Slicing the Onion: Anonymous Routing Without PKI

Sachin Katti
skatti@mit.edu

Dina Katabi
dk@mit.edu

Katarzyna Puchala
kpuchala@mit.edu

Abstract– Recent years have witnessed many proposals for anonymous routing in overlay peer-to-peer networks. The proposed protocols either expose the receiver and the message content, or require the overlay nodes to have public-private key pairs with the public keys known to everyone. In practice, however, key distribution and management are well-known difficult problems and have crippled any widespread deployment of anonymous routing. This paper uses a combination of information slicing and source routing to provide anonymous communication in a way similar to Onion Routing but without a public key infrastructure (PKI).

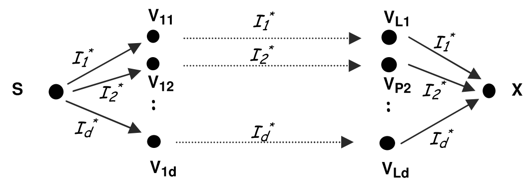

Figure 1—Node S sends a confidential message to X by splitting the information content into multiple pieces, each follows a disjoint path to X. Only X receives enough information bits to decode the original message.

1 INTRODUCTION

Anonymous routing plays a central role in private communication. Its applications range form file sharing to military communication, and include anonymous email, private web browsing and online voting. Traditionally, anonymous routing has required the help of a trusted third party, which either acts as a centralized proxy [1, 3], or provides the sender with the public keys of a list of willing relays [2, 9]. However, the recent success of peer-to-peer systems has evoked interest in using them as anonymizing networks. Indeed, the large number of nodes (a few millions [16]) and the heterogeneity of their location, communication patterns, political background and local jurisdiction make these networks ideal environments for hiding anonymous traffic. Many systems have been designed to exploit peer-to-peer overlays in anonymous communication, including Tarzan [11], AP3 [17], MorphMix [19] and Cashmere [22]. However, these systems either expose the receiver and message content (Crowds [18]), or require a trusted public key infrastructure (PKI) to distribute the public keys of each node in the peer-to-peer network.

But why is PKI problematic for peer-to-peer anonymizing networks? The first issue is key distribution [4]. Prior work assumes the sender knows a priori the public keys of all relay nodes, but does not elaborate on how they are obtained [11, 17, 22]. Limiting an anonymous routing overlay to nodes that know each others' public keys via an out-of-band channel results in very small overlays that cannot hide the identity of the communicators. One may assume that a trusted third party generates all keys and distributes them to the nodes, a constraint hard to satisfy in a large peer-to-peer network, where the trust model may differ from one node to another. Also, it opens up the system to attacks on the key distribution procedure and compulsion attacks[1] that force the key originator to disclose the keys under the threat of force or if required by a court order [13, 14]. Indeed, some countries have provisions that allow them to legally request the decryption of material or the handing over of cryptographic keys [5, 10]. Additionally with time, an increasing fraction of the keys can get stolen off the hard disk of compromised machines. This necessitates the existence of key management and update protocols, complicating the problem further. Finally, PKI makes anonymous multicast difficult as all recipients of a multicast message have to share the same public private key pair.

This paper shows how to perform *Onion Routing* without public key cryptography. Onion Routing [12] is at the heart of most prior work on peer-to-peer anonymizing networks [9, 11, 17, 22]. It uses a form of source routing, in which the IP address of each node along the path is encrypted with the public key of its previous hop. This creates layers of encryption–layers of an onion. To send a message, each node decrypts one layer, discovers its next hop, and forwards the message. Thus each relay node knows only its previous and next hops; it cannot tell the sender, the receiver, the path, or the content of the message. Our scheme provides similar anonymity but without PKI.

Our approach is based on the simple but powerful idea of *Information Slicing*. To provide anonymous communication, each node along the path, the destination included, needs a particular piece of information, which should be hidden from other nodes in the network. For example, the destination needs to learn the content of the message without revealing that content to other nodes, while each intermediate relay needs to learn its next hop without other nodes in the network knowing that information. We divide the information needed by a particular node into many small random pieces. These information pieces are then delivered along disjoint paths that meet only at the intended node. Thus, only the intended node has enough bits to decode the information content. We call this approach information slicing because it splits the information traditionally contained in an onion peel (i.e., the ID of the next hop) into multiple pieces/slices.

Anonymity via slicing is not as straightforward as it sounds. To send a particular node the identity of its next hop along different anonymous paths, one needs to anonymously tell each node along these paths about its own next hop. Without careful design, this may need an exponential number of paths. Our keyless onion routing algorithm provides efficient information slicing using a small constant number of paths.

