$\mathcal{P}^5$: A Protocol for Scalable Anonymous Communication *

Rob Sherwood Bobby Bhattacharjee Aravind Srinivasan
University of Maryland, College Park, Maryland, USA
{*capveg, bobby, srin*}*@cs.umd.edu*

Abstract

We present a protocol for anonymous communication over the Internet. Our protocol, called $\mathcal{P}^5$ (Peer-to-Peer Personal Privacy Protocol) provides sender-, receiver-, and sender-receiver anonymity. $\mathcal{P}^5$ is designed to be implemented over the current Internet protocols, and does not require any special infrastructure support. A novel feature of $\mathcal{P}^5$ is that it allows individual participants to trade-off degree of anonymity for communication efficiency, and hence can be used to scalably implement large anonymous groups. We present a description of $\mathcal{P}^5$, an analysis of its anonymity and communication efficiency, and evaluate its performance using detailed packet-level simulations.

1 Introduction

We present the Peer-to-Peer Personal Privacy Protocol ($\mathcal{P}^5$) which can be used for scalable anonymous communication over the Internet. $\mathcal{P}^5$ provides sender-, receiver-, and sender-receiver anonymity, and can be implemented over the current Internet protocols. $\mathcal{P}^5$ can scale to provide anonymity for hundreds of thousands of users all communicating simultaneously and anonymously.

A system provides *receiver anonymity* if and only if it is not possible to ascertain who the receiver of a particular message is (even though the receiver may be able to identify the sender). Analogously, a system provides *sender anonymity* if and only if it is not possible for the receiver of a message to identify the original sender. It is not possible to provide perfect anonymity in a communication system since it is usually possible to enumerate all possible senders or recipients of a particular message. In general, the degree of sender/receiver anonymity is measured by the size of the set of people who *could* have sent/received a particular message. There have been a number of systems designed to provide receiver anonymity [2, 5], and a number of systems that provide sender anonymity [8, 9]. In these systems, individual senders (or receivers) cannot determine the destination (or origin) of messages beyond a certain set of hosts in the network.

Our system, $\mathcal{P}^5$, provides both receiver and sender anonymity and also provides sender-receiver anonymity. Specifically, we assume that an adversary in our system may passively monitor every packet on every link of a network, and is able to correlate individual packets across links. Thus, the adversary can mount any passive attack on the underlying networking infrastructure. However, the adversary is not able to invert encryptions and read encrypted messages. The adversary can also read all signaling messages in the system. Our system provides receiver and sender anonymity under this rather strong adversarial model and provides the sender-receiver anonymity (or *unlinkability*) property. Thus, the adversary cannot determine if (or when) any two parties in the system are communicating. $\mathcal{P}^5$ maintains anonymity even if one party of a communication colludes with the adversary who can now identify specific packets sent to or received from the other end of the communication). Unlike previous known solutions, $\mathcal{P}^5$ can be used to implement a scalable wide-area system with many thousand active participants, all of whom may communicate simultaneously.

1.1 A naive solution

Consider a global broadcast channel. All participants in the anonymous communication send fixed length packets onto this channel at a fixed rate. These packets are encrypted such that only the recipient of the message may decrypt the packet, e.g., by using the receiver's published public key. Assume that there is a mechanism to hide, spoof, or re-write sender addresses, e.g., by implementing the broadcast using an application-layer peer-to-peer ring, and that all messages are sent to the entire group. Lastly, every message is hop-by-hop encrypted, and thus, it is not possible to map a specific incoming message to a node to a particular outgoing message. (In essence, every node acts as a *mix* [1]). It is possible that a node may not be actively communicating at any given time, but in order to maintain the fixed communication rate, it would have to send a packet anyway. Such a packet would be a *noise* packet, and any packet destined for a particular receiver would be a *signal* packet.

This system provides receiver anonymity, since the sender does not know *where* in the broadcast group the receiver is or

*The first author is with the Department of Computer Science, University of Maryland at College Park. The second and third authors are with the Department of Computer Science and the University of Maryland Institute for Advanced Computer Studies, University of Maryland at College Park. This work was supported by a grant (ANI 0092806) from the National Science Foundation.

which host or address the receiver is using; the sender only knows that the receiver is part of the broadcast group. This system also provides sender anonymity, since all messages to a given receiver (in case of a ring) come from a single upstream node, and the receiver cannot determine the original sender of a message. Lastly, this solution also provides unlinkability from a passive adversary since the adversary is not able to gain any extra information from monitoring any (or all) network links. For example, suppose node a is sending messages to node b. The adversary sees the same number of messages from node a whether it were conversing with b or not, and all of the messages are sent to the same broadcast address. Similarly, whether a talks to b or not, b receives the same number of messages from a over any suitably large interval. Note that the adversary is not able to *trace* a message from the sender to a receiver or vice-versa because of the hop-by-hop encryption, and thus, even if one end of a communication colludes with the adversary, the anonymity of the other party is not compromised.

This naive solution does not scale due to its broadcast nature. As the number of people in the channel increases, the available bandwidth for any useful communication decreases linearly, and end-to-end reliability decreases exponentially. It is possible to increase the bandwidth utilization and reliability by limiting the number of people in a broadcast group, but then two parties who want to communicate may end up in different groups.

$\mathcal{P}^5$ is based upon this basic broadcast channel principle; we scale the system by creating a hierarchy of broadcast channels. Clearly, any broadcast-based system, including $\mathcal{P}^5$ will not provide high bandwidth efficiency, both in terms of how many bits it takes a sender–receiver pair to exchange a bit of information, and how many extra bits the network carries to carry one bit of useful information. $\mathcal{P}^5$ allows users to choose how *inefficient* the communication is, and provides a scalable control structure for securely and anonymously connecting users in different logical broadcast groups. We present an overview of $\mathcal{P}^5$ next.

1.2 Solution overview

$\mathcal{P}^5$ scales the naive solution by creating a broadcast hierarchy. Different levels of the hierarchy provide different levels of anonymity (and unlinkability), at the cost of communication bandwidth and reliability. Users of the system locally select a level of anonymity and communication efficiency and can locally map themselves to a level which provides requisite performance. At any time, it is possible for individual users in $\mathcal{P}^5$ to decrease anonymity by choosing a more communication efficient channel. (Unfortunately, it is not possible to regain stronger anonymity). Obviously, it is possible to choose a set of parameters that is not supported by the system (e.g., mutually incompatible levels of bandwidth utilization and anonymity).

Chaum introduced sender anonymity in [1], and sender-receiver anonymity (as the dining cryptographers problem) in [2]. The solution presented in [2] provides sender, receiver anonymity and unlinkability. However, it does not provide a bandwidth vs. communication efficiency trade-off, and one "conversation" can be only sustained at any one time. The closest prior work to $\mathcal{P}^5$ is [5]. This system also allows a trade-off between bandwidth and communication efficiency, but like in [2], in this system only one sender-receiver pair may simultaneously communicate in this system. Thus, these systems cannot be used to implement large anonymous communication groups. A number of recent protocols [8, 4] also provide anonymous communication over the Internet. These protocols have the same underlying systems assumptions as $\mathcal{P}^5$; however, unlike $\mathcal{P}^5$ these protocols cannot withstand an all-powerful passive adversary.

1.3 Roadmap

The rest of this paper is structured as follows: we discuss related work in Section 2. We describe the $\mathcal{P}^5$ algorithm in detail in Section 3, and present a set of analytic bounds on performance in Section 5. In Section 6, we analyze results from packet-level $\mathcal{P}^5$ simulator. We discuss future work and conclude in Section 7.

2 Related Work

We discuss prior work related to $\mathcal{P}^5$. We begin with a discussion of the Dining Cryptographers problem, and discuss some other systems that provide anonymity over the Internet.

