

Hush: A Scalable And Efficient Anonymization Overlay

Abstract

Anonymous communication over an open network such as the Internet has long been recognized as a desirable feature, and there have been several proposals for building overlay networks for this purpose (Chaumian Networks [1], Onion Routing [2], Tarzan [3], Crowds [4], DC-Nets [5], Herbivore [6]). These networks are however not feasible in practice for one or more of the following reasons: poor scalability, low efficiency, and no long-term anonymity. This work describes Hush, an anonymization overlay, that addresses these issues using probabilistic forwarding schemes based on hash functions. We analyze the security properties of Hush to derive its anonymity guarantees against various classes of adversaries and compare it with existing anonymization overlays. We then further extend our analysis techniques to comment qualitatively and quantitatively on the capability and limitations of different classes of anonymization overlays. The scalability and efficiency of Hush make it a practical proposition for applications like anonymous web browsing and anonymizing peer-to-peer networks.

1 Introduction and Motivation

Anonymous communication refers to communication between two parties when at least one of the parties is unaware of the identity of the other. This is a highly desirable feature in the Internet to allow freedom of expression and to thwart monitoring and censorship. Anonymity is unfortunately not supported in the current Internet, since intermediary parties in the communication can aid one of the parties in discovering the identity of the other party.

Overlay networks consisting of interested end-hosts provide a mechanism for anonymous communication over an open network like the Internet. The design of secure anonymization overlays has been an active topic of research, but the search for an *scalable* and *efficient* anonymization overlay has proven elusive. Overlays based on relaying (forwarding through intermediary nodes) like Chaumian networks [1], Onion Routing [2], Tarzan [3], Crowds [4] are not scalable and are susceptible to the predecessor attack [7], while approaches based on the DC-Nets primitive [5] like Herbivore [6] are not efficient (especially at high utilization).

In this work, we propose Hush, a scalable and efficient overlay that uses probabilistic forwarding to provides anonymity to the sender. We address the scalability concern by developing new mechanisms for providing anonymity over a bounded-degree graph. We address efficiency by using relaying as a primitive and we also illustrate later that the anonymity versus efficiency tradeoff is acceptable. To defend against the predecessor attack, our mechanisms uses hash functions to represent forwarding rules, which makes it difficult for adversaries to breach the anonymity properties even in the long-term. We also use consistent hashing [8] to defend against an adversary that induces artificial churn (arrival and departure of nodes). Hush also does not require any form of public key distribution.

We compare Hush to existing schemes using a probability metric proposed in [4] and an entropy metric proposed in [9]. We characterize the anonymity guarantees provided by an overlay by analyzing its short-term and long-term anonymity against various kinds of adversaries. The former models an adversary who attempts to find the source of a single communication session, while the latter allows the adversary to observe the set of communication sessions initiated by the source over a long period of time.

Hush provides better short-term anonymity than existing anonymization overlays, when state restrictions are imposed and a public key infrastructure (PKI) is not available. It provides a long-term anonymity level of at least plausible deniability, which is better than all existing relaying-based anonymization overlays (however DC-Net based schemes provide superior long-term anonymity compared to Hush). Hush also provides greater efficiency compared to DC-Net based overlays under similar state restrictions. Hush is suitable for applications like anonymous web-browsing, and we also show how it can be incorporated into peer-to-peer networks to build applications like anonymous subscription networks, anonymous searching.

Our contributions in this work are three-fold. Firstly, in analyzing the anonymity of Hush, we characterize the tradeoff between state, performance, and anonymity in relaying based schemes. Our analysis techniques for this purpose may be of independent interest. Secondly, our hash based mechanisms can be used to thwart the predecessor attack in relaying schemes. Lastly, we provide a discussion of the capabilities and limitations of relaying and DC-Net based anonymization overlays. In particular, we prove that perfect anonymity is not possible in a complete graph in the asymptotic setting using Chaumian-like forwarding schemes, even with a public key infrastructure and perfect mixes (mixes on which no timing attacks are possible).

The organization of the rest of the paper is as follows. Section 2 explains our notions of a anonymization overlay, adversary model, and anonymity metrics. The shortcomings of existing anonymization overlays are discussed in Section 3. Hush's architecture is explained in Section 4 and the forwarding mechanism for Hush is designed and analyzed in Section 5. Numerical

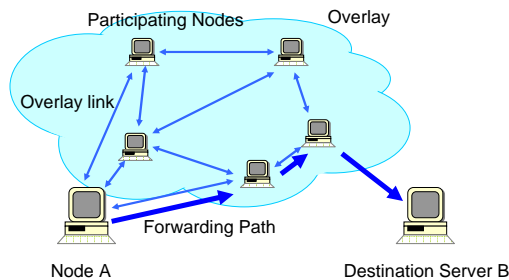


Figure 1: An Anonymization Overlay

results for the anonymity in Hush are presented in Section 6. We then place Hush in the context of existing schemes in Section 7, and in the context of peer-to-peer networks in Section 8. We finally summarize our contributions and present our conclusions in Section 9.

2 Definitions and Background

We provide the background for the rest of the paper in this section by discussing our notions of an anonymization overlay, the various kinds of adversaries that we model, and the anonymity metrics we use.

2.1 Framework of an Anonymization Overlay

An overlay network consists of a set of end-hosts that wish to obtain some functionality not available in the current Internet. An anonymization overlay (depicted in Figure 1) consists of end-hosts (also referred to as nodes) that forward packets for each other in order to achieve anonymous communication to other nodes in the Internet. In this figure, the *initiator* node A wishes to make an anonymous connection to the *destination server* B. It does so by choosing a *forwarding path* through the overlay and sending the data along this path. Note that the destination server need not support anonymous browsing and need not be a part of the overlay itself. A typical anonymization overlay consists of three main components:

- **Topology Maintenance:** These are the protocols used to construct and maintain the overlay network. This involves coordination among nodes in the overlay to maintain overlay links between them and to accommodate churn.
- **Forwarding Mechanism:** This consists of the rules used for forwarding data along the overlay. This includes the route selection procedure, route setup, data forwarding rules, route teardown etc. The anonymity guarantees offered by a system follow from these mechanisms.
- **Defenses against various forms of side-channel attacks:** These include defenses against a link-level adversary (exchanging cover traffic), data sanitation (removing identifying information such as cookies from web traffic) etc.

2.2 Adversary Model

We define our adversary model by characterizing different kinds of adversaries against an anonymization overlay. We refer to an adversary who attempts to influence the topology maintenance mechanism to gain advantage in some fashion, as a *topology-corrupting* adversary. An adversary who attacks the forwarding mechanism is the most commonly considered kind, and we refer to her as simply an adversary. We will not explore adversaries that attack the third component in detail, since those aspects of the overlay are mostly orthogonal to the topology maintenance and forwarding mechanism.

With respect to this work, we assume that all adversaries in the system collude with one another. We use the term adversary to refer to a single member of this collusion or to the collusion itself (the correct definition will be clear from context). We always assume that the destination server is corrupt, because that imposes the strict constraint that the destination is unaware of the source’s identity. One useful categorization of adversaries is based on how nodes are corrupted in the system. An *static* adversary controls a fixed set of nodes, while a *adaptive* adversary can corrupt new nodes at some finite rate (by using buffer overflow exploits etc). Another difference is between *passive* and *active* adversaries: the former can only observe traffic, while the latter can also inject traffic into the system. Also, we assume that the adversary cannot break cryptographic primitives, in particular, hash functions.

2.3 Anonymity Metrics

The anonymity metric we are interested in is *sender anonymity* [4] which is a measure of the anonymity level guaranteed to a sender. One such measure is the probability of identification (PoI) which is defined as follows. Let the source S initiate a set of sessions, and let the adversary employ her strategy to assign a probability to every node A in the system based on

her observations of this set of sessions. This probability is the adversary’s estimate of the probability that A is the source of these sessions. When the adversary is allowed access to a set of sessions collected over a long period of time, this probability determines the long-term anonymity level. When only a single session is available to the adversary, this probability defines the short-term anonymity level.

We use two metrics for characterizing the anonymity level: $PoI(S)$ is defined as the expected probability assigned to the *true* source S . Clearly, the lower the PoI, the stronger the anonymity provided by the system. Another recently proposed anonymity metric [9] is the expected *entropy* of the probability distribution inferred by the adversaries: the higher the entropy, the more “random” the distribution, and the higher the anonymity. An anonymization scheme is said to offer *plausible deniability* if the PoI is less than one (in the adversary’s view, there is a non-zero probability that another node is the initiator).

3 Related Work

We categorize existing anonymization overlays into two classes for ease of exposition. The first category includes relaying based networks like Chaumian networks, Crowds, Tarzan etc. The second category includes networks designed with scalability as a primary goal such as DC-Nets, Herbivore etc.

3.1 Relaying Based Networks

Chaumian mixes is among the first proposals for relaying based anonymization networks, which however rely on the assumption that every node knows the public key of every other node in the system. This was later leveraged for Transport Control Protocol (TCP) relaying in Onion Routing. These mixes do not scale well and also require costly public key operations. Tarzan is a peer to peer solution that does not require a public key infrastructure, but requires knowledge of all nodes in the network. Crowds is based on probabilistic forwarding and does not require any public key operations or distribution, however it requires complete state as well. Thus, scalability is a clear concern in all of the above anonymization networks since per-node state is expensive to maintain in large scale networks. The forwarding mechanism in these systems is susceptible to the predecessor attack [7] and they have poor long-term anonymity. In Chaumian mixes, it is assumed that the sender knows the public key of the destination, therefore the adversary cannot use packet contents to deduce that two requests are from the same source. This however is infeasible in an anonymization overlay for browsing and many applications: maintaining a public key database for every possible destination server is impractical. We will now discuss Crowds in detail, since our work is closest in spirit to Crowds.

3.1.1 Crowds

Crowds is specifically designed for web browsing and provides sender anonymity. The topology maintenance protocol in Crowds is simple: a designated node (*blender*) maintains a list of member nodes (*jondos*) in the overlay, and a joining node retrieves this list from the blender. It then establishes overlay paths to all other nodes. The forwarding mechanism is as follows: when a node wishes to establish an anonymous connection to a server, it forwards the request directly to the server with a probability $(1 - p_f)$ (p_f is the forwarding probability). With probability p_f , it relays the request to one of its neighbors (chosen uniformly at random). The neighbor now repeats the same procedure, and this process continues until the request is forwarded to the destination.

Crowds suffers from the disadvantages highlighted earlier: scalability and poor long-term anonymity. Scalability is clearly a concern since every node maintain state about all nodes. We will now describe the predecessor attack in [7] against long-term anonymity. Even a single adversarial node in the system can identify the source of a sufficiently large series of sessions, providing she can identify that these requests are initiated by the same source (which is possible to do if there is no end-to-end encryption between the destination server and the originating node). This can be done in a variety of ways in practice: for example, an adversary can use some identifying information in the data (username, cookie data) etc. Even if no such application data is available, an adversary who observes multiple back-to-back requests to the same destination can heuristically assume that they are both initiated by the same source. Given that the adversary has this capability, the attack is as follows: suppose that the source initiates a series of sessions and the adversary intercepts some of them. If she can identify that these sessions are in fact initiated by the same node, then given enough time, the source can be exposed. The method is to simply find that node which occurred the maximum number of times as the previous hop to the adversary. This attack works in Chaumian Networks, Crowds, and Onion Routing.