Apart from being keyless, our approach has the following additional advantages. It provides high degree of anonymity close to Chaum [7] mixes. It is also computationally efficient and can

[1]On the day of the paper submission deadline, August 1, a New York Times article detailed compulsion attacks on various anonymous filesharing services [14]!

address network churn and node failures.

2 Goals & Model

The objective of this work is to enable large and fully distributed peer-to-peer anonymizing networks. We focus on pragmatic anonymity for non-military applications, such as file sharing, private email and the communication of medical records. These applications strive for privacy but can deal with low probability of information leakage.

We assume an adversary who can observe some fraction of network traffic, operate relay nodes of his own, and can compromise some fraction of the relays. We do not protect against a global attacker who can snoop on all links. Though such an adversary is usually assumed when analyzing theoretical anonymity designs, all practical low-latency anonymizing systems, ours included, do not protect against such an adversary [9, 11, 17, 19, 22]. Also, similar to prior work [9, 11, 17, 22], we generate enough cover traffic to prevent simple traffic analysis attacks.

We also assume the sender can send from multiple IP addresses, and a secure channel like `ssh` is available between them. Many people have Internet access both at home and at work/school, and thus, can send from different IP addresses. Alternatively, the sender may have both DSL and cable connectivity. Or, he may belong to a multi-homed organization. For example, each of the authors has Internet access at home, as well as at school and on Planetlab machines. We believe that a large number of Internet users can send from multiple accounts with different IP addresses. An attacker may try to correlate IP addresses belonging to the same sender. However, in all of the examples above the IP addresses used belong to different domains. Additionally, most broadband providers and companies utilize NAT, preventing the association of an IP address with a particular user.

Last, we assume either the sender knows the receiver's key, *or* the attacker cannot snoop on all links leading to the receiver.

3 Example of Anonymous Routing with Information Slicing

We start with an example, while leaving the details of our routing protocol to §4. In onion routing, a node learns its next hop from its parent. Though the parent delivers this information to its child, it cannot access it because the information is encrypted with the child's public key. In the absence of keys, the path cannot be included in the message as that allows any intermediate node to learn the whole path from itself to the receiver. We need an alternative method to tell a node about its next hop without revealing it to other nodes, particularly the parent node.

How to preserve anonymity without a PKI? Fig. 2 shows an example keyless anonymous routing graph. Assume the sender has access to two IP addresses S and S'. To send an anonymous message to node R, the sender, in Fig. 2, has picked a few relay nodes at random. It has arranged them, with the receiver, into 3 stages (path length $L = 3$), each containing 2 nodes (split factor $d = 2$). The 0'th stage is the source stage itself. Each node in this graph is connected to every node in its successive stage. Also, note that the receiver node (the solid node labeled R) is randomly assigned to one of the stages in the graph.

The sender in Fig. 2 wants to send each relay the IP address of its next hop by splitting this information over 2 paths. The sender could have split each IP address to its most significant and least significant words. This however is undesirable as most significant word may indicate the owner of the IP prefix. Instead the sender transforms the IP addresses of the relay nodes by multiplying each address by an *invertible matrix* A of size $d \times d$. For example, assume V_l and V_h are the the low and high words of the IP address of node V; the sender splits the IP address as follows:

$$\begin{pmatrix} V_L \\ V_H \end{pmatrix} = A \begin{pmatrix} V_l \\ V_h \end{pmatrix} \quad (1)$$

and sends V_L and V_H to V's parents along two different paths.

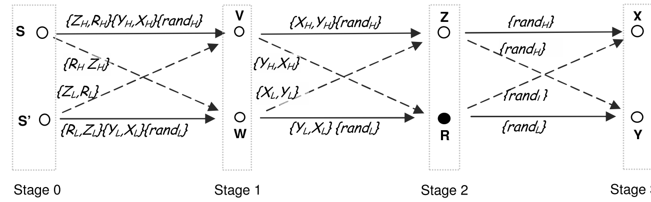

Figure 2—An example of anonymous routing with information slicing. Nodes S and S' are controlled by the sender. A message like $\{Z_l, R_l\}$ refers to the low-order words of the IDs of nodes Z and R, *rand* refers to random bits.

Fig. 2 shows how messages are forwarded such that each node knows no more than its direct parents and children. Consider an intermediate node in the graph, say V. It receives the message $\{Z_H, R_H\}\{X_H, Y_H\}\{rand_H\}$ from its first parent S. It receives $\{Z_L, R_L\}$ from its second parent S'. After receiving both messages, V can discover its children's IP addresses as follows:

$$\begin{pmatrix} Z_l & R_l \\ Z_h & R_h \end{pmatrix} = A^{-1} \begin{pmatrix} Z_L & R_L \\ Z_H & R_H \end{pmatrix} \quad (2)$$

But V cannot tell the children of its children (i.e., the children of nodes Z and R) because it misses half the bits in these addresses, nor does it know the rest of the graph. The same argument applies to other nodes in the graph.