2.1 Dining Cryptographers and Mixes

Dining Cryptographers The Dining Cryptographers (DC-net) protocol [2] provides sender anonymity under an adversary model similar to $\mathcal{P}^5$. DC-Net assumes a public key infrastructure, and users send encrypted broadcasts to the entire group, thus achieving receiver anonymity. However, unlike $\mathcal{P}^5$, all members of the group are made aware of *when* a message is sent, so Dining Cryptographers does not have the same level of sender-receiver anonymity. Also, in DC-net, only one user can send at a time, so it takes additional bandwidth to handle collisions and contention [10]. Lastly, a DC-net participant fixes its anonymity vs. bandwidth trade off when joining the system, and there are no provisions to rescale that trade off when others join the system.

Mixes A *mix* is a process that provides anonymity via packet re-shuffling. Mixes were introduced by Chaum in [1]. Mixes work best in series, and need a constant amount of traffic to avoid delay while preserving anonymity. $\mathcal{P}^5$ does both by creating a hierarchy of mixes, and the constant stream of signal and noise packets serve to keep the mixes operational.

2.2 Recent Internet-based Anonymous Communications Work

We describe four anonymity protocols that can be implemented over the Internet

Xor-Trees Like $\mathcal{P}^5$, Xor-Trees [5] provides sender-, receiver-, and sender-receiver anonymity. However, unlike $\mathcal{P}^5$, Xor-Trees do not admit a per user anonymity vs. communications efficiency trade off. Also, like in DC-net, only a single user may send at any one time in an Xor-Tree. Thus, in an Xor-Tree, performance degrades due to collisions as the number of users increase.

Crowds, Hordes, and Onion Routing Both Crowds [8] and the more recent Hordes [9] provide sender anonymity. The basic idea in both these systems is similar to Onion Routing [6], in which messages between communicating users are routed on an application-layer overlay using paths different than the shortest path. The receiver cannot resolve the sender of a particular message since messages take different, potentially randomly chosen, routes through the network. However, neither system can provide anonymity when confronted by a passive observer who can mount statistical attacks by tracing and correlating packets throughout the network. None of these systems provide receiver anonymity.

FreeNet Freenet [4] provides an anonymous publish-subscribe system over the Internet using an application-layer overlay, much like $\mathcal{P}^5$. However, FreeNet is designed for anonymous storage and retrieval, and the anonymity issues for such a system are different than a system like $\mathcal{P}^5$ that provides anonymity when communicating parties are on-line. There is no notion of noise or signal, etc., and the major issues in FreeNet are decoupling/hiding authorship from a particular document, and providing fault-tolerant anonymous availability for a set of static documents.

3 The $\mathcal{P}^5$ Protocol

$\mathcal{P}^5$ is based upon public-key cryptography. $\mathcal{P}^5$ does not require a global public-key infrastructure; however, we do assume that if two parties wish to communicate, they can ascertain each other's public keys using an out-of-band mechanism.

Assume N individuals[1] wish to form an anonymous communication system using $\mathcal{P}^5$. Assume each of these $\mathcal{P}^5$ users (or group members) have public keys $K_0, \ldots K_{N-1}$. $\mathcal{P}^5$ will use these N public keys, called *communication keys* to create a *logical broadcast hierarchy*.

3.1 The $\mathcal{P}^5$ logical broadcast hierarchy

The $\mathcal{P}^5$ logical broadcast hierarchy is a binary tree ($\mathcal{L}$) which is constructed using the public keys $K_0, \ldots, K_{N-1}$. Each node of $\mathcal{L}$ consists of a bitstring of a specified length. We present the algorithm assuming each node of $\mathcal{L}$ contains both a bitstring and a bitmask. The bitmask specifies how many of the most significant bits in the bitstring are valid. Though not strictly necessary, the addition of the bitmask will significantly ease our exposition. We use the notation (b/m) to represent the contents of a group, where b is the bitstring, and m is the number of valid bits.

The root of $\mathcal{L}$ consists of the null bitstring and a zero length mask. We represent the root with the label $(\star/0)$. The left child of the root contains the group $(0/1)$ and the right child is $(1/1)$. The rest of the tree is constructed as shown in Figure 1. For example, note that group $(0/1)$ represents the bitstring 0 and the group $(00/2)$ represents the bitstring 00.

Each group in $\mathcal{L}$ corresponds to a broadcast channel in $\mathcal{P}^5$. A message sent to a group is (unreliably) forwarded to a subset of all members of the system. Suppose user A sends a message to group (b/m). This message will be forwarded to user B in group (b'/m') if and only if the k most significant bits of b and b' are the same, where k is defined to be $\min\{m, m'\}$. We call this common prefix testing the "min-common-prefix check".

Thus, a message sent to a group (b/m) is sent to three distinct regions of the $\mathcal{L}$ tree:

- **Local:** A message sent on (b/m) is broadcast to all members of the (b/m) group.
- **Path to root:** For each $m' < m$, this message is also broadcast to all members of the group $(b_{m'}/m')$, where $b_{m'}$ denotes the m'-bit prefix of b.
- **Subtree:** Lastly, for all $m'' > m$, this message is also sent to all groups $(b| \star /m'')$, where $b| \star /k$ is any bitstring of length k that begins with the string b.

For example, any message sent to the root $(\star/0)$ is forwarded to all members of all groups. Any message sent to $(0101/4)$ is sent to members of the 0101 group (local). This message is also sent to all members of the $(010/3)$ group; all members of the $(01/2)$ and $(0/1)$ groups, and the all members of the $(\star/0)$ group. Further, this message is sent to every member of the $(0101 \star /k)$ groups, where $k > 4$, and $0101\star$ is any bitstring that starts with 0101. In general, when a message is sent to group (b/m), members of groups (b'/m') are forwarded this message if and only if the nodes of $\mathcal{L}$ corresponding to (b/m) and (b'/m') have an ancestor-descendant relationship. Note that these broadcast groups should be implemented as peer-to-peer unicast trees in the underlying network (and not multicast trees). These channels may lose messages and require no particular consistency, reliability, or quality-of-service guarantees. We describe the precise networking and systems requirements of $\mathcal{P}^5$ and underlying protocols in Section 4.3.

Each user in the system joins a set of such broadcast groups. In general, communication efficiency increases as the groups's mask size increases; however, as we shall see, this increase in efficiency comes at an expense of reduced anonymity. The depth of the $\mathcal{L}$ tree is defined at run-time and depends on the number of people in the system (N), and on the security parameters chosen by individual users. However, we need to fix a maximum depth $\mathcal{L}_d$ of this tree: choosing this parameter, a-priori, is not difficult, since such a system can accommodate approximately $2^{\mathcal{L}_d} \cdot k$ users, where k is the least number of people in any channel. In our implementation, we have chosen $\mathcal{L}_d$ to be 32.

[1] It is entirely possible that the N keys belong to n different individuals, such that $n < N$. We discuss this issue in Section 3.5.

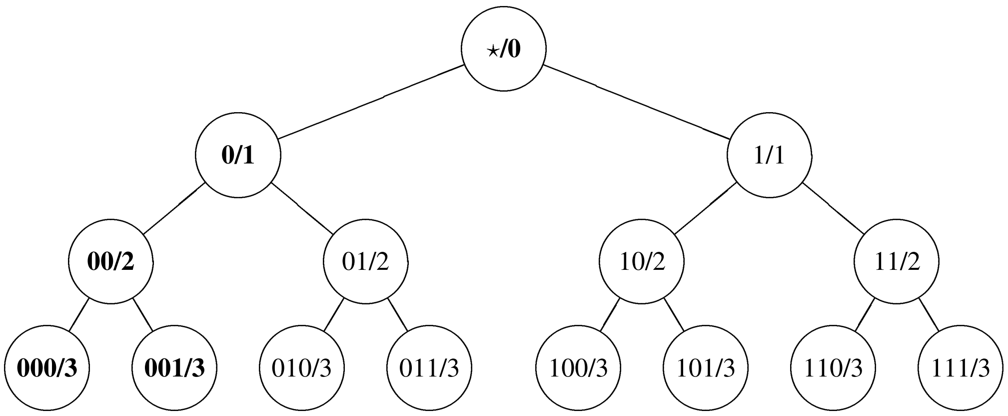

Figure 1: Form of the $\mathcal{P}^5$ logical broadcast tree ($\mathcal{L}$). The effective broadcast channels of a user in group $(00/2)$ is shown in boldface.