3.2 Scalable Anonymization Overlays

Existing schemes that have tried to achieve scalability suffer from one of two drawbacks: lack of analytical proofs for the anonymity or poor efficiency. A recent proposal for a scalable anonymization network is Herbivore which is based on DC-Nets. DC-Nets allow a set of k participants in a clique that have shared secrets with each other, to communicate anonymously amongst

themselves (it provides both sender and receiver anonymity). This primitive is extended in Herbivore by organizing a topology that divides participating nodes into cliques. In both DC-Nets and Herbivore, every message transmission involves coordination between all the nodes in a clique, since the protocol proceeds in rounds. The efficiency of such a network is characterized as the number of anonymous bits as a fraction of the number of bits transmitted. Even with mechanisms for reservation as in Herbivore, efficiency decreases as the number of simultaneous senders within the clique increases. Thus, anonymous bandwidth available per participant is low in practice.

P5 [10] is recently proposed broadcast based protocol which requires that the destination server be a participant in the overlay and its public key is known to the sender, since this precludes communication to arbitrary end-hosts. Freenet [11] is a scalable solution, however its anonymity guarantees have not been analyzed so far. Another drawback is that it combines the notions of anonymity and search, which makes it unsuitable for pure anonymization networks. As we will show later, our approach can be applied for both purposes with provable guarantees. Recent work on mixes with restricted routes [12] uses expander graphs and derives its security properties. However, they do not consider adversaries in the overlay: they assume that the only adversary is the destination server. Also, it is not clear how to maintain the expander graph property in a dynamic network with joins and leaves.

4 Architecture of Hush

This section describes the two main architectural components of Hush: topology maintenance and data forwarding. We will also discuss attacks on these components and mechanisms to defend against them. The requirements for these protocols is scalability and efficiency. The main difference from existing proposals is that it uses an un-structured network topology, which has lower overhead and easier to maintain under churn. This also allows flexibility for each node to choose the number of neighbors on its own.

4.1 Topology Maintenance

Since one of Hush's goals is scalability, the topology maintenance in Hush is based on popular un-structured peer to peer overlays like Gnutella. Also, these overlays can handle churn (arrival and departure of nodes) with minimal overhead and this is ideal for maintaining the topology in dynamic environments.

Nodes use a gossiping protocol to learn about new nodes in the system and to gain new neighbors. Each node maintains a list of overlay neighbors that it freely dispenses to any other requesting node. Since each node might have its own limitations (bandwidth etc) on how many neighbors it can support, we use an explicit *Request Neighbor* message that a node sends to a prospective neighbor. If the latter is willing to take on an additional neighbor, then it responds with a *Neighbor Ack* message that indicates that an overlay link is now established. This approach is scalable because each node needs to only maintain state about a limited number of neighbors of its choice. In Hush, each node specifies its neighbor requirements with two parameters: *min-degree* and *max-degree*. Min-degree refers to the number of overlay neighbors it would like to maintain, and Max-degree is the maximum number of overlay neighbors it can support. For robustness, each node also maintains a set of candidate neighbors so that it can use them as backups, if some of its current neighbors fail. Each node tries to acquire at least Min-degree neighbors, and will accept at most Max-degree neighbors. This style of gossiping protocol is used in unstructured overlays, and is known to have low overhead. This mechanism also offers flexibility to each node to its neighbor set as per its anonymity requirements and bandwidth limitations.

4.2 Data Forwarding Mechanism

The gossiping mechanism thus establishes an overlay network where each node has a bi-directional link to its neighbors. We now discuss the forwarding mechanism that routes traffic over this network. These mechanisms are based mostly on existing work. In schemes like Onion Routing, the initiator has a complete knowledge of the graph, and thus, it can specify the entire relaying path without any information from other nodes. In Hush however, such an approach is ruled out due to scalability constraints: each node has a knowledge of only its immediate neighbors. Therefore, Hush uses an approach similar to Crowds: each hop makes a local decision on the next hop. Unlike Crowds however, in Hush, each hop chooses the next hop from its *neighbors*, not all the nodes in the system. Thus, when the source needs to set up an anonymous path, it chooses a next hop according to some forwarding mechanism, sets up the path with the next hop, which repeats the process. The setup phase involves negotiating flow identifiers, setting up session encryption keys etc. This process repeats until some node decides to forward the request to the destination. Soft state is maintained in order to send data over the reverse path. The exact mechanism used to determine the next hop is one of the most important factors that determine the anonymity guarantees of the system, and we discuss this mechanism in detail later.

4.3 Dealing with Adversaries

We now consider adversaries that attack the topology maintenance protocol and the data forwarding mechanism. One possible aim of topology corrupting adversaries is to launch target attacks against a particular node, by increasing the fraction of adversaries in the victim’s neighbor table. This provides the adversary with greater information about packets relayed by the victim. The gossiping mechanism is resistant to these kind of attacks, since choices of a neighbor are made randomly. However, an adversary can in effect, generate a number of fake nodes, and use this to increase its presence in neighbor tables on honest node. This is the well-known Sybil attack ([13]) and there is no known way to defend against this attack in a decentralized environment. One way to limit the adversary’s capability to create fake identities, is to identify a node by its IP address of the node along with the port number. This identifier can be verified by sending a nonce at the specified location: only if the node can return the nonce, it is allocated that identifier. This limits the adversary’s capability to the number of IP addresses (and port numbers) she owns. Another possible way to limit the adversary is for a node to choose its neighbors with a diverse set of IP prefixes, so that an adversary is limited by the number of distinct prefixes she owns (this is suggested in Tarzan).

An adversary can also launch a denial of service attack by simply dropping packets. This practice can be discouraged using the following scheme: each node can maintain a reputation value for each of its neighbors, and forward requests only to those neighbors who have a positive reputation value. Schemes for distributed reputation management (as in [14]) require distribution of public keys, and are therefore are not applicable in this scenario. Also, in our setting, there is an incentive for an adversary to forward requests: it reduces the number of rounds for the predecessor attack and for adaptive adversary attacks. Thus, if the aim of the adversary is to discover the identity of the source, her strategy would be to forward all requests, so that her reputation value goes up. Therefore, this simple scheme where each node maintains a reputation value for its neighbors, suffices in Hush. Note that a newly added neighbor will be given a small credit so that she can improve his reputation value by forwarding packets.

4.4 Orthogonal Issues

There are other orthogonal issues here that have been well-investigated in literature and are beyond the scope of this work. The forwarding mechanism can be implemented at the application, transport, or network layer: a new relaying path can be selected for every packet, every new TCP session, or every new application session. There are several trade-offs involved in this decision: no state is required if implemented at the network layer, but TCP performance suffers due to packet re-ordering. TCP level forwarding is likely to be more useful because of the support for duplex traffic, but is suitable only for short-lived flows like HTTP. We leave open the choice of the layer at which Hush is implemented, and will henceforth simply refer to a “request” being forwarded (for a better understanding of such issues in relaying TCP traffic, refer ToR [15], the second-generation onion router specifically designed for TCP).

We assume cover traffic is exchanged between a node and all of its neighbors to defend against link-level adversaries. The cover traffic required is manageable in Hush since each node is free to choose its number of neighbors. Overlay neighbors can also encrypt the traffic between them by a key exchange during overlay link establishment. If timing attacks are important, then each node can implement mixing (an example of a possible mix is [16]). We also assume that application-level sanitization is done at the source. Also note that if a path break occurs in the relaying path, the node upstream of the break is responsible for establishing a new path (a standard mechanism to avoid attacks by adversaries joining and leaving frequently). Since public key distribution is required for content anonymity, Hush does not support content anonymity for scalability concerns (a node needs to maintain a public key for every node in the network).

5 Forwarding Mechanism

In this section, we specify how the choice of the next hop is made at every node during the forwarding phase. Our approach to the design of the forwarding scheme is as follows: we construct a series of forwarding schemes, where each one is an improvement over the earlier ones in some security aspect. This organization was chosen so as to provide an insight into the design. The important requirements for the forwarding scheme are scalability, efficiency, long-term anonymity, and resistance to topology corrupting and dynamic adversaries.

We first address the scalability concern by a Crowds-like forwarding scheme which however has no long-term anonymity. We then discuss the preferential forwarding scheme which provides long-term anonymity against a passive adversary. Finally, we discuss two hash-based schemes that provide long-term anonymity against an active adversary as well.

5.1 Crowds-like Forwarding

We design a scalable forwarding mechanism by modifying the crowds forwarding scheme to operate over an *incomplete* graph. Our modification is as follows: a node forwards a request to the destination with probability $(1 - p_f)$, otherwise it forwards it

to a randomly chosen *neighbor* (in Crowds, it can forward to any node in the system). Note that Crowds's forwarding rule has been analyzed only on a complete graph, and not on a bounded degree graph as in Hush.

We will analyze the security of this forwarding scheme by characterizing the adversary strategy in two steps: that invoked to identify the originator of a single request, and that for a series of requests that are known to be initiated by the same source. As discussed before, we refer to the anonymity of the source in the first scenario as *short-term* anonymity and in the second as *long-term* anonymity. We assume that the adversary has complete knowledge of the topology (this assumption is necessary in our overlay since gossiping reveals the topology to adversaries).

5.1.1 Short-Term Anonymity

We model the overlay network as a directed graph $G = (V, E)$ where the vertices correspond to the nodes, and the edges to overlay links. This graph G includes a node D for the destination, and every other node has an edge incident on D (this is because every node can choose to forward to the destination directly). Let the adjacency matrix be T and let $A \subset V$ be the set of adversaries in the system. Then, the modified forwarding rule is as follows: node S forwards the request to the destination server with probability $(1 - p_f)$ and to each of its neighbors with probability $p_f/d(S)$ where $d(S)$ is the degree of node S .

Suppose a request initiated from a source S is intercepted by the adversary and let A_{int} be the first adversary on the path. Due to the memoryless probabilistic forwarding, any other adversaries on the path do not provide any additional information about the source. Note that A_{int} could be D , the destination server itself, if the request was not intercepted by any other adversary enroute. In this scenario, the adversary is aware of only the previous hop R from which she received the request. In order to assign probabilities to each of the nodes X in $(V - A)$, she needs to calculate the probability $P(X|R) = \Pr[X \text{ is initiator} | R \text{ is the previous hop}]$. Note that this probability is conditioned on all the information that the adversary has about the source, and thus, this is the best possible strategy. Any other information that the adversary may have, say, the number of adversaries on the path, does not reveal any information about the source (this can be proved easily in the information theoretic sense from the memoryless nature of the forwarding scheme).

