Optimizing Packet Scheduling and Path Selection for Anonymous Voice Calls

David Schatz
Technische Universität Ilmenau
Germany
david.schatz@tu-ilmenau.de

Michael Rossberg
Technische Universität Ilmenau
Germany
michael.rossberg@tu-ilmenau.de

Guenter Schaefer
Technische Universität Ilmenau
Germany
guenter.schaefer@tu-ilmenau.de

ABSTRACT

Onion routing is a promising approach to implement anonymous voice calls. Uniform-sized voice packets are routed via multiple relays and encrypted in layers to avoid a correlation of packet content in different parts in the network. By using pre-built circuits, onion encryption may use efficient symmetric ciphers. However, if packets are forwarded by relays as fast as possible—to minimize end-to-end latency—network flow watermarking may still de-anonymize users. A recently proposed countermeasure synchronizes the start time of many calls and batch processes voice packets with the same sequence number in relays. However, if only a single link with high latency is used, it will also negatively affect latency of all other calls. This article explores the limits of this approach by formulating a mixed integer linear program (MILP) that minimizes latency “bottlenecks” in path selection. Furthermore, we suggest a different scheduling strategy for voice packets, i.e. implementing independent de-jitter buffers for all flows. In this case, a MILP is used to minimize the average latency of selected paths. For comparison, we solve the MILPs using latency and bandwidth datasets obtained from the Tor network. Our results show that batch processing cannot reliably achieve acceptable end-to-end latency (below 400 ms) in such a scenario, where link latencies are too heterogeneous. In contrast, when using de-jitter buffers for packet scheduling, path selection benefits from low latency links without degrading anonymity. Consequently, acceptable end-to-end latency is possible for a large majority of calls.

CCS CONCEPTS

• Security and privacy → Privacy-preserving protocols.

KEYWORDS

Anonymity, Onion Encryption, Circuits, VoIP

ACM Reference Format:

David Schatz, Michael Rossberg, and Guenter Schaefer. 2021. Optimizing Packet Scheduling and Path Selection for Anonymous Voice Calls. In *The 16th International Conference on Availability, Reliability and Security (ARES 2021)*, August 17–20, 2021, Vienna, Austria. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3465481.3465768>

This work is licensed under a Creative Commons Attribution-NonCommercial International 4.0 License.

ARES 2021, August 17–20, 2021, Vienna, Austria
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9051-4/21/08.
<https://doi.org/10.1145/3465481.3465768>

1 INTRODUCTION

Protecting communication *metadata* may be as important as protecting the confidentiality of communication content. For example, if Alice regularly contacts a counseling helpline, the pure fact about this relationship may allow to infer sensitive information. In such a context (and in many others), it may further be convenient for Alice to use Voice over IP (VoIP) instead of mail or chat. However, providing *anonymous* voice calls is challenging due to their strict latency requirements: For interactive calls, mouth-to-ear latency should not exceed 400 ms to provide acceptable call quality [11].

To provide anonymous VoIP, *onion routing*, like implemented by Tor [8], is a promising approach: Application data is split into uniform-sized packets, e.g. by using a constant bitrate codec, encrypted in multiple layers and forwarded across relays. By setting up so-called *circuits* beforehand, onion-encryption of voice packets only requires symmetric ciphers, minimizing processing delay. Unfortunately, using Tor for VoIP has various drawbacks: First, Tor deliberately only defends against a weak threat model. Most importantly, malicious relays have various options to confirm that they are on the same circuit, e.g. by network flow watermarking [25]. Second, a recent evaluation of VoIP over Tor measured a high median end-to-end latency slightly above 400 ms [21] (two circuits, three hops each). This is because path selection does not consider latency between relays [7].

Yodel [13] is a recent proposal that provides strong relationship anonymity for VoIP. For this, start and end time of calls are synchronized and voice packets are grouped in communication rounds, allowing for batch-processing like in classical mix networks [4]: All packets generated within the same round (i.e. the same sequence number) are collected in a batch, shuffled, and forwarded to the next hop at fixed time points. Moreover, missing packets are replaced by dummies to defeat watermarking attacks that drop packets. However, Yodel does not consider latency between mixes for path selection either. As a result, the minimum end-to-end latency is bounded by the maximum latency between any two mixes, multiplied by circuit length. This is because each mix has to wait for all packets of previous hops to arrive for batch processing.

One option to solve Yodel’s problem is to avoid high latency links for path selection. However, it is not obvious how to retain two other important properties while doing so:

- (1) Path selection should be “balanced” in a way that each mix has to handle a number of circuits proportional to their capacity (bandwidth/computational power). Otherwise, bottlenecks may further increase latency or packet loss.
- (2) If attackers know some prefix/suffix of a circuit, e.g. because they control it themselves, the number of possible endpoints should still be high to provide good anonymity.

In this article, we investigate the limits of Yodel’s approach (batch processing) by formulating a MILP that optimizes end-to-end latency subject to the stated constraints for path selection. Furthermore, we propose an alternative strategy for packet scheduling at mixes: Instead of batch processing all circuits, they are handled independently. To still defeat watermarking attacks, mixes implement a *de-jitter buffer* for each circuit and pad missing packets with dummies. However, the approach comes with a drawback as well: The buffer (in ms) has to be random to avoid correlation of ingress and egress timing. For perfect anonymity guarantees, the buffer interval needs to be a uniform random value chosen from [0ms; 20ms], with 20 ms being the sampling interval of VoIP packets. To compensate for the additional delay, we formulate a MILP that minimizes the average latency of selected paths. To compare the two approaches in a realistic context, we solve the MILPs using latency and bandwidth datasets obtained from the Tor network. Our own latency measurements are also publicly available [20]. Our results show that in such a scenario, link latencies are too heterogeneous for batch processing. In result, end-to-end latencies below 400 ms cannot be reliably achieved. In contrast, the additional latency introduced by de-jitter buffers can be compensated for by taking advantage of low latency links during path selection, without degrading anonymity. Consequently, voice calls with acceptable latency and good anonymity are feasible in a Tor-like network.

The remaining article is structured as follows: Sec. 2 defines objectives for anonymous voice calls and our threat model. Related work is discussed in Sec. 3. Sec. 4 includes our general system architecture, details about the scheduling strategies and our MILP formulations for deriving a suitable path selection. In Sec. 5, we present and discuss the results obtained by solving the MILPs on different datasets. We conclude our article in Sec. 6.

2 OBJECTIVES AND THREAT MODEL

This section defines our objectives and threat model.