You might be wondering how the graph in Fig. 2 will be used to send the actual message to node R. Indeed, as it is, R does not even know it is the intended receiver. But this is easy to fix. In addition to sending each node its next hop IPs, we send it: (1) a key and (2) a flag indicating whether it is the receiver. Similar to the next hop, the key and the flag are also split along disjoint paths, and thus inaccessible to other nodes. All keys are useless/invalid except for the receiver's key (the key at node R). Now every node along the path knows its next hops. Further, the receiver shares a secret key with the sender. The sender encrypts its message with the receiver's key, splits the message as before and sends it on the forwarding graph. All relay nodes can see the encrypted message but only the receiver will be able to decrypt it.

4 An Information Slicing Protocol

We use the intuition from the previous section to construct an anonymous routing protocol based on information slicing.

(a) Per-Node Information: Let x be one of the nodes in the forwarding graph. I_x is the information the sender needs to *anonymously* deliver to node x. I_x consists of the following fields:

- *Nexthop IPs.* The IP addresses of the d children of node x.

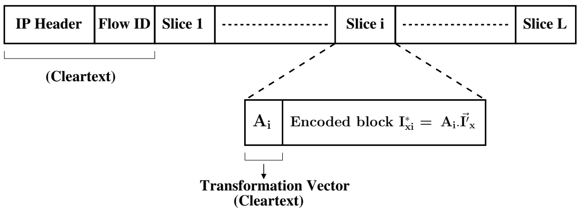

Figure 3—Packet Format. Each packet contains L information slices.

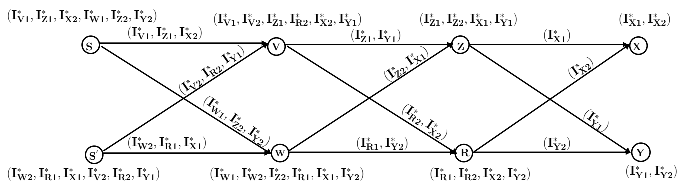

Figure 4—An example showing how to split information slices along disjoint paths. R is the receiver, S and S' are the senders.

Algorithm 1 Information Slicing Algorithm

Pick Ld nodes randomly including the destination
Randomly organize the Ld nodes into L stages of d nodes each
for Stage $l = L$ to $l = 0$ **do**
  for Node x in stage l **do**
    Assign to node x its own slices $I^*_{xk}, k \in (1, \ldots, d)$.
    for Stages $m = l - 1$ to $m = 1$ **do**
      Distribute slices $I^*_{xk}, k \in (1, \ldots, d)$ uniformly among the d nodes in stage m, assigning one slice per node
    end for
  end for
  Connect every node in stage $l-1$ to every node in stage l by a directed edge going towards l
  for every edge e **do**
    Assign the slices which are present at both the nodes at the endpoints of the edge e to the packet to be transmitted on e.
  end for
end for

- *Nexthop flow-ids.* These are d 64-bit ids whose values are picked randomly by the sender and are to be put in the clear in the packets going to the corresponding d next-hops. The sender ensures that different nodes sending to the same next hop put the same flow-id in the clear. This allows the next-hop to determine which packets belong to the same flow. The flow-id changes from one relay to another to prevent the attacker from detecting the path by matching flow-ids.
- *Receiver Flag.* This flag indicates whether the node is the intended receiver.
- *Secret Key.* The sender sends each node along the path a secret key which can be used to encrypt any further messages intended to this node. If the receiver flag is set, the source will encrypt the data intended for the receiver using this key.

(b) Creating Information slices: The node information I_x is chopped into d blocks of $\frac{|I_x|}{d}$ bits each and a d length vector $\vec{I'_x}$ is constructed. Further, $\vec{I'_x}$ is transformed into coded *information slices* using a full rank $d \times d$ random matrix A as follows[2]

$$\vec{I^*_x} = \begin{pmatrix} A_1^T \\ \vdots \\ A_d^T \end{pmatrix} \vec{I'_x} = A\vec{I'_x} \tag{3}$$

We call the elements in $\vec{I^*_x}$ *information slices*. We also add to information slice I^*_{xi} the row of the matrix A which created it i.e. A_i. The sender delivers the d slices to node x along disjoint paths.

(c) Packet Format Fig. 3 shows the format of a packet used in our system. In addition to the IP header, a packet has a flow id, which allows the node to identify packets from the same flow and decode them together. The packet also contains L slices. The first slice is always for this node (i.e., the receiver of the slice). The other slices are for nodes downstream on the forwarding graph.

