SN Computer Science (2022) 3: 341
https://doi.org/10.1007/s42979-022-01231-9

ORIGINAL RESEARCH

Hydra: Practical Metadata Security for Contact Discovery, Messaging, and Voice Calls

David Schatz[1] · Michael Rossberg[1] · Guenter Schaefer[1]

Received: 5 October 2021 / Accepted: 24 May 2022 / Published online: 18 June 2022
© The Author(s) 2022

Abstract
Protecting communications' metadata can be as important as protecting their content, i.e., recognizing someone contacting a medical service may already allow to infer sensitive information. There are numerous proposals to implement anonymous communications, yet none provides it in a strong (but feasible) threat model in an efficient way. We propose Hydra, an anonymity system that is able to efficiently provide metadata security for a wide variety of applications. Main idea is to use latency-aware, padded, and onion-encrypted circuits even for connectionless applications. This allows to implement strong metadata security for contact discovery and text-based messages with relatively low latency. Furthermore, circuits can be upgraded to support voice calls, real-time chat sessions, and file transfers—with slightly reduced anonymity in presence of global observers. We evaluate Hydra using an analytical model as well as call simulations. Compared to other systems for text-based messaging, Hydra is able to decrease end-to-end latencies by an order of magnitude without degrading anonymity. Using a dataset generated by performing latency measurements in the Tor network, we further show that Hydra is able to support anonymous voice calls with acceptable quality of service in real scenarios. A first prototype of Hydra is published as open source.

Keywords Metadata security · Strong anonymity · Onion routing · Voice over IP

Introduction

Instant messaging services like Signal and WhatsApp have become a ubiquitous utility for human-to-human communications. Due to their IP-based design, they are able to support a wide variety of applications, including text-based chats, file transfer and voice calls. However, such services also raise privacy concerns, like a (de-)centralized provider collecting communication *content* and *metadata* on a large scale. And while confidentiality of communication content may easily be achieved by end-to-end encryption, metadata is far more challenging to protect. For one, providers may still easily record communication metadata. For another, third parties may observe size and timing information of (encrypted) IP packets in different sections of the network to infer communication relationships. Unfortunately, communication relationships may also leak sensitive communication content, for example, when users contact a specialized counseling or medical service [1].

A promising concept to protect metadata are *communication mixes*, first introduced by David Chaum [2]. Instead of trusting a single provider, packets are routed via multiple mixes, which are deployed in a distributed fashion. At the same time, a fixed uniform size is enforced for all packets by padding or splitting of application layer messages. Each packet is encrypted in layers, one for each mix, which are removed along the way. In result, correlation of packets based on size or encrypted content is not possible. To defeat attacks based on timing of packets entering and leaving an observed mix, packets are collected in batches and forwarded in random order. Unfortunately, powerful

This article is part of the topical collection "Information Systems Security and Privacy" guest edited by Steven Furnell and Paolo Mori.

✉ David Schatz
david.schatz@tu-ilmenau.de

Michael Rossberg
michael.rossberg@tu-ilmenau.de

Guenter Schaefer
guenter.schaefer@tu-ilmenau.de

[1] Telematics/Computer Networks Research Group, Technische Universität Ilmenau, Ilmenau, Germany

attackers like a global observer are still able to infer communication relationships by performing *disclosure attacks* [3, 4]. Such attacks exploit user churn and correlate the sending/receiving behavior of users over time. To mitigate disclosure attacks, users should appear to be sending and receiving packets even if they are not in an active communication, which requires dummy packets [4].

Modern *anonymity systems* that build upon communication mixes may further be categorized by the cryptographic primitives they use for layered encryption. First, there are many recent systems [5–7] that aim at providing metadata security for short text messages even in the presence of a global observer, often called *strong anonymity*. As originally proposed by Chaum, they use asymmetric cryptography for layered encryption of every packet. While this avoids any setup or (long-term) state at mixes, it is not computationally efficient, especially when combined with many dummy packets. Furthermore, the missing connection context at mixes prevents tunnelling of connection-oriented applications like Voice over IP (VoIP) in a secure way: Attackers could use a simple form of network flow watermarking [8] that drops packets at the sender and correlates the resulting "gaps" in the packet flow at the receiver.

Both problems may be tackled by *onion routing*, which became popular in the Tor network [9]: Before any application data is sent, users select a fixed path of mixes and use them to setup a *circuit*. While circuit setup still uses asymmetric cryptography for key exchange, remaining packets may be onion-encrypted using efficient symmetric ciphers. Furthermore, mixes may use their state about a circuit to defeat network flow watermarking [10–13]: A deterministic schedule for packet sending times at each mix defeats attacks based on correlation of packet timings, and gaps can be detected and padded with dummy packets ("padded circuit"). However, existing circuit-based systems also have various drawbacks like missing defense against watermarking [9], weak anonymity in the presence of malicious mixes [11] or weak *location anonymity* [10, 12] (IP addresses are leaked to communication partners, facilitating geographical tracking [1]).

In this article, we present Hydra, a circuit-based system that overcomes the flaws of similar designs and is able to efficiently provide strong anonymity for a wide variety of applications. For this, all users are synchronized to periodically setup circuits, for example, once every ten minutes. A packet-based rendezvous mechanism between circuit endpoints provides location anonymity and allows the same circuit to be used to send contact requests and text messages to different users, inspiring the name Hydra. To defeat both network flow watermarking and disclosure attacks, circuits are padded by mixes and users. As long as users do not participate in a communication or only send text messages, the padding rate is low, in the order of 3 pps (packets per second), for efficiency. To also support voice calls (and small file transfers), existing circuits may be "upgraded" to a higher padding rate. Note that this comes at the cost of reduced robustness against disclosure attacks because churn of upgraded circuits is expected to be higher. Furthermore, path selection for circuits has to consider mix capacities and latencies between mixes to provide good quality of service (QoS) for interactive voice calls [13]. For this, we propose a novel approach to determine a suitable probabilistic path selection by solving a minimum cost flow problem.

The rest of this article is structured as follows: in the next section, we define our objectives and threat model. Based on the objectives, related work is discussed in the subsequent section. Thereafter, design details of Hydra are presented. We evaluate Hydra in the penultimate section using an analytical model and call simulations. In the final section, we conclude our article.

The article is an extended version of [14]. It adds support for voice calls by introducing a protocol to "upgrade" circuits to a higher padding rate. Furthermore, path selection is tuned to consider mix capacities and latencies between mixes to meet the strict QoS requirements for interactive voice calls. Our extended evaluation also reflects this addition, most importantly by studying the refined path selection. Additionally, the protocol for contact discovery is improved compared to the original version of our article. Further note that the minimum cost flow problem we introduce for path selection is a special case of the mixed integer linear program (MILP) we defined in [13]. It is tailored for usage in Hydra and is far more efficient to solve than the generic version.

System Objectives and Threat Model

This section defines both the functional and non-functional objectives for the Hydra system. Furthermore, we define our threat model.

Functional Objectives

From a functional perspective, we want to provide the following applications, similar to popular messengers like Signal and WhatsApp.

1. **Contact Discovery.** Users must be able to discover contacts that are also using Hydra. Contact discovery should be based on long-term *user pseudonyms*.
2. **Messaging.** After contact discovery, users must be able to send messages to their contacts. While we focus on text-based messages, the transfer of small files like pictures or voice messages should also be supported. When

recipients are offline, the system must provide storage for later delivery.

3. **Voice Calls.** Users must be able to initiate a voice call with their contacts.

Non-functional Objectives

In addition to the functional objectives, the following non-functional objectives shall be achieved. Whenever objectives are in conflict, suitable trade-offs have to be achieved.

1. **Confidentiality and Integrity.** Just like existing messengers, an anonymous messaging system must provide end-to-end security for all messages and voice calls. That is, the confidentiality and integrity of all end-to-end packets must be protected by cryptographic mechanisms.
2. **Anonymity.** First, *relationship anonymity* shall be provided. That is, communication relationships (including more fine-grained metadata like frequency and duration) shall be hidden from third parties. Moreover, some users may also require *location anonymity*: Neither third parties nor untrusted contacts shall be able to infer the mapping of user pseudonyms to their IP address.
3. **QoS**. When both sender and recipient of a message are online at the same time, end-to-end latency for delivery shall be in the order of seconds to enable interactive chats. For voice calls, QoS requirements are more strict: Mouth-to-ear latency should ideally not exceed 150 ms, whereas a latency of more than 400 ms usually is not acceptable for interactive voice calls [15]. Additionally, packet loss shall be below 5% [16].
4. **Efficiency and Scalability.** The amount of system resources (like number and capacity of mixes) to support a given number of active users with acceptable QoS shall be low. Moreover, *horizontal scalability* is required. That is, it should not only be possible to support more users by upgrading the capacity of existing system entities like mixes (vertical scalability), but also by adding more system entities. For users on mobile devices, energy efficiency is important to not degrade battery life.
5. **Robustness.** The system shall be robust in the presence of node churn, which includes both users and system entities like mixes.

Threat Model

We assume powerful external attackers according to the Dolev–Yao model [17]. That is, they are able to observe, drop, delay, manipulate, replay, and forge packets anywhere in the network. However, they cannot bypass cryptographic protections. Furthermore, we assume that a limited number of system entities and users are malicious and cooperate with external attackers.

Related Work

Basically, there are three different techniques that may be used to implement anonymity systems, as depicted in Fig. 1: Communication mixes [2], Dining Cryptographer networks (DC-nets) [18], and private information retrieval (PIR) [19]. However, existing anonymity systems that are based on DC-nets or PIR have inherent scalability issues due to intense usage of broadcast communication or computational expensive operations. For example, Riposte [20] implements text-based, anonymous broadcasting using PIR techniques, but introduces end-to-end latencies in the order of hours for one million users. Addra [21] introduces a more efficient PIR scheme to implement anonymous voice calls, but only supports a few thousand users with acceptable latency. There also is an approach to use (DC-nets) for voice calls [22], but it only supports a few hundred users.

In contrast, the concept of communication mixes is more promising to implement interactive anonymous communications for millions of users. The main idea is to unlink sender and recipient of a packet by relaying it via multiple mixes. A fixed uniform size for all packets and layered encryption (also called *onion encryption* throughout the article) defeats correlation of packet size and content by external observers and malicious mixes. To also defeat attacks that correlate the timing of packets entering and leaving a mix, most systems use some sort of batch processing: Instead of forwarding packets as fast as possible, mixes collect many packets in a batch before forwarding them in shuffled order. When

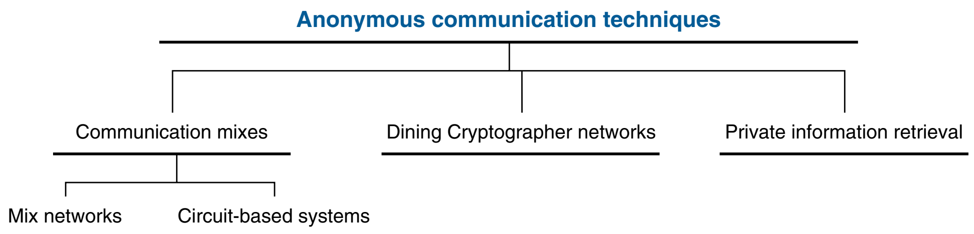
Anonymous communication techniques
Communication mixes
Dining Cryptographer networks
Private information retrieval
Mix networks
Circuit-based systems

Fig. 1 Categorization of existing anonymous communication techniques

assuming a global observer, users also have to send dummy packets to defeat or mitigate *disclosure attacks* [3, 4], which analyze the sending/receiving behavior of users over time. Based on these core principles, many anonymity systems were proposed in recent years.

Mix Networks

To provide anonymity for (short) text messages, many recent proposals share a core concept [5–7, 23, 24]: Onion-encrypted packets are used to read and write messages from and to *mailboxes*, providing location anonymity for all users. For end-to-end delivery of messages, randomized mailbox identifiers known by both sender and recipient are used. To defeat disclosure attacks, the systems use periodic and synchronized communication rounds. In each round, every user is asked to send a packet, even if he is not in an active conversation. In combination with using asymmetric cryptography for onion encryption, this limits scalability: Supporting millions of users with low end-to-end latency is only possible by deploying many powerful mixes.