This probability computation is simple in Crowds due to the symmetry of the complete graph, but it requires a slightly more complicated analysis in Hush. Firstly, note that since we assume that the destination is an adversary, every request originated by the source is intercepted by some adversary. To calculate $P(X|R)$, the adversary needs to estimate the fraction $P_{X,R}$ of requests that are originated by X and relayed to R . We present the pseudo-code for the adversary strategy in Algorithm 1 below:

Algorithm 1 Short-term Adversary Strategy

- 1: Given Graph $G = (V, E)$, the set of adversaries A , and an adversary A_{int} which received the request from previous hop R .
 - 2: Convert G to $G' = (V', E')$ as follows: $V' = V \setminus A$, E' is the set of edges induced by V' in G .
 - 3: **for** every neighbor i of A_{int} in G **do**
 - 4: $V' = V' \cup \{A_{(int,i)}\}$, $E' = E' \cup \{(i, A_{(int,i)})\}$
 - 5: **end for**
 - 6: Let T' be adjacency matrix of G' . Compute S^1 , the one-hop forwarding probability matrix, as follows: $S_{i,j} = T_{i,j} * f_{i,j}$ where $f_{i,j} = (1 - p_f)$ if $(A_{int} = D) \wedge (j = A_{int,i})$ and $p_f/d_G(i)$ otherwise ($d_G(i)$ is the degree of i in G).
 - 7: Compute $P = S^1 + S^2 + S^3 + \dots = S^1(I - S^1)^{-1}$ (assuming S^1 is not singular).
 - 8: $P(i|R) = \frac{P_{i,A_{(int,R)}}}{\sum_{j \in (V-A)} P_{j,A_{(int,R)}}$.
-

The reasoning behind this algorithm is as follows: Step 2 removes the adversaries from the graph, and Steps 3-5 modify the graph such that every honest node i that is a neighbor of A_{int} in G , now has an outgoing edge to a newly added node $A_{int,i}$ (which has no outgoing edges or any other incoming edges). This step implies that $A_{int,i}$ is a dead-end, a fact we will use later. Step 6 computes the transition matrix of the modified graph, and Step 7 computes the transitive closure. Step 8 finally uses the probabilities computed by the transitive closure to compute the $P(i|R)$. The correctness of the algorithm is proved in the following lemma:

Lemma 1. *Algorithm 1 computes $P(i|R) \forall i \in (V - A)$ (given that the adversary has no prior information on the source of the intercepted request)*

Proof. We first compute $P(i|R)$, the probability that i originated the request which was relayed through R .

$$P(i|R) = \frac{\Pr[i \text{ relayed to } R]}{\Pr[R \text{ relayed a request}]} = \frac{P_O(i)P_{i,R}}{\sum_{j \in (V-A)} P_O(j)P_{j,R}} = \frac{P_O(i)P_{i,A_{(int,R)}}}{\sum_{j \in (V-A)} P_O(j)P_{j,A_{(int,R)}}} \quad (1)$$

where $P_O(i)$ is the apriori probability that i originated the request, and $P_{i,R}$ is the probability that a request originated by i is relayed to R . The last step follows since $P_{i,A_{(int,R)}} = cP_{i,R}$ where c is a constant that depends only on R . Since the adversary

has no apriori knowledge of $P_O(i)$, she uses the uniform distribution $P_O: P_O(i) = P_O(j) \forall i, j$. Now it remains to compute $P_{i,A(int,R)}$. Define $L_{i,j}$ as the probability that a request originated by i passes through j . Note, that $P_{i,A(int,R)} = L_{i,A(int,R)}$ since $A(int,R)$ can occur on a relaying path only once (it has no outgoing edges). Now, $L_{i,j}$ can be written as:

$$L_{i,j} = \sum_{k=1}^{\infty} L_{i,j,k} = \sum_{k=1}^{\infty} (S^k)_{i,j}$$

where $L_{i,j,k}$ is the probability that j occurs as the k^{th} node on the relaying path from i , S is the forwarding matrix as defined in Step 7, and $(S^k)_{i,j}$ is the (i, j) element of the matrix S^k . S has been defined in Step 7 using $d_G(i)$, the degree of node i in G and $f_{i,j}$ represents the probability that j is chosen as the next hop. Thus, it can be seen that Algorithm 1 implements the steps outlined in the proof. It is also easy to see how this procedure can be modified if the adversary has an apriori distribution on the origin of the packet: she can use the apriori probability distribution $P_O(i)$ in Equation 1. \square

This analysis is applicable to probabilistic forwarding over an *incomplete* graph. We will later show using numerical results, that the short-term anonymity due to such probabilistic forwarding is acceptable.

5.1.2 Long-Term Anonymity

Suppose the adversary intercepts multiple requests from the same source (possibly to multiple destinations). This is the more interesting case introduced in [7] and intuitively the adversary can correlate the information from the multiple requests in order to identify the source. We outline the optimal adversary long-term strategy for the *incomplete* graph (over the complete graph, the adversary strategy is much simpler: the node which occurs most frequently as the previous hop is the source).

Firstly, observe that since the destination is also an adversary, the total number of requests initiated by the source is known. Define $N = \{n : (n, a) \in E, a \in A, n \in V \setminus A\}$, this represents the neighbor set of all the adversaries pooled together. Then, for a given set of requests identified to be initiated by the same source, the adversary knows R_i , the fraction of requests that had $i \in N$ as the previous hop. We call the vector $R = (R_i) \forall i \in N$ as the identity vector for this set of requests.

For every request intercepted by the adversary, the previous hop is the only information available to the adversary about the source (as discussed in the previous section). Since clearly the order of arrival of these requests does not matter, the only information to be gained from a set of requests is the identity vector R . Note that the total number of requests observed in a given time period gives information only about the initiator's request rate. This rate is not useful to the adversary in identifying the source since mixing by honest nodes hides this information. The best possible adversary strategy is then to compute the probabilities conditioned on the event that the observed identity vector is R .

The adversary strategy is as follows: she computes what we refer to as the *identity vector* of each node in $(V - A)$ and the nodes whose identity vector match the identity vector for the set of requests are the only possible sources. The identity vector of node i is defined as $E_i = (P_{i,j}) \forall j \in (V - A)$, where $P_{i,j}$ is the probability that a request originated by i is relayed through j (as defined in the previous section). The intuition for the definition of the identity vector E_i is as follows: let node i initiate a number of requests. Then, the distribution of these requests across the nodes in N converges to E_i in the long run. This distribution E_i can be thought of as the identity of i .

The identity vector of node i , E_i , can be computed by slightly modifying the construction in the single request case: one simply needs to contract all the adversary nodes to a single node (along with all the edges) and then carry out Algorithm 1. More formally, modify $G = (V, E)$ to $G'' = (V'', E'')$ as follows: $V'' = (V \setminus A) \cup \{\hat{A}\}$, E'' includes the edges induced in G by $(V \setminus A)$ in addition to the set of edges $\{(i, \hat{A}) : (i, j) \in E, i \in V \setminus A, j \in A\}$. Now, Algorithm 1 is computed on G'' , and at the completion of Step 8, the matrix P is computed, and this can be used to compute the identity vector of each node in $(V - A)$. The proof that this computes the required identity vector is similar to the proof for the short-term adversary strategy.

Note that it is possible for two different nodes to have the same identity vector E , for example, if all paths from A to other nodes pass through B (and there is no adversary on any path from A to B), so that $P_{A,N} = cP_{B,N}$, for some constant c . The other possible reason is symmetry: if both A, B are nodes located symmetrically with respect to the adversary, then they have the same identity vector.

All nodes with the same identity vector form an equivalence class (called anonymity sets, in [5]), and the adversary cannot distinguish between them even in the long run. Thus, the greater the size of a anonymity set, the better the long-term anonymity of nodes in the anonymity set. We measured this long-term anonymity in randomly generated 100–node graphs with varying number of adversaries and varying number of edges. We found that, even with a few adversaries in the system, the equivalence classes had size 1: in other words, every node had a unique identity vector. In sparse graphs, the size is greater, though this size drops rapidly as the number of edges in the graph increases (even if the number of edges is of the order of the number of vertices, so that the graph is connected, the anonymity is poor). Thus, a crowds-like scheme does not provide any long-term anonymity.

5.2 Preferential Forwarding

We now propose the preferential forwarding scheme to provide better long-term anonymity than simple crowds-like forwarding. The intuition behind this scheme is based on the observation that the adversary long-term strategy exploited the fact that each node chose its next hop *randomly* among its neighbors. One way to defend against this attack is that each node keep its forwarding vector secret. Consider a scheme where each node randomly chooses a probability vector V where V_i is the probability that it relays a request to its i^{th} neighbor, and $(1 - \sum_i V_i)$ is the probability that it forwards the request directly to the destination. In our schemes $(1 - \sum_i V_i) = (1 - p_f)$, so that the number of hops within the overlay is bounded. The short-term anonymity of this scheme is the same as that of the crowds-like forwarding scheme, since the probability that a node forwards a *single* request to a given neighbor is exactly the same as before.

With respect to long-term anonymity, the previous analysis no longer holds: the forwarding vector V need not choose the next hop from among its neighbors *uniformly*. We now discuss the optimal adversary strategy when the adversary is *passive* (does not inject any new packets). The intuition is that since the adversary does not know the forwarding vector of the nodes (because she does not probe the network), she has to consider all possible choices of the forwarding vector for every node in the system. Let the adversary observe the distribution of i 's requests across her neighbors and denote this distribution by R . The *exact* adversary strategy is to solve the following optimization problem (this adversary strategy is also the best possible since R represents completely the adversary's knowledge of the source). First modify the graph by removing outgoing edges from every adversarial node (this imposes the constraint that the previous hop is observed by the *first* adversary on the path). Define P_i to be the forwarding vector of node i with entries $\sum_{j \in N(i)} P_{i,j} = 1$ where $N(i)$ is the neighbor set of i (which includes the destination server). Define S as the transition matrix $[P_1 P_2 \cdots P_n]^T$. Then, the probability $P(i|R)$ that i has the identity vector R is proportional to the number of solution points in $P = P_1 \times P_2 \times \cdots \times P_n$ under the following set of constraints:

$$\forall j \sum_{i \in N(i)} P_{j,k} = 1, \quad [R_1 R_2 \cdots R_n]^T = S(I - S)^{-1}, \quad \frac{R_i}{\sum_j (R_i)_j} = R$$

The above problem is simply the derivation in Section 5.1.2 phrased in terms of solving a set of equations for the identity vector. The last step simply normalizes the R_i vector to obtain the identity vector. Note however, in these equations, we have treated the probability space as consisting of a finite number of elements: this is an approximation to the continuous case. The exact adversary strategy is to solve this equation for every node i to obtain R_i , the probability that node i has the identity vector R . The probability that i is the originator is simply $P(i|R)/(\sum_j P(j|R))$ (if the adversary has no apriori information about the source). We do not know whether it is possible to compute these probabilities efficiently (either exactly or within some approximation factor). However, observe that in most typical graphs, every node in the graph has a non-zero probability of having any particular identity vector, and thus even in the long-term, an adversary cannot say with *certainty* that any single node is the initiator. Thus, preferential forwarding has at least plausible deniability [4] against a passive adversary even in the long term.