3.2 Mapping users to $\mathcal{L}$

We use a secure public hash function ($H(\cdot)$) to map users to a $\mathcal{L}$ node (group). Consider user A, with public-key K_A. Assume $b_A = H(K_A)\ mod\ 2^{\mathcal{L}_d}$. User A will join some group of the form b_A/m. The length of the mask m is chosen independently and randomly by user A according to a local security policy, as described in Section 3.4. The choice of the m parameter should be secret, and it should not be possible to determine which *precise* group a user is joined to. Thus, given a public key, it is public knowledge which set of groups a user *may* be in, but it is difficult to determine which *specific* group in this set the user has chosen.

Suppose users A and B are mapped to some arbitrary groups in the tree $\mathcal{L}$. We say a channel c is common between A and B if and only if messages sent to c are forwarded to both A and B. Suppose A and B join groups (b_A/m_A) and (b_B/m_B) respectively, and assume both know each other's public key. Since A knows K_B, it can determine b_B; however, A does not know the value of m_B. Even without any knowledge of the masks, A and B can begin to communicate using the $(\star/0)$ channel. Unfortunately, this communication channel can be quite lossy since messages have a higher probability of getting lost in the channels "higher" up in the $\mathcal{L}$ tree and $(\star/0)$ is the most lossy communication channel of all.

The communication efficiency can be improved as follows. Instead of sending messages through $(\star/0)$, A could try to send messages on to some group (b_B/m), $m > 0$. B would receive these messages, and *may* reply back to A. However, in doing so, B sacrifices some anonymity since A can now map B to a smaller set of users. (A maps B to a smaller set using a "Difference Attack" described in Section 3.5). In general, B can choose to communicate back to A using any length mask; however, longer masks trade-off anonymity for communication efficiency. B can selectively "trust" some users and reveal longer masks for these trusted users, but in general, the anonymity of B is bounded by the longest mask that it has revealed to any other member. Lastly, note that the communication efficiency is upper bounded by the smaller of the two masks revealed by either of the participants in a communication.

Suppose A and B have agreed upon the length of a mask (say m), and wish to communicate. This means A is joined to some group (b_A/m_A) and B is joined to some group (b_B/m_B), where both m_A and m_B are greater than or equal to m. They can now communicate by sending messages to a lowest common ancestor channel that has both (b_A/m_A) and (b_B/m_B) as descendants. However, if A and B only join one group each, and the public keys are uniformly hashed to channels on the $\mathcal{L}$ tree, then there is approximately 50% probability that $(\star/0)$ will be the only channel that A and B would have in common! Similarly, for any given node, an exponentially high number of nodes would be "farther" away on the logical broadcast tree, and in general, the communication on the system will not be very effective. Once again, we are reduced to using the inefficient global broadcast channel for most communications.

Our solution to this inefficient routing is as follows: each user joins a small number of groups on the logical tree. For each joined group, users generate another public-private key pair, called *routing key*. These routing keys are generated locally, and do not require any global coordination. In fact, it should not be possible to map a user's routing key to their communication key, otherwise, the user's anonymity can be compromised (using an "Intersection Attack", Section 3.5).

When user A joins a group c, it periodically sends a message to the channel listing other channels that it is joined to. This message serves as a routing advertisement, and will be used to efficiently send messages along the lower levels of the $\mathcal{L}$ tree. In general, the advertisements from a node contains the set of channels it can directly reach, the set of channels it can reach using one other node, and so on. In effect, these t routing keys generate "lateral" edges in the tree. In Section 5, we show that typically, each user needs to join only a few groups (≤ 3) for any two users in $\mathcal{P}^5$ to have short paths (≤ 2 channel crossings) between them with high probability.

We note that a user joins a set of groups only when it enters the system, and should not change the set of channels it is part of. Otherwise, once again, yet another intersection attack becomes feasible that can compromise their anonymity. Each group joined by a user corresponds to a one hop peering in the underlying network[2]; we call the set of these peerings the physical connectivity graph, and denote it with $\mathcal{P}$.

3.3 Signal and Noise

Our description of the $\mathcal{P}^5$ protocol is nearly complete: however, we still need a crucial piece. Assuming packet sources cannot be traced from the broadcast messages (See Section 4 for the precise packet format), the protocol as described provides sender and receiver anonymity. We assume that each messages is of the same size and is encrypted *per-hop*, and thus it is not possible to map an outgoing message (packet) to a specific packet that the node received in the past. However, a passive observer can still mount an easy statistical attack and trace a communication by *correlating* a packet stream from a communicating source to a sink.

Thus, we add the notion of noise to the system. The noise packets should be added such that a passive correlation attack becomes infeasible. There are many possible good noise-generation algorithms, and we use the following simple scheme.

Each $\mathcal{P}^5$ user, at all times, generates fixed amount of traffic destined to channel chosen uniformly at random. A packet transmitted from a node is one of the following:

- A packet (noise or signal) that was received from some incoming interface that this node is forwarding onto some other channel(s). (The precise forwarding rule for $\mathcal{P}^5$ is described in Section 4).
- A signal packet that has been locally generated.
- A noise packet that has been locally generated.

Note that to an external observer, there is no discernible difference between these three scenarios. In general, only the source and destination of a communication can distinguish between noise and signal packets. They are treated with equal disdain at all other nodes.

Message Dropping Algorithms In any communication system without explicit feedback, e.g. our channel broadcasts, message queues may build at slow nodes or at nodes with high degree. In $\mathcal{P}^5$, members may simply drop any message they do not have the bandwidth or processing capacity to handle. The global properties of the system depend upon how messages are dropped. We have considered two different dropping algorithms:

- Uniform drop: This is the simplest scheme in which messages from the input queue are dropped with equal probability until the input queue size is below the maximum threshold.
- Non-uniform drop: In this scheme, messages which are destined to a channel higher up in $\mathcal{L}$ are dropped preferentially.

  We have experimented with several variations of this scheme; the specific scheme which we use for our simulations drops packets destined for higher nodes with an exponentially higher probability.

  If most of the end-to-end paths in a $\mathcal{P}^5$ network can use "lateral" edges, i.e. between channels at the same logical height, then this scheme provides lower drop rate. However any communication that must use "higher" channels have proportionately high drops.

3.4 Anonymity Analysis

Assume node a has joined the $\mathcal{L}$ at node (b/m). Let $E\{(b/m)\}$ be the set of members who receive a broadcast message sent to channel (b/m). (Recall that this set includes all members of group (b/m), all members of the groups below (b/m), and all members of groups on the path to the root of $\mathcal{L}$).

Claim 3.1 *The anonymity of a node communicating using channel (b/m) is equivalent to the set of members who are part of $E\{(b/m)\}$.*

Proof.
We consider the sender-, receiver-, and sender-receiver anonymity cases separately.

- Sender Anonymity

  Sender anonymity is the size of the set of the nodes that *could* have sent a particular packet to a given host (say B). A receiver who can only monitor their own links cannot determine the source of a packet since this information is never included in the packet. A receiver, in collusion with a all-powerful passive adversary, however, can enumerate the set of nodes who *could* have sent the packet by computing the closure of the set of nodes who have a causal relationship with B. (In this case, nodes X and Y are causally related if X sent a packet to Y). This closure would be computed over some finite time window on the order of the end-to-end latency in the system.

  However, in $\mathcal{P}^5$, a user connected to (b/m) sends packets at a constant rate (signal or noise), and these packets are received by all users in $E\{(b/m)\}$. Thus, there is a causal relationship between every user in a broadcast group. However, inter-channel routers transmit packets between channels, and over time every node in the system is causally related to every other node in the system.

  Suppose a malicious receiver tries to expose a sender. It can, at best (assuming there are no other cross channel packets), causally relate packets to its *own* broadcast

[2]Clearly, these are one hop "transport level" peerings, and not one "physical hop" peerings.

group. Further, if the receiver is able to determine and compromise the router node, then the sender anonymity becomes the effective broadcast group of the sender.