(d) Constructing the Forwarding Graph: The sender constructs a forwarding graph which routes the information slices to the respective nodes along vertex disjoint paths, as explained in Algorithm 1. We demonstrate the algorithm by constructing such a graph in Fig. 4, where $L = 3$ and $d = 2$. We start with the 2 nodes in the last stage, X and Y. The sender assigns both the slices, I^*_{X1}, I^*_{X2} to X. Then it goes through the preceding stages, one by one, and distributes (I^*_{X1}, I^*_{X2}) among the 2 nodes at each stage; each node receives one of the slices. The path taken by slice I^*_{X1} to reach X can be constructed by tracing it through the graph. Slice I^*_{X1} traverses (S', W, Z, X), which is disjoint from the path taken by I^*_{X2}, i.e., (S, V, R, X). The source repeats the process for the slices of Y and every other node in every stage.

Slices are delivered in packets transmitted between nodes in successive stages. The slices a node sends to its downstream neighbor are the intersection of the sets of slices assigned to both nodes by Algorithm 1. E.g., for edge (V, R), the slices (I^*_{R2}, I^*_{X2}) are present at both nodes V and R. These slices are contained in the packet transmitted from node V to node R. The source determines the packet contents for every edge in the graph. The algorithm thus ensures that slices belonging to a node take vertex disjoint paths to the node.

(e) Decoding the Information slices: A node can decode its information from the d slices it receives from its parents. The first slice in every packet x receives is for itself. It consists of one of d-slices of x's information, I^*_{xi}, and the row of the transform matrix that helped create it, A_i. Node x constructs the $d \times 1$ vector $\vec{I^*_x}$ from the d slices it receives, and assembles a $d \times d$ matrix $A = [A_1 \ldots A_d]^T$ from the d transform rows in the slices. It then computes $\vec{I'_x}$ by inverting the matrix A i.e. $\vec{I'_x} = A^{-1}\vec{I^*_x}$. The node can recover its information from $\vec{I'_x}$ by concatenating the elements of the vector.

(e) Data Transmission: After the forwarding graph has been setup, the source first encrypts the data it wants to send to the receiver with the secret key it has assigned to it. Then it splits the message into d fragments, which it sends down the forwarding graph, as before. Since no other node knows the key used to encrypt the message, only the receiver can decrypt the data.[3]

5 Robustness to Churn and Traffic Analysis

(a) Resilience to Bitwise Linkability: Bitwise unlinkability ensures that input and output messages 'look' different. Thus, an attacker cannot identify a connection by matching the bits of the

[2] Elements of $\vec{I'_x}$ and A belong to a finite field F_{p^q} where p is a prime number and q is a positive integer. All operations are therefore defined in this field and differ from conventional arithmetic.

[3] Alternatively, once the forwarding graph has been set up and every node has its key, the source could use plain onion routing to transmit its messages.

Var	Definition
d	Split factor, i.e., the number of fragments a message is split to.
L	Path length, i.e., the number of relays stages along a path.
N	Number of nodes in the peer-to-peer network excluding the source stage.
f	Fraction of subverted nodes in the anonymizing network.
s	the maximum number of successive stages in the forwarding graph, whose nodes are known to the attacker.
S	The set of nodes in the s stages.

Table 1—Variables used in the paper.

incoming and outgoing packets at a node. We achieve this by making each relay node x multiply each information slice it receives with a random number p_x of its choice. In [15] we prove[4] that:

LEMMA 5.1. *Though each relay multiplies the information slices it receives with a random number of its choice, a node along the path can still recover its information without knowing the random multipliers used by its upstream parents.*

(b) Maintaining Constant Packet Size: Fig. 4 shows a clear deficiency: The number of slices in a packet decreases along successive relays, allowing the attacker to analyze the position of a relay on the graph by observing the packet size. To prevent this attack, we fix the number of slices in a packet to L. Unused slice slots are padded with random bits. Furthermore, except for the first slice in the packet, which contains information intended for the relay node itself, the source node is free to shuffle the arrangement of slices in the packets transmitted to the next hops. To do so, the source anonymously tells each relay how to place the slices in the outgoing packet and where to add random padding. The source includes a bitmap, the *splitting vector* for every outgoing packet. The vector specifies which incoming information slice should be placed in which slot of the outgoing packet. One slot in each outgoing packet is kept free for random padding. The splitting vectors are part of the per-node information and can only be recovered by the relay node itself. The source picks a shuffling which ensures that the first slice contains information for the recipient of the packet but is free in rearranging other slices.