This question of long-term anonymity against a passive adversary is not very important however, since an *active* adversary (which is a more realistic assumption) that can inject new traffic into the system is far more effective. Apriori, she is unaware of the identity vector E_i of each node i . However, a malicious neighbor of i can deliberately probe node i with requests (say, with a magic byte sequence) and then analyze the requests that she intercepts later on. These requests will induce a distribution across the neighbors of the adversary, and thus the adversary can obtain the vector E_i in the long run. Thus, even though the adversary is unaware of the forwarding matrix S^1 , she can easily deduce the identity vector of each node i by *probing*. Once the identity vector of a node is known, the node has no longer any long-term anonymity (assuming the size of its anonymity set is 1). Thus, this scheme fails against an active adversary.

One naive suggestion to fix this attack would be to insist that each node change its probability forwarding vector V periodically. However, this scheme only weakens the anonymity: it is now in fact susceptible to a passive adversary as well. This can be seen from the analysis in Section 5.1. If a node keeps changing its vector V periodically and randomly, over a long interval, the number of requests forwarded through each neighbor is the same. Thus, this is equivalent to the case discussed earlier, in that each neighbor is the next hop with equal probability. Then, over a long (but finite) run, the adversary can use it to identify the source by the identity vector method discussed earlier.

5.3 Neighbor Hash Forwarding

In this section, we devise a forwarding scheme that is resistant to active adversaries even in the long-term. The main idea is that each node chooses a forwarding vector that does not change over time, and also is unique for each of its neighbors (from which the request is relayed). The intuition is that the probing of a honest node A by an adversary B might reveal at most the forwarding vector that A uses for requests relayed from B (which is independent of the forwarding vector A uses for other neighbor's requests and its own requests).

The scheme is as follows: the neighbor to which a given request is forwarded is chosen depending on the previous hop (or the node itself, if it is the originator). We choose to use a cryptographically secure hash function [17] in order to summarize the forwarding scheme of a node. The details of the forwarding scheme are stated in Algorithm 2. The basic idea is that the identity of the previous hop is used as a input to a secret hash function. The output of the hash function is then used to select the next hop from the neighbor list of the node. Each node maintains two secret keys for this purpose.

Algorithm 2 Deterministic Neighbor Hash Forwarding

- 1: Given the set of neighbors $N = \{N_1, N_2, \dots, N_k\} \cup \{\hat{D}\}$ of a node A , two secret keys S_1, S_2 , previous hop R (if A is the originator, the previous hop is set to A itself), h a cryptographically secure keyed hash function.
 - 2: Hash the identifiers of the neighbors using a hash function keyed by S_1 : $N' = \{h_{S_1}(N_1), h_{S_1}(N_2), \dots, h_{S_1}(N_k), h_{S_1}(\hat{D})\}$.
 - 3: Hash the identifier of the previous hop using a hash function keyed by S_2 : $R' = h_{S_2}(R)$.
 - 4: Use consistent hashing to map R' to an element, say $x = h_{S_1}(N_i)$, in the set N' . Choose N_i to be the next hop.
-

Note that the set of neighbors N includes a special node \hat{D} in place of the destination: in other words, if the algorithm chooses to forward to \hat{D} , the request is sent to the destination server. If a certain fixed forwarding probability p_f is desired, then there are several \hat{D} in the neighbor set, such that the probability of forwarding to the destination is p_f . This modification is implicit in our future schemes. Secondly, note that it is important that the identifiers of the nodes be unforgeable. Otherwise, an adversary could simply forge a honest node's, say B 's, identifier in order to learn about the forwarding rule that A uses for B . This requirement is satisfied by using the IP address and port number combination as an identifier, as discussed in section 4.3.

Consistent hashing [8] is used to map the output of the hash function R' to an element in the set N' . In other words, the element in the set N' that is numerically greater than and closest to R' is chosen as the next hop (wraparound is used in the numerical comparison). The purpose of consistent hashing is to avoid an “artificial churn” attack which is as follows. Consider an adversary that repeatedly joins and leaves the network, inducing artificial churn (departure and arrival of nodes) in the network. Such an adversary should not be able to cause the forwarding vector of a node to converge to a uniform distribution. Consider, for example, a forwarding scheme where the set of neighbors is maintained as a list of size s , and the hash value $h_{S_2}(R) \bmod s$ is used to index into this list to compute the next hop. If a set of neighbors leave, then it is possible in certain configurations for the new forwarding rule to be independent of the old forwarding rule. For example, say the identifier space is the set of integers $\{1, 2, \dots, pq\}$ (where p, q are two primes such that $p > q$). Now suppose that $p - q$ nodes out of a neighbor set of p nodes leave the network. Then, since $x \bmod p$ and $x \bmod q$ are independent (when x is unknown), the new forwarding rule is independent of the old one. This essentially means that it is possible for the adversary to force the node to choose a independent forwarding vector. Such an adversary can force multiple choices of the forwarding vector, and thus cause convergence of the long-term forwarding vector to the uniform vector. Consistent hashing is resistant to this attack: if an adversary leaves the network, the only difference is that requests that would have been forwarded to her, will now be forwarded to the next numerically greater element in the set N' . Thus, the new probability distribution is not independent of the old one (this feature also offers automatic route maintenance). Consistent hashing also ensures that the load is balanced across all neighbors of a node. However, natural churn still degrades the anonymity, and intersection attacks [18] might still reveal the source.

The determinism in this scheme implies that *all* requests initiated by a given node are relayed through a fixed node, say F , and to the adversary A_{int} . This trivially means that the long-term anonymity is the same as the short-term anonymity. There are several drawbacks of using a deterministic scheme: all requests initiated by a source take the same path in the absence of joins and leaves. This leads to several undesirable properties: some nodes always suffer a high latency due to long paths, some nodes always relay through an adversary etc. This also makes certain nodes susceptible to attacks by adaptive adversaries [7] in a very few rounds. An adaptive adversary corrupts the previous hop from which she intercepts the request and can identify the source after intercepting multiple requests. Some sources that are unlucky to have short relaying paths will be discovered in a few rounds by an adaptive adversary. To address these concerns, we propose a probabilistic variant in Algorithm 3.

The main difference between the probabilistic and deterministic version is that R' is used to generate a probability distribution instead of being used directly to choose the next hop. For maintaining consistent hashing in the probabilistic variant, first a probability distribution over the identifier space I is generated, and then used to generate a distribution over the neighbors. Note that I can be a randomly chosen subset of the identifier space. This can be used to reduce the computation involved in forwarding. In the probabilistic neighbor hash scheme, no node will have to suffer long relaying paths or short relaying path all the time. Probabilistic neighbor hash requires $O(k)$ computation where k is the number of neighbors, which is feasible in our case, since each node can choose k for itself.

Algorithm 3 Probabilistic Neighbor Hash Forwarding

- 1: Given the set of neighbors $N = \{N_1, N_2, \dots, N_k\} \cup \{\hat{D}\}$ of a node A , two secret keys S_1, S_2 , previous hop R (if A is the originator, the previous hop is set to A itself), h a cryptographically secure keyed hash function.
- 2: Hash the identifiers of the neighbors using a hash function keyed by S_1 : $N' = \{h_{S_1}(N_1), h_{S_1}(N_2), \dots, h_{S_1}(N_k), h_{S_1}(\hat{D})\}$
- 3: Hash the identifier of the previous hop using a hash function keyed by S_2 : $R' = h_{S_2}(R)$.
- 4: Generate a probability distribution $F : I \mapsto [0, 1]$ seeded by R' where I is the identifier space (say, using a pseudo-random generator seeded with R').
- 5: Use consistent hashing to assign the forwarding probability per node, $F'(i)$, as the sum of the probabilities assigned to the identifiers in I between $h_{S_1}(N_i)$ and the next numerically greater identifier in the set N' .
- 6: Sample according to the probability distribution F' in order to choose the next hop.

Table 1: Kinds of Adversaries Vs. Counter-measure

| Classification | Capability | Counter-Measure |
|--------------------|-----------------------------------|-----------------------------|
| Passive Adversary | Can Only Observe Traffic | Preferential Forwarding |
| Active Adversary | Can Inject Traffic | Neighbor Hash |
| Adaptive Adversary | Can Corrupt New Nodes every Round | Probabilistic Neighbor Hash |
| Artificial Churn | Malicious Joins and Leaves | Consistent Hashing |

5.3.1 Security Analysis

First, we analyze the short-term anonymity of the probabilistic neighbor hash scheme against a passive adversary. Our security proofs rely on the following property of the hash function h : it is computationally infeasible for any adversary to find the value of $h_S(x)$ given values of $h_S(x')$ for (polynomially) several $x' \neq x$. This assumption is referred to as the random oracle model [19] in cryptography literature. The passive adversary has no information about the forwarding vectors of any node, and thus the short-term anonymity of this scheme is the same as in the case of the crowds-like forwarding scheme. Similarly, the long term anonymity of this scheme against a passive adversary is the same as that of the preferential forwarding scheme. This follows from the random oracle model since we can approximate the hash function to a random function.

Now we consider an active adversary. Define $F_{i,j}$ as the forwarding vector that i uses for requests relayed from j . With respect to active adversaries, notice that any number of probes sent by the adversary cannot determine the vector $F_{i,i}$ because this vector is not used unless i originates the packet. Thus, the active adversary cannot determine any information about the forwarding rules used by a node for its own requests. However, the adversary can gain some information about the forwarding rules $F_{i,j}$ for $i \neq j$. Thus, though the adversary cannot compute the identity vector of a given node *exactly* by probing, it can however obtain a probability distribution for the identity vector. We could not characterize this partial information obtained, and so only conclude that this scheme has at least plausible deniability.

