The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems

MATTHEW K. WRIGHT, MICAH ADLER, and BRIAN NEIL LEVINE
University of Massachusetts Amherst
and
CLAY SHIELDS
Georgetown University

There have been a number of protocols proposed for anonymous network communication. In this paper we investigate attacks by corrupt group members that degrade the anonymity of each protocol over time. We prove that when a particular initiator continues communication with a particular responder across path reformations, existing protocols are subject to the attack. We use this result to place an upper bound on how long existing protocols, including Crowds, Onion Routing, Hordes, Web Mixes, and DC-Net, can maintain anonymity in the face of the attacks described. This provides a basis for comparing these protocols against each other. Our results show that fully-connected DC-Net is the most resilient to these attacks, but it suffers from scalability issues that keep anonymity group sizes small. We also show through simulation that the underlying topography of the DC-Net has affects the resilience of the protocol: as the number of neighbors a node has increases both the communications overhead and the strength of the protocol increase.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*

General Terms: Security

Additional Key Words and Phrases: privacy, anonymity, anonymous communication, predecessor attack

This work is supported in part by grants from the National Science Foundation under awards EIA-0080119, ANI-0087482, and ANI-0296194 and in part by grant 2000-DT-CX-K001 from the U.S. Department of Justice, Office of Justice Programs. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of the National Science Foundation or the Department of Justice.
Authors' addresses: M. Wright, University of Massachusetts, Dept. of Computer Science, Amherst, MA 01003, USA; email: mwright@cs.umass.edu; M. Adler, University of Massachusetts, Dept. of Computer Science, Amherst, MA 01003, USA; email: micah@cs.umass.edu; B. N. Levine, University of Massachusetts, Dept. of Computer Science, Amherst, MA 01003, USA; email: brian@cs.umass.edu; C. Shields, Georgetown University, Dept. of Computer Science, Washington, DC 20057, USA; email: clay@cs.georgetown.edu;

Portions of this paper have appeared earlier as Wright, Adler, Levine, and Shields. [2002].

1. INTRODUCTION

A variety of methods have been proposed to provide anonymous communication over the Internet. Previous protocols include DC-Net [Chaum 1988], Crowds [Reiter and Rubin 1998], Hordes [Shields and Levine 2000], APFS [Scarlatta et al. 2001], Onion Routing [Reed et al. 1998; Syverson et al. 2000], and Web Mixes [Berthold et al. 2000]. Each of these works include insightful analysis of possible attacks against their protocol. In their paper on Crowds, Reiter and Rubin describe an attack that allows an attacker to guess the initiator of an anonymous connection. The guess can be made based on information about the *predecessor* on the path of proxies. The presence of the attack caused the designers of Crowds to modify their protocol, which helped to ward off, but failed to eliminate, the threat. Syverson, et al. [2000] later examined a related attack that worked successfully against some configurations of Onion Routing and required timing analysis.

In this paper, we define the *predecessor attack*, a generalized version of these attacks. In this attack, the attacker tracks an identifiable stream of communications over a number of *rounds* (e.g., path reformations in Crowds). In each round, the attacker simply logs any node that sends a message that is part of the tracked stream. The attack does not always require analysis of the timing or size of packets (although that can speed up the attack), but instead exploits the process of path initialization. We look more carefully at the implications of the attack and its variations on protocols for anonymity.

In constructing the attack and analysis, we make several simplifying assumptions about protocol operation. Specifically, our major assumptions are:

(1) The subset of nodes that forward an *initiator's* messages are chosen uniformly at random;
(2) Initiators make repeated connections to specific *responders*, which are nodes outside the group performing the protocol.

In order to determine the length of time required for the attack, we also assume that initiators do not leave the group until the attackers are successful.

These assumptions are necessary for the proof we provide that shows that the attack works in all cases, and they are also critical to our analysis of the bounds on the time required for a successful attack. After describing the attack, we examine our assumptions and the effect that relaxing those assumptions has on the effectiveness of the attack. This includes examining defenses protocol designers might use to improve resistance to the predecessor attack.