The referenced systems differ in path selection for onion-encrypted packets, mailbox management, and defense against active attacks: Whereas Vuvuzela [23] uses a fixed mix cascade, which does not scale horizontally, others use some sort of *stratified topology*. That is, mixes are organized in ordered layers (not necessarily disjoint) and users select one mix per layer at random.

Mailbox identifiers in Vuvuzela, Stadium [24], and Karaoke [5] are based on out-of-band key exchange and subsequently derived tokens that are valid for one round and one pair of users. As a result, contacts may only exchange messages when both participate in the same round (no offline storage). Furthermore, location anonymity with regard to malicious contacts may slowly degrade: When attackers successfully receive a message from a targeted user, they know that he must have send a packet this round. Consequently, disclosure attacks by a cooperating global observer are possible due to inevitable user churn. In contrast, AnonPoP [6] uses one pseudo-random mailbox identifier for many rounds and different contacts. XRD [7] uses static mailboxes, addressed by the public key of users.

Different countermeasures are employed to defeat or mitigate active attacks like dropping or manipulating packets. Additional cover traffic may be generated by mixes to increase uncertainty of attackers [5, 23, 24]. Misbehaving mixes may be detected by various techniques like verifiable shuffles [7, 24, 25] (strong guarantees, computationally expensive), trap messages [25] or bloom filters [5] (weaker guarantees, more efficient).

cMix [26] splits every communication round into a pre-computation phase and a real-time phase. While the real-time phase only requires symmetric cryptography, the frequency of rounds is limited by the computational expensive pre-computation. Consequently, the average end-to-end latency is dictated by waiting for the next round to start. Furthermore, cMix does not scale horizontally because the protocol only works when using a fixed mix cascade.

Riffle's [27] design for sending messages to mailboxes is able to reduce the number of asymmetric cryptographic operations. After a setup phase, packets are onion-encrypted with a symmetric cipher for multiple rounds. However, receiving messages with location anonymity relies on broadcast or PIR, limiting scalability.

In contrast to previous systems, Loopix [28] does not rely on synchronized rounds. Instead, users send packets according to a Poisson process, injecting dummy packets if no real message is available. Each user may set its own trade-off between latency and overhead by adjusting the rate of the process. However, sending less packets on average also facilitates disclosure attacks [4]. Another drawback of Loopix is that location anonymity requires a trusted service to act as offline storage.

Apart from their extensive usage of asymmetric cryptography, systems discussed so far share another limitation: They conceptionally cannot support connection-oriented applications like voice calls with strong anonymity. Because mixes lack any connection context, they are not able to inject dummy packets that are forwarded to connection endpoints. Consequently, network flow watermarking attacks [8] based on packet drop are possible.

Circuit-based Systems

Both problems can be solved by setting up so-called circuits beforehand. For this, users select a path of mixes that will be used for the entire connection. At circuit setup, they establish a connection context with each mix on the path, including symmetric keys which may be used for onion encryption of application data.

Tor [9] is the most prominent example for using circuits. Unfortunately, it deliberately does not provide strong anonymity: Timing of circuit setup may be correlated in different sections of the network and established circuits do not employ countermeasures against network flow watermarking.

TARANET [10] establishes circuits at network layer. Circuit setup uses batch processing at each hop and network flow watermarking is defeated by padding every circuit to a static rate. However, users only setup circuits when in an active communication, facilitating disclosure attacks. Furthermore, TARANET requires initiators of a connection to know the network address of their communication partners (no location anonymity).

Herd [11] and Aqua [29] use padded circuits to specifically provide anonymous voice calls and file sharing, respectively. Furthermore, users are asked to send packets with a constant rate, defeating disclosure attacks. However, timing of circuit setup is not hidden from malicious mixes.

Yodel [12] provides voice calls with strong anonymity. Similar to mix networks, they use a round-based approach, expecting users to create a circuit every round even when not in an active call. To defeat network flow watermarking, mixes use batch processing which is synchronized by sub-rounds (dictated by the voice codec). If packets are lost or arrive too late on a circuit for a sub-round, mixes inject dummy packets. A drawback of Yodel is that users directly attach to the circuit endpoint of their contacts when in an active call. Consequently, malicious users can select a cooperating mix as endpoint to break location anonymity. Another drawback is the purely random path selection, which does not work well together with batch processing across all circuits: If only a single circuit uses a high latency link, QoS of all voice calls is equally degraded. Furthermore, bandwidth or processing bottlenecks are introduced when not all mixes offer an identical capacity.

In [13], we show how to improve Yodel's path selection by avoiding high latency links and considering mix capacities. We further suggest an alternative technique to defeat network flow watermarking for voice calls: Instead of batch processing all circuits, mixes implement a de-jitter buffer for each circuit individually. The drawback of this approach is that it requires a randomly selected delay for all packets on a circuit at each mix to still securely mix packet flows of all circuits. However, our evaluation shows that this delay can be compensated by preferring low latency links during path selection. In comparison, de-jitter buffers offer a better trade-off between end-to-end latency and anonymity. Unfortunately, suitable path selection probabilities are determined by solving MILPs, which is not practical for hundreds of mixes.

Lessons Learned

The crucial points that can be learned from existing approaches are:

1. Using synchronized communication rounds is promising to implement text-based messaging with strong anonymity (defeating disclosure attacks). However, using asymmetric cryptography for onion encryption of every (dummy) packet is too inefficient.
2. Setting up circuits that can be used for many packets greatly improves efficiency and allows to defeat network flow watermarking attacks. However, circuit setup must be synchronized to avoid timing correlation.
3. To implement anonymous voice calls with acceptable QoS, path selection for circuits must consider mix capacities and pairwise latencies.
4. Users must not directly attach to circuit endpoints of their contacts. Otherwise, location anonymity can easily be broken.
5. When using mailboxes for end-to-end delivery, mailbox identifiers must be valid for more than one round to provide offline-storage.

To the best of our knowledge, Hydra is the first system that considers *all* these aspects at the same time.

System Design

This section first gives an overview of Hydra's functionality. Subsequently, important details are defined in respective subsections.

Overview

Hydra is organized into periodic communication *epochs*, which in turn are divided into a *setup* and a *communication phase*. During epoch setup, all users are synchronized to setup padded circuits. Available mixes for an epoch are published by a *directory service*. After circuit setup, the communication phase is further divided into multiple synchronized *communication rounds*. In each round, every circuit is used to both send ("upstream direction") and receive ("downstream direction") one *circuit cell* to implement text-based messaging. For end-to-end delivery, cells are forwarded between circuit endpoints (exit mixes) by a distributed rendezvous service. Basically, the rendezvous service implements a publish/subscribe protocol based on special *tokens* as shown in Fig. 2. If a subscription is found, the rendezvous service forwards the cell to the subscribed exit mix. Otherwise, the cell is dropped. The exit mix in turn injects the payload into the corresponding downstream circuit. If multiple cells are received for a circuit in the same round, they are queued and injected in subsequent rounds.

In addition to text-based messaging, users can update their circuit at the end of each communication round. A circuit update either enables or disables the possibility to use the circuit for voice calls in parallel to "normal" text-based communication. We denote such "upgraded" circuits as *VoIP circuits*. Nevertheless, such circuits may also be used

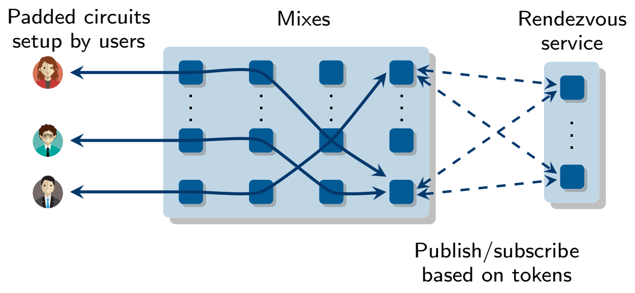
Padded circuits setup by users
Mixes
Rendezvous service
Publish/subscribe based on tokens

Fig. 2 Overview of Hydra's design [14]

to tunnel other applications, like an interactive chat session or file transfers.

For synchronization of epochs and within communication rounds, we assume clock synchronization between all users and mixes with an accuracy of ≈ 10 ms. For example, this may be implemented via NTP or GPS. Note that loss of clock synchronization may increase packet loss, but does not affect security. We further assume that all users setup circuits using the same circuit length l and denote the different positions on a circuit as *layers*. To defeat any attacks based on packet timing, forwarding of circuit setup packets, circuit cells, and circuit update packets between adjacent layers is synchronized by time as shown in Fig. 3. That is, mixes collect all packets/cells from the previous layer, remove (or add in downstream direction) one layer of onion encryption and forward all in shuffled order at a fixed point in time. During normal communication, mixes further use their circuit context to detect packet loss and inject dummy cells accordingly. This also includes exit mixes injecting dummy cells in the downstream direction if a user does not receive a message. When a circuit is used for VoIP in parallel, two (up- and downstream) additional constant packet flows of ≈ 50 pps each are sent—not shown in Fig. 3. However, forwarding of voice packets is not synchronized between different circuits, but instead uses a local de-jitter buffer.

In summary, four different packet types are involved:

1. *Circuit setup packets* establish the circuit context (including shared keys) at mixes. Additionally, they contain a set of tokens a user wants to subscribe to during the epoch. It is the only packet type that requires asymmetric cryptography for onion encryption.
2. *Circuit cells* are mainly used to transport text-based messages.
3. *Circuit update packets* are used to enable/disable the parallel usage of a circuit as VoIP circuit.
4. *VoIP cells* are used to transport application data on VoIP circuits (not restricted to voice data).

We simply write *cell* for brevity when it is clear from the context whether its a circuit or VoIP cell.

Time synchronization and consequently the duration of one epoch is further defined by the following parameters:

1. The intervals Δt_s, Δt_c, and Δt_u control the synchronized forwarding between adjacent layers, for circuit setup packets, circuit cells, and circuit update packets, respectively. Ideally, they match the time that is necessary to receive and process (onion encryption, random permutation) all packets at this layer. Then, latency to traverse a complete circuit is minimized while avoiding packet loss at the same time. Consequently, the parameters are dynamically set by the directory service to react to a varying number of circuits and potentially changing mix resources in different epochs.