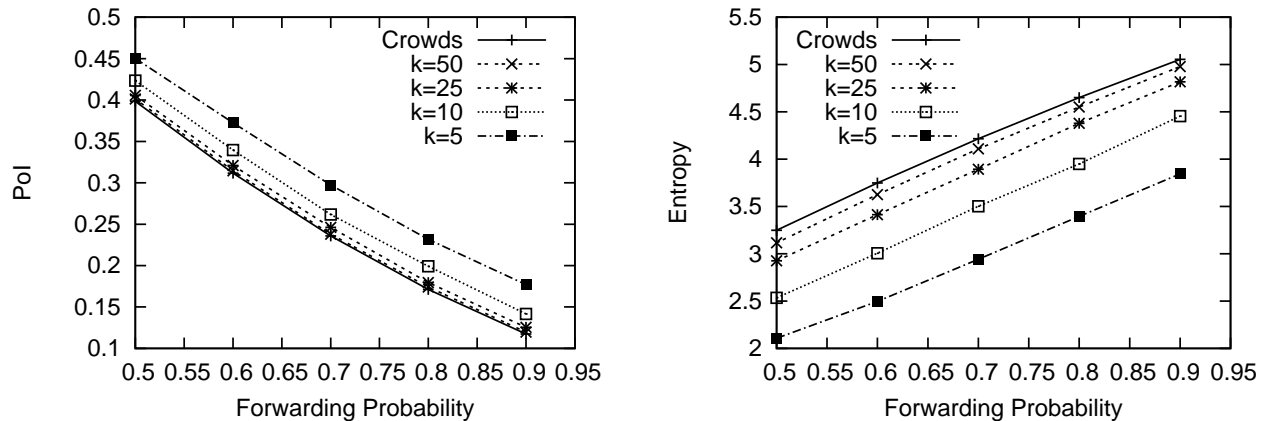


Figure 2: Anonymity Performance Tradeoff (a) PoI Vs. Probability of Forwarding (b) Entropy Vs. Probability of Forwarding

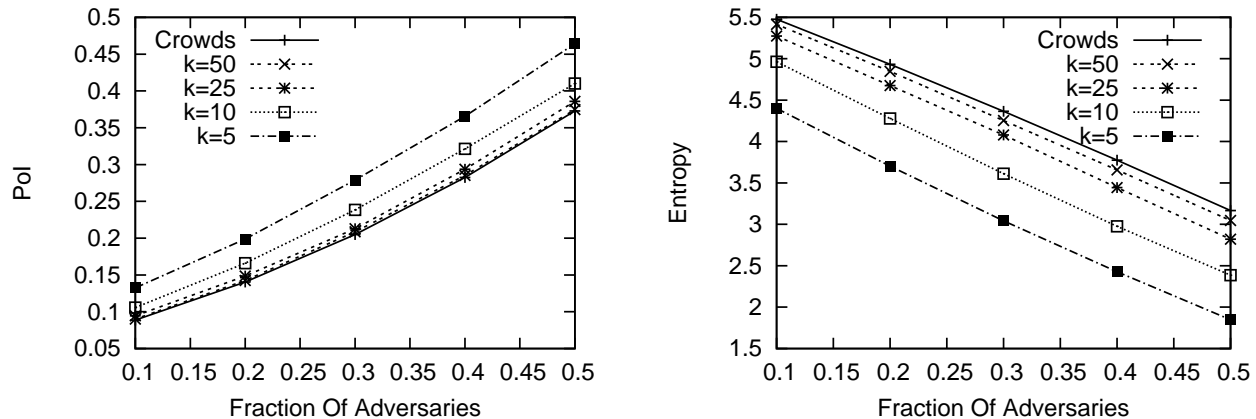


Figure 3: Varying Number of Adversaries (a) PoI Vs. Fraction of Attackers (b) Entropy Vs. Fraction of Attackers

6 Numerical Results

In this section, we present numerical results for the short-term anonymity of the forwarding schemes in Hush. With respect to the long-term anonymity, we know that Hush provides at least plausible deniability: the exact adversary strategies are too expensive to simulate. For these experiments, we generated a random topology, and then simulated the adversary strategy to quantify the level of anonymity. Through these results, we wish to quantify the tradeoff between anonymity, state (number of neighbors), and performance (in terms of number of hops). Unless otherwise stated, these results are derived from a 100 node regular graph (the required matrix operations are too expensive for larger graphs). We present two anonymity metrics, Probability of Identification (PoI) and entropy. These metrics can be easily derived from our analysis of the adversary strategies. Both metrics are averaged over all nodes (due to the regularity of the graph, all nodes have the same metric on average). We have also assumed that the adversary has no a priori knowledge of traffic patterns.

We generated k -regular graphs, and measured the variation of anonymity level with the probability of forwarding, for a constant number of attackers (25%). To illustrate the tradeoff against state, we plotted this variation for various values of k ($k = 100$ represents the complete graph as in Crowds). The results for PoI and entropy are illustrated in Figure 2 (a) and (b) respectively. Clearly, as the probability of forwarding p_f increases, the average hop length increases, as does the anonymity level. These graphs illustrate that for a logarithmic reduction in state, the reduction in anonymity level is considerably lower both for the PoI and the entropy metrics. These graphs illustrate for a hop length of around 5 or 6, the PoI with $k = 10$ is only 50% more than that with $k = 100$. The entropy metric is a richer metric that includes information about the probability distribution: for example, in Crowds ($k = 100$) with a hop length of 10, the entropy metric is 5 as against a maximum possible entropy of $\log_2(75)$ (little more than 6). The entropy metric remains above 4 even with as few as 10 neighbors. This graph clearly highlights the tradeoff of state with short-term anonymity possible in Hush.

To illustrate the variation of anonymity level against varying fraction of adversaries, we obtained the plots in Figure 3. These results make a similar point to the previous graph: the savings in state is considerably greater than the loss in anonymity. There is little reduction in either anonymity metric despite a 4-fold decrease in the amount of state. Most of the performance issues in Hush relating to efficiency (such as TCP performance over multiple hops) are similar to those of relaying schemes like Onion Routing which have been well-studied in [2], [15]. The scalability of Hush follows from the the scalability of unstructured peer-to-peer networks (for example, Gnutella is known to have tens of thousands of users [20]).

7 Comparison with Existing Schemes

In this section, we attempt to place Hush in the context of existing schemes (Table 2 provides a summary of this discussion). We first compare Hush with relaying based overlays, then with more recent anonymization networks based on DC-nets. We also attempt to place the existing anonymization overlays in context with one another.

7.1 Chaumian Networks

Most of the existing relaying based overlays are based on either Crowds-like Networks (probabilistic forwarding) or Chaum-like Networks (Onion Routing, Tarzan etc). We have discussed the former in detail before, we now address the latter. Chaumian networks require a Public Key Infrastructure (PKI), which is one of the reasons they are not suitable for a dynamic overlay environment. The presence of a PKI allows the initiator to choose a path of *constant* length through the overlay, by encrypting

Table 2: Comparison with Existing Anonymization Overlays

| Scheme | Short-Term Anonymity | Long-Term Anonymity | Scalability | PKI required |
|-------------------|----------------------|------------------------------|----------------|--------------|
| Chaumian Networks | Best | Poor | $O(n)$ state | Yes |
| Onion Routing | Best | Poor | $O(n)$ state | Yes |
| Crowds | Moderate | Poor | $O(n)$ state | No |
| Tarzan | Best | Poor | $O(n)$ state | No |
| Freenet | Not Analyzed | Not Analyzed | Yes | No |
| Herbivore | Moderate | Same as short-term Anonymity | Low efficiency | No |
| P5 | Tunable | Tunable | Low Efficiency | No |
| Hush | Moderate | Plausible Deniability | Yes | No |

the path using public key techniques. This leads to better short-term anonymity compared to probabilistic forwarding like in Crowds, Hush (the analysis of the short-term anonymity of the Chaumian network is presented in Appendix 7.1). We now illustrate the limitation of Chaumian-like schemes. Define relaying scheme as path-agnostic if all relaying paths of the same length have the same probability of being chosen: both Crowds and Chaumian networks are path-agnostic. We extend the analysis technique to show the following theorem:

Theorem 1. *In a complete graph on N nodes, perfect anonymity is not possible using path-agnostic relaying schemes if the fraction of adversaries $\alpha > \frac{1}{2} - \sqrt{\frac{1}{4} - \frac{1}{N}}$.*

Corollary 1. *Perfect Anonymity is not achievable using path-agnostic relaying schemes in the asymptotic setting*

Proof. Refer Appendix A.

Without a PKI, choosing a path of *constant* length would *necessarily* leak information about the source. Intuitively, the argument is as follows. Consider an initiator setting up a relay path through L other nodes (for some constant L). The path setup mechanism operates hop by hop: the path to hop i is established, and i^{th} hop is then instructed to extend the path to the $(i+1)^{\text{th}}$ hop (which is chosen by the initiator). If the initiator does not have the public key of hop $(i+1)$, then it cannot communicate the identity of the $(i+2)^{\text{th}}$ hop to hop $(i+1)$ without hop i learning the identity as well. This means that the first adversary on the path can deduce the number of nodes that follow her on the path. Since the path length is a constant, she can deduce its position on the path i . In particular, if $i = 1$, she knows definitely that the previous hop is the source. Even if $i > 1$, this leaks some information about the source in an *incomplete* graph. Thus, a constant length relaying technique requires Chaumian-like public key encryption in order to provide its strong anonymity properties. Tarzan, an anonymization overlay based on Chaumian networks, does away with the requirement for a PKI, by performing public key distribution along with the topology maintenance algorithm. However, this implies that each node has to maintain public keys for every other node (even if it does not have every node as its neighbor).

When the length of the path cannot be encoded in this fashion, then the *memoryless* probabilistic forwarding (as in Crowds or Hush) is one possibility. Another possibility is that the hop length can be chosen *randomly* according to some distribution by the initiator. This means the adversary does not know exactly the distance of the source from itself: she can only obtain a probability distribution. Note however that the initiator has to choose this distribution *without* a knowledge of the complete topology (due to scalability concerns), and it seems a hard problem to choose some “optimal” distribution without such knowledge. To the best of our knowledge, this possibility has not been considered in the literature. The point we wish to make from this discussion however is that the presence of a PKI allows constant length paths, and consequently better short-term anonymity.

7.1.1 Improved Chaumian Networks

In Chaumian-like networks, the path selection mechanism makes it susceptible to the predecessor attack. The only requirement for this attack to work is that the adversary be able to identify requests from the same source, which is possible since the request is sent unencrypted to the destination server. Even if session keys are established to encrypt the request, the request leaves the overlay in the plain, and therefore the last hop node can identify requests from the same source, say using the contents of the request. The path selection is to blame here: the random selection can be modified to use secret keys as in Hush. Thus, our techniques can be used to patch up other proposals.

As an example, we illustrate how to defend against the predecessor attack in Chaumian-like networks. Assume a public key infrastructure is available (say, a trusted third party which can return the public key of a specific node). Then, our solution would be to have the initiator node maintain secret keys for each of the other nodes in the system, and use these to choose a

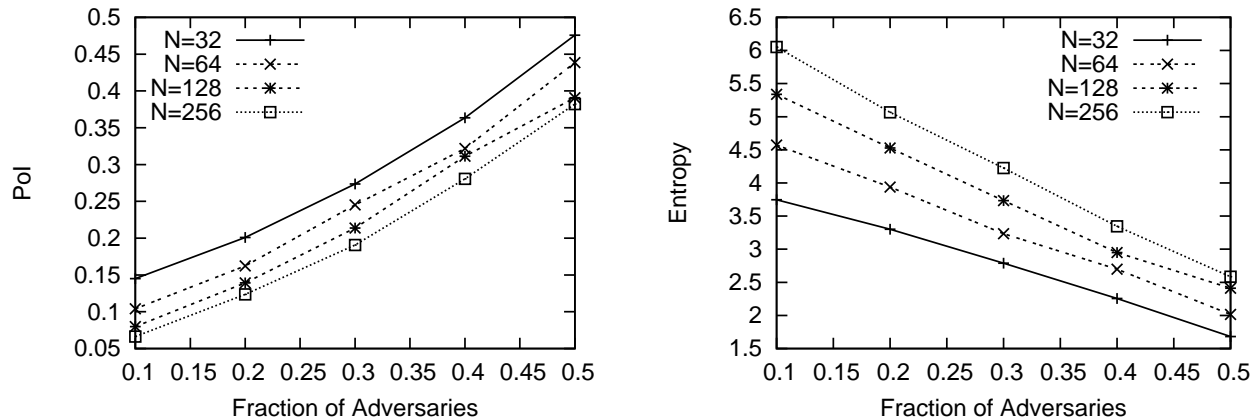


Figure 4: DHT-like Environment (a) PoI Vs. Fraction of Attackers (b) Entropy Vs Fraction of Attackers

random route (as per the probabilistic hash scheme). Once the route is chosen, the initiator can query the trusted third party for the public keys of the nodes involved, and set up a relaying path as in Chaum’s scheme. Note that in order to avoid leaking information to the third party, the node may store the public keys of all nodes in the system.