Specifically, we make the following contributions:

—We define a class of anonymous protocols in which all currently proposed protocols may be placed and prove that the entire class is subject to the predecessor attack.
—We derive upper bounds on the amount of resources an attacker requires to significantly degrade the anonymity of users of the DC-Net, Onion Routing, and Mix-Net (e.g., Webmixes) protocols, extending Reiter and Rubin's analysis of Crowds.
—We show for the first time an attack on DC-Net based on a ring-topology that proves it is weaker than all other protocols. We use simulation to show how

adding neighbors in the topology greatly improves the robustness of DC-Net. This robustness can be achieved without resorted to the fully-interconnected topology that is easily subject to denial-of-service attacks if only one node chooses to not participate.

—We show that modifying path lengths in Mix-Net systems has limited value and exposes users to attacks that work faster than would be possible with fixed path lengths.

This work has implications not only for anonymous users with recurring Internet connections, but also for protocols that provide responder anonymity [Goldberg and Wagner 1998; Scarlatta et al. 2001] that want to support servers that remain available for long periods of time.

The next two sections overview related work. We prove a theorem in Section 3 stating a generalized attack undermines anonymous protocols. We analyze attacks against specific protocols in Section 4, and we focus more specifically on DC-Nets in Section 6. Section 5 discusses the merits of these protocols in light of these attacks, and Section 7 concludes.

2. BACKGROUND

Previous work on anonymous communication over the Internet has been extensive (e.g., [Chaum 1988; Reed et al. 1998; Reiter and Rubin 1998; Shields and Levine 2000]). A good survey of previous work is presented by Martin [1999]. There have been efforts to directly compare or analyze those techniques, or analyze the variety of attacks that may reduce the anonymity of a protocol's user over time [Shields and Levine 2000; Berthold et al. 2000; Syverson et al. 2000]. Reiter and Rubin [1998] have described an attack in the context of Crowds by which sufficiently powerful attackers can *degrade* the anonymity of a user. A related attack has been described by Syverson, et al. [Syverson et al. 2000] for Onion Routing. These attacks form the basis of our analysis in this paper. We distinguish our work from those analyses at the end of this section.

Anonymous protocols route messages from the *initiator* of a connection that wishes to remain anonymous to an overt *responder*, which is a node that does not participate in the protocol. A common feature of all proposed protocols for anonymity, some of which we review here, is the selection of a *path* through which messages are routed or the selection of a group of *nodes* that work together to send messages.

2.1 Crowds

Reiter and Rubin developed Crowds [1998], which uses a group of nodes that serve as proxies for a given initiator from the group. An initialization message is routed from the initiator to a series of proxies, forming a path for all future messages from the initiator. Upon receiving this message, each proxy decides, based on a *probability of forwarding* (p_f), whether to extend the path through another proxy chosen at random with uniform probability or to become the last node on the path and communicate with the responder directly. This path is maintained for a limited period of time, after which all paths must be reformed. The time limit allows nodes that join the protocol to add their paths at the same time as all other nodes; otherwise new paths may be easily attributed to recently joined nodes.

2.2 Onion Routing

Onion Routing, by Reed, et al. [1998], is similar to Crowds in that an initial message forms a path of proxies through which the initiator sends its future messages. The protocol gets its name from its method of encrypting the initial packet and the address of the proxies at each hop on the path with the public key of the previous step. This scheme results in layers of encryption that are peeled off at each step in order to determine the next address to send to on the path. This requires the initiator to predetermine the entire path. We will explore the benefits of initiator-determined paths and layered encryption in Section 4.

2.3 Mix-Net

A number of protocols for anonymity, Webmixes [Berthold et al. 2000], ISDN-Mixes [Pfitzmann et al. 1991], Stop-and-Go-Mixes [Kesdogan et al. 1998], Onion Routing, and others, have been based on David Chaum's anonymous email solution: a network of *mixes* [Chaum 1981]. We refer to a Mix-Net as protocol that uses Onion Routing's layered encryption and also employs *mixing* techniques to thwart timing analysis. Such mixing techniques include sending messages in reordered batches, sending dummy messages, and introducing random delays. It is beyond the scope of this paper to study which protocols effectively stop timing attacks, or, more generally, the effectiveness of the various mixing techniques. Instead, we consider an idealized Mix-Net protocol that guarantees that timing analysis will be effectively stopped. Onion Routing provides no defenses against timing analysis, and we show the difference that this makes against the predecessor attack in Section 4.

