methods for hiring additional mobbers: (i) the usage of monetary compensation where each participating user would be rewarded with some form of stake, such as cryptocurrency [41, 42]. As such, every mobber would essentially be mining when participating in a flashmob. (ii) The usage of gamification where users would grind reputation for different factions that simulate distinct system-set traffic patterns. As such, a user would have the desire to grind rating through flashmob participation as a way to display trustworthiness to potentially mobbers looking for a group.

3) Resisting Sybil attacks:: An adversary with significant computation power might own several devices acting like malicious mobbers, a realistic scenario in modern times

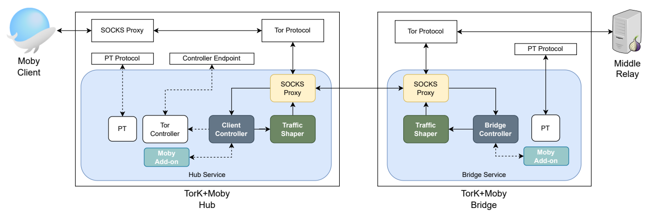


Figure 5. Overview of Moby's components with TorK

with surveillance agencies creating bridges in anonymity networks [43]. This would decrease the anonymity of a set or even allow to infer the identity of a user (if all the mobbers in a flashmob are sybil actors). To address this problem, Moby may allow "closed" groups where every mobber is invited by the flashmob organizer therefore establishing a trusted group. Each user would be defined by a static-roster of identifier tokens listing all members. A downside of this approach is that it reduces the chances of finding mobbers to participate in flashmobs. To support "open" groups we could build upon Sybil-resistant schemes such as those based on social networks [44].

D. Architectural Components

Figure 5 depicts the major internal subcomponents of the Moby add-on and the TorK implementation, which comprises hub and bridges. To route Tor traffic through TorK, the hub must run set up to use the TorK pluggable protocol and run Tor. Upon starting, Tor spawns the TorK hub service, the Moby add-on and a SOCKS [45] proxy to receive Tor traffic. Henceforth, Tor cells generated by the client can then be received by the hub service, packed into TorK frames and sent to the bridge according to a specific traffic shaping function. Conversely, TorK bridges extract the Tor cells embedded into TorK frames and deliver these cells to the Tor network.

Since TorK is a Tor pluggable transport, any TCP/IP application that places requests through Tor can reap the benefits of our system. To entirely control a Tor circuit, TorK instrumentally reroutes streams to a given circuit only when authorized by bridges. To perform this rerouting operation, we leverage the Tor Controller specification to build a Tor controller that exposes functions capable of creating, closing, and attaching streams to circuits.

Moby, once instanced will leverage a CLI interface allowing the user to send commands to the bridge the client is currently connected to, and create a flashmob that future users, once connected to this same bridge might be interested in joining. The Moby component for the bridge not only receives commands to instantiate a flashmob, but is the responsible process to coordinate mobbers to start traffic generation and block traffic if the number of messages reaching the bridge is below a pre-configured anonymity threshold.

We implemented Moby prototype on top of TorK for GNU/Linux. Moby includes both a Client and Bridge (Server) component. The prototype was written in C++ using OpenSSL[1] and Boost[2] libraries.

Moby leverages the six main components of TorK. Each component whether aims at implementing an Tor API, in which TorK can use to setup Tor or implements TorK architectural components: indistinguishable and unlinkable channels, k-anonymity through k-circuits, the Moby protocol, and finally, measures to withstand against active attacks. The following subsections present implementation details, issues and challenges faced for each component.

E. Moby and the Internet of Things (IoT)

Since the main cause of deanonymization for Moby is the churn of mobbers, to protect Alice in the advent of sensitive website accesses, we have performed a practical deployment of Moby using RPi microcomputers acting as Moby hubs. These devices are intended to function as mobbers being deployed behind the residential network of Moby users and generating traffic once a flashmob begins.

The reason behind this deployment was that differently from desktop, laptop computers and mobile phones, Internet of Things (IoT) devices such as RPis are meant to be online during long periods running relatively simple, non-intensive computational tasks whilst being available almost 24/7. Differently from computation servers, RPis offer low energy consumption and are practical to deploy, in an almost plug-and-play fashion.