Improving the scalability of Chaumian networks with Hush-like techniques seems to be a harder problem, even with the presence of a PKI that can be queried for public keys. The main problem is that since the initiator does not know the topology of the graph, it has to rely on other nodes to choose paths. The first adversary A_{int} on the path can deliberately return a path containing only adversaries. Then, the adversary can discover its position on the path, and since Chaumian networks use a constant length, gain some information about the source.

7.2 DC-Net Based Schemes

In this section, we discuss DC-Net based schemes (like Herbivore) and P5. Herbivore partitions the nodes into cliques of size k which follow a DC-Net like protocol between them. Their proposal for routing to a destination outside the overlay is as follows: the node chooses a random node in its clique, routes to it anonymously, and that node then relays it on to the destination. The main disadvantage of Herbivore is that efficiency drops heavily at high utilization. We will show using a simple analysis of Herbivore and numerical results in Appendix C to show that for the same state, Hush provides superior short-term anonymity and efficiency, while Herbivore has superior long-term anonymity. Hush is also more flexible than Herbivore in that the anonymity is a function of both the forwarding probability and state. Also in Hush, the latency goes up linearly with the number of hops, whereas, in Herbivore, the latency goes up exponentially with the size of clique (due to collisions). P5 suffers from efficiency problems similar to Herbivore (in fact, it has lower efficiency).

Based on our analysis of relaying and DC-Net based schemes, we can compare relaying based overlays with DC-Net like overlays. The former are based on the relaying primitive and have good efficiency while the latter are based on the DC-Net primitive and have better long-term anonymity. It is possible to use Hush-like techniques to build a hybrid overlay: like Herbivore, the nodes are partitioned into cliques which operating using the DC-net primitive. The cliques can now establish links to each other to form a topology like Hush. The forwarding scheme is now as follows: the initiator chooses a random neighbor in its clique as the next hop. This next hop then forwards it to nodes in neighboring cliques in a probabilistic fashion like Hush. This hybrid scheme clearly has at least the long-term anonymity of Herbivore, and better short-term anonymity than Herbivore. We believe it also requires smaller clique size for the same level of short-term anonymity and thus greater efficiency. The exact quantitative behavior can be derived using our techniques for analyzing an incomplete graph’s anonymity properties, along with the analysis of Herbivore.

8 Application to Peer-to-Peer Networks

In this section, we demonstrate the usefulness of Hush-like techniques in peer-to-peer networks. Peer-to-peer networks have been an active topic of research in the networking community in the past few years, and two of the chief design requirements of such networks have been scalability and handling node churn. Under such requirements, maintaining $O(n)$ state is impractical, and most existing anonymization overlays are not feasible. However, Hush’s goals are similar, and incorporating the forwarding schemes of Hush into existing peer-to-peer networks is a practical alternative. We illustrate the application of our schemes in the two kinds of P2P networks: Structured P2P networks (DHTs) and Unstructured P2P networks.

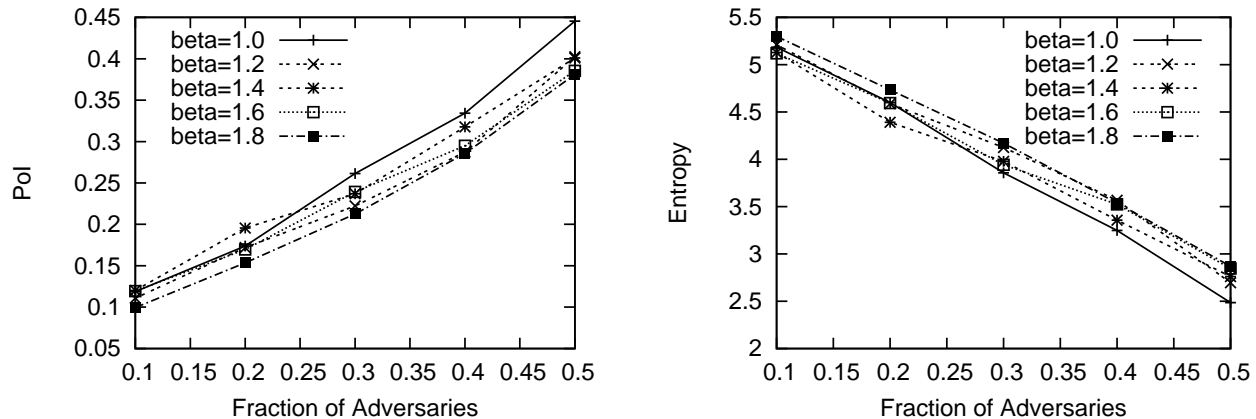


Figure 5: Gnutella-like Environment (a) PoI Vs. Fraction of Attackers (b) Entropy Vs Fraction of Attackers

8.1 Distributed Hash Tables

A DHT is a dynamic overlay network of peer nodes that offers a lookup service. It allows two operations: *lookup(key)* and *insert(key,value)*. These two operations enable nodes to efficiently store data under a keyword in the overlay network and allows such data to be accessible using the keyword. The topology of these nodes is organized in such a fashion, so as to index the data stored within the network. DHTs are typically k -regular graphs where the topology maintenance algorithm and data storage algorithms offer guarantees on the performance on these operations. Most DHTs are optimized towards reducing per-node state, typically requiring about $O(\log(n))$ state where n is the number of nodes in the system. An example is Chord [21] which uses $O(\log(n))$ state to route a lookup in $O(\log(n))$ hops. DHTs can be used as a substrate for a wide variety of applications including event notification [22], distributed filesystems [23], database systems [24], keyword based file searching, publish-subscribe systems etc.

Some of these applications could benefit from the following feature: the initiator of the insert or lookup operation is anonymous to other nodes. This allows applications like anonymous keyword-search and anonymous subscription etc. There are two possible ways of incorporating our forwarding schemes in a DHT. Since the gossiping protocol in Hush has low overhead, the nodes can organize an overlay alongside the DHT, for purposes of anonymization. Instead of initiating connections to a destination server, the final hop within the anonymization overlay can perform the lookup/insert on behalf of the originator. Another possible way is to use the DHT neighbors itself as neighbors in the anonymization overlay, though this has security problems if the DHT chooses neighbors based on proximity.

To illustrate the anonymity levels achievable by our schemes in a DHT-like environment where the number of neighbors is $O(\log(n))$, we obtained the plots in Figure 4. In these plots, the forwarding probability was set at $1 - (1/\log(n))$, so that the number of hops within the overlay is logarithmic. These results indicate the levels of anonymity achievable in a DHT. In fact, as the number of nodes increases, the anonymity level increases (this is due to the increase in hop length within the overlay). This suggests that these mechanisms are suitable for incorporating in a DHT.

8.2 Unstructured P2P Networks

An unstructured P2P network (like say, Gnutella) is one which is more loosely organized, and the topology maintenance is usually of the gossiping style. For this reason, they are believed to be better suited for environments with frequent churn (departure and arrival of nodes). The downside is that the search operation is more expensive, involving some form of controlled flooding. These networks are typically used for file-sharing today.

In such an environment, anonymity is clearly desirable (for the users), and our forwarding schemes can be applied to this network as well. These networks usually follow a power-law graph [20] where $d_k \propto k^{-\beta}$ (d_k is the probability a node has degree $\geq k$, and β is a parameter ≥ 1). We evaluated our schemes in such an environment in Figure 5 by plotting the variation of anonymity against the fraction of adversaries in a 100-node graph. We generated graphs following different power-law parameters β : greater the value of β , more the number of edges, and greater the variation in degree among nodes. These graphs show that the anonymity level in these environments is comparable with a similar DHT-environment (whose degree is set to be the average degree of the power-law graph). Despite the variation in degrees of nodes in a power-law graph, our forwarding scheme performs well, and thus, appears suitable for unstructured P2P networks as well.

9 Conclusions

The main contribution of this work is the proposal of Hush, a scalable and efficient anonymization network. We address three specific issues in the design of Hush. Firstly, we characterize the tradeoff between state, performance, and anonymity in probabilistic forwarding schemes. Secondly, Hush also provides a defense against the predecessor attack, which is applicable to most of the existing relaying based schemes. This defense is based on using hash functions to make forwarding decisions. Hush provides both scalability and efficiency and this make it suitable for building dynamic overlays for web browsing etc. Our analysis techniques for the short-term and long-term anonymity of Hush may be of independent interest. We also extend this analysis to shed light on the anonymity tradeoffs offered by relaying and DC-Net based overlays. Lastly, we show that our forwarding scheme can be “plugged” into existing peer-to-peer networks to offer anonymity as an useful feature for several large-scale distributed applications.