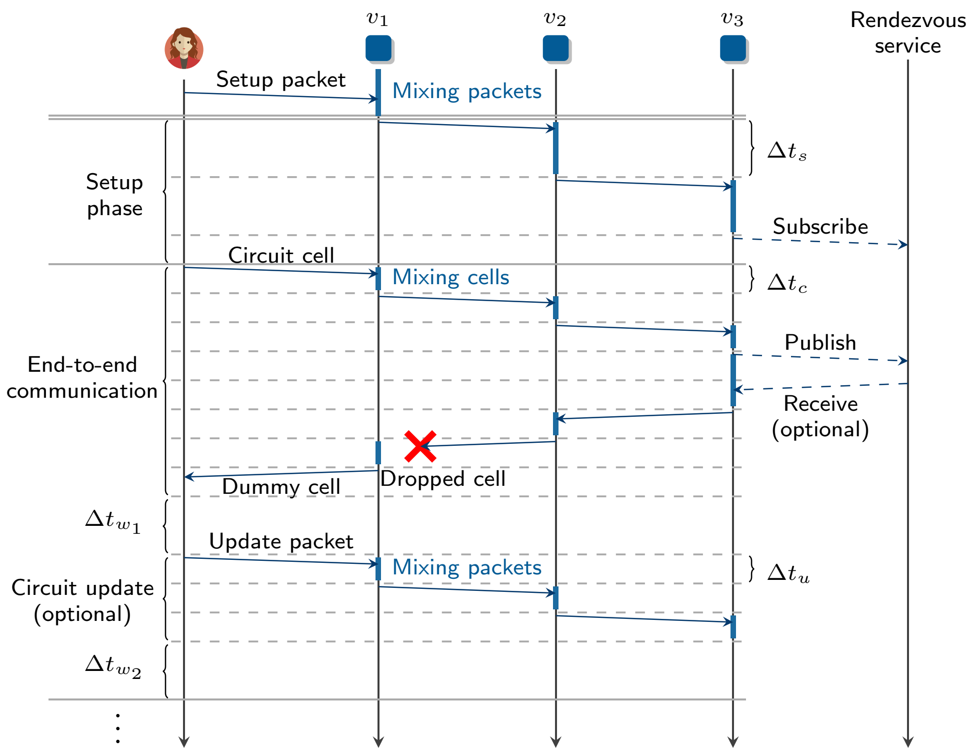

Fig. 3 Synchronization during one epoch, showing circuit setup and the first communication round for a circuit length of $l = 3$ (extended from [14])

2. A waiting time Δt_{w_1} between normal communication and circuit update. Its main purpose is to let users react on incoming calls before potentially updating their circuit.
3. A waiting time Δt_{w_2} between circuit update and the next communication round. In combination, the waiting time $\Delta t_w = \Delta t_{w_1} + \Delta t_{w_2}$ in each round tunes a trade-off between overhead and end-to-end latency: When Δt_w is high, less dummy cells have to be sent and users on mobile devices potentially benefit from improved battery management. However, average end-to-end latency is increased because users have to wait for the next round to send a message.
4. The number k of communication rounds during one epoch. It tunes a trade-off between efficiency and robustness to mix churn: While long epochs reduce the usage of asymmetric cryptography, circuits can not be re-established until the next epoch if mixes fail.

To allow seamless transitions between epochs, the total waiting time $k \times \Delta t_w$ during an epoch is used to process the circuit setup of the next epoch. Consequently, $k \times \Delta t_w$ has to be long enough to establish all circuits in time and therefore also is set dynamically by the directory service.

Directory Service

The directory service is responsible for collecting information about the set V of mixes that are available for upcoming epochs. The registration process for mixes is out-of-scope for this article. For example, Hydra could be a voluntary-based system like Tor.

Based on the mixes and their properties, the directory service also determines suitable probabilities for path selection and suitable values for the parameters of upcoming epochs. The required properties for each mix $v \in V$ are shown in Table 1.

Note that both the capacity of mixes (especially transmission rates) and their pairwise latencies affect path selection. Consequently, they have to be measured in a secure way by the directory service. However, solutions are orthogonal to the remaining design of Hydra and therefore out of scope for this article. FlashFlow [30] (bandwidth measurements) and Treeple [31] (latency measurements) are promising approaches to consider in future work. Both are specifically designed to use a decentralized approach. Consequently, the directory service itself may be implemented in a decentralized way, avoiding a centralized provider to be trusted for distributing unbiased path selection probabilities.

We further assume the transmission rates to reflect full duplex operation. The processing rates also reflect any additional overhead, for example, key lookups and thread synchronization.

Path Selection

We first explain our goals for probabilistic path selection and subsequently define a minimum cost flow problem (MCFP). The optimal flow of this problem can directly be mapped to optimal (with regard to our model) probabilities for path selection. As usual, we denote the first mix on a circuit as *entry mix* and the last mix as *exit mix*. Note that path selection yields circuits of length l. However, it might be beneficial to be able to use a different trade-off between anonymity and latency for VoIP calls, compared to text-based messaging. Therefore, when using a circuit as VoIP circuit in parallel, only its *prefix* of length $l' \le l$ will be used for voice calls, while text-based messaging still uses the full circuit. We denote the last hop on the VoIP circuit (on layer l') as *VoIP exit mix*.

Goals

Our probabilistic path selection should achieve the following properties:

Table 1 Properties of mix $v \in V$

Symbol	Property
ip_v	Network address and port(s)
$\mathrm{pk}_{e,v}$	Ephemeral public key for epoch e
$\beta_{\mathrm{s},v}$	Transmission rate for setup packets in pps
$\beta_{\mathrm{c},v}$	Transmission rate for circuit cells in pps
$\beta_{\mathrm{u},v}$	Transmission rate for circuit update packets in pps
$\mu_{\mathrm{s},v}$	Processing rate for setup packets in pps
$\mu_{\mathrm{c},v}$	Processing rate for circuit cells in pps
$\mu_{\mathrm{u},v}$	Processing rate for circuit update packets in pps
$\rho_{\beta,v}$	Fraction of total bandwidth that is utilized by one VoIP circuit
$\rho_{\mu,v}$	Fraction of total processing power that is utilized by one VoIP circuit
$w_{v,u}$	Transport network latency to all other mixes $u \in V$

1. At each layer, the expected number of circuits n_v a mix $v \in V$ has to handle should be proportional to its capacity. Otherwise, synchronized forwarding between adjacent layers would result in "bottlenecks": Mixes that still have to process packets while other mixes are idle already unnecessarily increase end-to-end latency.
2. Links with a very high latency ($> w_{\max}$) should be avoided because they increase end-to-end latency of text-based messaging.
3. There should be a good "mixing" of circuits. That is, each (honest) mix on a path should maximize the uncertainty about the next hop of a targeted circuit.
4. Mixes should not be used on adjacent layers for the same circuit.
5. Given all other constraints, the average transport network latency of a selected path should be minimized, especially to provide acceptable QoS for voice calls.

Unfortunately, latency and anonymity are in conflict here. To maximize uncertainty for attackers, path selection would have to be uniform at random, excluding any possibilities to optimize latency. Consequently, we aim for a reasonable trade-off. For this, we focus on location anonymity because it is only protected by a single circuit (compared to relationship anonymity, which is protected by two circuits). When attackers try to break location anonymity of a user, they try to track its circuit in downstream direction. Therefore, when looking at a mix v at layer i, we enforce that there should be at least $d_{\min}$ possible previous hops in layer $i-1$. Furthermore, the posteriori probability distribution for selecting one out of the $\geq d_{\min}$ previous hops should be close to uniform. Consequently, a trade-off between anonymity and latency may be tuned by adjusting the parameter $d_{\min}$.

To further specify our goal regarding mix capacities, recapitulate that a mix actually has a total of eight values that describe its capacity (transmission and processing rate for four different packet types each). However, all transmission rates are proportional to each other. For example, if a mix can transmit twice as much circuit cells per second compared to another mix, the same proportion applies for other packet types. Similar, we assume the same to be true for processing rates (although there might be small differences in practice due to varying hardware support for symmetric/asymmetric cryptography on mixes). Consequently, we focus on the transmission and processing of circuit cells because this impacts achievable end-to-end latency of text-based messaging the most (via Δt_c). That is, the effective capacity a_v of mix v is the minimum of its transmission and processing rate for circuit cells:

$$a_v = \min\left\{\beta_{c,v}, \mu_{c,v}\right\} \tag{1}$$

We further denote its relative capacity (compared to the total system capacity) as b_v:

$$b_v = \frac{a_v}{\sum_{u \in V} a_u}. \tag{2}$$

Minimum Cost Flow Problem

To achieve our goals, the probability distribution for selecting the entry mix $v \in V$ of a circuit can be defined straight forward to match its relative capacity (random variable X_1):

$$\Pr\left(X_1 = v\right) = b_v. \tag{3}$$

To determine optimal probabilities for the second hop, we model the situation as a MCFP in a bipartite graph $G = (L, R, E)$ as depicted in Fig. 4. There is one node for each mix in L and R:

$$L = \left\{l_v \mid v \in V\right\} \tag{4}$$

$$R = \left\{r_v \mid v \in V\right\}. \tag{5}$$

Furthermore, there is an edge $(l_u, r_v) \in E$ for all pairs of mixes u, v with $u \neq v$. The cost (weight) of each edge matches the latency of the corresponding link:

$$w(l_u, r_v) = w_{u,v} \tag{6}$$

Next, the flow constraint is as follows: Each node $l_v \in L$ is a source node with a supply of b_v, its probability to be selected as entry mix. On the other side, each node $r_v \in R$ is a sink with a demand of b_v. Then, we can map a feasible flow f to path selection probabilities for the second hop (random variable X_2), given a fixed entry mix u:

$$\Pr\left(X_2 = v \mid X_1 = u\right) = \frac{f(l_u, r_v)}{b_u}. \tag{7}$$

In combination with the demand of b_v for r_v, the absolute probability to select a mix v at layer 2 again matches its relative capacity b_v:

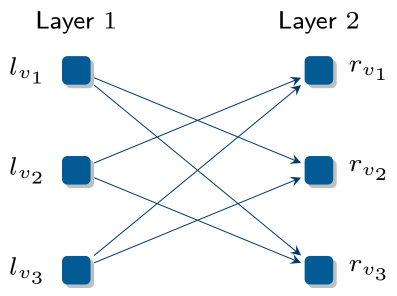

Fig. 4 Example for the MCFP for determining optimal path selection probabilities, mix set $V = \left\{v_1, v_2, v_3\right\}$

$$\Pr\left(X_2 = v\right) = \sum_{u\in U} b_u \times \frac{f(l_u, r_v)}{b_u} = \sum_{u\in U} f(l_u, r_v) = b_v. \tag{8}$$

The capacity of edges in the flow network can be used to model our remaining constraints:

$$c(l_u, r_v) = \begin{cases} 0 & \text{, if } w_{u,v} > w_{\max} \\ \frac{1}{d_{\min}} \times b_v & \text{, else.} \end{cases} \tag{9}$$

That is, edges may not be used if the corresponding link latency is too high. Furthermore, the minimum in-degree $d_{\min}$ for each node in R is enforced by limiting the flow on each adjacent edge. In many cases, this is expected to also result in a close to uniform distribution of the flow entering $r_v \in R$ because using more than $d_{\min}$ edges would increase the cost of the flow (but it might be necessary to some extend due to capacity constraints).

In result, we can determine optimal path selection probabilities by solving the MCFP and using the optimal flow $\hat{f}$ in Equation (7). For example, the problem can be solved via linear programming. To also determine optimal probabilities for subsequent hops, we could model the situation between other adjacent layers using the same idea. Luckily, all layers yield the exact same MCFP, so that we can reuse the same optimal flow $\hat{f}$. That is, given the previous hop u_{i-1}, the next hop is selected according to the following probability distribution (random variable X_i) for $2 \le i \le l$:

$$\Pr\left(X_i = v \mid X_{i-1} = u_{i-1}\right) = \frac{\hat{f}(l_{u_{i-1}}, r_v)}{b_{u_{i-1}}} \tag{10}$$

Note that the MCFP might not have a feasible solution in two cases:

1. There exists a mix v with a very high capacity compared to all other mixes:

$$b_v > \frac{1}{d_{\min}} \times \sum_{u\in V\setminus\{v\}} b_u \tag{11}$$

   Then, we artificially decrease the capacity of v. While this also reduces the total capacity of the system, it increases anonymity if v is malicious.
2. The constraint $w_{\max}$ for the maximum link latency is too low. In this case, we solve the MCFP with increased values for $w_{\max}$ to find a feasible upper bound $w'_{\max}$.

Setting Parameters

The directory service has to dynamically set the parameters Δt_c, Δt_u, Δt_s and k for upcoming epochs in an automated way. For this, it first has to estimate the expected number n_v of circuits a mix $v \in V$ has to handle *per layer*. Similar, the *total* number n'_v of VoIP circuits that v has to handle has to be estimated. That is because VoIP circuits permanently reduce the effective transmission rate and processing power of mixes, regardless of their position on the VoIP circuit. The estimation can be done as follows:

1. We expect all users of Hydra to establish a circuit in as many epochs as possible. Consequently, the total number of circuits should only gradually change over time. For example, this allows to use a moving average of the number of circuits in recent epochs to estimate the load in upcoming epochs.
2. Given our probabilistic path selection, n_v can be modelled as a random variable that follows a binomial distribution (a user either selects mix v at layer i or not and all n users select paths independently). Consequently, the 99.9% percentile of the resulting binomial distribution can be used as an practical upper bound for n_v.
3. Again using a moving average, the fraction f of *all* circuits that are used for voice calls can be estimated, yielding $n'_v = f \times l' \times n_v$. Note the additional factor of l' because each VoIP circuit is handled by l' mixes.