Through the usage of RPis, deployed in residential homes, intended to be online and running Moby client 24/7, the availability of mobbers is highly increased. The main reason for this being clients that run Moby on their personal devices and have signed up for a future flashmob might be offline during the time period of the performance. Lest the occasion of power outtages and network failures the probability of churn resides on the client willingly quit the flashmob the node was signed for.

For Moby to be functional behind the Network Address Translation (NAT) of residential routers some adjustments were made. In order for the Moby bridge to instruct mobbers to start generating cover traffic, these devices have to be reachable from outside the NAT. For this goal, we have made use of Tailscale [46] – a plug-and-play secure VPN service – allowing mobbers to expose a public virtual IP address accessible to the Moby bridge.

With Tailscale, every user will now expose an extra virtual network interface to the outside world. These virtual network interfaces will act as a regular interface, thus giving these machines a reachable IP. Not only that, but it facilitates

[1]https://www.openssl.org/
[2]https://www.boost.org/

Device	ISP	Modem	Wired	Avg. Tput. (Mbps)	Avg. Latency (ms)
rpi01	ISP$_1$	Fiber	WiFi	**85.51** $\pm$ **2.48**	**46.24** $\pm$ **1.77**
rpi02	ISP$_1$	Fiber	Ethernet	**194.96** $\pm$ **2.43**	**44.69** $\pm$ **0.30**
rpi03	ISP$_2$	Fiber	Ethernet	**203.43** $\pm$ **3.42**	**41.91** $\pm$ **1.35**
rpi04	ISP$_3$	Coaxial	Ethernet	**17.13** $\pm$ **2.35**	**49.18** $\pm$ **7.26**
rpi05	ISP$_3$	Coaxial	Ethernet	**21.38** $\pm$ **0.82**	**52.07** $\pm$ **25.70**
rpi06	ISP$_2$	Fiber	Ethernet	**117.90** $\pm$ **0.84**	**49.01** $\pm$ **3.37**
rpi07	ISP$_2$	Fiber	Ethernet	**92.92** $\pm$ **1.94**	**57.58** $\pm$ **165.15**
rpi09	ISP$_1$	Fiber	WiFi	**92.71** $\pm$ **0.78**	**118.94** $\pm$ **339.71**
rpi10	ISP$_2$	Fiber	Ethernet	**119.01** $\pm$ **0.48**	**42.49** $\pm$ **4.91**

Table I
CONNECTIVITY CHARACTERISTICS AND NETWORK PERFORMANCE OF THE DEPLOYED RPI MOBY NODES.

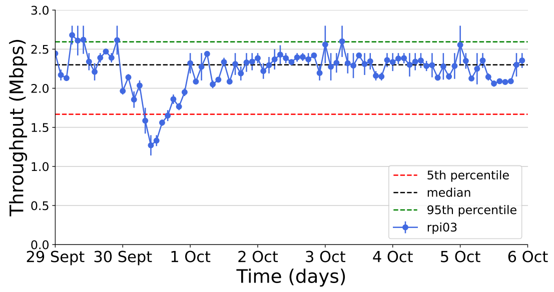


Figure 6. Weekly Moby throughput as experienced by *rpi3*.

reaching any of these devices, no matter where they are located, let the goal be to SSH into the RPis or use them for a distribution application such as Moby.

Nevertheless, it is important to mention that, although the Moby bridge used the Tailscale IP addresses to contact mobbers, once the performance starts the client nodes will route traffic regularly (through the bridge and relays) devoided of any VPN tamper. As such, Tailscale is used exclusively for the process of broadcasting flashmob management instructions (start and finish) to otherwise unreachable nodes behind a home network.

Not only, did Tailscale provide public addresses for mobbers behind NATs, now reachable from Moby bridges outside of this VPN, but it allowed for every device connected to the same Tailscale network instance to also communicate with one another. This helped shaping the idea of plug-and-play Moby-ready device which one would connect to the network and be instantly reachable from every other flashmob member.

IV. EVALUATION

In this section we show the experimental work we have conducted. We split the experimental work in two: (i) the real world deployment of Moby using microcomputers, and (ii) evaluation of Moby against statistical disclosure attacks.

A. Availability in Real-World Deployment

To mimic a set of people interested in using Moby, we performed a practical deployment of Moby using 9 RPi devices acting as TorK hubs, each provisioned with a 4-core Broadcom BCM2711 CPU and 2GB of RAM. These devices were distributed among different members of our research group and installed in households across the metropolitan area of Lisbon. This distribution enabled us to gather a representative sample of the variety of ISPs and Internet connections' performance expected to be found in the homes of individuals interested in using Moby.