References

- [1] D. Chaum, “Untraceable electronic mail, return addresses, and digital pseudonyms,” *Communications of the ACM*, vol. 24, no. 11, pp. 84–90, 1981.
- [2] P. F. Syverson, D. M. Goldschlag, and M. G. Reed, “Anonymous connections and onion routing,” in *IEEE Symposium on Security and Privacy*, (Oakland, California), pp. 44–54, 1997.
- [3] M. J. Freedman and R. Morris, “Tarzan: A peer-to-peer anonymizing network layer,” in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, (Washington, D.C.), November 2002.
- [4] M. K. Reiter and A. D. Rubin, “Crowds: anonymity for Web transactions,” *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–92, 1998.
- [5] D. Chaum, “The dining cryptographers problem: Unconditional sender and receiver untraceability,” *Journal of Cryptology*, vol. 1, no. 1, pp. 65–75, 1998.
- [6] S. Goel, M. Robson, M. Polte, and E. G. Sirer, “Herbivore: A Scalable and Efficient Protocol for Anonymous Communication,” Tech. Rep. 2003-1890, Cornell University, Ithaca, NY, February 2003.
- [7] M. Wright, M. Adler, B. Levine, and C. Shields, “An analysis of the degradation of anonymous protocols,” in *ISOC Symposium on Network and Distributed System Security*, February 2002.
- [8] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy, “Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web,” in *ACM Symposium on Theory of Computing*, pp. 654–663, May 1997.
- [9] G. D. Andrei Serjantov, “Towards an information theoretic metric for anonymity,” in *Privacy Enhancing Technologies (PET 2002)* (R. Dingledine and P. Syverson, eds.), Springer-Verlag, LNCS 2482, 2002.
- [10] R. Sherwood, B. Bhattacharjee, and A. Srinivasan, “P5: A protocol for scalable anonymous communication,” in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, May 2002.
- [11] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, “Freenet: A distributed anonymous information storage and retrieval system,” *Lecture Notes in Computer Science*, vol. 2009, pp. 46+, 2001.
- [12] G. Danezis, “Mix networks with restricted routes,” in *Privacy Enhancing Technologies (PET 2003)* (R. Dingledine, ed.), Springer-Verlag, LNCS 2760, March 2003.
- [13] J. Douceur, “The sybil attack,” in *Proc. IPTPS 2002*, (Cambridge, MA), March 2002.
- [14] R. Dingledine, M. J. Freedman, D. Hopwood, and D. Molnar, “A Reputation System to Increase MIX-net Reliability,” in *Proceedings of Information Hiding Workshop (IH 2001)* (I. S. Moskowitz, ed.), pp. 126–141, Springer-Verlag, LNCS 2137, April 2001.
- [15] R. Dingledine, N. Mathewson, and P. Syverson, “Tor: An anonymizing overlay network for tcp.” CodeCon 2004.
- [16] A. Pfitzmann, B. Pfitzmann, and M. Waidner, “ISDN-mixes: Untraceable communication with very small bandwidth overhead,” in *Proceedings of the GIITG Conference on Communication in Distributed Systems*, pp. 451–463, February 1991.
- [17] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook Of Applied Cryptography*. CRC Press, 1997.
- [18] O. Berthold and H. Langos, “Dummy traffic against long term intersection attacks,” in *Privacy Enhancing Technologies (PET 2002)* (R. Dingledine and P. Syverson, eds.), Springer-Verlag, LNCS 2482, 2002.

- [19] M. Bellare and P. Rogaway, “Random oracles are practical: A paradigm for designing efficient protocols,” in *ACM Conference on Computer and Communications Security*, (Fairfax, Virginia, USA), pp. 62–73, 1993.
- [20] S. Saroiu, P. K. Gummadi, and S. D. Gribble, “A measurement study of peer-to-peer file sharing systems,” in *Proceedings of Multimedia Computing and Networking 2002 (MMCN '02)*, (San Jose, CA, USA), January 2002.
- [21] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, “Chord: A scalable Peer-To-Peer lookup service for internet applications,” in *Proceedings of the 2001 ACM SIGCOMM Conference*, pp. 149–160, 2001.
- [22] L. F. Cabrera, M. B. Jones, and M. Theimer, “Herald: Achieving a global event notification service,” in *Proc. of the IEEE Eighth Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, (Elmau, Germany), May 2001.
- [23] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, “Wide-area cooperative storage with CFS,” in *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01)*, (Chateau Lake Louise, Banff, Canada), Oct. 2001.
- [24] R. Huebsch, J. M. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica, “Querying the internet with pier,” in *Proc. VLDB 2003*, (Berlin), 2003.
- [25] M. Rennhard and B. Plattner, “Practical anonymity for the masses with mix-networks,” in *Twelfth IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, (Linz, Austria), pp. 255–262, June 2003.

A Perfect Anonymity

In this section, we will prove that perfect anonymity is impossible in a *complete* graph for a certain class of relaying based schemes we refer to as *path-agnostic* even with the presence of a PKI and perfect mixes. This class of schemes includes Chaumian Networks, Crowds, Tarzan etc. We are not aware of any proof in existing literature of the same. The proof is simple and relies on the symmetry of the complete graph. The main implication of this theorem is that relaying based schemes like Chaumian Networks and Crowds cannot provide perfect anonymity on a complete graph.

We define a relaying based scheme to be path-agnostic if the probability that a request from a source is relayed along a path depends only on the length of the path and not on the nodes comprising the path. This implies that the length of the path follows some distribution, and all routes of the same length at the originator are equally probable. This class does not include DC-net like schemes (which are based on coordinated relaying), or Hush (where the identity of the nodes on the path determines the probability that the path is chosen).

This hop length distribution can arise in a variety of ways. In Crowds, the distribution is geometric due to independent forwarding by each node. In Chaumian networks, the hop length is constant. In general, if a PKI is available, the initiator could choose any arbitrary distribution without leaking any information to adversaries by encoding the path in a Chaumian-like fashion.

We define perfect anonymity as follows: the PoI is $1/n$ where n is the number of honest nodes in the system. This means that the probability assigned by the adversary to the true source, is the same as that assigned to every other node in the system. Note that this proof studies the theoretical feasibility of perfect anonymity (unlike [25] which argues that it requires too much overhead in practice). This holds for both short-term and long-term anonymity measures. Also, this analysis applies to the *complete* graph only, and not to bounded degree graphs, like in Hush or Freenet.

Theorem 2. *In a complete graph on N nodes, perfect anonymity is not possible using path-agnostic relaying schemes if the fraction of adversaries $\alpha > \frac{1}{2} - \sqrt{\frac{1}{4} - \frac{1}{N}}$.*

Corollary 2. *Perfect Anonymity is not achievable using path-agnostic relaying schemes in the asymptotic setting*

Proof. Let us assume that the initiator establishes relaying paths with hop lengths that follow a distribution h : $h(i)$ is the probability that the hop length is i (i denotes the number of overlay nodes through which the request is relayed before being forwarded to the destination server). Denote the total number of nodes in the system by N , the number of honest nodes by n , and the fraction of adversaries by α . We first prove the following lemma:

Lemma 2. *In a path-agnostic scheme, if a request is intercepted by an adversary, then the adversary strategy assigns a greater probability to the previous hop from which the request was intercepted (as compared to the other honest nodes).*

Consider a source that initiates a request. If the request is not intercepted by an adversary before it is relayed to the destination, then the adversary has no information about the source. On the other hand, consider the case when there is at least one adversary on the path. Let A_{int} be the first adversary on the path, which knows the previous hop R . Now, we consider what information is revealed about the source from the other adversaries on the path. Assuming that each honest node mixes the traffic such that

no timing attacks are possible, then only the adversaries that *immediately* follow A_{int} on the relaying path are aware that this request had previous hop R . Due to mixing, if any honest node is present on the path between two adversaries, then the two adversaries cannot verify that they are indeed relaying the same request. Thus, if the path consists of $A_{int}, A_1, \dots, A_{j-1}$ before it encounters a honest node, only the adversaries $A_{int}, A_1, \dots, A_{j-1}$ know that the previous hop is R . Note that A_{j-1} could be D , the destination server.

Therefore, the adversary strategy of A_{int} is conditioned on two facts: the previous hop is R and there are j adversaries following it on the path (since the public key encryption hides the length). This is the only information available to the adversary about the initiator of the request. Thus, the adversary strategy is to obtain a probability distribution of the distance of the source from the first adversary on the relaying path. This strategy to determine the number of hops holds due to two reasons: the symmetry of the complete graph and the assumption that the scheme is path-agnostic. Intuitively, the assumption that the scheme is path agnostic allows the adversary to obtain a probability distribution for the distance of the source, and then use that to compute the distribution of the source itself. This is similar to the fashion in which the route itself is chosen: the hop length is chosen first and then a route of that hop length is chosen uniformly at random.

The fact that there are j adversaries on the path implies that the relaying path is of length at least j . Then, the probability $P(i, j)$ that the source is j hops away from the first adversary A_{int} in the path is:

$$\begin{aligned} P(i, j) &= \Pr[\text{source } i \text{ hops away} \mid j \text{ adversaries}] = \frac{\Pr[\text{source } i \text{ hops away, } j \text{ adversaries}]}{\Pr[j \text{ adversaries}]} \\ &= \frac{\sum_{k=(i+j-1)}^{\infty} F(i, j, k)}{\Pr[j \text{ adversaries}]} \end{aligned}$$

where $F(i, j, k)$ is the probability that the relaying path is of length k and there are j consecutive adversaries on the path with the first one at position i and $\Pr[j \text{ adversaries on path}]$ refers to the probability that there are $(j - 1)$ consecutive adversaries following the first one on the path. The parameter k runs from $(i + j - 1)$ to ∞ because the path length is at least $(i + j - 1)$. It is easy to see that:

$$F(i, j, k) = h(k) \times (1 - \alpha)^{(i-1)} \alpha \times \alpha^{(j-1)} (1 - \alpha)$$

This equation includes the probability that the hop length is k , the probability that the first adversary is at position i , and the probability that there are exactly $(j - 1)$ adversaries following her. The last term $(1 - \alpha)$ is omitted if $A_{j-1} = D$, since the adversary knows that the path terminates at D .

Thus, given the fact that there are j adversaries on the path, the adversary infers that the probability that the source is at a distance i from the first adversary on the path as $P(i, j)$. Now, notice if $i = 1$, there is only node in the system at distance 1 (along the path) from the adversary: the previous hop R to the first adversary. For $i \geq 2$, the source could be any honest node in the network with equal probability (again using symmetry). Thus, the adversary strategy is to assign a probability of $P(1, j) + (1 - P(1, j))/n$ to the previous hop R , and a probability of $(1 - P(1, j))/n$ to each of the other $(n - 1)$ honest nodes in the system. Now, notice from the equations, that $P(1, j) > 0$ for any distribution ($j \geq 1$, and $P(1, j) = 0$ only if $h(0) = 1$ which means there is no anonymity). This means that the previous hop always has a higher probability assigned it than any other node in the system, and thus this probability distribution is not a uniform distribution over the honest nodes. This proves the required lemma.

Lemma 3. *In any relaying based scheme over a complete graph, given that a request is intercepted by the adversary, the source is more likely to be the previous hop to the adversary than any other honest node, provided $\alpha > \frac{1}{2} - \sqrt{\frac{1}{4} - \frac{1}{N}}$.*

Note that this lemma holds for all forwarding schemes, not just path-agnostic schemes. This lemma is a generalization of the proof in [7] (which however holds for all α): the lemma implies that the source has poor long-term anonymity provided the adversary fraction exceeds the specific value.

Denote the probability that the source is the previous hop to the adversary by P_h (the subscript h is a reminder that this probability depends on the distribution of the hop length). Now, by symmetry, the probability Q_h that some other honest node is the previous hop to the adversary can be written as: $Q_h = (1 - P_h)/(n - 1)$. Now, notice that P_h can be written as:

$$\begin{aligned} P_h &= \alpha + (1 - \alpha)Q_{h_2} \\ h_2(i) &= \frac{h(i+1)}{1 - h(1)} \end{aligned} \tag{2}$$

Note that h_2 is a transformed version of h : $h_2(i)$ is the probability that there are i remaining hops on the path, given that the path has at least 1 hop. The first equation follows by considering two cases. If the source forwards the request to an adversary,

then it is observed as the previous hop. If not, then observe that the next hop now behaves like a source whose hop length follows the distribution h_2 . In this case, the probability that the original source will be the previous hop is Q_{h_2} by symmetry. Now, notice that if $\alpha > 1/n$, then $P_h > 1/n$, which proves the required lemma. It can be verified that if $\alpha \in [\frac{1}{2} - \sqrt{\frac{1}{4} - \frac{1}{N}}, \frac{1}{2} + \sqrt{\frac{1}{4} - \frac{1}{N}}]$, then $P_h > 1/n$. However, observe from equation 2, that if $\alpha > 1/2$, then $P_h > 1/2 \geq 1/n$. Thus, combining the two inequalities, if $\alpha > \frac{1}{2} - \sqrt{\frac{1}{4} - \frac{1}{N}}$, then we have $P_h > 1/n$ which implies $P_h > Q_h$.