2.4 DC-Net

Another solution for anonymous communication, called DC-Net [Chaum 1988], has each participant share secret coin flips with other participants in pairs. The parity of the flips a participant has seen is then announced to all other participants. Since each flip is announced twice, the total parity should be even. To send a message, a participant incorrectly states the parity seen. This causes the total parity to be odd, which indicates transmission of a bit. No one except the initiator knows who sent the message, unless all of the nodes who flipped coins with the sender reveal their coin flips among themselves. Various techniques are available to handle collisions similar to media access control protocols for link layer networking [Bertsekas and Gallager 1987].

Any node may launch a denial-of-service attack by choosing to send a message every round of coin flips. Such a node is as anonymous as any initiator, and therefore cannot be simply detected and denied access. Strategies have been developed by Waidner and Pfitzmann [1989a] to detect such an attacker, but at the cost of a constant multiple of an already high message overhead.

Our work shows that attacks against DC-Net are extremely low-cost when participants are arranged in a logical ring (see [Schneier 1996]). Attacking a DC-Net with fully connected participants is much harder, requiring unreasonable resources on the part of the attacker. However, we argue that this arrangement creates substantial overhead that scales poorly with the number of participants. Moreover, DC-Net suffers from easily performed denial-of-service attacks that require even more overhead to avoid effectively. Our approach is probabilistic, and contrasts with epistemic approaches to understanding anonymity taken in the past [Syverson and Stubblebine 1999].

2.5 Comparison with Related Work

Reiter and Rubin were the first to identify the predecessor attack [1998]. In their initial analysis, they provide analysis that could be used to derive a bound on the number of rounds (i.e., path reformations) required for the attack to work with high likelihood for crowds.

Syverson, et al. identified a related attack for Onion Routing [2000]. Their analysis determined that the attack, complemented with timing analysis, succeeds with probability $(c/n)^2$, where c is the number of attackers, and n is the total number of nodes. They also recognized that the attack fails when the first onion router on the path is a trusted node.

Berthold, et al. discuss an intersection attack against the anonymity groups that arise when multiple mix routes are chosen [2000]. In this attack, the different anonymity groups are intersected with each other to shrink the number of possible initiators.

Raymond also discusses an intersection attack based on observations of user activity [2001]. Only the active users can be the initiator, and the set of active users changes over time. Intersecting the sets of active users reduces the set of possible initiators. We explore this idea further in Section 4.

More recently, Shmatikov used formal analysis and model checking to verify the efficacy of the predecessor attack against Crowds [2002]. Due to the high processing requirements of model checking, he was only able to demonstrate probabilities of attacker success for small numbers of nodes, i.e., twenty or fewer. In this paper, we use simulation to extend these results to thousands of nodes with an acceptable loss of precision.

This work differs from previous work in several ways. In this paper, we formally prove the attack Reiter and Rubin identified is successful against all existing anonymous protocols. Furthermore, we extend their analysis to calculate resources required to attack other protocols, which allows a quantitative comparison of the robustness of the protocols.

We extend the analysis of Onion Routing to show how long the attack will take. Additionally, we show how Mix-Nets hold a substantial advantage over Onion Routing in defending against this attack. We show the same for Onion Routing over Crowds.

Additionally, we analyze Mix-Nets in several different scenarios, including variable and fixed path lengths. Varying path lengths, which may seem like a good method of confusing attackers, fails to significantly increase the security against the predecessor attack. In fact, we show that using a Crowds-like approach of varying path lengths, as proposed by Syverson, et al. [2000], exposes users to much greater security risk.

We also show that the ring-based version of DC-Net described by Chaum (and later by Schneier [1996]) is easily attacked. Only one known variation on DC-Net is safe from the attack, though it is the most expensive protocol and subject to simple and anonymous denial-of-service attacks.

Since the preliminary version of this work appeared [Wright et al. 2002], we have added to our study of the predecessor attack.

In this article, we include a deeper examination of the topology of peers using DC-Net than just rings and fully-interconnected meshes. (See Section 6.)

Our APFS protocol presents a solution to the predecessor attack for responder-anonymous an mutually-anonymous applications. In that work, peers cooperate to provide a service (such as a file-sharing index or web server); the exact peer that provides the service changes before the predecessor attack is likely to be successful. The service remains in place, but the exact peer providing it does not.