1) Raw network performance: Table I details the connectivity characteristics of each RPi machine. The table also shows the average throughput and latency obtained by each node over the course of a week-long measurement where throughput and latency statistics were obtained every two hours. These statistics were obtained by reaching out to a public server under our control, hosted in Frankfurt, using the `httping` and the `iperf3` utilities, respectively. We did not send traffic through Tor or Moby during these measurements, i.e., we intended to measure the raw network performance of these devices. As seen in the table, our measurements reflect an heterogeneous landscape of network performances, where the throughput of RPi nodes ranges from 17 Mbps to 203 Mbps. Interestingly, we see that even the low performance nodes achieve a sufficient throughput to sustain bandwidth-hungry applications, like video streaming.

2) Moby performance: We used our RPi deployment to run periodic Moby rounds based on a k-circuit (k=9) while using a traffic shaping rate of $\approx$5 Mbps, i.e., equivalent to 720p video streaming. Every two hours, the RPis engaged in a flashmob where device *rpi3* sent real data, in the form of an `iperf3` measurement, and all other nodes sent chaff. In this experiment, all RPi nodes were connected via the same Tor circuit; we selected the nodes composing this circuit by picking a Tor middle relay and exit relay with an uptime larger than 40 days and a total advertised throughput over 20 Mbps.

Figure 6 shows our weekly throughput measurements conducted over TorK. The figure reveals that Moby traffic was rather stable, achieving a median throughput of 2.3 Mbps. Some of the more meaningful throughput drops we were able to observe (e.g., in September 30th), may be explained by the fact that the operator of *rpi3* was working from home and attending back-to-back videoconferencing meetings throughout the day, using another machine. This suggests that the congestion caused by other bandwidth-hungry applications' traffic in the network adds to the variability of Moby's performance.

B. Resistance Against Statistical Disclosure

One of the main components of our work is to evaluate flashmobs composed within Moby against possible intersection attacks and statistical disclosure on the anonymity sets. Since with the RPi experiment we have concluded that Moby devices work under high availability scenario, the next sections will focus on: (i) evaluating the original statistical disclosure attack against a theoretical scenario with node failure rates similar to the ones of real-world deployment, and (ii) evaluating the same attack for Moby using a bridge oracle.

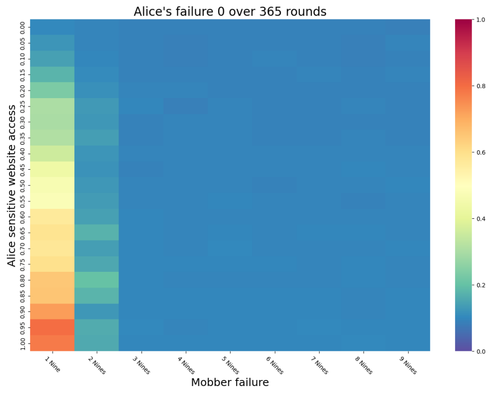


Figure 7. Highly available users with 100% chance of failures being permanent

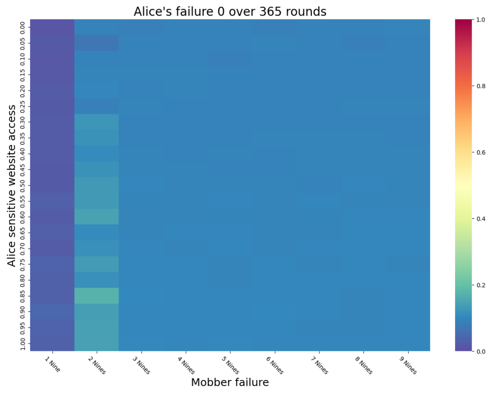


Figure 8. Highly available users with 100% chance of failures being permanent

1) High Availability Scenario: Figure 7 shows a highly available scenario for flashmob compositions. Each square on the map will represent the attacker's probability of inferring Alice's website accessing profile to a sensitive destination – WikiLeaks. Each member of the flashmob – $\mathbb{K}$ – will pick a website at random from the fixed website set – $\mathbb{M}$ – during rounds and access it until the round is over. The *x* axis, represent the failure rate of mobbers using the number of nines. The *y* axis, represents the probability of Alice accessing WikiLeaks with a higher probability than any website from $\mathbb{M}$. The number of $\#\mathbb{K}$ is 100, and the number of $\#\mathbb{M}$ is also 100.