Setting Δt_c

To recapitulate, Δt_c is the interval between the forwarding of circuit cells on two adjacent layers. It must be large enough to allow all mixes on a layer to receive and process all cells from the previous layer. On the other side, it should not significantly exceed this bound to minimize end-to-end latency.

For layer $i + 1$, we can now estimate the time it takes to receive and process all cells from layer i using a simplified model of the transmission process as shown in Fig. 5 (the same model applies to the downstream direction from layer $i + 1$ to i).

First, we have to compensate for the potentially inaccurate clock synchronization between mixes. That is because a mix in layer i could start sending its circuit cells too late if its clock is behind in time. This time is bounded by the

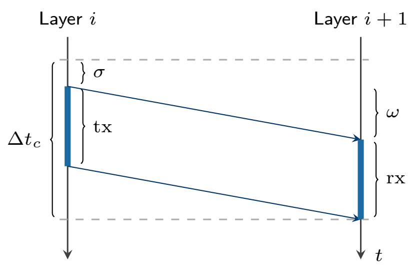

Fig. 5 Simplified model for the transmission of circuit cells from layer i to layer $i + 1$

(assumed) maximum clock offset σ between any pair of mixes.

Next, all mixes transmit their circuit cells in parallel and the total transmission time can be determined:

$$\mathrm{tx} = \max_{v \in V} \left\{ \frac{n_v}{\beta_{\mathrm{c},v} \times (1 - n'_v \times \rho_{\beta,v})} \right\} \tag{12}$$

Next, we assume that all circuit cells have the same propagation delay in the transport network. This propagation delay is bounded by the maximum latency ω across all links that are selectable for a circuit. Consequently, mixes in layer $i+1$ start to receive and process the circuit cells after time $\sigma + \omega$. We assume that receiving new cells and processing of already received cells can be done in parallel, so that the total time at mix v is either dictated by v's bandwidth or processing power:

$$\mathrm{rx}_v = \max \left\{ \frac{n_v}{\beta_{\mathrm{c},v} \times (1 - n'_v \times \rho_{\beta,v})}, \frac{n_v}{\mu_{\mathrm{c},v} \times (1 - n'_v \times \rho_{\mu,v})} \right\}. \tag{13}$$

Because all mixes work in parallel again, the total time for receiving and processing at layer $i+1$ is

$$\mathrm{rx} = \max_{v \in V} \left\{ \mathrm{rx}_v \right\}. \tag{14}$$

Note that this assumes that there is no (significant) idle time at mixes during the receiving/processing phase. Especially, mixes should send circuit cells in a "round robin" fashion to next hops to avoid idle times at the beginning of this phase. Further, note that $\mathrm{tx} \leq \mathrm{rx}$.

In summary, we get the following lower bound for setting Δt_c:

$$\Delta t_c \geq \sigma + \omega + \mathrm{rx}. \tag{15}$$

***Setting* Δt_u**

Determining a suitable value for Δt_u is analogous to Δt_c. However, not all users update their circuit every round. Consequently, the estimation for the number of affected circuits can be lower, for example, based on the moving average of circuit updates per round in recent epochs.

***Setting* Δt_s *and* k**

Setting Δt_s also is analogous to Δt_c. For setting k, recapitulate that the total waiting time Δt_w during epoch e should be long enough to run the setup phase of epoch $e+1$ (dictated by the circuit length l and Δt_s):

$$k \times \Delta t_w \geq (l-1) \times \Delta t_s \tag{16}$$

$$\Leftrightarrow \quad k \geq \left\lceil \frac{(l-1) \times \Delta t_s}{\Delta t_w} \right\rceil. \tag{17}$$

Rendezvous Service

The mixes in V also implement the distributed rendezvous service. We call them rendezvous nodes when acting in this role. The rendezvous service is a publish/subscribe protocol based on *tokens*. That is, an exit mix can subscribe to a token on behalf of a user and subsequently receive all circuit cells that are addressed to this token. Users may use both their setup packet and subsequent circuit cells to subscribe to tokens. As implementation of tokens, we use unsigned 64 bit numbers.

For a distributed implementation, the responsible rendezvous node for each possible token t must be determined deterministically. We suggest that in each epoch e the directory service sorts the mixes $v \in V$ by their ephemeral public key $\mathrm{pk}_{e,v}$. Since we use unsigned numbers as tokens, the *index* of the responsible rendezvous mix in the sorted list may be determined by $t \bmod |V|$. When a circuit cell is published, the responsible rendezvous mix checks if there are any subscriptions to the token contained in the cell and forwards it to *all* subscribed exit mixes. However, the cell will not be injected back on the circuit that it originated from.

Applications

Using circuits for all communications, combined with the distributed rendezvous service, Hydra is able to support a wide variety of applications with strong anonymity. Similar to popular messengers (which do not provide anonymity), Hydra supports user registration, contact discovery, text-based messaging, and voice calls. The following subsections further define the implementation of these applications.

User Registration

We denote the set of users as U. Each user $u \in U$ has a long-term pseudonym nym_u and a long-term key pair $(\mathrm{pk}_u, \mathrm{sk}_u)$. For example, users may reuse PGP keys if available or generate a new key pair upon joining Hydra.

If desired by a user u, he may then upload the mapping $(\mathrm{nym}_u, \mathrm{pk}_u)$ to a public contact service. To protect his location anonymity while uploading, u may use his circuit cells, addressed to a reserved token of the contact service. However, he should not do so in the first epoch he participates. Otherwise, new uploads to the contact service can be correlated to users setting up a circuit for the first time, breaking location anonymity. Furthermore, he has to generate (or request) and upload a cryptographic binding bind_u of nym_u to pk_u to defeat impersonation. We enforce no specific implementation, but assume that possible contacts of u can verify the binding. For example, a fingerprint of pk_u could be used and verified by meeting in person.

Registration at the contact service is optional. Users may also decide to publish their public key out-of-band (like on their personal/professional website) or only by meeting in person.

Contact Discovery

The objective of contact discovery is to establish a shared secret between an initiator $a \in U$ and a responder $b \in U$. Similar to user registration, contact discovery is obsolete if a and b meet in person. We start with some preliminary thoughts for the responder of contact requests before defining the actual protocol.

Responder View

To be able to receive contact requests, each user $b \in U$ has a *contact token* ct_b that is derived from pk_b. Furthermore, b has to subscribe to ct_b on a regular basis. However, care has to be taken to protect his location anonymity while doing so. A straight forward approach would be to (effectively) have a one-to-one mapping from pk_b to ct_b, for example, using a 64 bit hash value, and subscribe to ct_b every epoch. Unfortunately, attackers could then potentially disclose the mapping of user b to his contact token (and consequently to his public key and pseudonym) by blocking all his network packets (and only his). If they further control the responsible rendezvous node for ct_b, they observe the missing subscription. Note that using a cryptographic hash function for deriving ct_b does not help to keep pk_b secret because attackers could simply hash all public keys. We defeat the attack by two mechanisms:

1. We artificially introduce collisions for contact tokens using a hash function with $x \ll 64$ bit of output (padding with zeros to be used as token).
2. Users randomly decide whether they subscribe to their contact token in an epoch, for example, with a probability of 50%.

The parameter x should be small enough that with high probability, at least one other user will also subscribe to b's token even if b is blocked. Combined with random subscriptions, attackers cannot reliable infer which contact token is missing by counting subscriptions to different tokens.

However, there are two more aspects to consider for this approach:

1. Unrelated users will also receive the contact requests for b. Therefore, contact requests have to be encrypted using pk_b in such a way that only b will detect it as a valid request. More problematic is the fact that users cannot receive another circuit cell in the same round. Consequently, x should not be too small either. For example, it could dynamically be set (based on the expected number of users for upcoming epochs) so that ≈ 100 users share the same contact token on average.
2. Because b does not subscribe to ct_b every epoch, contact requests are also forwarded to the public contact service for later delivery.

Initiator View

To be able to contact a user b via Hydra for the first time in epoch e_a, the initiator a has to know the public key pk_b. For example, a could download it from the public contact service if the pseudonym nym_b is known. To protect relationship anonymity, a must use his circuit for this.

Next, a generates a secret s and encrypts it with pk_b in a way that b can validate it, for example by adding a Message Authentication Code (MAC) generated with s. Furthermore, he uses s as seed for a hash chain that is advanced every epoch, starting at epoch e_a as depicted in Fig. 6. The hash chain is used to derive session keys and *rendezvous tokens* for further communication with b. Each rendezvous token $\mathrm{rt}_{e,a,b}$ is only valid for one epoch e and for one pair of users a, b. Then, a sends the encrypted secret s and the epoch e_a to b using a circuit cell addressed to ct_b. Furthermore, he starts to subscribe to the generated rendezvous tokens.

As soon as b receives the request, he can initialize the same hash chain and subscribe to the same rendezvous tokens. Then, a and b could start normal communication by sending circuit cells addressed to $\mathrm{rt}_{e,a,b}$ and using the session keys for end-to-end protection. However, all session keys are derived from s, which does not provide any *forward secrecy* in case sk_b is compromised. Consequently, we suggest that a and b first run a key exchange protocol that provides a new shared key s' with forward secrecy. Then, they can incorporate s' into the hash chain (or setup a new one). And only then, a should send his own pseudonym to b to also protect relationship anonymity with forward secrecy.

Text-Based Messaging

Text-based messaging between contacts a and b is straight forward: Given the shared hash chain, they derive and

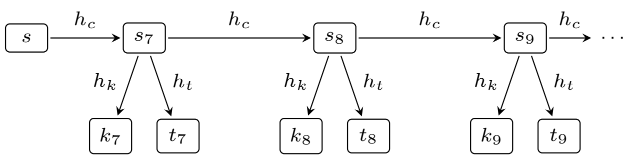
$$\begin{tikzcd}
\boxed{s} \arrow[r, "h_c"] & \boxed{s_7} \arrow[r, "h_c"] \arrow[d, "h_k"'] \arrow[dr, "h_t"] & \boxed{s_8} \arrow[r, "h_c"] \arrow[d, "h_k"'] \arrow[dr, "h_t"] & \boxed{s_9} \arrow[r, "h_c"] \arrow[d, "h_k"'] \arrow[dr, "h_t"] & \cdots \\
& \boxed{k_7} \quad \boxed{t_7} & \boxed{k_8} \quad \boxed{t_8} & \boxed{k_9} \quad \boxed{t_9} &
\end{tikzcd}$$

Fig. 6 Hash chain shared by users a and b, starting at epoch $e_a = 7$. Three different hash functions/key derivation functions are involved to advance the chain (h_c), to derive session keys (h_k), and to derive rendezvous tokens $t_e = \mathrm{rt}_{e,a,b}$ (h_t)

subscribe to the same rendezvous tokens $\mathrm{rt}_{e,a,b}$. Consequently, circuit cells addressed to $\mathrm{rt}_{e,a,b}$ can be used to tunnel messages. Furthermore, cells are encrypted and authenticated end-to-end using session keys derived from the hash chain. When users are temporarily offline, their entry mix acts as offline storage.

While the baseline protocol is straight forward, there are two aspects that may need special treatment if required by a user:

1. While end-to-end latency is minimized when a circuit cell sent by a is delivered to b in the exact same round, it might slowly degrade a's location anonymity. When b is malicious, he could cooperate with a global observer to track which users are online whenever he receives a cell from a (anonymity set). Because of inevitable user churn, intersection of anonymity sets weakens location anonymity. To counter the attack, we allow users to instruct an arbitrary mix on their circuit to delay the next cell they send by a random number of rounds (and insert a dummy cell instead).
2. Because all messages a user sends/receives are multiplexed over one circuit, users might experience increased end-to-end latencies when they are in many conversations. If users do not mind leaking that they potentially have many contacts, they can setup multiple circuits per epoch to increase throughput.