2.1 Objectives

Besides functional requirements (call signaling, end-to-end delivery of voice packets), an anonymous VoIP system shall meet the following non-functional requirements:

2.1.1 Anonymity. First of all, *relationship anonymity* is required, i.e. communication relationships shall be hidden from third parties. Second, *location anonymity* may be required by (some) users [15]: IP addresses shall not be leaked to (potentially malicious) communication partners.

2.1.2 Quality of Service (QoS). The QoS of individual voice calls shall be acceptable. Most notably, the mouth-to-ear latency of calls should ideally not exceed 150 ms. A latency of more than 400 ms usually is unacceptable for an interactive conversation [11]. Furthermore, packet loss and jitter (variation in latency) shall be low [12].

2.1.3 Efficiency. The processing and bandwidth overhead shall be low to support as many active calls as possible for fixed system resources. Besides minimizing costs, e.g. for running mixes, supporting more calls naturally increases anonymity sets. For circuit-based designs, path selection shall be balanced in a way that each mix

handles a total number of circuits proportional to his capacities. Otherwise, bottlenecks are created, degrading QoS of affected calls.

2.1.4 Practicability. While not in the focus for this article, packet scheduling and path selection have to be implemented to be of any use. Consequently, any assumptions that are hard to implement in practice should be avoided.

Trade-Offs. Unfortunately, both low end-to-end latency and efficiency are conflicting objectives with regard to anonymity [6]: For example, increasing circuit length may strengthen anonymity in presence of malicious relays, but latency and overhead is also increased. Consequently, practical trade-offs are required.

2.2 Threat Model

With regard to relationship anonymity, we assume a global external attacker as well as a limited fraction of malicious mixes. In addition, malicious users may cooperate with others to attack the location anonymity of communication partners. The capabilities of the stated attackers are assumed to follow the Dolev-Yao model [9], i.e. they are able to observe, manipulate, delay, drop, and generate packets, but they cannot break cryptographic protections.

While disclosure attacks (analyses of changing anonymity sets over time) may be performed by (extremely powerful) global observers, they are out of scope for this article. Known countermeasures, e.g. occasional or even permanent fake calls [13, 14] are orthogonal to packet scheduling and path selection.

3 RELATED WORK

We focus on approaches that rely on *onion routing* in some form. While there is a proposal to use Dining Cryptographer networks for VoIP calls [10], it can only handle a few hundred users due to inherent scalability issues. The core concept of onion routing is that users setup so-called *circuits* before sending any application data. During circuit setup, symmetric keys are exchanged with each relay on the selected path. Subsequently, onion-encryption uses efficient symmetric ciphers to exchange packets of uniform size over all circuits. If mutual location anonymity is required, caller and callee may individually setup a circuit, meeting at some rendezvous point. The main differences of proposals arise from the following aspects:

- Path selection for circuits
- Coordination of circuit setup from different users
- Scheduling of egress packets on circuits

Tor [8] is the most prominent implementation of onion routing. While it is mainly designed for anonymizing web traffic, voice packets may also be tunneled via Tor’s circuits [21]. Unfortunately, Tor has various drawbacks: Most notably, circuit setup packets and application data packets are relayed as fast as possible. While this is beneficial for circuit setup time and end-to-end latency, it allows various timing-based attacks. For example, external attackers or malicious relays may perform network flow watermarking to track circuits [25], even if the marked flow passes several honest relays. And while packets are forwarded as fast as possible, end-to-end latency is still comparatively high for VoIP [21]. This is because path selection only considers bandwidth of relays (to avoid bottlenecks), but not latency between relays [7].

There are various suggestions to take latency information into account for Tor’s path selection [1, 2, 16, 22]. However, they do not consider bandwidth and thus potentially introduce bottlenecks. Congestion-aware path selection [24] tries to improve end-to-end latency by re-actively avoiding circuits via congested relays. However, they deliberately do not consider latency between relays.

Yodel [13] is specially designed for anonymous VoIP. They suggest to synchronize many calls to start at the same time. Then, circuit setup packets and all voice packets with identical sequence numbers are batch processed as in classical mix networks [4]. Missing packets (e.g. deliberately dropped by attackers) are replaced by dummies. However, path selection is purely random. In combination with batch processing, this introduces a bottleneck for latency: The minimum end-to-end latency is bounded by the maximum latency between any two mixes, multiplied by circuit length. While this bottleneck latency is comparatively low (45 ms) in their evaluation setup (virtual machines in the US and Europe only), it will be too high in voluntary-based deployments like Tor. Another drawback of Yodel is that users directly attach to circuit endpoints selected by their communication partners. Consequently, location anonymity is easily broken by selecting a cooperating endpoint.

TARANET [5] operates on network layer and thus can also anonymize VoIP. It defeats timing-based attacks on circuit setup (by batch processing) and data packets. In contrast to Yodel, watermarking attacks are defeated by enforcing a deterministic rate of egress packets on a per circuit basis. Furthermore, any link between two mixes is padded to a static (for some period) rate. While this avoids latency bottlenecks like in Yodel, it is unclear how the concrete schedule of packet sending times (per circuit and per link) is determined. We assume that individual packets will suffer from some waiting time at each hop to avoid direct correlation of ingress and egress packets. For path selection, it relies on “normal” source routing with anonymous path queries, e.g. as offered by some future Internet architectures. While this would minimize latency between mixes, i.e. autonomous systems (ASs), it does not provide location anonymity: Users need to know the network address of their communication partners. Furthermore, it is very unlikely that ASs will offer onion encryption functionalities soon.

Herd [14] hides circuit setup for VoIP from observers by using a static link padding between any pair of mixes. Unfortunately, malicious mixes may still be able to correlate setup timing and infer that they are on the same circuit. Voice packets are also assigned to the statically padded time slots, but no specific algorithm is presented. Path selection always uses four mixes, distributed to two “trust-zones”, selected by the caller and callee, respectively. Latency between two mixes of the same trust-zone is assumed to be very low, e.g. by deploying the trust-zone in the same data center. Consequently, Herd is able to achieve acceptable QoS for VoIP. Nevertheless, only using mixes from exactly two jurisdictions provides limited protection in case of compromise of one site.

Schatz et al. [18] improve another aspect of Herd: Instead of using a full mesh for static padding, they suggest to pad fewer links in total. Due to the statistical multiplexing gain, more active calls can be supported (fewer calls are blocked due to saturated links). They find suitable topologies by solving MILPs. But afterwards, paths are selected uniformly at random out of all paths with acceptable latency, creating bottlenecks if mix capacities are not homogeneous.