As we can observe for a failure rate of 10% (1 Nine), and with a probability of 100% of failure resulting in a user not coming back, the attacker infers with a $\approx$10% difference the WikiLeaks accessing profiling of Alice, when compared to her real accessing profile. However, even if every failure means permanent churn in the flashmob, for (2 Nines) the attacker can barely infer the probability of Alice accessing the sensitive website being monitored.

As such, we can conclude that highly available scenarios can concur in great protection for flashmob participants as long as mobber failure probability is no more than 10%. Furthermore, as shown, not only does the failure probability of mobbers influence the attacker readings, but the analysis for temporary versus permanent churn, has shown us that, the lower the probability of permanent churn, the better Alice is protected against a global passive adversary.

2) Oracle and Minimum Threshold: We can observe from the previous experiment that the amount of offline mobbers not only conditions the plausible deniability of Alice, but also for greater probabilities of permanent churn, the attacker is able to greatly approximate the real accesses profile with the estimated one. As such, and inspired by Wolinksy [22], when traffic is routed through a Moby bridge, this component can access how many messages it did receive and implement a minimum threshold of online mobbers – $\mathbb{O}$ – to prevent the deanonymization of users, thus acting like an Oracle.

Figure 8 implements a minimum flashmob threshold of $\mathbb{O} = 4$ for a flashmob. If during a certain communication round the Moby detects that no more than 4 messages were sent to the destination websites no traffic will be routed. Following the evaluation on previous sections, the parameters remain the same for the $\mathbb{K}$, websites being accessed $\mathbb{M}$ and probability of a failure resulting in permanent churn.

In Figure 8 we can observe that for a failure rate of 10% (1 nine), with every failure resulting in permanent churn, regardless of Alice's sensitive website accessing tendencies, barely any message is being routed through Moby. Although the attacker cannot profile Alice, this poses an usability issue. Nonetheless, for a mobber failure rate of 1% (2 Nines) the results are much more encouraging. Not only, does the attacker have trouble profiling Alice's WikiLeaks accesses (compared to her real probability) but many more rounds are not being blocked by the Moby bridge Oracle.

As such, we can conclude that for Moby flashmobs to provide usability a degree of usability, the flashmob users have to possess highly availability (above 90% per round) and a low probability of failures resulting in permanent churn. With these conditions known, we can greatly vary the size of the flashmob sizes and the $\mathbb{O}$ online mobber threshold. We leave for future work evaluating the correlation between flashmob sizes, availability and the minimum flashmob threshold necessary to route traffic between bridges.

V. Conclusions

Nowadays, attacking anonymity networks is becoming easier for state-level adversaries. Although recent works such as TorK add k-anonymity and prevent traffic flow correlation on Tor (the most popular anonymity network in usage as the time of writing), such system remains vulnerable to intersection attacks and statistical disclosure against a global passive adversary. In this work, we presented Moby, aimed at providing Tor a network of Moby bridges protecting whistleblowers from intersection attacks and statistical disclosure. This is achieved by introducing k-anonymous flashmobs scheduled once a user wants to protect herself from global passive adversaries.

Acknowledgments

We are grateful to Vítor Nunes and Kevin Gallagher for the fruitful discussions and valuable insights during the preparation of this report.

References

[1] R. Dingledine, N. Mathewson, and P. F. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium, August 9-13, San Diego, California, USA*, 2004.

[2] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous Connections and Onion Routing," *IEEE Journal on Selected Areas in Communications*, 1998.

[3] Tor Project, "Tor FAQ," https://2019.www.torproject.org/about/overview.html.en, accessed: 2021-09-10.

[4] J. Raymond, "Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems," in *Proceedings of the International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, California, USA, July 25-26*, 2000.

[5] S. J. Murdoch and G. Danezis, "Low-Cost Traffic Analysis of Tor," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy (S&P 2005), 8-11 May, Oakland, California, USA*, 2005.

[6] J. Barker, P. Hannay, and P. Szewczyk, "Using Traffic Analysis to Identify the Second Generation Onion Router," in *Proceedings of the 9th IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, Melbourne, Australia, October 24-26*, 2011.