(c) Resilience to Churn and Failures: Instead of slicing the per-node information into d independent pieces which are all necessary for decoding, we use $d' > d$ dependent slices. Replace Eq. 3 with:

$$\vec{I_x^*} = A'\vec{I_x'} \tag{4}$$

where A' is a $d' \times d$ matrix with the property that any d rows of $A^{'}$ are linearly independent. The source picks d' disjoint paths to send the message. The intended node can recover its information from any d out of d' slices that it successfully receives. Hence we can tolerate $d^{'} - d$ node failures at each stage.

6 SECURITY ANALYSIS

Instead of standard key-based encryption, our scheme uses information slicing. To understand the security obtained with such encryption, we estimate the amount of information a malicious node can glean from the messages it receives. We borrow the following definition from [6, 21].

Definition A function f is packet independent (pi)-secure if for all v and a uniformly distributed message block $\vec{x} = [x_1, x_2, \ldots, x_n]$ $Pr(x_i = v) = Pr(x_i = v|f(\vec{x}))$.

LEMMA 6.1. *Our information slicing algorithm is* pi-*secure.*

The proof[4] is in [15]. In our case $f(\vec{x})$ represents any set of atmost $(d-1)$ coded information slices. A *pi*-secure information slicing algorithm implies that to decrypt a message, an attacker needs to obtain all d information slices; partial information is equivalent to no information at all.

7 ANONYMITY ANALYSIS

We would like to understand the degree of source and destination anonymity provided by our scheme and its dependence on parameters like L, d, and fraction of subverted nodes in the network, f. To simplify the analysis, we assume that L is constant and known to the attacker. We also assume the source picks the relays randomly from the set of all nodes in the network, and every node appears only once in the anonymity graph. These assumptions degrade anonymity, making the results lower bounds. We evaluate the anonymity using a combination of analysis and simulation. We use 1000 different random assignments of malicious nodes to estimate $s = g(L, d, N, f)$, the maximum number of consecutive relay stages known to the attacker (the attacker knows the IPs of the nodes in these stages). Given a value of s, we have closed-form solutions for the anonymity of the source and destination, as explained in §7.2 and §7.3.

7.1 Anonymity Metric

We define the anonymity of a system as the amount of information the attacker is missing to uniquely identify an actor's link to an action—e.g., uniquely identify the sender or the destination of a message. The anonymity of a system is typically measured by its entropy [20, 8],[5] and is usually expressed in comparison with the maximum anonymity possible by such a system, i.e.:

$$Anonymity = \frac{H(x)}{H_{max}} = \frac{\sum_x -P(x)log(P(x))}{log(N)}, \tag{5}$$

where N is the total number of nodes in the network and $P(x)$ is the probability of a node being the source/destination, and $H_{max} = log(N)$ is the maximum entropy which occurs when the attacker has no information. For example, the source is perfectly anonymous when it is equally likely to be any node in the network, in which case $P(x) = \frac{1}{N}$ and the $Anonymity = \frac{H(x)}{H_{max}} = 1$.

7.2 Source Anonymity

Source anonymity depends on the probability of attackers identifying the nodes in stage 0 (i.e. the sender stage) since they know it is controlled by the source. We distinguish two cases:

Case 1: All nodes in stage 1 are malicious. In this case, the attacker can decode the entire graph, discover she controls the first stage, and thus the previous stage has to be the source stage. The probability of Case 1 occurring is very low, $P(Case1) = f^d$, but the anonymity of the source is 0.

[4]Due to space constraints the proofs of Lemmas 5.1 and 6.1 can be found in our technical report [15] at http://nms.lcs.mit.edu/∼sachin/slicing.html

[5]The entropy of a random variable x is $H(x) = -\sum_x P(x)log(P(x)$, where $P(x)$ is the probability function.

Case 2: Some nodes in stage 1 are not malicious. Although the attacker cannot decode the entire graph, she still knows about many nodes in the graph. Since flow-ids change every hop, malicious nodes can collude only when they are in successive stages in the graph; otherwise they would not know whether they belong to the same forwarding graph. Assume s is the largest number of successive stages known to the attacker. The attacker's best guess is to consider the nodes in the first stage in the chain s to be the source stage. The first stage necessarily has no malicious nodes, since if it did the previous stage would be known to the attackers and s would not be the longest chain. Let Γ be the set of nodes in the first stage in the chain s. The probability the first stage the attacker knows about is stage 0 is $\frac{1}{L-s}$.[6] Thus, if $x \in \Gamma$, then $P(x = src) = \frac{1}{L-s}$. The rest of the probability is divided equally between non-malicious nodes $\notin \Gamma$. The number of such nodes is $N(1-f) - |\Gamma|$. Thus, the probability a node x is the source:

$$P(x = src) = \begin{cases} \frac{1}{(L-s)} & x \in \Gamma \\ (1 - \frac{1}{L-s})\frac{1}{N(1-f)-|\Gamma|} & \text{otherwise} \end{cases} \quad (6)$$

The length of the chain s is estimated via simulation. Anonymity can then be easily computed using Eq. 5.