Conclusion. In summary, there are two options to defeat network flow watermarking: On the one side, Yodel suggests to batch process voice packets with the same sequence number. However, this is susceptible to high latency links that are used for path selection as they also affect all other calls. Nevertheless, end-to-end latency might still be improvable by avoiding such links. Our idea is to formulate this objective as a MILP to learn if it is possible to achieve end-to-end latencies of less than 400 ms when batch processing.

On the other side, TARANET and Herd suggest to defeat watermarking more efficiently by removing potential watermarks on circuits independently. For this, they suggest to enforce a deterministic timing of egress packets, but do not present a specific algorithm. This makes it impossible to reason about potentially introduced delays at each mix. Inspired by their idea, we present a specific algorithm in this article: implementing de-jitter buffers at each mix. Similar to batch processing, we determine a lower bound for end-to-end latency in this case by solving a MILP.

4 SYSTEM MODEL

We mainly aim at finding (theoretically) optimal path selection strategies for the two introduced packet scheduling strategies: batch processing and de-jitter buffers. Nevertheless, neither path selection nor scheduling can be evaluated independently from some other important aspects. Therefore, we first define our general architecture, including some justified assumptions that are necessary for practical deployment in future, but are out of scope for this article. Then, we present details for the two packet scheduling strategies. Next, we develop a MILP that models constraints for path selection, namely balance according to mix capacities and robustness with regard to malicious mixes. For batch processing, the objective function is to minimize the maximum latency of a circuit (and consequently for all circuits). When using de-jitter buffers, we minimize the average latency instead. The optimal solution of each MILP can directly be mapped to a probabilistic path selection algorithm.

4.1 General Architecture

In the following, we define our general system model and declare some justified assumptions for the remaining article.

4.1.1 Directory Service. Similar to Tor [17], we assume the existence of a de-centralized directory service that collects any information that is relevant for path selection. In our case, this includes mix capacities and latencies between any pair of mixes. This information can be modeled as a weighted graph $G = (V, E = V \times V, w, c)$. Mix capacities are represented by node weights $c : V \rightarrow \mathbb{R}$ and latency by link weights $w : E \rightarrow \mathbb{R}$. For brevity, we write c_v for $c(v)$ and $w_{u,v}$ for $w((u, v))$ from now on. Based on G as input, we expect the directory service to determine a probabilistic path selection strategy and distribute the resulting probabilities to users.

We further assume that bandwidth is the limiting factor for mix capacities, e.g. not onion-encryption using AES. Consequently, our approach requires secure measurements (or estimations) of bandwidths and latencies to minimize the impact of malicious mixes. However, solutions are orthogonal to path selection and are therefore out of scope for this article. Note that promising approaches exist, e.g. FlashFlow [23] for bandwidth measurements and Treeples [3] for latency estimation.

Figure 1: An established call between Alice and Bob, using two circuits with $l = 4$ mixes each.

4.1.2 Call Setup. We assume that both caller and callee want their location anonymity to be protected. Consequently, both have to individually setup a circuit before any voice data can be transferred. We further assume that dialing and circuit setup may be realized without leaking additional metadata, yet we consider metadata leaked by the established call itself (if any). Especially, we assume that circuit setup of different calls is synchronized in some way. Otherwise, attackers may simply track a newly established call by observing an increased number of packets per second on used links. For example, Hydra [19] supports synchronized circuit setup and anonymous dialing with large anonymity sets. Furthermore, additional data transfer during dialing is supported. Consequently, the caller may send his circuit endpoint, called rendezvous mix, and a rendezvous token to the callee. Then, the callee may send the rendezvous token to his circuit endpoint and ask him to join the two circuits, similar to Tor’s rendezvous mechanism [8].

Fig. 1 shows an overview of an established call. Note that both circuits are bidirectional, i.e. used for voice packets from Alice to Bob and vice versa. Unidirectional circuits, i.e. four circuits in total, would about double the chance of successful attacks. In result, each circuit actually handles *two packet flows*: An “upstream” flow directed towards the rendezvous mix and a “downstream” flow directed towards the receiving user.

Further, we assume the circuit length parameter l to be the same for all users. We denote the different positions on a path as *layers*. For now, we impose no further restrictions on these layers, i.e. every mix $v \in V$ may potentially be used on every layer.

4.1.3 Dummy Packets. Any packet scheduling resilient to watermarking requires mixes to inject dummy packets if no real packet arrives in time. These dummies must not be distinguishable from real packets at any other mix of the two involved circuits. Otherwise, watermarking attacks based on packet drop would still be possible. As in Yodel, we suggest to not use any integrity protection for voice packets at mixes, only end-to-end between users. Then, mixes may simply randomize the content of dummy packets. In combination with end-to-end encryption by caller and callee, both voice packets and dummy packets appear to have a random content at all mixes – even at rendezvous mixes, where all layers of onion-encryption have been removed. Note that the sizes of all voice packets and dummy packets have to be identical. To naturally achieve this, we suggest to use a constant bitrate codec for voice packet generation.

Figure 2: Voice packet format on circuits.

Figure 3: Batch processing of all packets with sequence number r at layer i , in upstream direction.

4.2 Packet Scheduling

This section presents further details about the two packet scheduling strategies we evaluate: Batch processing of all circuits and de-jitter buffers for each individual circuit. Both share the same packet format and replay protection.

4.2.1 Packet Format and Replay Protection. For both strategies, voice payload is encapsulated into packets as depicted in Fig. 2. As usual, circuit identifiers are selected by each mix individually during circuit setup. Each mix on a circuit looks up the corresponding key material for onion-encryption based on circId, removes or adds one layer of encryption and rewrites the circId.

Replay protection is based on the sequence number seqNo. Without this protection, replayed packets could be used to track a circuit because they take the same path and are transformed using the same keys. In addition to dropping packets with an old seqNo, it is very important that onion encryption/decryption also depends on the seqNo. Otherwise, attackers could replay the same payload, but forge the seqNo to appear new because of the absence of integrity protection at mixes. One way to achieve this requirement is to use the seqNo as part of the nonce (or initialization vector) during onion encryption/decryption. While the replayed payload is forwarded in this case (and a benign packet with the same seqNo is dropped), its bit pattern changes unpredictably.

When using de-jitter buffers, we further require the seqNo to be rewritten at every mix to avoid correlation of ingress and egress packets. For this, every mix shifts the sequence numbers of each packet flow by an *offset* before forwarding, using addition modulo 2^b (b bit sequence numbers). By deriving these offsets from the key material of corresponding circuits, they are unknown to attackers, yet users are able to derive the correct nonces for onion encryption. Furthermore, sequence numbers of the same flow remain consecutive at each mix and replay protection works as described.