[7] Tor Project, "Traffic Correlation Using Netflows," https://blog.torproject.org/traffic-correlation-using-netflows?page=1, accessed: 2021-10-29.

[8] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson, "Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, Berlin, Germany, November 4-8*, 2013.

[9] K. Gallagher, "How Tor helped catch the Harvard bomb threat suspect," https://www.dailydot.com/crime/tor-harvard-bomb-suspect/, 2016, accessed : 2021-11-02.

[10] R. Lakshmanan, "Russia Blocks Tor Privacy Service in Latest Censorship Move," https://amp.thehackernews.com/thn/2021/12/russia-blocks-tor-privacy-service-in.html, accessed: 2021-12-12.

[11] Wired, "Russia's Internet Censorship Machine Is Going After Tor," https://www.wired.com/story/russia-block-tor-censorship/amp, accessed: 2021-12-12.

[12] D. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Scalable Anonymous Group Communication in the Anytrust Model," in *Proceedings of the 5th ACM European Workshop on Systems Security, April 10*, 2012.

[13] H. Corrigan-Gibbs and B. Ford, "Dissent: Accountable Anonymous Group Messaging," in *Proceedings of the 17th ACM Conference on Computer and Communications Security, Chicago, Illinois, USA, October 4-8*, 2010.

[14] L. Barman, I. Dacosta, M. Zamani, E. Zhai, A. Pyrgelis, B. Ford, J. Feigenbaum, and J. Hubaux, "PriFi: Low-Latency Anonymity for Organizational Networks," *Proceedings on Privacy Enhancing Technologies*, 2020.

[15] C. Chen, D. E. Asoni, A. Perrig, D. Barrera, G. Danezis, and C. Troncoso, "TARANET: Traffic-Analysis Resistant Anonymity at the Network Layer," in *Proceedings of the 2018 IEEE European Symposium on Security and Privacy, London, United Kingdom, April 24-26*, 2018.

[16] A. Kwon, D. Lu, and S. Devadas, "XRD: Scalable Messaging System with Cryptographic Privacy," in *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation, Santa Clara, California, USA, February 25-27*, 2020.

[17] A. Kwon, D. Lazar, S. Devadas, and B. Ford, "Riffle: An Efficient Communication System With Strong Anonymity," *Proceedings of the Privacy Enhancing Technologies*, 2016.

[18] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. F. Syverson, "Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, Berlin, Germany, November 4-8*, 2013.

[19] V. Nunes, "Hardening Tor against State-Level Traffic Correlation Attacks with K-Anonymous Circuits," Master's thesis, Instituto Superior Técnico, Universidade de Lisboa, 2021.

[20] G. Danezis and A. Serjantov, "Statistical Disclosure or Intersection Attacks on Anonymity Systems," in *Proceedings of the 6th International Conference on Information Hiding, Toronto, Canada, May 23-25*, 2004.

[21] S. Oya, C. Troncoso, and F. Pérez-González, "Meet the Family of Statistical Disclosure Attacks," *Computing Research Repository*, 2019.

[22] D. I. Wolinsky, E. Syta, and B. Ford, "Hang With Your Buddies to Resist Intersection Attacks," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, Berlin, Germany, November 4-8*, 2013.

[23] J. Hayes, C. Troncoso, and G. Danezis, "TASP: Towards Anonymity Sets that Persist," in *Proceedings of the 2016 ACM Workshop on Privacy in the Electronic Society, Vienna, Austria, October 24-28*, 2016.

[24] E. G. Sirer, S. Goel, M. Robson, and D. Engin, "Eluding Carnivores: File Sharing With Strong Anonymity," in *Proceedings of the 11st ACM SIGOPS European Workshop, Leuven, Belgium, September 19-22*, 2004.

[25] L. Melis, G. Danezis, and E. D. Cristofaro, "Efficient Private Statistics with Succinct Sketches," in *Proceedings of the 23rd Annual Network and Distributed System Security Symposium, San Diego, California, USA, February 21-24*, 2016.

[26] D. Lazar and N. Zeldovich, "Alpenhorn: Bootstrapping Secure Communication without Leaking Metadata," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, Savannah, Georgia, USA, November 2-4*, 2016.