Now, we can prove the theorem by combining the two lemmas. Given that a request is first intercepted by j consecutive adversary, the probability P assigned to the source is given by:

$$P = P_h(P(1, j) + \frac{(1 - P(1, j))}{n}) + (1 - P_h)\frac{(1 - P(1, j))}{n}$$

The probability Q assigned to any other honest node is given by:

$$Q = Q_h(P(1, j) + \frac{(1 - P(1, j))}{n}) + (1 - Q_h)\frac{(1 - P(1, j))}{n}$$

Combining the two lemmas along with these two equations, implies that $P > Q$, which in turns proves that source is assigned a greater probability (in expectation) than any other honest node when there are j adversaries on the path. Since this inequality holds for all j , the required theorem follows. The corollary follows by observing the limit as $n \rightarrow \infty$. \square

B Chaumian Networks

In this section, we compare the anonymity provided on a complete graph by a probabilistic forwarding scheme like Hush, Crowds with that provided by a Chaumian Network. We have seen earlier Chaumian networks and Crowds both have poor long-term anonymity unlike Hush. We now provide a simple analysis for the short-term anonymity of Chaumian Network and then compare with probabilistic forwarding in Crowds (note that Crowds has already been compared to Hush). The purpose of this analysis is to illustrate the advantage of constant length relaying as in Chaumian networks over probabilistic forwarding. This analysis is similar to the general analysis in Appendix A. Consider a Chaumian network (a complete graph on N nodes) which uses a hop length of L . Let the fraction of adversaries be α and the number of honest nodes be n . Suppose an adversary A_{int} intercepts a request and the previous hop is R . As in the previous case, due to mixing, only adversaries immediately following A_{int} on the path (excluding the destination), say A_1, \dots, A_{j-1} , can provide useful information about the source. The adversary strategy then is as follows. Firstly, the probability $p_{1,j}$ that the initiator is a single hop away from it on the path can be shown to be:

$$\begin{aligned} p_{1,j} &= \Pr[\text{source is 1 hop away} | j \text{ adversaries on path}] = \frac{\Pr[\text{source is 1 hops away}, j \text{ adversaries on path}]}{\Pr[j \text{ adversaries on path}]} \\ &= \frac{\Pr[\text{first adversary at } 1^{st} \text{ hop}] \Pr[(j-1) \text{ adversaries beyond } 1^{st} \text{ hop}]}{\sum_{k=0}^{k=L} \Pr[\text{first adversary at } (k+1)^{th} \text{ hop}] \Pr[j \text{ adversaries on path} | \text{first adversary at } (k+1)^{th} \text{ hop}]} \\ &= \frac{\alpha \times (\alpha)^{j-1}}{\sum_{i=0}^{i=(L-j)} (1-\alpha)^i \alpha^j} = \frac{\alpha}{1 - (1-\alpha)^{(L-j+1)}} \end{aligned}$$

In this equation, $\Pr[j \text{ adversaries on path}]$ refers to the probability that there are $(j-1)$ consecutive adversaries following the first one on the path. Thus, due to symmetry, the adversary assigns a probability $p_{1,j} + (1 - p_{1,j})/(n-1)$ to the previous hop and $(1 - p_{1,j})/(n-1)$ to the other honest nodes. Now, the PoI assigned to the source S is given by:

$$\begin{aligned} PoI &= \frac{(1-\alpha)^L}{n} + \sum_{j=1}^{j=L} (F_j p_{1,j} + \frac{\Pr[j \text{ adversaries on path}](1 - p_{1,j})}{n-1}) \\ F_j &= \Pr[j \text{ adversaries in path}, S \text{ is previous hop to first adversary}] \end{aligned}$$

The above equation follows by conditioning on the number of consecutive adversaries following the first adversary. We lower bound F_j by α^j by neglecting the probability that the initiator can appear more than once in the path.

We used these equations to compare the the short-term anonymity of Crowds and Chaumian Networks in Figure 6. This figure plots the anonymity level of both networks (each containing 100 nodes) for varying hop lengths. The hop length L is specified within parentheses in the legend of the graph (for crowds, L is the average hop length). This illustrates the advantage of using a constant length path versus probabilistic forwarding. The intuition is that using a constant length has poor anonymity only

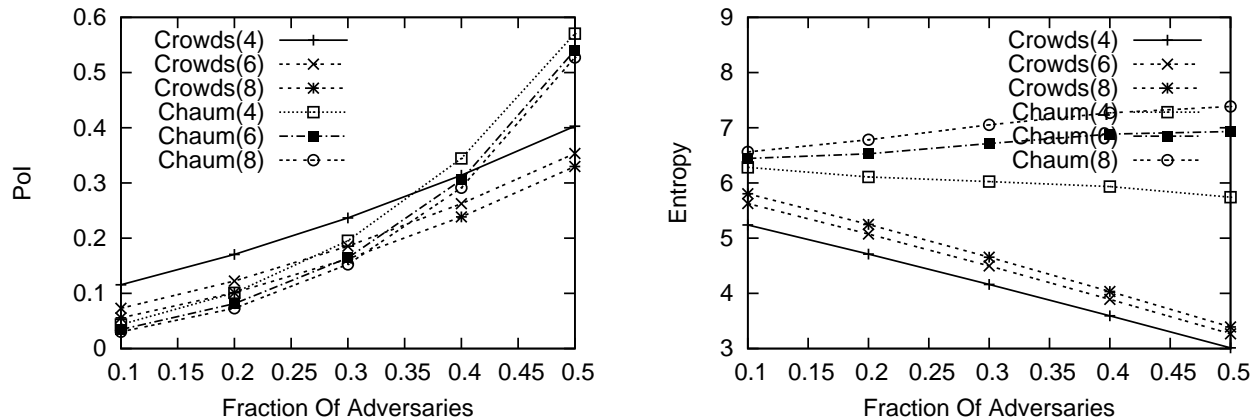


Figure 6: Crowds vs. Chaumian Networks (a) PoI Vs. Fraction of Attackers (b) Entropy Vs Fraction of Attackers

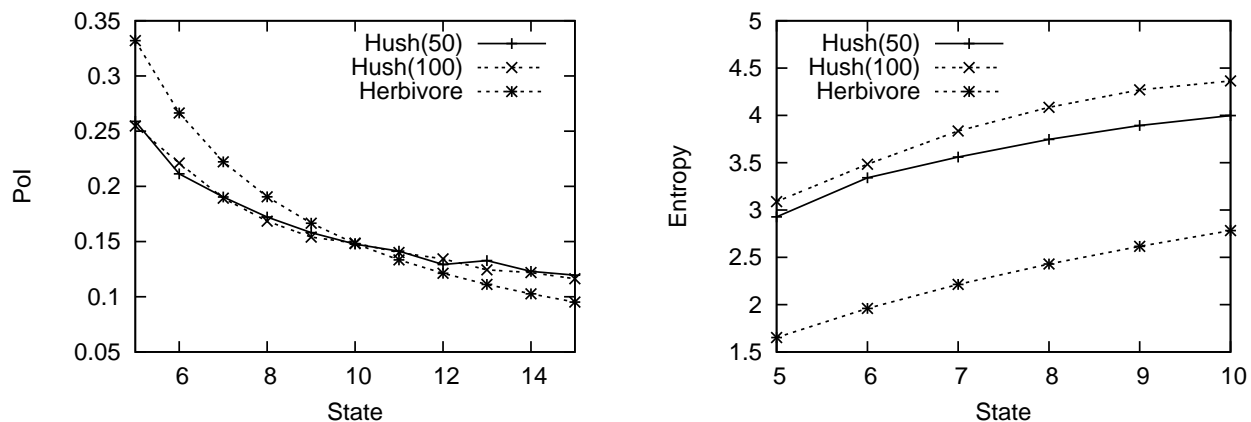


Figure 7: Hush Vs. Herbivore (a) PoI Vs. State (b) Entropy Vs State

if the first hop is an adversary: otherwise, the adversary has no information about the source. Using probabilistic forwarding however, the adversary can gain some information about the source irrespective of her position in the path. The entropy metric for Chaumian networks in fact appears to increase slightly with the fraction of adversaries: this is because of the nature of the metric itself. The entropy metric measures the degree of “randomness”, rather than the probability assigned to the source. Note however, that using a constant length path requires a PKI and complete knowledge of the topology.

C DC-Net Based Schemes

In this section, we compare the anonymity provided by Hush with that provided by a DC-Net based scheme, Herbivore. We first analyze the adversary strategy in Herbivore, and then compare it with Hush. The short-term anonymity analysis for Herbivore is quite simple. Consider a honest node A in a clique C of size k that initiates a request. It chooses a random neighbor in the clique say B , and forwards the request to her anonymously. B in turn forwards it to the destination D (who is an adversary). If there are j adversaries in the clique C , then the probability assigned to A is simply $1/(k - j + 1)$ (since B cannot be the initiator and neither can the adversaries). Thus, the PoI of the source can be computed by conditioning on the number of adversaries in the clique and using the above formula.

We have compared Hush and Herbivore by simulating the adversary strategy in Figure 7. To ensure that both systems use the same state, the clique size in Herbivore was chosen to be k , the number of neighbors maintained per node in Hush is $(k - 1)$. and the forwarding probability in Hush is chosen such that the average number of hops is k . The second requirement follows since assuming there are no collisions involved in the clique, k nodes are involved in the transmission in Herbivores. In particular, the bandwidth is limited by the slowest of these k nodes, and the same restriction is imposed in Hush, by requiring that the request be forwarded through k hops. The figures shows two plots for Hush, for graphs of size 50, 100 respectively. The results show that with respect to short-term anonymity, Hush outperforms Herbivore especially with respect to the entropy metric. The main reason is that in Herbivore, irrespective of the number of nodes in the system, the anonymity level is set at a constant value k . If

one clique forwards a request to the destination, then only nodes within that clique are suspect. There is no forwarding between nodes in different cliques. Hush on the other hand spreads the probability along all the nodes (to varying extents) and thus has better entropy metric in the short-term anonymity. Also, notice that the PoI in Hush is nearly independent of the number of nodes, while the entropy metric anonymity increases with the number of nodes. Herbivore's long-term anonymity is the same as its short term anonymity, since the destination sees requests originating from the same clique. A small modification is required in the proposal in [6] to achieve this: the initiator should choose the next hop from all the nodes in the clique uniformly at random (including itself). We were not able to compute an efficient adversary strategy for the long-term anonymity in Hush, and hence cannot compare it with Herbivore on that aspect.