In case the receiver cannot compromise the channel router node, the sender's anonymity is the size of the entire system, even in the presence of an all-powerful passive adversary.

- Receiver Anonymity

  When A sends a packet to B at channel (b_B/m_B), every member of $E\{(b_B/m_B)\}$ receives the packet. From the perspective of an external observer, the behavior of the system is exactly the same whether B receives the packet or not. Thus, B's receiver anonymity is exactly equivalent to the set of all users who receive the packet, namely $E\{b_B/m_B\}$.

- Sender-receiver anonymity

  Since all nodes in the system send at a constant rate, and all packets are pair-wise encrypted between each hop, we claim that it is impossible for a passive observer to distinguish noise from signal packets. Since the observer cannot distinguish signal packets, it cannot discern if or when A communicates, and thus, it cannot determine when A is communicating with any other node B.

□

Assume that the rate at which some user A sends packets does not change when it is sending signal versus noise packets. In this case, the distribution of packets, whether they are signal packets or noise, does not affect the security of the node. Thus, a nice property of our system is that the anonymity of any node A depends only upon the length of the mask that A is willing to respond to.

3.5 Attacks

In this section, we outline a number of attacks that a system like $\mathcal{P}^5$ must guard against, and show why $\mathcal{P}^5$ is invulnerable to all these attacks.

- **Correlation Attack:** We have already alluded to this attack in which a passive observer is able to (statistically) track signal packets from a source to a destination, thus violating sender-receiver anonymity. In $\mathcal{P}^5$ the noise packets thwart this attack.

- **Intersection Attack:** If an adversary knows that a user is two different sets U and V, then the anonymity of the user is reduced to $U \cap V$. If users are uniformly distributed across such sets that can be intersected, then the anonymity for any user in these reduces exponentially with the number of intersecting sets. For example, suppose users communicate using both their routing keys and their communication key. With each key, there is a corresponding set of users who may own that key. This leads to an intersection attack. This is the reason a user communicates with only one key in $\mathcal{P}^5$, and routing keys cannot be mapped back to the communication keys. Note that this is also the reason users cannot increase their anonymity beyond the smallest set they have ever been mapped to.

- **Difference Attack:** If an adversary can map the user to some set and can assert that the user is not in some other set, then it can map the user to the difference between these two sets.

  For example, suppose user A has revealed an m bit mask. In this case, it should not respond or react to packets sent to any group (b/m') where $m' > m$. If A does respond to such packets, then A is divulging where in the $\mathcal{L}$ it is *not*. That is because the receiver set of $E\{(b/m')\}$ is smaller than the receiver set $E\{(b/m)\}$, and now an adversary knows that A is not in the set $E\{(b/m)\} - E\{(b/m')\}$.

- **DoS attack:** Suppose a malicious user wants to reduce the efficiency of the system by sending a large number of useless packets. $\mathcal{P}^5$ can withstand this type of an attack since we impose a per-link queue limit, and all the extra packets from the malicious user will be dropped at the very first hop. Note that even the local broadcast group is not affected by a DoS attack as long as the first non-colluding hop correctly implements its queue limits.

- **Mob attack:** In this case, a (set of) malicious users (the mob) collude to try to expose some user A. These users can all join the same channel as A, and reduce the efficiency of the channel. This can cause A to expose more bits of its mask or cause other legitimate users to leave the channel. In either case, the mob has reduced A's anonymity to the set of remaining legitimate users.

  This is a difficult attack to handle in a system which provides anonymity, since it is (hopefully) not possible to map public keys back to individuals. In $\mathcal{P}^5$, this attack can be handled by choosing a security parameter larger than the size of largest mob.

  Note that in practice, it may be possible for a single attacker to spoof multiple addresses. However, since we use unicast and each member of the group must communicate with others, all these addresses must actually exist on the network. In practice, however, it is unlikely that an attacker can co-opt addresses from many different Autonomous Systems (ASs); and A can thwart this attack by ensuring that there are enough different ASs represented in the channel that it responds on. However, an all-powerful *active* attacker can mount this attack from enough different ASs to expose any user. Thus $\mathcal{P}^5$ is susceptible to adversaries who can actively manipulate (e.g. by generating packets from arbitrary ASs) large parts of the network.

4 Details

In this section, we present details from our implementation of $\mathcal{P}^5$. We have implemented $\mathcal{P}^5$ in a packet level simulator, but the details from our implementation would be useful in a "real" implementation as well.

Packet Format We use fixed length packets of size 1 KB. The fixed packet length is used to eliminate any information an adversary can gain by monitoring packet lengths.

The $\mathcal{P}^5$ header only contains the identifier for the first hop destination channel (a (b/m) pair). (It could, equivalently, also contain the ultimate destination, but we chose the first-hop destination in cleartext option). In our implementation, b is a 32 bit unsigned integer, and m is a 6 bit integer. If the packet is a signal packet, the rest of the packet is encrypted using the next hop receiver's public key. Since packets may need to be encapsulated, and each packet is the same size, each packet also contains a padding field which is the size of the $\mathcal{P}^5$ header.

The data part of the $\mathcal{P}^5$ packet contains a set of fixed size "chunks" each of which is encrypted with the receiver's public key. These chunks are formed naturally by many public-key encryptions. The decrypted data part of the $\mathcal{P}^5$ packet contains a checksum which the receiver uses to determine whether a packet is destined for itself or not. Each chunk can be decrypted independently; thus, a receiver does not need to decrypt an entire noise packet, it can discard a packet as soon as the first chunk fails its checksum. For efficiency, the first chunk of a signal packet may also include a symetric cipher key for use in decrypting the other chunks, as symetric cipher decryptions tend to be faster than asymetric ones.

Regardless of whether a packet decrypts properly, the receiver schedules each packet for further delivery within the local channel using the forwarding rule described below.

The first chunk of a signal packet contains an encrypted bit which determines whether a packet should be forwarded onto some other channel, or whether the packet is destined for the current node. It also contains a channel identifier for the ultimate destination for the packet, which the current node uses to choose an outgoing channel.

When a node receives a packet with the "forward" bit set, it interprets the rest of the data as another $\mathcal{P}^5$ packet, and if possible, forwards it onto the specified channel. In the forwarding step, the process at a channel router is different depending on whether the packet is at its ultimate channel or not:

- If the packet is not at its final channel, the current node replaces the first chunk with a new chunk in which the "forward" bit is set, sets the proper ultimate destination channel, and encrypts this chunk with the public key of the next hop.

  If the packet is already is the destination channel, then the data part of the packet is already formatted with the proper address and has a valid first chunk encrypted by the public key of the intended recipient. The current node adds a last chunk at end of the packet with random bits to increment the packet length to the fixed system size.

If the "forward" bit is not set, then this signal packet is delivered locally.

Forwarding within a channel Since each logical channel is a tree, each node can use the following simple forwarding algorithm to forward a packet p sent to some arbitrary (b/m) channel on $\mathcal{L}$.

> Forward p to a peer on channel c iff only if p did not come in on c and if c passes the min-common-prefix check with respect to (b/m).

Note that it is important that the output order of the packets not be determined by the input order, else it becomes possible to correlate packets across successive nodes and trace communication between two parties. In other words, each node should act like a *mix* [1].

4.1 Member Security and Join Procedure

Analogous to the definition in [8], we define anonymity for a user A as the set S of users in the group who are "indistinguishable" from that A, i.e., no other user or a passive adversary can resolve messages from A to a granularity finer than S.

We assume that each user u requires a minimal acceptable level of anonymity, i.e. each user requires their corresponding S set to be of a minimum size. We call this minimum set size the *security parameter*, and denote it with ψ_u. Each user u may also define a maximum required level of security (i.e. a maximum size of the corresponding S set) since this provides a bound on the communication inefficiency. We call this the *efficiency parameter*, and denote it with η_u. User A joins group (b/m), s.t. $\psi_A \le |E\{(b/m)\}| \le \eta_A$ in the following manner:

- A initially joins $(\star/0)$. If $\psi_A \le |E\{\star/0)\}| \le \eta_A$, we're done.