7.3 Destination Anonymity

Destination anonymity depends on the probability the attacker assigns to each node being the destination. In contrast to the source, the destination can be at any stage $i > 0$. Again, we distinguish two cases:

Case 1: All the nodes in some stage i upstream of the destination are attackers. The attacker can decode the downstream graph and discover the intended destination. Assume the destination is in stage $j+1$. Then the probability that an entire stage before stage $j+1$ consists of attacker nodes is given by $\binom{j}{1}f^d$. Since the destination could be in any stage with equal probability $1/L$, the overall probability is given by

$$P(Case1) = \frac{1}{L}\sum_{1\leq j\leq (L-1)}\binom{j}{1}f^d = (\frac{L-1}{2})f^d. \quad (7)$$

The probability of Case 1 occurring is low, but when it occurs, the anonymity is 0.

Case 2: When the attacker cannot decode the part of the graph containing the destination, she can still try to infer the destination from among the nodes it knows to be on the graph. Let s be the largest number of consecutive stages whose nodes are known to the attacker. Call the set of nodes in these s stages S. There are sd nodes in S, among which $sd(1-f)$ nodes are non-malicious. Since the destination can be in any stage in the graph, the probability it is in S is $\frac{s}{L}$. Each non-malicious node $x \in S$ is equally likely to be the destination, $P(x = dst) = \frac{s}{L}\frac{1}{sd(1-f)} = \frac{1}{Ld(1-f)}$. The remaining probability is divided equally among the $(N-sd)(1-f)$ non-malicious nodes outside S. Thus:

$$P(x = dst) = \begin{cases} \frac{1}{Ld(1-f)} & x \in S \\ (1 - \frac{s}{L})\frac{1}{(N-sd)(1-f)} & x \notin S \end{cases} \quad (8)$$

Given $P(x = dst)$, destination anonymity is computed using Eq. 5.

[6]Note that the total number of stages including the source stage is L+1. The attacker knows s stages, out of which the last $s-1$ cannot be the source stage.

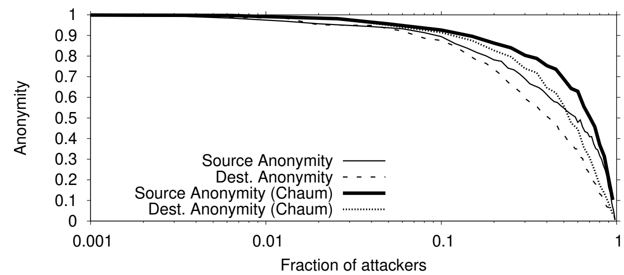

Figure 5—Source and destination anonymity as functions of the fraction of malicious nodes in the network ($N = 10000, L = 8, d = 3$)**.**

7.4 Simulations

We complement the analysis in §7.2 and §7.3 with simulation. The analysis is for a particular s, but s will change depending on the assignment of malicious nodes and the parameters of the system. We use a large number of simulations to discover the distribution of s. In each simulation, we randomly pick Nf nodes to be controlled by the attacker. Then we pick Ld nodes randomly and arrange them into L stages of d nodes each. We randomly pick the destination out of the nodes on the graph. We then identify the malicious nodes in the graph and analyze the part of the graph known to attacker, as follows. First, we check if we are in Case 1, which results in zero anonymity. If we are not in Case 1, we compute the probabilities of each node being the source or the destination according to Eqs. 6 and 8.[7] Given, this probability we compute the anonymity using Eq. 5. The procedure is repeated 1000 times and the average anonymity is plotted. We explore how anonymity changes with the various parameters.

(a) Fraction of Malicious Nodes: Fig. 5 plots the anonymity of the source and destination as functions of the fraction of attackers in the graph, for the case of $N = 10000, L = 8, d = 3$. The anonymity is very high when less than 20% of the nodes in the network are malicious. As the fraction of malicious nodes increases beyond 50%, the anonymity falls. Destination anonymity drops faster with increased f because discovering the destination requires the attacker to control any stage upstream of the destinations, while discovering the source requires the attacker to control stage 1, in particular. The figure also compares the anonymity in the information slicing scheme with that in Chaum Mixes [7], showing that despite the lack of PKI, the anonymity in our scheme is close to that in Chaum Mixes and similar to other practical peer to peer anonymizing systems [22].