We examined the assumptions of this work in a followup paper [Wright et al. 2003]. In that work we described a possible defense that comes from breaking the assumption of uniformly random path selection. Our analysis shows that the defense improves anonymity in the static model, where nodes stay in the system, but fails in a dynamic model, in which nodes leave and join. Additionally, we use the dynamic model to show that the intersection attack creates a vulnerability in certain anonymous communciations. We also presented simulation results that show that attack times are significantly lower in practice than the upper bounds given by this work. To determine whether users' web traffic has communication patterns required by the attacks, we collected and analyzed the web requests of users. We found that, for our study, frequent and repeated communication to the same web site is common.

Finally, in later work [Levine et al. 2004] we further examined the robustness of mix systems again timing attacks to clarify the threat they pose. We proposed defensive dropping as a technique to thwart timing attacks. Through simulations and analysis, we show that defensive dropping can be effective against attackers who employ timing analysis.

3. A GENERAL ANALYSIS

In this section, we define a model of anonymous protocols and an attack on such protocols. We then prove a theorem stating that the generic attack works on all protocols in the model when specific conditions are met.

There are two major assumptions that the attack requires to succeed.

—First, that there is a recurring connection between some party that initiates the sending of a message and the receiver of that message.

—Second, that there is information available to the attacker in the transported packets that uniquely identifies this recurring connection.

To justify the assumption that the connection recurs frequently, note that in the case of web browsing (which was the main intended use of Crowds), users often return to the same site [Harmon 1998]. We further support this claim in our related work [Wright et al. 2003]. Onion Routing was designed to support a wider variety of Internet connections (including HTTP, FTP, NNTP, and raw sockets) [Syverson et al. 2000], a number of which encourage types of recurring activity from users other than web browsing: USENET news-reading, ssh or telnet connections to remote accounts, on-going email correspondence, and IRC or other chat programs.

In many of these applications, the user may typically provide some identifying information with each transaction or session. For example, login names, user IDs, web cookies, and email addresses all provide unique identifiers at the application layer. While a user that is attempting to be anonymous would probably be careful to avoid these in many circumstances, she may allow such identifiers for some applications where it is needed or where there is trust. A user of an anonymous protocol may hope that the protocol itself will provide privacy while taking advantage of such identifiers (e.g., a Hotmail email account). We show that such a user is vulnerable to tracking.

Even without these identifiers, a user might still be tracked if the tracked connections are with unique responders. Attackers might even find a distinct pattern in the user's communications that could make the session trackable.

3.1 Model

A *protocol* is a series of instructions that a set of nodes (i.e., hosts) on a network can follow to hide the origin of their users' communications. A *participant* is a node that follows the protocol to send messages anonymously and to assist others in sending their messages anonymously. An *attacker* is a participant that collects data from its interactions with other participants in the protocol and may share its data with other attackers. We only consider peer-to-peer systems for simplicity, but attackers need not act as full participants to be effective. The attack works as long as attackers can send and receive messages in the protocol.

A participant that initiates the sending of a message is known as an *initiator*. The intended receiver is known as a *responder* and is not a participant. We refer to a *session* as continuing communications between an initiator and a responder. We use the term *sender* to refer strictly to a node that sends a packet directly to another node or to the responder; the term *receiver* strictly refers to a node that accepts packets from a sender as part of the protocol. The receiver of any packet can determine the identity of the sender and the sender of a packet knows the identity of the receiver; we equate IP addresses with identity.

We will show that sufficient attackers can collect enough information over time to compromise the anonymity of an initiator. Specifically, we show that, with time, attackers can indefinitely increase the probability that they can identify the initiator for a given session.

When a single message is transmitted from the initiator, participants send packets to each other. We refer to the *active set* for a given message as the set of all participants that send or receive any of these packets. Note that this means that the initiator is always in the active set. We denote the active set by A. In addition, there is some total order, Π, on the packets used in sending a message from the initiator to the responder; Π represents the global order of peers based on when the packets are received. This total order may be influenced by both the protocol, as well as the behavior of the network, since the network may deliver some packets faster than others.