When using batch processing, sequence numbers do not have to be rewritten because packets from all circuits are synchronized by sequence number at every mix as described next.

4.2.2 Batch Processing. First, recall that only calls starting at the same time are mixed when using batch processing. Every mix has one batch for each combination of layer $1 \leq i \leq l$, sequence number r (depending on the duration of the synchronized calls) and flow direction (upstream, downstream). In each batch, the mix collects and transforms any packets that arrive before a fixed timeout,

Figure 4: De-jitter buffers for three flows at one mix. Details are shown for flow f and its packets with sequence numbers r and $r + 1$. Egress sequence numbers are offset by o .

removes duplicates and adds dummy packets on a per flow basis. Then, they are forwarded in shuffled order as depicted in Fig. 3. Consequently, the complete packet schedule across all mixes is dictated by the set of timeouts for all batches. The main limit for achievable end-to-end latency is that the timeouts for the same sequence number r on two adjacent layers (e.g. $i - 1, i$ for the upstream direction) must not be too close to each other to avoid packet loss. Especially, the interval between them must be larger than the maximum latency $w_{\max}$ of all possible links between corresponding layers according to path selection. Otherwise, packets on this link do not have a chance to arrive in time and will always be dropped.

Besides the theoretical lower bound, there are some practical drawbacks of batch processing that may further increase latency. First, jitter should also be considered when setting the interval between timeouts for the same sequence number on adjacent layers. Similar, clock offset of adjacent mixes has to be considered. This implies that low end-to-end latency with low packet loss is only possible if mixes have tightly synchronized clocks (in the order of a few ms). Last, mixes might have to send multiple batches “in parallel” according to the schedule, introducing additional queuing delay. That is because end-to-end latency is higher than the voice payload (in ms) in each packet and mixes are used at all layers. For example, a mix might have to send the batch for layer i , seqNo = r and layer $i + 1$, seqNo = $r - 1$ at the same time (upstream direction).

4.2.3 De-jitter Buffer. The idea for the de-jitter buffer strategy is depicted in Fig. 4. As for batch processing, each circuit is divided into its two packet flows (upstream, downstream), which are scheduled independently. Now, each mix v locally divides time into uniform intervals of size Δ_v that equals the duration of voice data encoded in each packet. Then, both the ingress and egress timing of a packet flow f may be modeled as a periodic signal. We denote the offset between the start of each interval and the corresponding ingress packet of f as its *ingress phase* ϕ_f^- . Analogously, ϕ_f^+ denotes its *egress phase*. From the point of view of v , the ingress phase of flow f is fixed. It is determined by the egress phase of f at its previous hop u and the latency $w_{u,v}$ between the two mixes. In practice, u and v could for example exchange corresponding egress phases during circuit setup, in combination with a fresh RTT measurement. Alternatively, the arrival of the first voice packet on flow f could be used to set ϕ_f^- . To still mix different flows (three in our example, Fig. 4), the egress phase ϕ_f^+ must be selected uniformly at random. If $\phi_f^+ < \phi_f^-$, then each packet of the flow has to be delayed to the next interval. Consequently, the additional delay introduced for flow f is Δ_v in the worst case, and $\Delta_v/2$ on average.

Figure 5: Flow network model for path selection. Example with $n = 3$ mixes and $l = 3$ layers.

Similar to batch processing, the model does not capture some practical aspects. First, as the name de-jitter buffer suggests, jitter has to be considered, e.g. by adapting the ingress phase accordingly. Second, the scheduled transmission times of different packets could overlap here as well. But while this slightly increases the delay at one hop, it is automatically handled by the next hop because the ingress phase already reflects this delay.

4.3 Path Selection

In this section, we first define a special *flow network* and show how specific flows in this network can be mapped to a probabilistic path selection algorithm. To further find an optimal flow, we introduce the corresponding MILP formulation, including constraints that shall provide a reasonable amount of anonymity in presence of malicious mixes. Finally, we add further constraints (e.g. for balanced path selection) and define objective functions (minimize latency), depending on the scheduling strategy.

4.3.1 Flow Network Model. Probabilistic path selection may be modeled by a flow in a flow network as depicted in Fig. 5. The network consists of $l \times n + 1$ nodes, with l being the length of circuits and $n = |V|$ the number of mixes. There is one node for each combination of mix and layer, and one special source node s , corresponding to a user. Links only exist from s to the first layer and between any pair of nodes on adjacent layers. Unlike “traditional” flow networks, links do not have a capacity. Now, the basic idea to map a flow in the given network to a probabilistic path selection is to use a flow demand (value) of 1 and let s be the source of the flow. The target of the flow is no single node, but the last layer in its entirety, which represents all possible rendezvous mixes. The flow is characterized by its value on all possible links:

- Values $x_{s,v}$ for links from s to all mixes $v \in V$.
- Values $x_{u,v,i}$ for links between layers, $u, v \in V, 1 \leq i \leq l - 1$.

Given a fixed flow, the corresponding path selection algorithm iteratively selects mixes as follows (*all users use the same flow*):

- (S1) The entry mix is randomly selected according to the following distribution over all mixes $v \in V$ (random variable X_1):

$$\Pr(X_1 = v) = x_{s,v}$$

- (S2) For $2 \leq i \leq l$: Given the currently last hop $u_{i-1} \in V$ on the selected path, the next hop is selected according to the following conditional distribution over all mixes $v \in V$ (random variable X_i):

$$\Pr(X_i = v | X_{i-1} = u_{i-1}) = \frac{x_{u_{i-1},v,i-1}}{\sum_{w \in V} x_{u_{i-1},w,i-1}}$$

$$\sum_{v \in V} x_{s,v} = 1 \quad (1)$$

$$x_{s,v} = \sum_{u \in V} x_{v,u,1} \quad \forall v \in V \quad (2)$$

$$\sum_{u \in V} x_{u,v,i} = \sum_{u \in V} x_{v,u,i+1} \quad \forall v \in V, 1 \leq i \leq l-2 \quad (3)$$

Figure 6: Flow conservation constraints.

$$x_{v,v,i} = 0 \quad \forall v \in V, 1 \leq i \leq l-1 \quad (4)$$

$$z_v \in \{0, 1\} \quad \forall v \in V \quad (5)$$

$$x_{s,v} \geq z_v \times a_{\min} \quad \forall v \in V \quad (6)$$

$$x_{s,v} \leq z_v \quad \forall v \in V \quad (7)$$

$$x_{u,v,i-1} \leq \frac{1}{d} \sum_{w \in V} x_{w,v,i-1} \quad \forall u, v \in V, 2 \leq i \leq l \quad (8)$$

Figure 7: Anonymity constraints.