Voice Calls

When a user a wants to call his contact b in epoch e, signaling can use the same protocol as text-based messaging:

1. The invitation to the call is sent using a circuit cell addressed to $\mathrm{rt}_{e,a,b}$. It includes the network address ip_{v_a} of a's VoIP exit mix v_a and a fresh, random rendezvous token t.
2. User a updates its circuit to be used for VoIP in parallel. The update packet includes t.
3. If b accepts the call, he also updates his circuit and sends both t and ip_{v_a} to his VoIP exit mix v_b.
4. Mix v_b sends t to v_a to permanently connect the two circuits. In result, voice packets do not have to use the rendezvous service for end-to-end delivery.
5. During the call, users apply authenticated end-to-end encryption using the session key from their shared hash chain.
6. After the call, both users may disable their VoIP circuit again or potentially reuse it to connect to a different call (again by sending an update packet).

If b does decline the call (using text-based messaging) or does not react in time, a aborts the protocol and updates its circuit accordingly.

Note that using Hydra's VoIP feature does not provide the same strong anonymity as text-based messaging. First, VoIP circuits potentially have a shorter length of l'. This increases the chance of attackers to narrow down possible endpoints of VoIP circuits. Especially, the probability for a majority of mixes on the circuit (or even all) to be malicious increases. Second, churn for VoIP circuits is higher compared to "normal" circuits. In result, disclosure attacks on relationship and location anonymity are facilitated.

To counter the latter and to make Hydra more user friendly at the same time, other applications may be tunneled using VoIP circuits as well. For example, contacts may setup a call to be used as an interactive chat session with very low end-to-end latency or to transmit small files like photos. For the latter, an untrusted offline storage could be used as endpoint for the "call" in case the recipient is temporarily offline. Further countermeasures, for example fake calls, are out of scope for this article and have to be studied in future work.

Circuit Design

This section presents details about our implementation of circuits, especially regarding the structure of packets and their onion encryption. Moreover, we show how dummy circuits created by mixes may further strengthen anonymity.

The objective of circuits is to unlink users from packets sent during one epoch. The main threats during an epoch are malicious mixes and external attackers that control links of circuits. We already discussed that network flow watermarking attacks are defeated by synchronized forwarding between layers and injecting dummy packets if necessary. Additionally, mixes implement a de-jitter buffer for each VoIP circuit. Nevertheless, circuit design has to consider two more possible threats:

1. When attackers replay packets, they must not be forwarded using the same cryptographic transformation as the original packet. Otherwise, they can be tracked by their recurrent ciphertext.
2. Attackers must not be able to manipulate (onion-encrypted) packet content in a way that is detectable even after the packet has passed an honest mix, often called *tagging attack*.

Circuit Setup

The objective of circuit setup is to establish a shared secret s_i with each mix $v_i, 1 \leq i \leq l$ on a selected path. Furthermore, the exit mix subscribes to a set of tokens

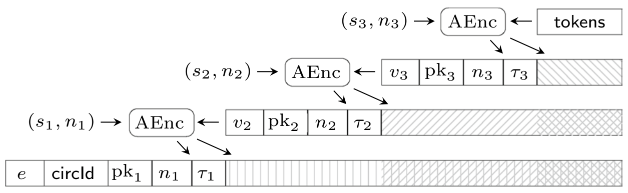

Fig. 7 Authenticated onion encryption of a circuit setup packet on path (v_1, v_2, v_3) [14]

at the rendezvous service on behalf of the user. Because the directory service publishes the ephemeral public keys pk_{v_i} of each mix, users are able to pre-compute all secrets s_i. For this, a user generates own ephemeral key pairs $(\mathrm{pk}_i, \mathrm{sk}_i)$, $1 \le i \le l$. In result, he is able to prepare a single onion encrypted setup packet to establish the circuit and subscribe to tokens as depicted in Fig. 7.

For onion encryption, we require the use of an authenticated encryption scheme. Consequently, users also add an authentication τ_i (128 bit) after each layer of encryption. We further recommend using a symmetric cipher based on s_i, which usually requires an additional initialization vector (IV) or nonce value n_i (96 bit) that is also added to the layer. Moreover, a user adds his own public key pk_i and the address of the next hop v_i to each layer. When aiming for IPv6 support, network address and port require a total of 144 bit and IPv4 addresses must be mapped to IPv6 to achieve uniform packet sizes. Finally, the current epoch number e and a random circuit id is added to the setup packet before it is sent to the entry mix v_1. To keep setup packet size uniform across users, we suggest to use 256 tokens for now, padding with dummy tokens if necessary. If users have more contacts, they may also use circuit cells to subscribe to more tokens. Furthermore, token order is randomized to avoid rendezvous tokens of the same contact to be correlated across epochs. Using 4 B epoch numbers, 8 B circuit ids, and public keys of size x, the total size of the setup packet sent by a user is

$$2042\,\mathrm{B} + l \times (46\,\mathrm{B} + x). \tag{18}$$

When mix v_i receives the packet, he first uses his secret key sk_{v_i} and the public key pk_i contained in the packet to derive s_i. Together with n_i, he is then able to remove one layer of encryption, check the authentication tag τ_i and extract the next hop v_{i+1} or the set of tokens in case of the exit mix. Before forwarding, the circuit id is replaced by a new random id and the mix stores the mapping of the two circuits ids to s_i, previous hop, and next hop. Furthermore, v_i removes v_{i+1}, pk_i, n_i, and τ_i from the packet. Padding is not necessary because the size of all setup packets at this layer decreases uniformly and a mix knows its layer i on the circuit because of the timing anyway. If authentication fails, the mix drops the packet and creates a dummy circuit instead, starting at layer i. While the circuit is not utilizable by the user in this case, it still gets padded with dummy circuit cells up to layer i (and back). Similar, the dummy circuit created by the mix is padded to the exit mix (and back).

Authenticated encryption naturally defeats tagging attacks. Replay protection can be implemented efficiently using a bloom filter on the authentication tags. If attackers try to "hide" a replay by manipulating the tag, the packet still is dropped because authentication fails. Furthermore, the bloom filter may be cleared every epoch because replays from older epochs also lead to failed authentication (mixes use a fresh key pair every epoch).

Circuit Cells

The packet format of circuit cells is shown in Fig. 8. Most importantly, they include an onion-encrypted payload and token, which are used for end-to-end delivery of text messages. The payload size of 240 B should be a reasonable trade-off between overhead of dummy cells and usability for (compressed) text messages. Of course, larger messages can be split to multiple cells or transferred using a VoIP circuit.

Additionally, cells may indicate one of the following commands, interpreted by the mixes (and rendezvous node) in upstream direction:

1. Delay the *next* circuit cell in upstream direction by args rounds. This may increase location anonymity of users as discussed previously.
2. The indication that the circuit cell contains a contact request. Then, the responsible rendezvous node forwards the circuit cell to the contact service for later delivery if no subscription is found.
3. Subscribe to args more tokens stored in the payload.

For onion encryption, we use a symmetric cipher *without authentication*. Consequently, mixes are able to inject randomized dummy cells that are indistinguishable from real cells. However, care has to be taken to not allow tagging attacks in upstream direction: For example, when using a block cipher in counter mode of operation, the cell is xored with a key stream. Consequently, attackers would be able to flip specific bits in the plaintext of cmd, args, token without detection. Then, malicious exit mixes could check if they receive a valid cmd or known contact token with the same bits flipped. We counter

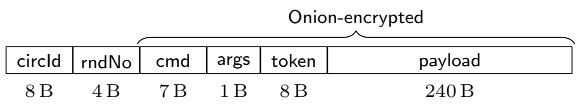

Fig. 8 Packet format of circuit cells [14]

this using a *wide block cipher* with a block size of 256 B. In result, onion encryption works on a single block and the complete cell changes in an unpredictable way at honest mixes if only a single bit is flipped.

For replay protection, mixes use the communication round number rndNo and simply drop cells with an old one (injecting a dummy cell instead). But again, care has to be taken because cells are not authenticated: Attackers could simply manipulate the round number to appear new. To counter this, we also require the used block cipher to be *tweakable*. Similar to an IV/nonce, a tweak is an additional, non secret input for the cipher. By using rndNo as tweak, the complete cell changes in an unpredictable way at honest mixes if the rndNo is manipulated.

VoIP Cells

VoIP cells, shown in Fig. 9, are a stripped version of circuit cells. All VoIP cells have a uniform size that depends on the used VoIP codec. A payload size in the range of 32 B to 128 B seems like a reasonable trade-off between codec quality, usability for file transfers and overhead. Because we expect the payload to be end-to-end encrypted (or randomized for dummy cells) at all times, tagging attacks are ruled out from the start and we can use an arbitrary cipher that supports an IV (nonce/tweak). Similar to circuit cells, replay protection uses a sequence number seqNo which is used as the IV for the cipher. In contrast to round numbers in circuit cells, sequence numbers are initially selected at random by each mix when establishing a VoIP circuit. Afterwards, consecutive cells between two mixes also use consecutive sequence numbers. Note that users have to know which "offset" each mix on the circuit uses for transforming the sequence number (one offset per direction). Only then users can derive the correct IVs for onion encryption. For this, mixes use voice cells in downstream direction to notify the user as further described in the next subsection.

Circuit Update

Update packets on a circuit prefix with mixes v_i, $1 \leq i \leq l'$ use authenticated encryption similar to setup packets, shown in Fig. 10. However, update packets do not include data for a key exchange because mixes already know the shared secret s_i. Furthermore, the round number is used as IV/nonce to allow an easy replay protection similar to circuit cells (here, authentication fails upon manipulation). One command per layer is used to instruct mixes to either disable or enable the usage as VoIP circuit. Additionally, the circuit update packet can be used to optionally send a rendezvous token t and the VoIP exit mix r of another user to the VoIP exit mix of the affected circuit. This reflects our protocol for call signalling. For now, we use 2 B to encode the command, resulting in a total update packet size of $38\,\text{B} + l' \times 18\,\text{B}$.

When a mix v_i at layer i is instructed to enable VoIP support (and it was not enabled before), he starts sending a constant flow of VoIP cells to both the previous and next hop on the circuit. The packet rate matches the rate of the VoIP codec Hydra is configured to use, for example 50 pps. For each direction, the mix waits a random time drawn uniformly at random from the interval $[0;\Delta t_v]$ before sending the first packet, with Δt_v being the interval at which the voice codec generates packets. He further selects the 4 B sequence number for the two first cells uniformly at random. Subsequently, a deterministic sending interval of Δt_v is enforced, increasing the sequence number by one for each new packet. By the time he starts sending both packet flows, he is also expected to receive the first voice cells on the circuit from the previous hop. Then, he starts injecting the cells in his upstream packet flow, including dummy cells if the cell with the next expected sequence number does not arrive in time. Note that for injecting cells to the deterministic sending schedule, a random scheduling delay of $\Delta t_v/2$ is introduced on average. Using this algorithm, the mix effectively implements a de-jitter buffer with a random offset in timing and sequence number, unlinking ingress and egress flow. He further determines the offset in sequence numbers he uses when injecting upstream cells. Later in time, he will also start receiving cells from the next hop (or from another VoIP exit mix) in downstream direction and also determines the offset in sequence numbers for injecting downstream cells. Then, he may send both offsets to the user via a downstream cell, padded with zeros for detectability at the user. Because all mixes on previous layers $j < i$ already did send their offsets by this time, the user can derive the correct (egress) sequence number that mix v_i used, and successfully remove all layers of onion encryption.

When a mix is instructed to disable VoIP support, it simply stops sending both packet flows.