[27] N. Gelernter, A. Herzberg, and H. Leibowitz, "Two Cents for Strong Anonymity: The Anonymous Post-Office Protocol," in *Proceedings of the 16th International Conference on Cryptology and Network Security, Hong Kong, China, November 30-December 2*, 2017.

[28] S. Goel, M. Robson, M. Polte, and E. Sirer, "Herbivore: A Scalable and Efficient Protocol for Anonymous Communication," *Technical Report, Cornell University*, 2003.

[29] S. L. Blond, D. R. Choffnes, W. Caldwell, P. Druschel, and N. Merritt, "Herd: A Scalable, Traffic Analysis Resistant Anonymity Network for VoIP Systems," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, London, United Kingdom, August 17-21*, 2015.

[30] M. J. Freedman and R. T. Morris, "Tarzan: A Peer-to-peer Anonymizing Network Layer," in *Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington DC, USA, November 18-22*, 2002.

[31] H. Corrigan-Gibbs, D. Boneh, and D. Mazières, "Riposte: An Anonymous Messaging System Handling Millions of Users," in *Proceedings of the 2015 IEEE Symposium on Security and Privacy, San Jose, California, USA, May 17-21*, 2015.

[32] G. Danezis, "Statistical Disclosure Attacks," in *Proceedings of the International Conference on Information Security on Security and Privacy in the Age of Uncertainty, Athens, Greece, May 26-28*, 2003.

[33] N. Mathewson and R. Dingledine, "Practical Traffic Analysis: Extending and Resisting Statistical Disclosure," in *Proceedings of the 4th International Workshop on Privacy Enhancing Technologies, Toronto, Canada, May 26-28*, 2004.

[34] A. Serjantov, R. Dingledine, and P. F. Syverson, "From a Trickle to a Flood: Active Attacks on Several Mix Types," in *Proceedings of the 5th International Workshop on Information Hiding, Noordwijkerhout, The Netherlands, October 7-9*, 2002.

[35] F. Pérez-González and C. Troncoso, "Understanding Statistical Disclosure: A Least Squares Approach," in *Proceedings of the 12th International Symposium on Privacy Enhancing Technologies, Vigo, Spain, July 11-13*, 2012.

[36] D. Das, S. Meiser, E. Mohammadi, and A. Kate, "Anonymity Trilemma: Strong Anonymity, Low Bandwidth Overhead, Low Latency - Choose Two," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy, San Francisco, California, USA*, 2018.

[37] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a Type III Anonymous Remailer Protocol," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 11-14*, 2003.

[38] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and A. Johnson, "Dissent in Numbers: Making Strong Anonymity Scale," in *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation, Hollywood, California, USA, October 8-10*, 2012.

[39] H. Corrigan-Gibbs, D. I. Wolinsky, and B. Ford, "Proactively Accountable Anonymous Messaging in Verdict," in *Proceedings of the 22th USENIX Security Symposium, Washington DC, USA, August 14-16*, 2013.

[40] P. Samarati and L. Sweeney, "Generalizing Data to Provide Anonymity when Disclosing Information," in *Proceedings of the 17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, Seattle, Washington, USA, June 1-3*, 1998.

[41] A. Biryukov and I. Pustogarov, "Proof-of-Work as Anonymous Micropayment: Rewarding a Tor Relay," in *Proceedings of the 19th International Conference on Financial Cryptography and Data Security, San Juan, Puerto Rico, January 26-30*, 2015.

[42] T. Dinh, F. Rochet, O. Pereira, and D. S. Wallach, "Scaling Up Anonymous Communication with Efficient Nanopayment Channels," *Proceedings on Privacy Enhancing Technologies*, 2020.

[43] E. Snowden, "Tor Stinks," https://edwardsnowden.com/docs/doc/tor-stinks-presentation.pdf, accessed: 2021-11-04.

[44] D. N. Tran, B. Min, J. Li, and L. Subramanian, "Sybil-Resilient Online Content Voting," in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, Boston, Massachusetts, USA, April 22-24*, 2009.

[45] M. Leech, M. Ganis, Y.-D. Lee, R. Kuris, D. Koblas, and L. Jones, "Rfc 1928: Socks protocol version 5," https://tools.ietf.org/html/rfc1928, 1996, accessed: 2022-01-05.

[46] Tailscale Inc., "Tailscale - best vpn service for secure networks," https://tailscale.com/, accessed: 2022-10-25.