The defined path selection has the following important properties:

(P1) If attackers know that mix u_i was selected on a path at layer $i \geq 2$, the posterior probability distribution for the previous hop $v \in V$ is as follows:

$$\Pr(X_{i-1} = v | X_i = u_i) = \frac{x_{v,u_i,i-1}}{\sum_{w \in V} x_{w,u_i,i-1}}$$

(P2) Flow values on links equal the absolute probability for selecting the link $(u, v \in V, 1 \leq i \leq l-1)$:

$$\Pr(X_i = u \wedge X_{i+1} = v) = x_{u,v,i}$$

(P3) Total flow values at nodes in the flow network equal the absolute probability for selecting the corresponding mix $v \in V$ at the corresponding layer i :

$$\Pr(X_i = v) = \begin{cases} x_{s,v} & \text{for } i = 1 \\ \sum_{u \in V} x_{u,v,i-1} & \text{for } 2 \leq i \leq l \end{cases}$$

4.3.2 Basic MILP Formulation. To find a suitable flow (i.e. path selection probabilities) by solving a MILP, flow values $x_{s,v}$ and $x_{u,v,i}$ are interpreted as *variables* in the MILP. Then, flow constraints can be defined as depicted in Fig. 6: Constraint (1) fixes the total flow value to 1, constraints (2) and (3) realize flow conservation.

4.3.3 Anonymity Constraints. We now add constraints that shall guarantee some reasonable degree of anonymity, even if some mixes are malicious. The anonymity constraints are shown in Fig. 7 and are derived from the following thoughts.

First, the same mix should not be selected on two adjacent layers for the same path, realized by forcing the corresponding flow variables to be 0, see constraint (4). Note that the same mix v still can be selected multiple times for the same circuit, e.g. every second hop. Nevertheless, we do not expect this to be a problem because if there is a honest mix in between, v cannot infer that he is part of the same circuit multiple times. Completely avoiding duplicated mixes on paths would require a different model that basically enumerates all possible paths and consequently does not scale with l .

$$z_{u,v,i} \in \{0, 1\} \quad \forall u, v \in V, 1 \leq i \leq l-1 \quad (9)$$

$$x_{u,v,i} \leq z_{u,v,i} \quad \forall u, v \in V, 1 \leq i \leq l-1 \quad (10)$$

$$z_{u,v,i} \times w_{u,v} \leq y_i \quad \forall u, v \in V, 1 \leq i \leq l-1 \quad (11)$$

$$x_{s,v} = \frac{c_v}{C} \quad \forall v \in V \quad (12)$$

$$\sum_{u \in V} x_{u,v,i-1} = \frac{c_v}{C} \quad \forall v \in V, 2 \leq i \leq l \quad (13)$$

Figure 8: Constraints for modeling batch processing.

Next, it is import that each honest mix v introduces a reasonable amount of uncertainty for attackers at every layer i it is used. There are two ways to introduce such uncertainty:

- *Directly:* If mix v handles many circuits at layer i , they are all perfectly mixed – even if all use the same next hop.
- *Indirectly:* If there is a variety of possible next hops at v , more circuits may be mixed at the next layer. Ideally, the posterior probability distribution for the next hop should also be close to uniform in this case. Otherwise, attackers might still be able to correctly guess the next hop of a targeted circuit with high probability.

“Next hop” refers to the attackers’ perspective here: When attackers try to break the location anonymity of a user, they have to track its circuit in *reverse* direction, i.e. starting at the known rendezvous mix. Then, we distinguish three cases to derive MILP constraints:

- In the worst case, all mixes on a selected path are malicious and location anonymity is broken. While the probability for this to happen should be low, it cannot be modelled in our MILP because the set of malicious mixes is unknown.
- In the second-worst case, only the entry mix v is honest. Then, there is no next hop to indirectly mix more circuits, so v should (on average) handle many circuits at the entry layer. This is modeled by constraints (5), (6), and (7): Whenever a mix is used at entry position (binary indicator variables z_v), their probability for being selected as entry mix must be at least $a_{\min}$.
- If there is one honest mix v at any other position, i.e. at layer $2 \leq i \leq l$, constraint (8) enforces a minimum *in-degree* of d in the flow network at layer i , which corresponds to the *out-degree* in the tracked direction. Technically, the minimum degree d at v is enforced by restricting each link adjacent to v to have a flow less than $1/d \times f_v$, with f_v being the total flow at v . Because of property (P1), constraint (8) also forces posterior probabilities of next hops to be less than $1/d$.

We do not add constraints for explicitly protecting the relationship anonymity of a user because it is always at least as good as the location anonymity of his communication partner.

4.3.4 Batch Processing. The limit for end-to-end latency when using batch processing is determined by the maximum latency between layers. It can only be decreased by not using high latency links for realizing the flow in the network. To decide whether a link is used, binary indicator variables $z_{u,v,i}$ are necessary, see constraint (9) in Fig. 8. The flow may only use a link if the corresponding indicator variable allows it, see constraint (10). Furthermore, we

introduce one variable y_i , $1 \leq i < l$ that represents the maximum latency between layer i and $i + 1$. This is realized by constraint (11).

To avoid bottlenecks for batch processing, mixes should, on average, handle a total number of circuits proportional to their capacity *in every layer*. Let $C = \sum_{v \in V} c_v$ be the total capacity of all mixes. Then, using property (P3), this balance is expressed by constraint (12) for the entry layer and (13) for all other layers.

Finally, the objective function for minimizing latency on selected paths and consequently minimizing end-to-end latency is:

$$\min \sum_{1 \leq i \leq l-1} y_i \quad (14)$$

$$\text{subject to constraints (1) – (13)} \quad (15)$$

Note that the latency of the “rendezvous link” is not explicitly modeled because caller and callee select paths independently. It is determined by the maximum latency between any pair of rendezvous mixes. Latency between users and their entry mix is not considered either so that every user has the same probability of selecting a specific path.