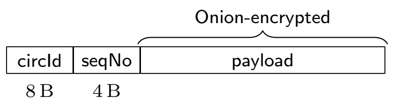

Fig. 9 Packet format of VoIP cells

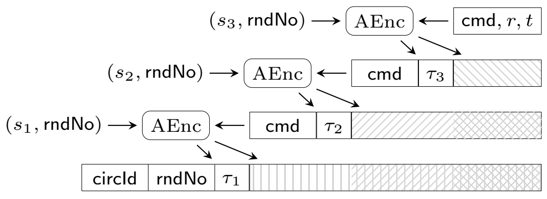

Fig. 10 Authenticated onion encryption of a circuit update packet on path (v_1, v_2, v_3)

Dummy Circuits

Mixes create a dummy circuit whenever authentication of a real setup packet fails. For this, they use the same path selection as users, with a reduced path length depending on their current layer. Additionally, mixes also proactively create dummy circuits to increase uncertainty for attackers when not many real circuits are established (for example, when bootstrapping Hydra). To maximize uncertainty, mixes create one additional dummy circuit for each possible link (according to path selection) that is not used by any other circuit. To further defeat flooding attacks (all but a few circuits are set up by attackers), mixes create at least one dummy circuit at each layer. Mixes also use their dummy circuits to send (fake) messages and contact requests to themselves to add cover traffic to/from the rendezvous service.

Evaluation

To evaluate Hydra, we first qualitatively discuss to what extend our objectives (section 2) are fulfilled. Subsequently, we quantitatively evaluate the achievable end-to-end latency and anonymity when using Hydra.

Qualitative Discussion

Note that our qualitative discussion concisely summarizes many arguments that already guided our design.

Functional Objectives

Hydra implements user registration, contact discovery and text-based messaging by tunneling all packets via circuits. End-to-end delivery is realized by the distributed rendezvous service via a publish/subscribe protocol. Offline storage is provided by the contact service (for contact discovery) and by entry mixes (text-based messages). Furthermore, voice calls are supported by "upgrading" circuits accordingly and by connecting the two circuit endpoints directly for end-to-end delivery. While we define no protocol for arbitrary message types, for example files and photos, larger messages may also be tunneled via VoIP circuits. A limitation of this approach is the limited bandwidth because VoIP circuits are optimized for efficient VoIP codecs. An idea for future work could be to introduce further circuit types similar to a VoIP circuit, offering higher padding rates or larger cells. Unfortunately, anonymity sets would be smaller for such circuits.

Non-functional Objectives

Confidentiality and Integrity

After contact discovery (or out-of-band key exchange), contacts initialized the same hash chain and can subsequently derive session keys for each epoch for authenticated end-to-end encryption of all message types and voice calls. Note that when the contact service is used to query public keys, secure handling of cryptographic bindings to pseudonyms is crucial to defeat impersonation.

Anonymity

The anonymity discussion is roughly sorted by the effort for potential attacks, from least to most.

Regarding local attackers, honest mixes perfectly unlink ingress packets from their corresponding egress packets: Packet content can not be correlated because of onion encryption of all packet types. Furthermore, circuit ids of all packets and sequence numbers of VoIP cells randomly change at every honest mix. Packet size cannot be correlated because it is uniform across all circuits at any given layer for each packet type. Timing information is removed by batch processing and synchronized forwarding in the case of setup packets, circuit cells, and circuit update packets. Timing information of VoIP cells is removed by implementing a de-jitter buffer for each VoIP circuit. Furthermore, the sending times of the first VoIP cell in each direction as well the "size" (in time) of each de-jitter buffer is randomized. Consequently, VoIP circuits that are established in the same round are perfectly mixed, even though each circuit uses its own de-jitter buffer. Last but not least, all packet types implement replay protection and protection from tagging attacks.

When circuit cells or VoIP cells are dropped by attackers, the resulting "gap" is padded by the next honest hop. In contrast, dropping circuit update packets results in a possible attack: If update packets on all but one circuit are dropped, the remaining packet can be tracked to disclose the path of the circuit because mixes do not send dummy update packets in our current design. However, there still is some protection for text-based messaging when VoIP circuits use a shorter path length of $l' < l$. Furthermore, it requires a *global active attack*. And while this is not ruled out by our threat model, we expect this to be very expensive for attackers (and most likely detectable).

When only few circuits are established, attackers drop setup packets or control a majority of circuits, "honest" circuits are still not distinguishable from the dummy circuit each honest mix creates. Even when only two honest users participate in Hydra (or all others are blocked), attackers can not tell whether they are communicating or not. That is because dummy circuits of mixes also create valid traffic from/to the rendezvous service. However, in this artificial case, the relationship of the two users is de-anonymized if they successfully setup a voice call.

Malicious mixes naturally do not unlink ingress and egress packets, because they can always share their internal mapping with external attackers. Therefore, we do not implement any mechanisms to detect misbehaving mixes (for example verifiable shuffles or trap messages). And while misbehaving mixes may simply disrupt service, similar disruption by external attackers can not be defeated anyway. If a majority of mixes on a circuit are malicious, they may be able to narrow down possible circuit endpoints. For example, if only the entry mix is honest, the anonymity set only contains users that use this entry mix. In the worst case, all mixes on a circuit are malicious and location anonymity is broken. However, relationship anonymity is still protected by the circuits of communication partners.

Disclosure attacks during text-based communication are expected to not have a high chance of success because user churn is expected to be low. Disclosure attacks on VoIP circuits are more promising and have to be studied in future (together with countermeasures like fake calls).

While outside the scope of this article, the directory service has to be implemented in a secure and decentralized way to provide unbiased path selection probabilities.

QoS

End-to-end latency for text-based messages is expected to be low because during one communication round, only symmetric ciphers are used. However, the artificial waiting time between rounds increases achievable latency. Unfortunately, waiting times are inevitably for energy efficiency on mobile devices and to run the setup phase of the next epoch. Suitable trade-offs have to be studied by our quantitative evaluation.

In contrast, VoIP calls (which may also be used for normal messaging) are expected to result in an end-to-end latency below 400 ms. While VoIP cells suffer an average scheduling delay of $\Delta t_v/2$ at each hop, for example, 10 ms for common codecs, path selection uses low latency links on average. Again, a quantitative evaluation will yield further insights. Packet loss is also expected to be low because our path selection utilizes mixes proportionally to their capacity, avoiding bandwidth and processing bottlenecks.

Efficiency and Scalability

Apart from circuit setup, which is only necessary once per epoch, onion encryption may use efficient symmetric ciphers. Furthermore, horizontal scalability is achieved using constant path lengths l and l' for circuits.

Robustness

While using long epochs increases efficiency (less asymmetric cryptography), failing mixes disrupt the communication on all circuits they are part of. Then, affected users cannot take part in any communication for up to two epochs. That is because their circuit for the next epoch might also use a failed mix, and the path cannot be changed anymore because the setup phase is already in progress. Consequently, the number of rounds during one epoch should not be longer than necessary for the next setup phase to finish.

Quantitative Evaluation

We aim to answer the following research questions with our quantitative evaluation:

1. How does Hydra's performance for text-based messaging compare to other systems that provide strong anonymity? Especially, we want to compare Hydra to Karaoke [5], one of the most efficient candidates today.
2. How long does one epoch have to be to allow the setup of the next epoch to finish in time?
3. Does Hydra achieve an acceptable end-to-end latency for voice calls, that is, below 400 ms?
4. Is anonymity degraded by latency-based path selection?
5. Is it efficient to solve the MCFP for determining path selection probabilities, even for large deployments of Hydra?

Default Parameters and Algorithms

If not stated otherwise, we use the default parameters and algorithms as listed in Table 2.

Note that Karaoke uses very long paths of $l = 14$ mixes to achieve a negligible probability for path compromise when 20% of mixes are malicious. For a practical deployment of Hydra we envision a reduced circuit length of $l \approx 8$. Then, the probability of selecting a completely malicious circuit is still fairly low and relationship anonymity is protected by ≈ 16 mixes (two circuits). As tweakable wide block cipher for circuit cells we use `Threefish-1024` in combination with 12 rounds of a Feistel network to double its block size to 256 B [32].

Comparison with Karaoke

The minimal and average end-to-end latency for text-based messages in Hydra are determined by static and dynamic parameters. The minimal end-to-end latency $\delta_{\min}$ equals the round-trip time for circuit cells (recapitulate Fig. 3):

$$\delta_{\min} = 2(l+1)\Delta t_c. \tag{19}$$

For users, the average end-to-end latency $\bar{\delta}$ is crucial. It additionally has to reflect the average waiting time for the next round to start when a user wants to send a message, which is half the duration of a complete round (normal communication, circuit update, and artificial waiting time):

$$\bar{\delta} = \delta_{\min} + \frac{1}{2}\left(\delta_{\min} + l' \times \Delta t_u + \Delta t_w\right) \tag{20}$$

We compare Hydra against Karaoke's empirical results obtained from a deployment of $|V| = 100$ Amazon AWS `c4.8xlarge` instances as mixes [5]. Furthermore, we

Table 2 Default parameters and algorithms

Parameter	Default value	Comments
Circuit length l	14	Path length used in Karaoke
VoIP circuit length l'	4	
Min. node degree $d_{\min}$	$0.2 \times \|V\|$	20% of the number of mixes
Max. latency $w_{\max}$ on links	100 ms	
Max. clock skew σ	10 ms	
Waiting time Δt_{w_1}	10 s	
Waiting time Δt_{w_2}	0 s	
VoIP codec interval Δt_v	20 ms	Packet rate of 50 pps
VoIP cell payload	64 B	
Key exchange	`x25519`	Diffie Hellman on Curve25519
Authenticated encryption	`AES-GCM-256`	For setup and update packets
Circuit cell encryption	`Threefish-1024`	With doubled block size [32]
VoIP cell encryption	`AES-CTR-256`	

fit their empirical results into our model for setting Δt_c, assuming that Hydra would have to use asymmetric cryptography (`x25519`) for every circuit cell. Then, Karaoke's performance can be modelled by setting $\Delta t_w = 0$ (they do not require waiting times) and $\Delta t_u = 0$ (they do not support VoIP). Note that we also multiply their empirical results (which correspond to the minimal end-to-end latency) by a factor of 1.5 to compensate the fact that on average, Karaoke's users have to wait half a round to be able to send their message just like in Hydra.

The performance characteristics of all AWS instances are assumed to be identical and are listed in Table 3. For transmission rates, we multiplied the available bandwidth ($10\,\text{Gbit}\,\text{s}^{-1}$) by a (conservative) factor of 0.5 to compensate for transport protocol overhead. To approximate processing power, we benchmarked single core performance of all relevant cryptographic algorithms on an Intel Core i7-7500U for one minute each, using Hydra's prototype code. We further multiplied the results by the thread count (18 for `c4.8xlarge`) and by an additional factor of 0.36 to compensate overhead like key lookup and thread synchronization, which fits the empirical results of Karaoke. Further note that Karaoke's experiment artificially introduced a latency of 50 ms between any pair of mixes.

We further assumed that all circuits are updated every round to get an upper bound for Δt_u. But even then, Δt_u is negligible because of the very efficient onion encryption of circuit update packets (`AES-GCM-256`).

The results of our comparison are shown in Fig. 11. First note that our model for Karaoke is a good approximation for their empirical results and thus can be used as an extrapolation of its performance. And as expected, using symmetric ciphers for onion encryption of circuit cells can significantly reduce the average end-to-end latency for large user populations. However, the end-to-end latency is comparatively high for small user populations due to the

Table 3 (Approximated) performance characteristics of an AWS `c4.8xlarge` instance (mix v) in the context of Hydra

Symbol	Value	Comment
$\beta_{s,v}$	200000 pps	Transmission rate for circuit setup packets
$\beta_{c,v}$	2332000 pps	Transmission rate for circuit cells
$\beta_{u,v}$	5682000 pps	Transmission rate for circuit update packets
$\mu_{s,v}$	131000 pps	Processing rate for setup packets
$\mu_{c,v}$	2333000 pps	Processing rate for circuit cells
$\mu_{u,v}$	19440000 pps	Processing rate for circuit update packets
$\rho_{\beta,v}$	1.216×10^{-5}	Fraction of transmission rate for one VoIP circuit
$\rho_{\mu,v}$	5.144×10^{-6}	Fraction of processing power for one VoIP circuit
$w_{v,u}$	50 ms	Latency to all other mixes $u \in V \setminus \{v\}$

inevitable waiting time between communication rounds. Furthermore, when using 3.5% of all circuits for VoIP in parallel, end-to-end latency drastically increases when the user population approaches 50 million users. This can be explained by the fact that for 50 million users, mixes then have to use ≈ 25% of their bandwidth solely for sending/receiving VoIP cells.