  If $\psi_A > |E\{\star/0)\}|$, A the entire system does not have enough members to provide the requisite anonymity. A forwards packets for other nodes and sends noise packets, but does not directly communicate with other nodes.

  If $|E\{\star/0\}| > \eta_A$, then this group has too many members, and A joins the appropriate channel of the form $(b/1)$.
- A repeats this procedure until it finds a (b/m) such that $\psi_A \le |E\{(b/m)\}| \le \eta_A$.

Clearly, it is possible to choose incompatible values ψ and η such that $\eta > |E\{(b/m)\}$ and $\psi > |E\{(b/m+1)\}|$. In this case, the user can either change their security or efficiency parameter or wait in channel $(b/m+1)$, until $\psi \le |E\{(b/m+1)\}|$.

In our simulations, each user can determine the number of people in a group by consulting an "oracle" which maintains an up to date list of channel memberships. In implementation, this information can be maintained in a secure distributed manner, either by the underlying application-layer multicast primitive, or at a well-known centralized "topology server".

The "topology server" construct is needed if it is not possible to infer approximate group sizes. The topology server keeps pairs of the form $\langle (b/m) : \text{IP address}\rangle$. It is possible for the topology server to expose a user by providing false information (reporting a group is large when it is in fact not). For extra security, the topology information can be replicated at l different topology servers. A user would only consider the minimum group size reported by all topology servers; this way, a user can withstand up to $l-1$ colluding malicious topology servers. Similar techniques can be used to handle malicious topology servers who return a value smaller than the actual group size (to make the communication inefficient). Lastly, note that the topology servers can also be used to find users on a specified channel, which is needed when a new user joins a channel.

4.2 Migration up and down $\mathcal{L}$

Suppose A is connected to (b/m), and initially $\psi_A \leq |E\{(b/m)\}| \leq \eta_A$. As users join and leave the system, it is possible for the channel (b/m) to violate A's security or efficiency parameter. If A's efficiency parameter is violated, A can migrate to group $(b/m+1)$.

Unfortunately, A does *not* re-gain any security by migrating "up" $\mathcal{L}$ (i.e. by decreasing m) since an intersection attack fixes S_A to the size of the smallest channel that A was part of since it initially joined. Thus, in common use, we assume users do not leave once they join and if a group becomes too small, all remaining users have to recreate a new hierarchy. Note that they must then use a new communication key, since a passive observer along with a colluding receiver can mount an intersection attack.

4.3 The Network Abstraction and Processing Requirements

In this section we describe the precise networking requirements of $\mathcal{P}^5$. It was our design goal for $\mathcal{P}^5$ to be easily implementable using the current Internet protocols, as such our networking requirements are meager.

$\mathcal{P}^5$ requires the implementation of broadcast channels in which the source address cannot easily be determined. This can be efficiently implemented using a application-layer multicast protocol. The transport level requirements of $\mathcal{P}^5$ are minimal, and UDP would suffice as the transport protocol for $\mathcal{P}^5$ edges on the $\mathcal{P}$ topology. Lastly, note that during normal operation, $\mathcal{P}^5$ members only remain joined to the same set of channels; they only change channels if the overall security policy changes or if the group dynamic changes drastically. Thus, the signaling load due to $\mathcal{P}^5$ is low, and since the topologies within each channel is relatively static, the $\mathcal{P}^5$ tree can be optimized to map efficiently on to the underlying physical topology.

Host Requirements $\mathcal{P}^5$ requires state, processing, and link bandwidth at each host. We discuss these requirements in turn:

- Suppose member A communicates using channel (b/m) at depth d in $\mathcal{L}$. In order to communicate within the group, A may have to maintain next hop information about 2^d groups. For small d (e.g. $d < 16$), the state requirements are minimal. Note that unlike an IP router, A does not have to search its "routing table" for every packet; it only searches this table when it initiates a new data connection. Thus, this table can be maintained in secondary storage.

  If d is large (e.g. $d \geq 24$), it may not be feasible for a node to maintain information about all 2^d peer channels. In this case, A could maintain a small cache of advertisements, and if a new communication requires a channel that is not in the cache, A would have to wait until it hears another advertisement for that channel.
- There is a single public-key decryption for every packet that member A receives . Further, A has to encrypt every signal packet during communication. However, in general, A does not *have* to encrypt noise packets; it is only necessary that the adversary not be able to distinguish noise packets from signal packets. Thus, it is feasible for A to generate noise packets using a good local random number source.
- The broadcast nature of $\mathcal{P}^5$ requires individual group members to (potentially) devote more bandwidth for communication than pure unicast (or systems such as Crowds [8]). However, the extra bandwidth directly results in enhanced anonymity, and, obviously, individual users may choose a more bandwidth efficient channel if they are willing to sacrifice some anonymity. We analyze the actual bandwidth usage and how it scales with number of group members and different security parameters in Section 6.

5 The Random Channels Model

In this section, we present an analysis of the paths in a $\mathcal{P}^5$ network. Let n be the number of users in the system, and suppose there are k channels available in total. For some integer t (which is typically small—at most 5), each user u independently chooses t random channels without replacement: we will denote this random set of t channels by $S(u)$. Two central parameters for us will be: (i) d, the maximum "hop-count" between any 2 users, which is the maximum communication distance between any two users, and (ii) L_{min} and L_{max}, the minimum and maximum load (number of users) on any channel. Note that L_{min} and L_{max} are different than the security parameters as they count only the set of users local to a channel, while the security parameters count the set of users in all channels that provide anonymity for a given user. We next discuss these two parameters.

The parameter d. We say that there is a path $u = u_0, u_1, u_2, \ldots, u_\ell = v$ between two users u and v, if for each i, users u_i and u_{i+1} choose a channel in common. The *length* of such a path is defined to be ℓ, and the minimum length of any path between u and v is the distance or hop-count between u and v; if there is no such path, then this hop-count is defined to be ∞. We are interested in bounding d, the maximum hop-count between any two users.

The load parameters L_{min} and L_{max}. Define the load on a channel to be the number of users who chose it among their t random channels; let L_{min} and L_{max} respectively be the minimum and maximum loads on any channel. A lower bound on the former is needed to guarantee the security of the system, and an upper bound on the latter is required to show that the bandwidth overhead is not significant.

Given the above discussion, our basic goals will be as follows. Clearly, we simultaneously want "small" d, a value of L_{min} that is "not too small", and L_{max} being "not too high". It is easy to see that the expected load on any given channel is exactly $t \cdot n/k$. Thus, n/k is a natural parameter to study our system with. So, we will consider scenarios where k is constrained to be at most some given value K, and n/k is required to be at least some given value λ. So, the primary parameters are K, t, and λ. Given these, and for any choice of (n, k) for which $k \leq K$ and $n/k \geq \lambda$, we aim to show that the system has the above-sketched satisfactory properties w.r.t. d, L_{min}, and L_{max}.

More concretely, we will proceed as follows. Fix $\epsilon = 0.3$, say. Let $\mathcal{A}_s$ denote the desirable event that "(i) all the channel loads are within $1 \pm \epsilon$ of the expected value $t\lambda$, and (ii) $d \leq s$". We derive the following sufficient conditions for $\mathcal{A}_s$ to hold (for $s = 2, 3$) with a probability of at least $1 - 10^{-4}$:

1. $s = 2$: two sufficient conditions are

   (P1) $t = 3$, $\lambda \geq 100$, and $K \leq 100$; or

   (P2) $t = 4$, $\lambda \geq 80$, and $K \leq 300$.

2. $s = 3$: two sufficient conditions are

   (P3) $t = 3$, $\lambda \geq 100$, and $K \leq 150$; or

   (P4) $t = 4$, $\lambda \geq 80$, and $K \leq 700$.

We now prove that these conditions are indeed sufficient, in the rest of this section.

5.1 Analysis Approach

In our analysis, we will frequently use the union bound or Boole's inequality: $\Pr[E_1 \vee E_2 \vee \ldots \vee E_s] \leq \sum_{i=1}^{s} \Pr[E_i]$.