4.3.5 De-Jitter Buffer. When using de-jitter buffers for scheduling, balance only has to be achieved in total, i.e. the number of circuits a mix has to handle regardless of its position has to be proportional to its capacity, again using property (P3):

$$x_{s,v} + \sum_{u \in V, 2 \leq i \leq l} x_{u,v,i-1} = \frac{c_v}{C} \times l \quad \forall v \in V \quad (16)$$

As objective function, we minimize the *average latency* on circuits, which requires no additional variables:

$$\min \sum_{u,v \in V, 1 \leq i \leq l-1} x_{u,v,i} \times w_{u,v} \quad (17)$$

$$\text{subject to constraints (1) – (8) and (16)} \quad (18)$$

Our rationale for being more “greedy” here is to compensate for the additional delays introduced by the de-jitter buffers. Again, the latency of the entry and rendezvous links is not modeled.

5 EVALUATION

We first qualitatively discuss to what extend our objectives, see Sec. 2, can be achieved in our model. For quantitative evaluation, we then solve the MILPs for varying datasets and parameters.

5.1 Qualitative Discussion

Note that the following discussion mainly is a summary of arguments that we also used in earlier sections to guide our design.

5.1.1 Anonymity. First, both packet scheduling strategies defeat network flow watermarking by enforcing a deterministic timing of egress packets. Potential gaps in packet flows are padded with dummy packets that are indistinguishable from voice packets.

Next, we discuss if the “greedy” path selection that minimizes circuit latencies gives attackers any significant advantage, given that they already tracked a circuit up to some honest mix v . We argue that this is not the case for two reasons: First, a solver of our MILP is expected to create a close to uniform posterior probability distributions across possible next hops. That is because constraint (8) forces a minimum node degree of d at v and further restricts the probability for each next hop to be less than $1/d$. In addition, the

solver will try to *only* use the d links with the lowest latency as long as all other constraints can be met. Consequently, posterior probabilities for most used links should be close to $1/d$, regardless of their latency. Second, when using de-jitter buffers, attackers might be able to rule out some next hops if the known path prefix up to v already has a high latency and attackers know that the *total* latency of the attacked circuit is low. However, only malicious users are able to measure circuit latency because sequence numbers are rewritten at each mix. Even then, we do not expect a RTT measurement solely based on voice data to be precise enough to be problematic. Nevertheless, an easy countermeasure for users that require location anonymity is to *locally* delay voice packets if they selected a path with very low latency.

5.1.2 Quality of Service (QoS). By minimizing latency in our MILPs, we are able to evaluate the theoretical limits of our approach. Our quantitative evaluation will have to show whether these lower bounds for end-to-end latency are below 400 ms.

5.1.3 Efficiency. Probabilistic path selection as modeled by our MILPs results in mixes to handle a total number of circuits proportional to their capacity, on average. Consequently, the probability for processing/forwarding bottlenecks is expected to be low while the network is not used to its overall capacity.

Overhead in circuit packets is low, e.g. using 8 B in total for the circuit identifier and the sequence number should suffice. Nevertheless, the total system capacity occupied by a single call is increased by 2l-times compared to a “normal” voice call. Therefore, the circuit length l should not be too long to keep the system efficient.

5.1.4 Practicability. With regard to packet scheduling, we expect de-jitter buffers to lead to fewer problems in practice because they mainly rely on local state. In contrast, batch processing requires tightly synchronized clocks and a global schedule for the different layers and sequence numbers.

Regarding path selection, MILPs are a useful tool to find limits, i.e. for end-to-end latency in our case, but solving them for hundreds or even thousands of mixes is not feasible: The number of variables and constraints grows quadratically with the number of mixes. Consequently, heuristics have to be developed after the most promising values for our system parameters have been identified.

5.2 Quantitative Evaluation

For quantitative evaluation, we solve the defined MILPs with varying parameters and for various latency and bandwidth datasets obtained from the Tor network. As solver, we use Gurobi 9.1.1. We aim to answer the following research questions:

- Can latency aware path selection achieve an end-to-end latency ≤ 400 ms and a reasonable high anonymity at the same time? Especially when attackers compromise mixes with a low average latency to other mixes?
- Can the additional delay at each mix caused by the de-jitter buffer scheduling strategy be compensated by using links with lower latency on average?
- How do heterogeneous mix capacities impact the path selection? And how is anonymity affected when attackers compromise mixes with large capacities?

5.2.1 The Dataset. From January 2019 to December 2020, we used the method described in [16] to estimate the pairwise latencies between 4102 Tor relays. We further used their consensus weight as an estimate for their bandwidth in kB s^{-1} [17]. Our raw dataset and further details about pre-processing are also publicly available [20]. To generate datasets for evaluation, we filtered 627 relays that are not able to relay at least 100 voice calls with 32 kbit s^{-1} each. We further filtered 2551 relays for which we did not have measurements to at least 90 % of all other relays, e.g. due to node churn during two years. Remaining missing measurements were completed by using the average latency of the two involved relays. In result, we obtained a “base” dataset with 924 stable relays, from which random subsets are drawn for evaluation.

To study networks with varying heterogeneity in mix capacities, subset generation is parametrized by a quantile q : Let lo_q be the q -th quantile of all capacities in the dataset and hi_q be the $(1 - q)$ -th quantile. Then, only mixes v with a capacity $\text{lo}_q \leq c_v \leq \text{hi}_q$ are considered for subsets. For each experiment, we randomly generated 32 datasets as input for the MILPs. Bar charts show the average results of the 32 datasets with 95 % confidence intervals.

5.2.2 Metrics. We use the following metrics for evaluation.

Latency Metrics. To measure end-to-end latency, we simulate m calls and use the average and 0.95-th quantile of all calls as metrics. Note that for batch processing, all calls have the same latency (for a fixed path selection). To find limits for latency, we use the theoretical latency in our model, including random delays for realizing de-jitter buffers, but do not capture potentially increased latency in practice as discussed in Sec. 4.2.2 and Sec. 4.2.3. Latency between users and their entry mix is not included in any case. Consequently, the theoretical latency should not be too close to 400 ms to compensate for potential bad access links of users.