To compare anonymity, we further evaluate the "mixing property" of path selection between Karaoke (uniformly at random) and Hydra. For this, we simulated a varying number of circuit setups and measured the relative (to all circuits) location anonymity set size for each circuit. The location anonymity set size includes all other circuits that are successfully mixed with a targeted circuit. We further use the 5% percentile across all circuits as the metric for path selection as a whole. Following the evaluation of Karaoke, we assume a global observer and 20% of mixes to be malicious. Creation of dummy circuits is deliberately turned off to solely compare path selection.

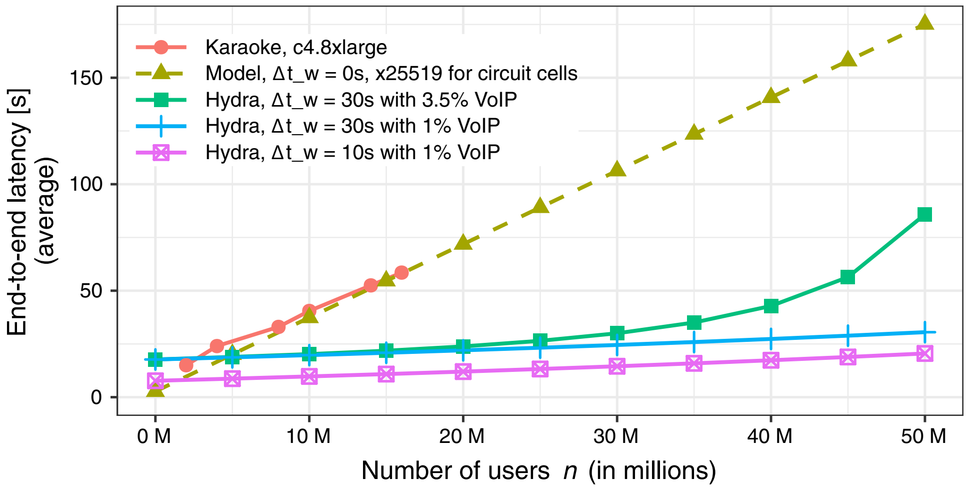

Fig. 11 Average end-to-end latency for Karaoke, Hydra, and an instantiation of Hydra with `x25519` for circuit cells to extrapolate the empirical results of Karaoke (updated from [14])

The results are shown in Fig. 12. While Karaoke's path selection has slight advantages when there are only a few circuits, both strategies approach a relative location anonymity set size of 1 when the total number of circuits is large enough. The advantage for fewer circuits may be explained by the increased uncertainty due to higher node degrees for path selection (99 versus 20).

Epoch Duration

The (minimum) duration d_e of one epoch is determined as follows (using the lower bound for the number k of communication rounds):

$$d_e = 2 \times k \times \left(\delta_{\min} + l' \times \delta_u + \Delta t_w\right) \tag{21}$$

$$\geq 2 \times \left\lceil \frac{(l-1) \times \Delta t_s}{\Delta t_w} \right\rceil \times \left(\delta_{\min} + l' \times \delta_u + \Delta t_w\right). \tag{22}$$

Note the factor of 2, which reflects the setup and communication phase.

With regard to mix capacities, we used the same scenario as for the Karaoke comparison (100 AWS `c4.8xlarge` instances as mixes). We further assumed that 3.5% of all circuits are used for VoIP again. Our results in Fig. 13 show a similar characteristic as our results for end-to-end latency with the same parameters: Epoch durations are practical for up to ≈ 40 million users, but increase drastically for larger user populations for the same reason as above.

VoIP Calls

To evaluate achievable end-to-end latency for VoIP calls in a realistic scenario, we use a latency dataset that we generated by extensive measurements in the Tor network. As bandwidth estimation in kbytes/second, we used their consensus weight [33]. We also published our dataset at [34] to facilitate future research on anonymous communications with strict latency requirements. Given the raw dataset, we further filtered all Tor relays that are not able to relay at least 100 voice calls with 32 kbits/second each. Furthermore, we filtered relays for which we did not have latency measurements to at least 90% of all other mixes. Missing latency measurements were completed using the average latency of the two corresponding relays. In summary, this results in a dataset containing 924 relays with their bandwidth and

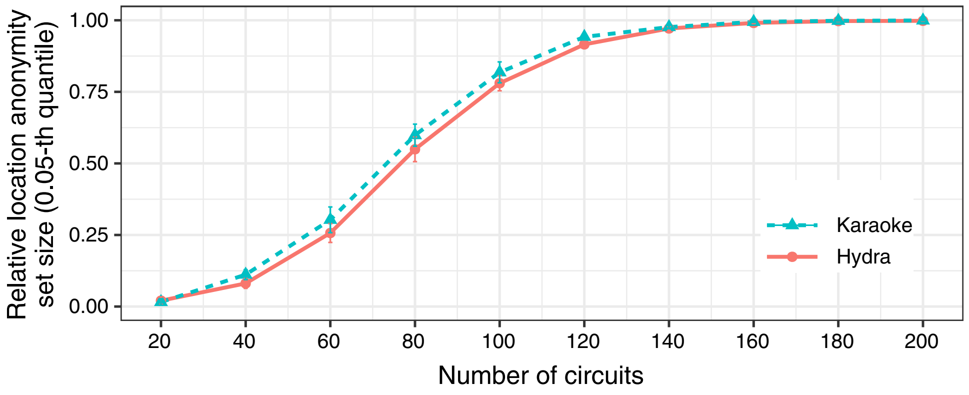

Fig. 12 Relative location anonymity set size for a varying number of circuits, comparing Karaoke's and Hydra's path selection

pairwise latencies. We further generated 32 subsets of size 100 uniformly at random to be used as mix set V for Hydra. If not stated otherwise, charts show the average metrics across the 32 subsets with 95% confidence intervals.

First, we simulated 1000 calls to get an overview of achievable latency and anonymity. As latency metric, we use the 95% percentile of the end-to-end latency across all calls. As anonymity metric, we use the (static) probability of selecting a completely malicious VoIP circuit (no location anonymity). For this, we assume a strong threat model: 20% of the mixes with the lowest average latency to all other mixes are compromised. Our results are shown in Figs. 14 and 15.

As expected, using a longer prefix of circuits for VoIP increases anonymity but also end-to-end latency. The same applies to the minimum node degree $d_{\min}$ for path selection. A good trade-off in this scenario is to use a prefix length $l' = 5$ combined with $20 \leq d_{\min} \leq 25$. Then, acceptable QoS is achieved for a large majority of calls and location anonymity is reasonable when considering the strong threat model. Naturally, relationship anonymity is better than location anonymity, but not shown here for brevity.

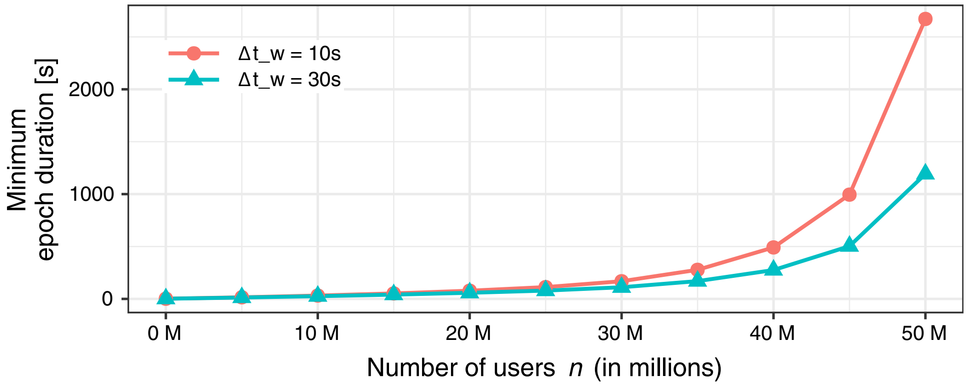

Fig. 13 Minimum epoch duration in Hydra, using 100 Amazon AWS `c4.8xlarge` instances as mixes (updated from [14])

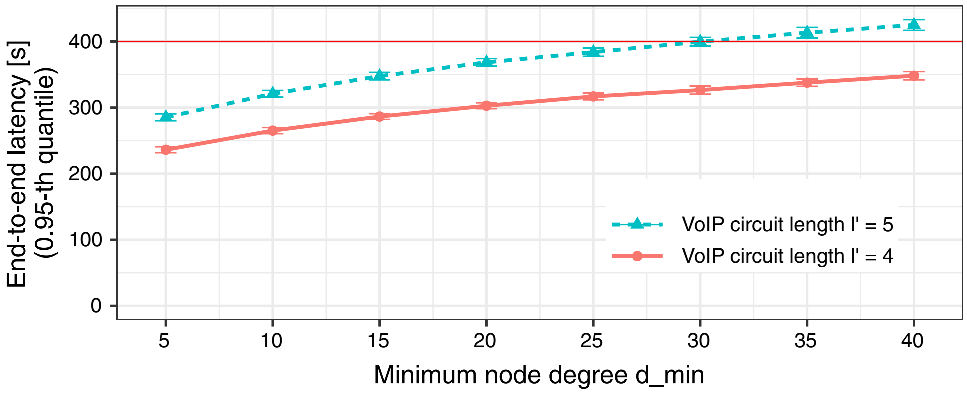

Fig. 14 End-to-end latency of voice calls for varying minimum node degrees $d_{\min}$ for path selection and a varying length l' of VoIP circuits

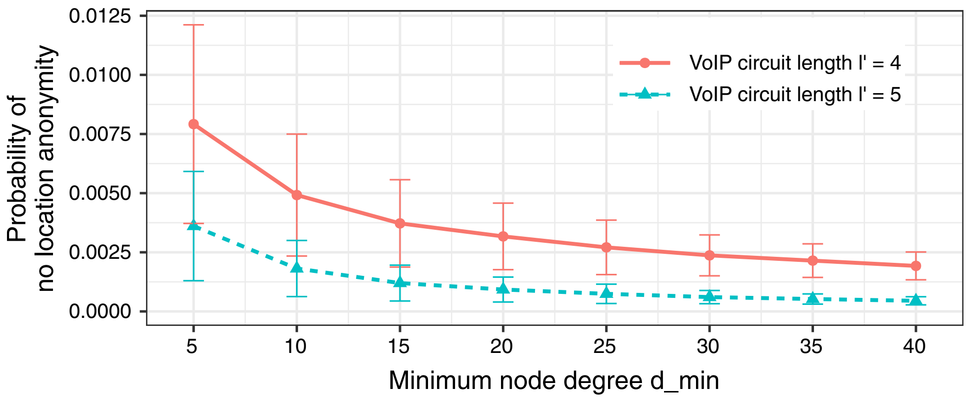

Fig. 15 Probability for selecting a completely malicious VoIP circuit (no location anonymity) for varying minimum node degrees $d_{\min}$ for path selection and a varying length l' of VoIP circuits

To further study the influence of latency-based path selection on anonymity, we compare two different attack strategies to compromise 20% of all mixes: One is to compromise the mixes with the best average latency to other mixes (as above) and one is uniformly at random. We simulated a varying number of calls, assuming all of them to start and stop at the same time to solely evaluate path selection. As metric, we use the 5% percentile of the relative location anonymity set sizes across all involved users. We fixed the VoIP circuit length to $l' = 5$ and the minimum node degree to $d_{\min} = 25$.

Our results in Fig. 16 (confidence intervals are too narrow to be visible) only show a slight advantage for the latency-based attack. Consequently, latency-based routing does not significantly degrade anonymity.