The parameters L_{min} and L_{max} are much more tractable than d, so we handle them first. Let $\exp(x)$ denote e^x. It is an easy consequence of large-deviations bounds such as the Chernoff-Hoeffding bounds [3, 7] that for any given channel c and any parameter $\epsilon \in [0, 1]$, its load $L(c)$ satisfies:

$$\begin{aligned}
&\Pr[L(c) \notin [(1-\epsilon) \cdot tn/k,\ (1+\epsilon) \cdot tn/k]] \\
&\quad \leq \quad \exp(-t\epsilon^2/2 \cdot (n/k)) + \\
&\quad\quad\quad \exp(-t(\epsilon^2/2 - \epsilon^3/6) \cdot (n/k)) \\
&\quad \leq \quad \exp(-t\lambda\epsilon^2/2) + \exp(-t\lambda(\epsilon^2/2 - \epsilon^3/6)).
\end{aligned}$$

Now, a simple application of the union bound yields

$$\begin{aligned}
&\Pr[(L_{min} < (1-\epsilon) \cdot tn/k) \bigvee (L_{max} > (1+\epsilon) \cdot tn/k)] \\
&\leq \quad K \cdot \exp(-t\lambda\frac{\epsilon^2}{2}) \ + K \cdot \exp(-t\lambda(\frac{\epsilon^2}{2} - \frac{\epsilon^3}{6}))] \quad (1)
\end{aligned}$$

We next turn to bounding d. We cannot directly draw on the rich random graphs literature, since we are working here with a certain model of random hypergraphs with possibly repeated hyperedges. We next study the two requirements of most interest: $d \leq 2$ and $d \leq 3$. As can be expected, the second case involves more work than the first. Our basic plan is as follows. Lemma 5.1 gives an upper-bound on $\Pr[d > 2]$, and Lemma 5.2 gives an upper-bound on $\Pr[d > 3]$. Then, letting $\overline{\mathcal{E}}$ denote the complement of event $\mathcal{E}$, we see by the union bound that $\Pr[\overline{\mathcal{A}_2}]$ is upper-bounded by the sum of (1) and the probability bound given by Lemma 5.1. Similarly, $\Pr[\overline{\mathcal{A}_3}]$ is upper-bounded by the sum of (1) and the bound given by Lemma 5.2. We shall do this "putting together" in Section 5.4.

5.2 The Requirement $d \leq 2$

Here, we want sufficient conditions for "$d \leq 2$" to hold with high probability. In other words, we want to show that for any two users, there is a path of length at most 2 between them. To do so, we fix distinct users u and v, and upper-bound the probability p that there is no path of length 2 between them; then, by the union bound, the probability of $d > 2$ is at most $\binom{n}{2} \cdot p$, since there are only $\binom{n}{2}$ choices for the unordered pair (u, v). Thus, we need to show that p is negligible in comparison with $(\binom{n}{2})^{-1}$, which we proceed to do now.

Our plan is to condition on the values of $S(u)$ and $S(v)$. For each such choice, we will upper-bound the probability that there is no user (among the remaining $(n-2)$) who chose a channel that intersects both $S(u)$ and $S(v)$. Then, the maximum such probability is an upper-bound on p. Fix $S(u)$ and $S(v)$. If $S(u) \cap S(v) \neq \emptyset$, then u and v are at distance 1; so suppose $S(u) \cap S(v) = \emptyset$. In particular, we may assume that $k \geq 2t$.

Consider any other user w. What is the probability of w's random choice $S(w)$ intersecting $S(u)$ and $S(v)$? The total number of possible choices for w is $\binom{k}{t}$. The number of possible intersection patterns can be counted as follows. Suppose $|S(w) \cap S(u)| = i$ and $|S(w) \cap S(v)| = j$, where $i, j \geq 1$ and $i + j \leq t$. The remaining $t - (i + j)$ elements of $S(w)$ are selected at random from outside $S(u) \cup S(v)$. Thus,

$$\Pr[(S(w) \cap S(u) = \emptyset) \bigvee (S(w) \cap S(v) = \emptyset)] = 1 - f(k, t),$$

where

$$f(k, t) = \frac{\sum_{i,j \geq 1;\ i+j \leq t} \binom{t}{i}\binom{t}{j}\binom{k-2t}{t-i-j}}{\binom{k}{t}}.$$

Thus, since different users w make their random choices independently, we get that $p \leq (1 - f(k,t))^{n-2} \leq \exp(-(n-2)f(k,t))$. Thus, as discussed above, a union bound yields

$$\Pr[d > 2] \leq (n^2/2) \cdot \exp(-(n-2)f(k,t)). \qquad (2)$$

In order to see what this bound says for various concrete values of our parameters K, λ, t, we develop:

Lemma 5.1 *Suppose* $\lambda \geq 2K/(t^3(t-1))$ *and* $K \geq 2^{t+2}$. *Then,* $\Pr[d > 2] \leq ((Ke\lambda)^2/2) \cdot \exp(-0.8t^3(t-1)\lambda/K)$.

Proof. **(Sketch.)** Since $f(k,t) \geq t^2\binom{k-2t}{t-2}/\binom{k}{t}$, bound (2) shows that

$$\Pr[d > 2] \leq (n^2e^2/2) \cdot \exp\left(-nt^2\binom{k-2t}{t-2}/\binom{k}{t}\right).$$

We can now do a calculation to show that subject to our constraints (which includes the constraint that $k \geq 2t$), this bound is maximized when $k = K$ and $n = \lambda K$. Further simplification then leads to the bound of the lemma.

□

5.3 The Requirement $d \leq 3$

We now adopt a different approach to get an upper-bound on $\Pr[d > 4]$. Let $\mathcal{C}$ denote our set of k channels. For a set $A \subseteq \mathcal{C}$ with $|A| = t$, define $Y(A)$ to be the set of all $a \in (\mathcal{C} - A)$ for which the following holds:

$$\exists x: \ (a \in S(x)) \bigwedge (S(x) \cap A \neq \emptyset).$$

In other words, $Y(A)$ is the set of channels that lie outside of A, but which lie in some set $S(x)$ that intersects A. Thus, we need to show that with high probability, at least one of the following four conditions holds for each pair of users u and v: (i) $S(u) \cap S(v) \neq \emptyset$; (ii) $S(u) \cap Y(S(v)) \neq \emptyset$; (iii) $Y(S(u)) \cap S(v) \neq \emptyset$; or (iv) $Y(S(u)) \cap Y(S(v)) \neq \emptyset$. To do so, we will instead show that with high probability, *all* $A \subseteq \mathcal{C}$ with $|A| = t$ will have $|Y(A)| > s$, where $s = \lfloor (k-2t)/2 \rfloor$. It can be verified that this implies that at least one of the conditions (i), (ii), (iii) and (iv) will hold for all u, v.

Fix $A \subseteq \mathcal{C}$ such that $|A| = t$. Let us bound $\Pr[|Y(A)| \leq s]$. Now, for any fixed $B \subseteq (\mathcal{C} - A)$ such that $|B| = k - t - s = \lceil k/2 \rceil$, a calculation can be used to show that

$$\Pr[Y(A) \cap B = \emptyset] \leq (\exp(-t^2/k) + 2^{-t})^{n-1}. \quad (3)$$

We summarize with

Lemma 5.2 *Suppose* $\lambda \geq 2K/(t^3(t-1))$ *and* $K \geq 2^{t+2}$. *Then,* $\Pr[d > 3] \leq 2^{K+2} \cdot K^t \cdot \exp(-t^2\lambda/2)$.