Let Π_i be the ith position within the ordering of packets received. There is always some position Π_I where the initiator first sends a message. In our analysis, we assume that the protocol and the network combine to give A and Π the following property: given that the initiator sends a packet in position Π_I, the participants

Table I. Table of variables.

I	Initiator of a connection.
R	Responder to a connection.
A	Set of participants used by I to forward I's messages.
A_{min}	Minimal number of participants in A needed to determine I and R.
n	Total number of participants.
c	Number of attackers.
T	Number of rounds.
Π	Total ordering of the packets used in sending a message.
l	Mix-Net or Onion Routing fixed path length.
p_f	Crowds probability of forwarding.

seen sending in the remainder of the positions are chosen uniformly at random, either with or without replacement. For simplicity, we only consider one packet at a time, but multiple packets can be sent with the same or similar orderings Π before a new random ordering is used.

Let A_{min} be the minimum of the number of attackers, over all active sets A and total orders Π, that occur with non-zero probability required to determine the initiator and the responder. For example, in Onion Routing, $A_{min} = 2$ when a timing attack can be mounted (see Section 4.2), since it is sufficient for one attacker to be the first participant on the path and another attacker to be the last participant on the path. Note that the attackers might not know that they have correctly identified the initiator in this case, as a peer cannot determine if it is the first node on the path. This is not a problem for the attack, as we will show. For the case of the ring-based implementation of a DC-Net, $A_{min} = 2$, since it is sufficient for the attackers to occupy the two positions on either side of the attacker.

Nodes do not have indefinitely stable connections to the network. When a node disconnects, any active sets that it was a part of become disconnected. We call this event a *reset* and assume that resets occurs repeatedly without end in the operation of any protocol. We call the period between resets a *round*. Partial resets are possible in some cases, but nodes joining the network cause full resets due to the creation of an entirely new path. Note that all active sets must be reset at the same time if any are reset, as it will otherwise be obvious who the initiator of the reset stream must be. If the nodes reset their active sets immediately when a new node joins or an existing node leaves, an attacker could then hasten the predecessor attack by increasing the rate of resets [Reiter and Rubin 1998]. The protocol cannot, however, make the rounds indefinitely long, as users would be without the service until the next reset. Since attackers can leave and join to force resets, and protocols are constrained from long delays between resets, we can expect that rounds occur with short, regular intervals. For our analysis, we only require that resets occur repeatedly.

We say that a protocol that satisfies all these constraints is a *Uniform Active Set Protocol* for anonymous communications. In the rest of this section, we only concern ourselves with such protocols.

The attack can identify *all* initiators that keep a session active with the responder R. In the case that multiple initiators contact a single responder, attackers will not be able to link specific data streams to each initiator unless there is information in at least one packet per round that distinguishes the sessions from each other. Even if such information is not available, the attack can, in some cases, be considered successful if an initiator is linked to a particular responder. This case, however, is more difficult to analyze, so we will assume that only one initiator maintains a session with a given responder. We shall refer to a single initiator of interest to the attackers, node I, who is communicating with responder R. Note that if information is available that distinguishes sessions with the same responder from each other, that is an equivalent case.

We assume that I contacts R in every round. If I does not contact R in every round, we only use the rounds where R is contacted. The duration of the attack will be increased by a factor equal to the ratio of total rounds to rounds in which I contacts R.

In summary, we consider three assumptions to be key to our later discussion:

—I maintains the session, using the protocol, without end. It may leave the protocol or temporarily halt communications with R, but it must always resume the session using the protocol.

—An attacker must be able to distinguish the messages corresponding to a given session.

—For each Π_i within a total order Π, a node must be selected uniformly at random. (We know of only one protocol that does not operate this way: the local configuration of Onion Routing, in which the first node is trusted; For a discussion of non-uniform path selection, see our related work [Wright et al. 2003].)

3.2 The Attack

The attack depends on the assumption that an initiator might choose to remain in contact with a responder for an extended period of time. In that case, the session between the initiator and responder is subject to a number of resets. With each reset, a new active set is constructed between the initiator and responder. For each active set, there must be some participant that forwards the message outside the anonymous group to the responder. When this happens, we assume that this participant is able to associate the message sent with a specific session. The basic idea is that whenever the attackers are able to determine the specific session, there is some first attacker that sees the message. Our attack rests on the fact that the initiator is more likely to send the message to that first attacker than any other participant.