Anonymity Metrics. To measure anonymity, we calculate static worst case probabilities, i.e. the probability for a user to select a completely malicious circuit (no location anonymity) and the probability for caller and callee to both select a completely malicious circuit (no relationship anonymity). We skip the latter when showing results; it equals the former to the power of 2. We further simulate m concurrent calls and use the *relative anonymity set size* as metric, i.e. if A is the location anonymity set of a user, we use $(|A|-1)/(2m-1)$. Possible values are in $[0; 1]$, with 1 being perfect anonymity. Again, results for relationship anonymity are omitted as it is guaranteed to be at least as good as location anonymity. Naturally, anonymity metrics depend on the set of malicious mixes $\mathcal{M} \subset V$. We assume a fraction f of mixes to be malicious, i.e. $|\mathcal{M}| = \lceil f \times |V| \rceil$, and compare three strategies to pick $\mathcal{M}$:

- (1) Uniformly at random
- (2) Mixes with the lowest average latency to all other mixes
- (3) Mixes with the highest capacity

5.2.3 Default Parameters. If not stated otherwise, the system and evaluation parameters are set according to Table 1.

5.2.4 Experiment 1: Overview.

Setup. For our first experiment, we are mainly interested in an overview how well the path selection strategies generated by our

Table 1: Default parameters

Parameter	Symbol	Default Value
Number of mixes	n	50
Simulated calls	m	1000
Voice payload per packet	Δ_v	20 ms
Circuit length	l	4
Minimum degree	d	16
Min. entry mix probability	$a_{\min}$	$1/n$
Fraction of malicious mixes	f	0.2
Attack strategy		by latency

MILPs behave. We generated datasets of size $n = 50$ and manually set the relative capacity c_v/C of all mixes $v \in V$ to $1/n$, i.e. assume completely homogeneous capacities. We set the quantile parameter to $q = 0.33$ to draw our datasets from a more homogeneous population as well. We further varied the circuit length from $l = 3$ to $l = 6$, resulting in 6 to 12 passed mixes for end-to-end delivery. We evaluated a very strong attacker model, with a fixed fraction of malicious mixes $f = 0.2$ and we assume that the mixes with lowest average end-to-end latency are malicious, as attackers will try to improve their position.

Results. The results when using batch processing are shown in Fig. 9. Unfortunately, the only parameter combination that is likely to achieve end-to-end latencies of less than 400 ms in this scenario is $l = 3, d = 4$. But then, location anonymity is weak: One out of 100 circuits will not provide any location anonymity at all. One reason for the high end-to-end latency, especially significant for short paths, is batch processing at rendezvous mixes: Because rendezvous mixes are fully meshed, the maximum latency between any two mixes dictates the bottleneck for all rendezvous links.

While bad rendezvous links cannot be avoided when using de-jitter buffers either, they only affect the corresponding calls. Furthermore, the average latency on circuits is minimized by our MILP. In combination, see Fig. 10, better trade-offs between end-to-end latency and anonymity are possible, despite the additional random delay at each mix. For example, using a circuit length of $l = 4$ or $l = 5$ in combination with a higher node degree $d \geq 12$ results in very good location anonymity while still being able to meet the latency requirement for a large majority of calls (0.95-th quantile). Note, the average end-to-end latency is lower but not shown here for brevity. Interestingly, increasing d in this scenario seems to yield better trade-offs in general. For example, when circuit length is set to $l = 4$, latency increases linear with d while anonymity improves exponentially. Especially if d is too low, the solver will greedily select very good links only, significantly increasing the chance that mixes with the lowest average end-to-end latency are connected. As a general rule of thumb, it might be a good idea to set $d > |\mathcal{M}|$ (which is 10 in this scenario), but this needs yet to be confirmed for larger networks. In practice, an upper bound for $|\mathcal{M}|$ can only be estimated, of course.

5.2.5 Experiment 2: Anonymity Set Sizes.

Setup. In this experiment, we take a closer look at achievable anonymity when using de-jitter buffers. We set $d = 16$ and only evaluate circuit lengths $l = 4$ and $l = 5$ because this combination

Figure 9: Latency and anonymity when using batch processing, for varying circuit lengths l and minimum node degrees d .

Figure 10: Latency and anonymity when using de-jitter buffers, for varying circuit lengths l and minimum node degrees d .

Figure 11: Relative location anonymity set sizes when using de-jitter buffers, for varying number m of concurrent calls.

offered the best trade-off in experiment 1. This time, we are interested in the relative size of anonymity sets for a varying number m of concurrent calls (and the 0.05-th quantile across these calls). Furthermore, we also increased $a_{\min}$ to $2/n$ for comparison.

Results. The results are shown in Fig. 11, the mentioned parameter a was used to set $a_{\min} = a/n$. When there are enough calls, i.e. $m \geq 400$, anonymity sets include the majority of users, especially for $l = 5$. If fewer calls can be mixed, especially for $m < 200$, differences between the parameter combinations are more significant: In this case, setting $l = 5$ improves the mixing properties of path selection significantly. Nevertheless, we expect that similar effects can be achieved by using dummy circuits. It is more important to achieve a high “peak” when there are a sufficient number of circuits. And this can also be achieved with $l = 4$ when increasing the parameter $a_{\min}$, which did not degrade latency. This is especially helpful to improve anonymity in the case where attackers are able to identify the entry mix of a user.

5.2.6 Experiment 3: Heterogeneous Mix Capacities.

Setup. In our final experiment, we use the relative mix capacities as determined by the datasets, with varying heterogeneity by using

different quantiles q to consider for generation. To compensate for the heterogeneous capacities at entry layer, we lowered $a_{\min}$ to the lowest relative capacity across all mixes for each dataset. While we focused on a single attack strategy before, we evaluate all three suggested this time. Remaining parameters are as in experiment 2.

Results. The results for this experiment are depicted in Fig. 12. First of all, attackers that control the mixes with the highest capacity are the most serious threat in this scenario. This is easily explained by the fact that in our strong threat model, the fraction $f = 0.2$ relates to the *number* of malicious mixes, not their forwarding capacity. When heterogeneity in mix capacities increases (q decreases), the 20 % most powerful mixes contribute a total capacity of more than 20 %. For example, using $q = 0.1$ results in a total malicious capacity of 46 % on average, i.e. making the attacker even more powerful. Consequently, the probability of selecting a completely malicious circuit is expected to be as high as 0.46^l , e.g. ≈ 0.04 for $l = 4$, see Fig. 12(a). Nevertheless, even such powerful attackers can be defeated to some extent by increasing the circuit length to $l = 5$, see Fig. 12(b). This still offers acceptable latency, see Fig. 12(c), but naturally decreases efficiency because there will be two more copies of each voice packet.

Except for the very powerful attack, there is no significant advantage for attackers when they control the mixes with the lowest average end-to-end latency as compared to random mixes. This is another indication that “greedily” selecting links with better latency *does not* degrade anonymity.

5.3 Conclusion

In conclusion, our experiments showed that using de-jitter buffers for packet scheduling offers the better trade-off in a Tor-like network. In such a scenario, especially rendezvous link latencies are too heterogeneous for batch processing to provide an acceptable

Figure 12: Anonymity and latency when using de-jitter buffers, for varying attacks and heterogeneity of mix capacities.

end-to-end latency. In contrast, the additional latency introduced by de-jitter buffers can be compensated by taking advantage of low latency links during path selection. When the minimum node degree is set high enough, this “greedy” strategy does not degrade anonymity. Last but not least, de-jitter buffers are also more practical as they mainly rely on local state of mixes.