(b) Splitting Factor: Fig. 6 plots source and destination anonymity as functions of the splitting factor. When f is low information leakage is primarily due to the malicious nodes knowing their neighbors on the graph, i.e., Case 2. In this case, increasing d increases the exposure of non-malicious nodes to attackers which results in a slight loss of anonymity. When f is high, information leakage is mainly due to attackers being able to compromise entire stages, i.e., Case 1. Hence, increasing d increases anonymity. Note that anonymity of 0.5 implies that attackers are missing half the information needed to decode the graph. Given that the size of the message used to describe the graph is large, attackers will

[7]Equations 6 and 8 assume the number of malicious nodes in S is equal to its expectation. In this section, we compute Anonymity by using the actual number of malicious nodes in S, in each simulation, and then averaging over 1000 simulations.

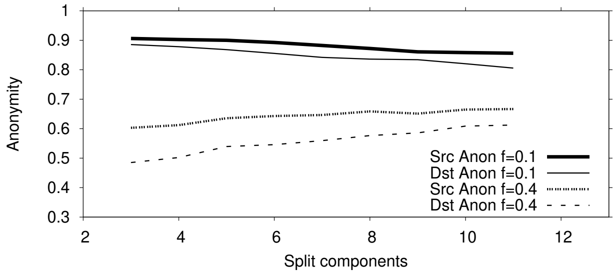

Figure 6—Source and destination anonymity as functions of the splitting factor ($N = 10000, L = 8$). For small f, increasing d decreases anonymity because it exposes more nodes to the attacker. For large f, the probability that attackers control an entire stage dominates (i.e., Case 1), hence increasing d increases anonymity. Anonymity of 0.5 is still quite high since the attackers are missing half the information necessary to decode the graph.

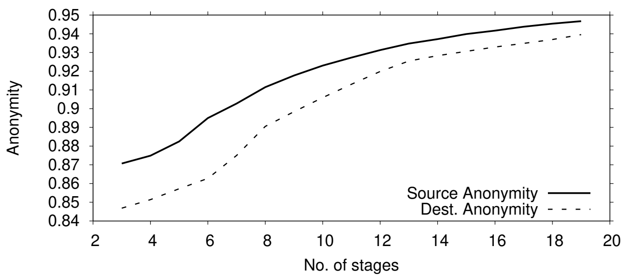

Figure 7—The anonymity of the source and destination increases with the path length ($N = 10000, d = 3, f = 0.1$).

Route Length & Split factor	Setup Latency (ms)	Standard Deviation (ms)
L=1, D=2	11.59	1.88
L=2, D=2	39.05	4.20
L=3, D=2	61.14	9.33
L=4, D=2	89.86	7.56
L=5, D=2	109.12	11.09

Table 2—Setup latency in milliseconds and its variance for the construction of multi-hop routes through pre-defined relays.

not have enough information when anonymity is 0.5.

(c) Path Length: Fig. 7 plots source and destination anonymity as functions of the path length L. Both source and destination anonymities increase with L. The attacker knows the source and destination have to be on the graph; putting more nodes on the graph allows the communicators to hide among a larger crowd.

8 PERFORMANCE

We have implemented our scheme in Python, and performed preliminary tests on a 100 Mbps switched network with the tested relay daemons running on 2.8 GHz Pentium boxes with 1 GB of RAM and a Linux 2.6.11 kernel. Table 2 shows route setup latency for different path lengths and split factor of 2. Setup latency is measured end-to-end from when the sender initiates route establishment, connects to the stage-1 relay processes, which process the next hop computation, store the forwarding information, and connect to their next hop relays, which repeat the procedure until all routing messages reach the receiver. On average, we incur a setup cost of 19 ms per hop. This figure suggests that the latency of the underlying network will dominate even during route setup. The table shows that the route setup latency incurred by our scheme is comparable to other anonymous routing protocols such as Tarzan [11], and is low enough to make it practical.

9 CONCLUSION

We have shown it is possible to design anonymizing peer-to-peer overlays that do not need a public key infrastructure (PKI). Our information slicing protocol can hide the source, the destination, the path, and the content of the message, even when the sender does not have the public keys of the nodes in the overlay. We believe this is an important step towards truly peer-to-peer anonymous communications; it obviates the need for a universal trusted PKI and avoids the difficulties of large scale key distribution in a global peer-to-peer network.

REFERENCES

[1] Anonymizer- Anonymous Web Surfing. http://www.anonymizer.com.

[2] MixMinion- Anonymous Remailer. http://www.mixminion.net.

[3] Safeweb- Anonymous Web Surfing. http://www.safeweb.com.

[4] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *CRYPTO '93*.

[5] Bob Sullivan. FBI software cracks encryption wall. www.msnbc.com/news/660096.asp?cp1=1.