MCFP Runtime

To solve the MCFP for determining optimal path selection probabilities, we formulated the problem as linear program and solved it with Gurobi [35] (version 9.11) on an Intel Core i7-6700. Using the same base dataset as in the last experiment, Fig. 17 shows the average solving time for up to $m = 900$ (the maximum size for our base dataset) mixes across 32 runs each, with the standard deviation as error bars. We furthermore solved the problem on an artificial dataset with $m = 3000$. And while the latter took roughly 8 min to solve, we do not expect this to be a problem in practice because the MCFP only has to be solved once per epoch and only if the mix set or mix properties changed compared to the previous epoch. Nevertheless, in future work, we study possible approximation algorithms to efficiently determine path selection probabilities for even larger networks, for example, at the scale of Tor ($m \approx 5000$).

Summary

Compared to a recent anonymity system for text-based messaging with strong anonymity, Hydra is able to improve end-to-end latencies by an order of magnitude without degrading anonymity. At the same time, Hydra is able to support anonymous voice calls with large anonymity sets and acceptable QoS even in the presence of many malicious mixes. Furthermore, our novel approach to determine path selection probabilities may efficiently be applied in real word scenarios.

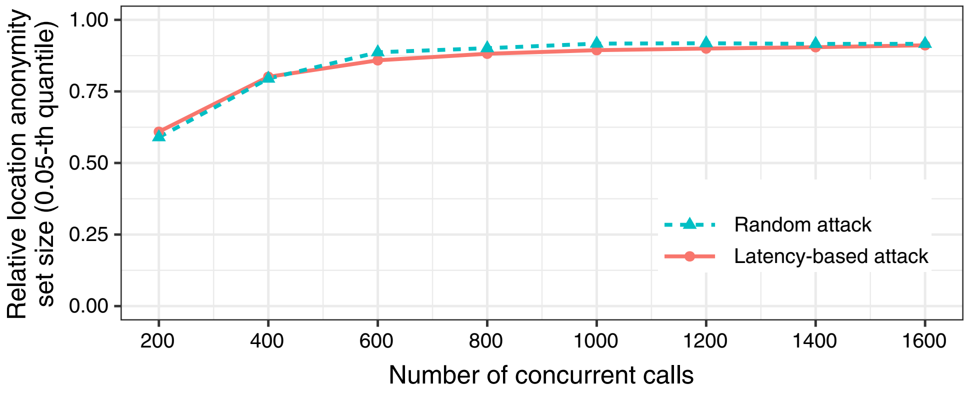

Fig. 16 Relative location anonymity set size for varying number of concurrent calls and two different attack strategies, for $l' = 5$ and $d_{\min} = 25$

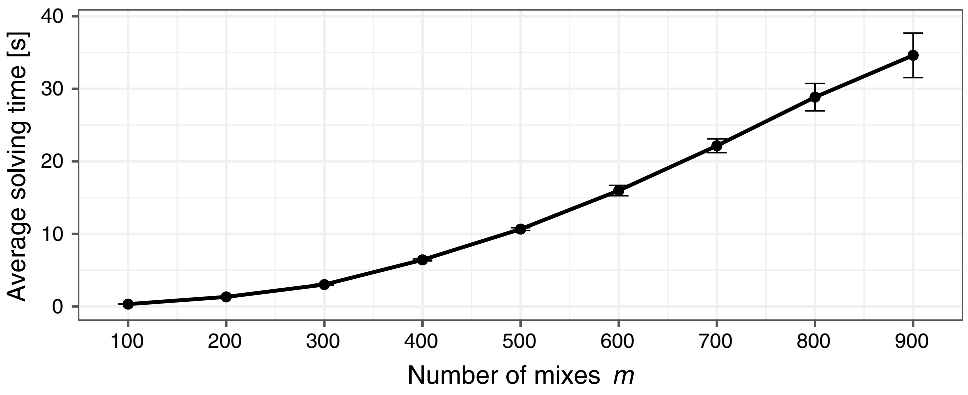

Fig. 17 Average time to solve the MCFP for determining path selection probabilities

Conclusion and Future Work

Using padded circuits in combination with latency-aware path selection allows Hydra to implement metadata security for a wide variety of applications. Compared to recent anonymous messaging systems with strong anonymity, Hydra is able to decrease end-to-end latencies for text-based messages by an order of magnitude without degrading anonymity. Our rendezvous mechanism furthermore avoids shortcomings of similar circuit-based designs. Most importantly, Hydra is able to efficiently provide strong location anonymity in addition to relationship anonymity. Circuits may further be upgraded to be used for voice calls, interactive chat sessions, or file transfers in parallel.

As future work, we plan to study the impact of disclosure attacks on the anonymity of VoIP circuits. Due to a higher churn of VoIP circuits, we expect disclosure attacks to be successful in practical deployments if no further countermeasures like fake calls are employed. We further want to implement Hydra as an open source project. A first prototype is available at https://github.com/hydra-acn.

Funding Open Access funding enabled and organized by Projekt DEAL.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Mayer J, Mutchler P, Mitchell JC. Evaluating the privacy properties of telephone metadata. Proc Nat Acad Sci. 2016;113(20):5536–41. https://doi.org/10.1073/pnas.1508081113.
2. Chaum D. Untraceable electronic mail, return addresses, and digital pseudonyms. Commun ACM. 1981;24(2):84–90. https://doi.org/10.1145/358549.358563.
3. Pham DV, Wright J, Kesdogan D. A practical complexity-theoretic analysis of mix systems. In: European Symposium on Research in Computer Security; 2011. pp. 508–527. https://doi.org/10.1007/978-3-642-23822-2_28.
4. Oya S, Troncoso C, Pérez-González F. Do dummies pay off? Limits of dummy traffic protection in anonymous communications. In: International Symposium on Privacy Enhancing Technologies; 2014. pp. 204–223. https://doi.org/10.1007/978-3-319-08506-7_11. Springer.
5. Lazar D, Gilad Y, Zeldovich N. Karaoke: distributed private messaging immune to passive traffic analysis. In: 13th USENIX OSDI; 2018. pp. 711–725.
6. Gelernter N, Herzberg A, Leibowitz H. Two cents for strong anonymity: the anonymous post-office protocol. PETS. 2016;2016(2):1–20.
7. Kwon A, Lu D, Devadas S. XRD: Scalable messaging system with cryptographic privacy. In: 17th USENIX NSDI; 2020. pp. 759–776.
8. Wang X, Chen S, Jajodia S. Tracking anonymous peer-to-peer VoIP calls on the internet. In: ACM CCS; 2005. pp. 81–91. https://doi.org/10.1145/1102120.1102133.
9. Dingledine R, Mathewson N, Syverson P. Tor: The second-generation onion router. In: 13th USENIX Security; 2004.
10. Chen C, Asoni DE, Perrig A, Barrera D, Danezis G, Troncoso C. TARANET: Traffic-analysis resistant anonymity at the network layer. In: IEEE EuroS &P; 2018. pp. 137–152. https://doi.org/10.1109/EuroSP.2018.00018.
11. Le Blond S, Choffnes D, Caldwell W, Druschel P, Merritt N. Herd: a scalable, traffic analysis resistant anonymity network for VoIP systems. ACM SIGCOMM. 2015;45(4):639–52. https://doi.org/10.1145/2829988.2787491.
12. Lazar D, Gilad Y, Zeldovich N. Yodel: Strong metadata security for voice calls. In: 27th ACM SOSP; 2019. pp. 211–224. https://doi.org/10.1145/3341301.3359648.
13. Schatz D, Rossberg M, Schaefer G. Optimizing packet scheduling and path selection for anonymous voice calls. In: ARES; 2021. https://doi.org/10.1145/3465481.3465768.
14. Schatz D, Rossberg M, Schaefer G. Hydra: Practical metadata security for contact discovery, messaging, and dialing. In: ICISSP; 2021. pp. 191–203. https://doi.org/10.5220/0010262201910203.
15. International Telecommunication Union. One-way Transmission Time, ITU-T recommendation G.114 edn; 2003. International Telecommunication Union.
16. Jung Y, Manzano C. Burst packet loss and enhanced packet loss-based quality model for mobile voice-over Internet protocol applications. IET Commun. 2014;8(1):41–9. https://doi.org/10.1049/iet-com.2011.0701.
17. Dolev D, Yao A. On the security of public key protocols. IEEE Trans Info Theory. 1983;29(2):198–208. https://doi.org/10.1109/TIT.1983.1056650.
18. Chaum D. The dining cryptographers problem: unconditional sender and recipient untraceability. J Cryptol. 1988;1(1):65–75. https://doi.org/10.1007/BF00206326.
19. Chor B, Goldreich O, Kushilevitz E, Sudan M. Private information retrieval. In: Proceedings of IEEE 36th Annual Foundations of Computer Science; 1995. pp. 41–50. IEEE
20. Corrigan-Gibbs H, Boneh D, Mazière, D. Riposte: An anonymous messaging system handling millions of users. In: IEEE SP; 2015. pp. 321–338. https://doi.org/10.1109/SP.2015.27.
21. Ahmad I, Yang Y, Agrawal D, El Abbadi A, Gupta T. Addra: Metadata-private voice communication over fully untrusted infrastructure. In: 15th USENIX OSDI; 2021.
22. Franck C, Sorger U. Untraceable voip communication based on dc-nets. arXiv preprint arXiv:1610.06549; 2016.
23. Van Den Hooff J, Lazar D, Zaharia M, Zeldovich N. Vuvuzela: Scalable private messaging resistant to traffic analysis. In: 25th

ACM SOSP; 2015. pp. 137–152. https://doi.org/10.1145/2815400.2815417

24. Tyagi N, Gilad Y, Leung D, Zaharia M, Zeldovich N. Stadium: A distributed metadata-private messaging system. In: 26th ACM SOSP; 2017. pp. 423–440. https://doi.org/10.1145/3132747.3132783.
25. Kwon A, Corrigan-Gibbs H, Devadas S, Ford B. Atom: Horizontally scaling strong anonymity. In: 26th ACM SOSP; 2017. pp. 406–422. https://doi.org/10.1145/3132747.3132755.
26. Chaum D, Das D, Javani F, Kate A, Krasnova A, De Ruiter J, Sherman AT. cMix: Mixing with minimal real-time asymmetric cryptographic operations. In: International Conference on Applied Cryptography and Network Security; 2017. pp. 557–578. https://doi.org/10.1007/978-3-319-61204-1_28.
27. Kwon A, Lazar D, Devadas S, Ford B. Riffle: an efficient communication system with strong anonymity. PETS. 2016;2016(2):115–34.
28. Piotrowska AM, Hayes J, Elahi T, Meiser S, Danezis G. The Loopix anonymity system. In: 26th USENIX Security; 2017. pp. 1199–1216.
29. Le Blond S, Choffnes D, Zhou W, Druschel P, Ballani H, Francis P. Towards efficient traffic-analysis resistant anonymity networks. ACM SIGCOMM. 2013;43(4):303–14. https://doi.org/10.1145/2534169.2486002.
30. Traudt M, Jansen R, Johnson A. FlashFlow: A secure speed test for Tor. arXiv preprint arXiv:2004.09583; 2020.
31. Chan-Tin E, Hopper N. Accurate and provably secure latency estimation with Treeple. In: NDSS; 2011.
32. Patarin J, Gittins B, Treger J. Increasing Block Sizes Using Feistel Networks: The Example of the AES. In: Cryptography and Security: From Theory to Applications, Springer; 2012. pp. 67–82. https://doi.org/10.1007/978-3-642-28368-0_8.
33. Tor Project. Tor Directory Protocol, Version 3. https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt, Accessed 19 May 2022.
34. Schatz D, Rossberg M, Schaefer G. Large-scale Latency Measurements in the Tor Network (v1.0); 2021. https://doi.org/10.5281/zenodo.4911583.
35. Gurobi Optimization, LLC. Gurobi Optimizer Homepage. https://www.gurobi.com, Accessed 19 May 2022.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.