Proof. **(Sketch.)** A union bound using (3) yields

$$\begin{aligned} \Pr[d > 3] &\leq \Pr[\exists A : \ |Y(A)| \leq s] \\ &\leq \binom{k}{t} \cdot \binom{k-t}{\lceil k/2 \rceil} \cdot (\exp(-t^2/k) + 2^{-t})^{n-1} \\ &\leq 2^k \cdot k^t \cdot (\exp(-t^2/k) + 2^{-t})^{n-1}. \end{aligned}$$

A calculation shows that subject to our constraints, this bound is maximized when $k = K$ and $n = \lambda K$. Further simplification completes the proof. □

5.4 Putting It Together

Now, as described at the end of Section 5.1, we just do routine calculations to verify the following. First, if $s = 2$ and any one of (P1) and (P2) holds, then the sum of (1) and the probability bound given by Lemma 5.1 is at most 10^{-4}. Similarly, if $s = 3$ and any one of (P3) and (P4) holds, then the sum of (1) and the probability bound given by Lemma 5.2 is at most 10^{-4}. This concludes our proof sketch about these sufficient conditions for $\mathcal{A}_2$ and $\mathcal{A}_3$ respectively to hold with high probability.

6 Simulation Results

In this section, we present results from a packet-level $\mathcal{P}^5$ simulator. Our simulator is written in C, and can simulate the entire $\mathcal{P}^5$ protocol with thousands of participants. We designed and implemented five basic experiments:

- Measure system performance as the number of participants increase; specifically, we measure the end-to-end bandwidth, latency, and packet drop rates as the number of users in the system is increased
- Measure the effect of the security and efficiency parameter on communication efficiency
- Estimate the amount of time it takes $\mathcal{P}^5$ systems of a given number of participants to converge (i.e. how long does it take a user to find a channel that satisfies their security and efficiency constraints)
- Measure the effects of different noise generation rates and queuing disciplines
- Measure how the system behaves when increasing numbers of nodes engage in end-to-end communication

Simulation Methodology For each experiment, we generated a random physical topology. We did not model different propagation delays between pairs of nodes; instead, we assumed unit propagation delay between any two nodes. Since all inter-node latencies are the same, our simulation proceeds using a synchronous clock. At every tick, all packets sent from every node is received at their destination.

We assume an unbounded input queue length and a bounded output queue. All packets received at a node at a given time step are processed. Some of these packets may be queued at appropriate output queues, and a subset of them may be delivered locally. Next, each node generates a set of outgoing packets and enqueues these on the output queues. We then impose the output queue limit and according to the queuing discipline, discard packets if any output queue is larger than its maximum specified size. Note that during the discard phase, the node does not discriminate whether it is dropping its own packets or packets from some other node. All remaining packets at an output queue are delivered in the next time step to the next hop node.

Since all packets queued at a node are delivered at the next time step, the output queue size serves as a measure of both the processing and bandwidth requirements at a node. We instrumented the simulator to record the end-to-end latencies (number of simulator ticks), drop rates and bandwidth, end-to-end hop counts, number of channel crossings, and convergence times. In the rest of this section, we report results from individual experiments.

6.1 Scalability

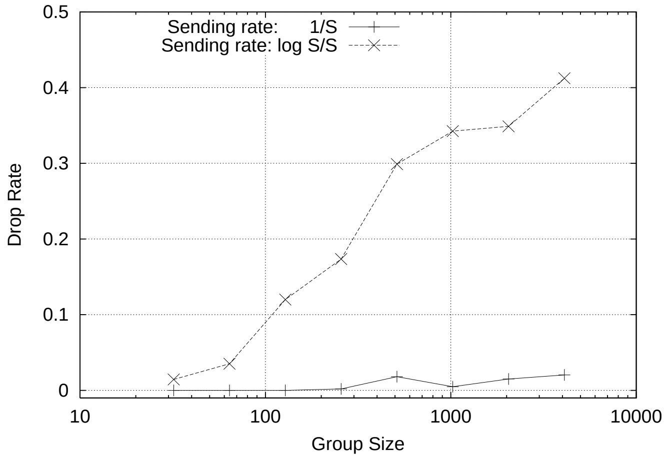

Figure 2: Loss rate vs. number of users

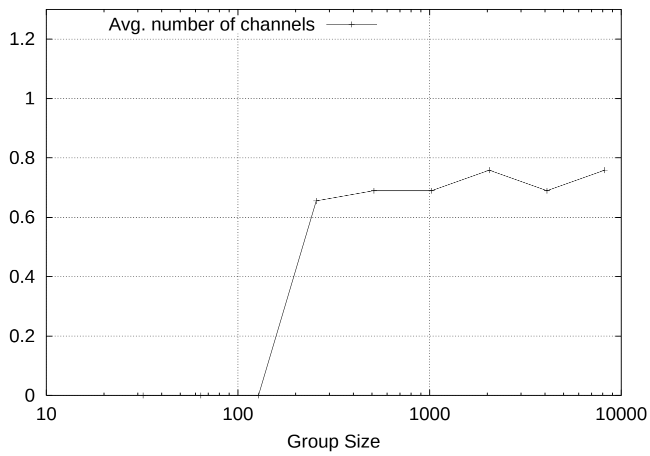

Figure 3: Average number of channels

In Figure 2, we plot the end-to-end loss rate as the number of users in the system is increased. In each case, there is only one pair of communicating nodes; all other nodes only send noise packets. For each group size, we chose ten different random seeds and created ten different topologies. For each topology, we chose three different sender-receiver pairs. Each point on Figure 2 is an average of all these runs (24 for each topology size). Unless otherwise noted, we choose the values of security parameters as $\psi = 100$, and $\eta = 300$, and implement non-uniform queuing. All nodes in the system were connected to two channels, i.e. they had one communication key and one routing key.

There are two different curves, each corresponding to two different sending rates. In the "Sending Rate=1/S" case, each node generates a packet at each time stop with probability $1/S$, where S is the size of its current broadcast group. Analogously, in the "Sending Rate=log S/S" case, each node generates a packet at each time stop with probability $\log S/S$. For both cases, the queue sizes at each node were very small: $\max\{10, \log S\}$. From the plot, it is clear that the $1/S$ sending rate can be sustained in the system, and almost no signal (or noise) packets are lost. However, as the sending rate is increased, the queues in the system are saturated, and drop rates increase with group size.

In Figure 3, we plot the average number of channels that the signal packets have to cross in order to reach their destination. Note that in this case, each user only connects to 2 channels. As predicted by the analysis in Section 5, the average channel level hop count is very small, and is less than 1 for all our runs. The average end-to-end hop count in these runs were ~ 13. The worst case inter-channel distance that we encountered in these runs was 2. This occurred in 4 out of 6000 signal packets sent.

Analysis In our simulations, the average size of each local broadcast group in these experiments is approximately 150, and the average queue size is around the minimum (10). In the worst case, each queue is always full, and the nodes have to handle two full queues worth (20 packets) per tick. Each tick in our system corresponds to an end-to-end propagation delay. Assume that this delay is on average on the order of 100ms. Thus, these nodes would have to handle up to 200 packets per second. At 1000 bytes per second, this translates to 1.6 Mbps of bandwidth and the ability to handle two hundred 1000 byte public-key decryptions per second. The processing capability required is trivial compared to the power of current processors. The bandwidth required is somewhat more of a concern; however, note that individual users can always reduce their bandwidth requirement by migrating "lower" in the tree. The "Sending Rate=1" corresponds roughly to each node sending at 16Kbps (with essentially no packet loss). If nodes are willing to incur higher packet loss, then the sending rate can be much higher, e.g. for the 8192 user case, users can send at up to 200 Kbps if they are willing to handle upto 40% packet losses. Note that these losses accumulate over about 13 hops, which is the average end-to-end path length in these simulations.

Thus, in a 8192 node $\mathcal{P}^5$ network, any subset users can *anonymously* communicate at hundreds of kilobits per second (with relatively high packet losses), if they invest approximately 2 Mbps of bandwidth. Clearly, the loss rate is high compared to the communication media we are used to, but we have to remember that in $\mathcal{P}^5$, the user is gaining anonymity of at least 100 other users. In a pure broadcast system with 8192 users and these same parameters, the average end-to-end loss rate would be about $1 - 10^{-39}$; communication in this system would, essentially, be impossible. (In such a system, to achieve a 50% average end-to-end loss rate, the sending rate per user would have to be $\sim$ 1 packet/8 seconds. Of course, each user is also gaining anonymity from *all* other users in the system).