[6] J. Byers, M. C. Cheng, J. Considine, G. Itkis, and A. Yeung. Securing Bulk Content Almost for Free. Journal of Computer Communications, Special Issue on Network Security.

[7] D. L. Chaum. Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms. Commun. ACM 21, 2(1981).

[8] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In *Proceedings of PET 2002*.

[9] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security 2004*.

[10] Foundation for Information Policy Research. Regulation of Investigatory Powers Information Centre. www.fipr.org/rip/.

[11] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of ACM CCS 2002*.

[12] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Hiding Routing Information. In *Proceedings of Information Hiding: First International Workshop*, 1996.

[13] Johan Helsingius. anon.penet.fi is closed! www.penet.fi.

[14] John Markoff. New File-Sharing Techniques Are Likely to Test Court Decision. The New York Times, Aug 1.

[15] S. Katti, D. Katabi, and K. Puchala. Slicing the Onion: Anonymous Routing Without PKI. MIT CSAIL Technical Report, http://nms.lcs.mit.edu/ sachin/slicing.html.

[16] Kazaa. http://www.kazaa.com/.

[17] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. Wallach. Ap3: A cooperative, decentralized service providing anonymous communication. In *Proceedings of the 11th ACM SIGOPS European Workshop*, September 2004.

[18] M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), June 1998.

[19] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *WPES 2002*, Washington, DC, USA.

[20] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *PET 2002*.

[21] D. Stinson. Something About All Or Nothing Transforms. In *Designs, Codes and Cryptography, 2001*.

[22] L. Zhuang, F. Zhou, B. Y. Zhao, and A. Rowstron. Cashmere: Resilient anonymous routing. In *NSDI 2005*.

APPENDIX

Proof of Lemma 5.1: Let the transformed information slices received at x be $(p_1 I^*_{x1}, \ldots, p_d I^*_{xd})$ where p_i represents the cumulative product of random numbers with which I^*_{xi} was multiplied along the path. The corresponding transformation code vector A_i is also multiplied with the same number p_i. Hence node x receives the following slices

$$\begin{pmatrix} p_1 I^*_{x1} \\ \vdots \\ p_d I^*_{xd} \end{pmatrix} = \begin{pmatrix} p_1 A_1 \\ \vdots \\ p_d A_d \end{pmatrix} \vec{I'}_x \tag{9}$$

Multiplying both sides of the original equation with the same invertible diagonal matrix

$$\begin{pmatrix} p_1 & 0 & 0 & \ldots \\ 0 & \ddots & 0 & \ldots \\ 0 & 0 & \ddots & 0 \\ 0 & \ldots & 0 & p_d \end{pmatrix} \begin{pmatrix} I^*_{x1} \\ \vdots \\ I^*_{xd} \end{pmatrix} = \begin{pmatrix} p_1 & 0 & 0 & \ldots \\ 0 & \ddots & 0 & \ldots \\ 0 & 0 & \ddots & 0 \\ 0 & \ldots & 0 & p_d \end{pmatrix} \begin{pmatrix} A_1 \\ \vdots \\ A_d \end{pmatrix} \vec{I'}_x \tag{10}$$

reduces the original transformation to Eq. 9. Thus both the transformations are equivalent and have the same solution. Since the original transformation A is invertible and $\vec{I'}_x$ can be recovered from that, the new transformed slices are equivalent which means that $\vec{I'}_x$ can be recovered by x from the received slices.

A PROOF OF LEMMA 6.1

Proof: Let $\vec{x} = [x_1, x_2, \ldots, x_n]$ be the original message. The m messages received at node i can be written as $A\vec{x} = \vec{b}$ where A is a $m \times n$ matrix, $\vec{b}$ is a m length vector and $m < n$. Pick $(m - n)$ components of $\vec{x}$ and set them to arbitrary values $\vec{v}$ and set the rest of the components of $\vec{x}$ to 0. Let this vector be $\vec{x'}$. Compute $\vec{b'} = \vec{b} - A\vec{x'}$. Eliminate the columns in A corresponding to the components of $\vec{x}$ which were set to arbitrary values. Let the resulting matrix be A'. A' is a $m \times m$ matrix of full rank since the messages received at the node are all independent of each other. Hence the matrix A' is invertible and therefore a unique solution to the equation $A'\vec{x'} = \vec{b'}$ exists. Hence for any arbitrary values $\vec{v}$ of the $(m - n)$ components picked out from $\vec{x}$ we can find a solution satisfying the constraints at each node. Since the components and their values were picked arbitrarily, knowledge of A doesnt add any information to the likely values of $\vec{x}$. Therefore $Pr(x_i = v) = Pr(x_i = v|f(\vec{x}))$ which proves that our information slicing algorithm is *pi*-secure.