Note that while we could only extensively evaluate a small network size of $n = 50$, first results for selected parameter values and up to $n = 90$ mixes coincide with our presented results. Still, solving our MILPs to determine a suitable path selection strategy is not viable to implement larger networks at the scale of Tor. For example, the average solving time for the MILP (de-jitter buffer, 32 runs) increased from 30 s for $n = 50$ to 938 s for $n = 90$. Consequently, heuristic solutions have to be developed in future.

6 CONCLUSION AND FUTURE WORK

In this article, we investigated limits for latency-aware path selection for anonymous voice calls. For this, we modeled two different packet scheduling strategies, both able to defeat network flow watermarking attacks, but introducing different constraints for path selection. We modeled these constraints by formulating MILPs that minimize latency on circuits yet provide anonymity at the same time. Our results show that batch processing of voice packets from different circuits is not able to reliably achieve acceptable latencies in voluntary-based systems like Tor. Link latencies are too heterogeneous and high latency links cannot be avoided entirely, equally degrading QoS for all calls. In contrast, removing potential watermarks on circuits by implementing independent de-jitter buffers at each mix is able to achieve latencies of less than 400 ms for a large majority of calls. In this case, path selection benefits from low latency links without degrading anonymity.

As future work, we study heuristics for solving the MILP that are able to efficiently generate path selection strategies for larger networks like Tor, i.e. in the order of 5000 mixes. In this course, reasonable values for the minimum node degree have to be studied, depending on the number of mixes. Then, in combination with an anonymous dialing system, anonymous voice calls with good QoS may be implemented for general availability.

REFERENCES

- [1] Masoud Akhond, Curtis Yu, and Harsha V Madhyastha. 2012. LASTor: A Low-Latency AS-Aware Tor Client. In *IEEE SP*. 476–490.
- [2] Robert Annessi and Martin Schmiedecker. 2016. Navigator: Finding Faster Paths to Anonymity. In *IEEE EuroS&P*. 214–226.
- [3] Eric Chan-Tin and Nicholas Hopper. 2011. Accurate and Provably Secure Latency Estimation with Treple. In *NDSS*.
- [4] David Chaum. 1981. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM* 24, 2 (1981), 84–90.
- [5] Chen Chen, Daniele E Asoni, Adrian Perrig, David Barrera, George Danezis, and Carmela Troncoso. 2018. TARANET: Traffic-Analysis Resistant Anonymity at the Network Layer. In *IEEE EuroS&P*. 137–152.
- [6] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. 2018. Anonymity Trilemma: Strong Anonymity, Low Bandwidth Overhead, Low Latency - Choose Two. In *IEEE SP*. 108–126.
- [7] Roger Dingledine and Nick Mathewson. 2019. Tor Path Specification. <https://gitweb.torproject.org/torspec.git/tree/path-spec.txt> (visited on 03/09/2021).
- [8] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-Generation Onion Router. In *13th USENIX Security*.
- [9] Danny Dolev and Andrew Yao. 1983. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory* 29, 2 (1983), 198–208.
- [10] Christian Franck and Uli Sorger. 2016. Untraceable VoIP Communication based on DC-nets. *arXiv preprint arXiv:1610.06549* (2016).
- [11] International Telecommunication Union. 2003. *One-way transmission time* (ITU-T recommendation G.114 ed.). International Telecommunication Union.
- [12] Yunchan Jung and Caroline Manzano. 2014. Burst packet loss and enhanced packet loss-based quality model for mobile voice-over Internet protocol applications. *IET Communications* 8, 1 (2014), 41–49.
- [13] David Lazar, Yossi Gilad, and Nickolai Zeldovich. 2019. Yodel: Strong Metadata Security for Voice Calls. In *27th ACM SOSP*. 211–224.
- [14] Stevens Le Blond, David Choffnes, William Caldwell, Peter Druschel, and Nicholas Merritt. 2015. Herd: A Scalable, Traffic Analysis Resistant Anonymity Network for VoIP Systems. *ACM SIGCOMM* 45, 4 (2015), 639–652.
- [15] Jonathan Mayer, Patrick Mutchler, and John C. Mitchell. 2016. Evaluating the privacy properties of telephone metadata. *Proceedings of the National Academy of Sciences* 113, 20 (2016), 5536–5541.
- [16] Andriy Panchenko and Johannes Renner. 2009. Path Selection Metrics for Performance-Improved Onion Routing. In *SAINT*. 114–120.
- [17] Tor Project. 2019. Tor Directory Protocol, Version 3. <https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt>.
- [18] David Schatz, Michael Rossberg, and Guenter Schaefer. 2017. Reducing Call Blocking Rates For Anonymous Voice Over IP Communications. In *IEEE ICUMT*. 377–385.
- [19] David Schatz, Michael Rossberg, and Guenter Schaefer. 2021. Hydra: Practical Metadata Security for Contact Discovery, Messaging, and Dialing. In *ICISSP*. 191–203.
- [20] David Schatz, Michael Rossberg, and Guenter Schaefer. 2021. Large-scale Latency Measurements in the Tor Network (v1.0). <https://doi.org/10.5281/zenodo.4911583>
- [21] Piyush Kumar Sharma, Shashwat Chaudhary, Nikhil Hassija, Mukulika Maity, and Sambuddho Chakravarty. 2020. The Road Not Taken: Re-thinking the Feasibility of Voice Calling Over Tor. *PETS* 2020, 4 (2020), 69–88.
- [22] Micah Sherr, Matt Blaze, and Boon Thau Loo. 2009. Scalable Link-Based Relay Selection for Anonymous Routing. In *PETS*. Springer, 73–93.
- [23] Matthew Traudt, Rob Jansen, and Aaron Johnson. 2020. FlashFlow: A Secure Speed Test for Tor. *arXiv preprint arXiv:2004.09583* (2020).
- [24] Tao Wang, Kevin Bauer, Clara Forero, and Ian Goldberg. 2012. Congestion-aware Path Selection for Tor. In *International Conference on Financial Cryptography and Data Security*. Springer, 98–113.
- [25] Xinyuan Wang, Shiping Chen, and Sushil Jajodia. 2007. Network Flow Watermarking Attack on Low-Latency Anonymous Communication Systems. In *IEEE SP*. 116–130.