In Section 6.3, we discuss the effects of varying the sending rate while the bandwidth and group sizes are fixed.

6.2 Convergence Times

Group Size	No. of rounds	Security Parameter	No. of rounds
64	0	16	6
128	0	32	5
256	1	64	4
512	2	128	3
1024	3	256	2
2048	4	512	1
4096	5		
8192	6		

Table 1: Convergence times

In Table 1, we present the convergence times for $\mathcal{P}^5$ in our simulator. There are results from two different experiments in the Table; in the experiment we fixed the security parameters (at $\psi = 100$, $\eta = 300$), and varied the number of users. In the second experiment, we fixed the number of users at 1024, and varied the security parameter (ψ). In all cases, we used $\eta = 3\psi$.

In all experiments, all the users join simultaneously at time 0. Each user migrates down the $\mathcal{L}$ tree in rounds. Each round consists of 10 simulator ticks, and each user only makes a single migration decision in any one round. As expected, the convergence times increase as the number of users increase (or the ψ parameter is decreased) since each user *settles* "lower" down in $\mathcal{L}$. However, in all cases the number of rounds to converge is given by $\max\{0, \ \log N - \log \psi\}$.

6.3 Noise and Signal Generation

In Figure 4, we vary the sending rate while keeping all other parameter fixed. We monitor a single sender–receiver pair, and report the observed packet loss. We use the two base sending rates from Section 6.1, and linearly increase these rates by the rate multipliers plotted on the x-axis, while keeping the link bandwidths constant. As expected, the drop rates increase linearly with increases in sending rate. Interestingly, the non-uniform drop rates perform slightly better as the sending rates increase.

In Figure 5, we repeat the same experiment and vary the number of sender-receiver pairs in the system. The rest of the users still generate noise at the same rate ($\log S/S$). We plot the average drop rate across all of the sender-receiver pairs. As expected, the drop rate is not affected by the number of sender–receiver pairs, and thus, no extra information is divulged to a

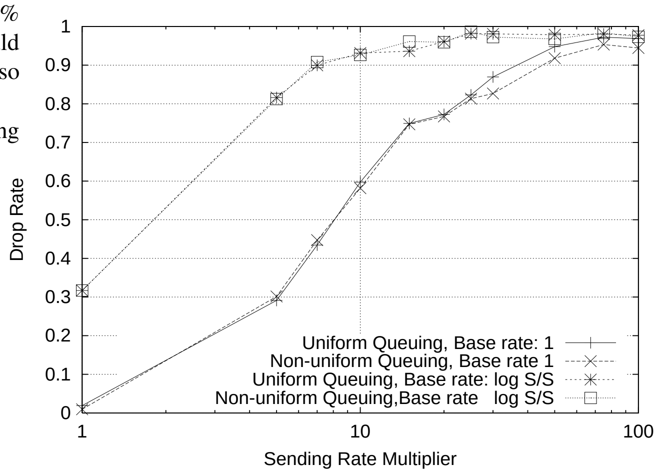

Figure 4: Loss rate vs. sending rate

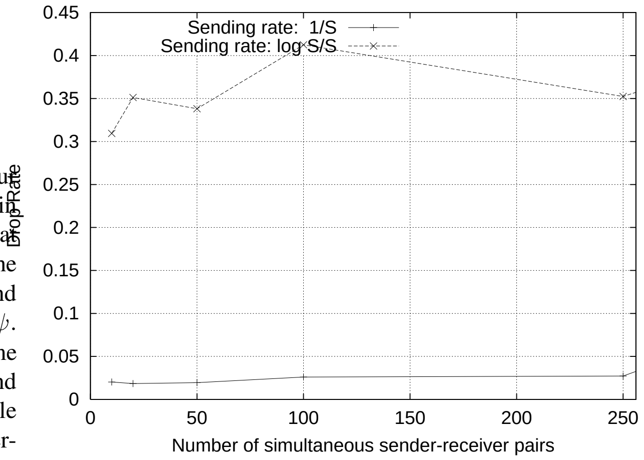

Figure 5: Loss rate vs. number of senders

passive observer when more people in the system communicate. We have also experimented with different values of ψ and η. As expected, the drop rate increases as users choose higher values of the security parameters, since they are mapped to larger broadcast groups.

7 Conclusions

We divide our conclusions for $\mathcal{P}^5$ in to two parts.

7.1 Observations

$\mathcal{P}^5$ is a protocol for anonymous communication over the Internet. $\mathcal{P}^5$ allows secure anonymous connections between a hierarchy of progressively smaller broadcast groups, and allows individual users to trade off anonymity for communication efficiency.

In developing $\mathcal{P}^5$, we found an interesting property relating communication latency, bandwidth usage, and anonymity. In general, we found it was easy to construct protocols that provided two out of these three properties, e.g., consider plain unicast communication: it provides low latency and high bandwidth usage, but does not provide anonymity. Now consider multicasting to a set (using per-source shortest path trees) in which the message is intended for only one member of the group. This solution provides low latency; however the bandwidth utility decreases as the anonymity and unlinkability increases. $\mathcal{P}^5$ has the interesting property that it allows individual users to trade-off these three properties on-line.

We designed $\mathcal{P}^5$ to be scalable and compatible with current Internet protocols. Our simulations show that $\mathcal{P}^5$ can scale to large groups, and our analysis shows that $\mathcal{P}^5$ will maintain its "short paths" property with very little extra overhead for extremely large groups. Our current work is to adapt lower overhead noise generation algorithms to further improve scalability; provide better reliability by considering more connected structures within individual groups; and to build a prototype for deployment around the Internet.

7.2 A Note on Ethics

There may be some questions about why a system like $\mathcal{P}^5$ is needed. Clearly, privacy over the Internet is an important and open issue, and $\mathcal{P}^5$ is a first step towards a truly scalable anonymous network layer over IP. There are a number of applications, e.g. anonymous web transactions and anonymous re-mailers, where sender- and receiver-privacy is all that is required. $\mathcal{P}^5$, however, also provides sender-receiver privacy, and like all technologies, this can be used in a malicious manner. We have decided to include sender-receiver privacy in $\mathcal{P}^5$ for the following reasons:

- We believe it is important to study these protocols, simply to learn what levels of anonymity are feasible over a public network such as the Internet.
- The protocol-steps in $\mathcal{P}^5$ that provide sender-receiver anonymity can be decoupled from the rest of the protocol, and $\mathcal{P}^5$ can be used in sender-, receiver-anonymity mode only. It is an orthogonal ethical (and possibly political) decision as to whether $\mathcal{P}^5$ should be implemented to provide sender-receiver anonymity.
- We describe an attack that can be mounted by an powerful active adversary, specifically an adversary who can inject packets on a arbitrary set of network links. $\mathcal{P}^5$ fails under such an attack.

Acknowledgments. We thank the referees for their helpful comments.

References

[1] David Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2), 1981.

[2] David Chaum. The Dining Cryptographers Problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[3] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. Annals of Mathematical Statistics, 1952.

[4] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *International Workshop on Design Issues in Anonymity and Unobservability, LNCS 2009*, 2001.

[5] Sholmi Dolev and Rafail Ostrovsky. *Xor-Trees for Efficient Anonymous Multicast Receiption*. Advances in Cryptography - CRYPTO 97, 1997.

[6] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42(2), February 1999.

[7] W. Hoeffding. Probability inequalities for sums of bounded random variables. American Statistical Association Journal, 58, 1963.

[8] Michael K. Reiter and Aviel D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[9] Clay Shields and Brian Neil Levine. A protocol for anonymous communication over the Internet. In *Proceedings of the 7th ACM Conference on Computer and Communications Security (CCS-00)*, pages 33–42, N.Y., November 1–4 2000. ACM Press.

[10] Michael Waidner and Birgit Pfitzmann. The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability. In J.-J. Quisquater and J. Vandewalle, editors, *Advances in Cryptology—EUROCRYPT 89*, volume 434 of *Lecture Notes in Computer Science*, page 690, April 1989.