We define $G(n, c, T)$ as the probability of correctly guessing the initiator after T rounds with n participants and c attackers working cooperatively.

THEOREM 3.2.1. *No Uniform Active Set Protocol for anonymous communcations can maintain $G(n, c, T) \leq \epsilon$ for any $\epsilon < 1$, for all $T \geq 0$ when $c \geq A_{min}$ and $c \geq 2$.*

PROOF . Consider a protocol P. The attackers attempt to determine the identity of initiator I, who is the only participant communicating with responder R.

We now show that the attackers will be able to increase $G(n, c, T)$ to be arbitrarily close to 1, and therefore larger than ϵ, given sufficient T. We consider two cases: nodes in the active set chosen with replacement and without replacement.
Case 1:

Given that nodes are choosen with replacement, we will see how the attackers log I more than any other node. In any round where the attackers can determine R (which occurs with positive probability) and where the configuration is such that they might correctly identify I, they log the participant who first sent a message to the attackers in that round. At any step, the attackers identify the participant that has been logged the largest number of times as the initiator.

In the case where an attacker receives the message Π_I, the attacker logs I as the predecessor. This occurs with positive probability. In the remaining cases, due to the uniformity assumption, all participants are logged with equal probability. Thus, the expected number of times that I is logged by the attackers is greater than the expected number of times that any other node is logged by the attackers. By the law of large numbers, as $T \rightarrow \infty$, I will appear more often than any other node. The probability that I is identified as the initiator will grow larger than any value of $\epsilon < 1$.

Case 2:

When nodes are chosen without replacement, there is a corresponding attack based on the fact that the initiator cannot appear in any other positions in its active set. There is a positive probability that the attackers will be in position to determine R and will send a message to another node in the active set. This occurs, for example, when an attacker receives Π_I from the initator and other attackers are in position to determine R (When $|A| = 2$, this will not occur, but there is only the initiator and one other node in the active set, and the attackers easily identify the initiator). When the attackers are in such a position, they can log the receiver of any message from an attacker. This node is guaranteed to not be the initiator. Every node, except the initiator, is selected to receive such a message with equal probability. Thus, as $T \rightarrow \infty$, all nodes except the initiator will be logged with probability $p \rightarrow 1$. Thus, the attackers can determine I by waiting until all other nodes are logged as not being the initiator.

Note that the attackers need not determine apriori whether they must apply the attack from Case 1 or from Case 2. The attackers may run both attacks simultaneously, logging both their predecessors and receivers. Either the predecessor log will identify the initiator, or the receiver log will identify every node that is not the initiator, given enough rounds, with high probability. □

3.2.1 *Degrading Unlinkability.* Instead of determining both the initiator and the responder in the communication, the attackers may only attempt to determine the initiator without linking her to any communication to a responder. The success of this attack, which is easier in some cases for the attackers, means that the initiators will no longer be anonymous and only the *unlinkability* of the initiators is maintained. That is, it is known to the attackers that the initiators are communicating, but the identity of the responders are not known. This information can be particularly useful when few initiators are communicating. It is an advantage for anonymous protocols wishing to maintain more than unlinkability to have a requirement that all participants form active sets in all rounds (and possibly fill them with null traffic under threat of monitoring by attackers). Proofs of these statements exist that are similar in construction to the above proof. In general, any non-uniformity in anonymous protocols can be exploited for attack.

Protocol	**Rounds to attack, with high probability**	**Rounds to attack, expectation**	**Work required of participants**	**Latency from I to R**
Crowds	$O\left(\frac{n}{c}\log n\right)$	$O\left(\frac{n}{c}\right)$	$O\left(\frac{1+1/n}{(1-p)^2}\right)$	$\left(\frac{p}{1-p}+2\right)$
Onion Routing	$O\left(\left(\frac{n}{c}\right)^2\ln n\right)$	$O\left(\left(\frac{n}{c}\right)^2\right)$	$O(l)$	$O(l)$
Mix-Net				
•fixed path length l	$O\left(\frac{n^l}{c^l}\ln n\right)$	$O\left(\frac{n^l}{c^l}\right)$	$O(l)$	$O(l)$
•variable path length	$O\left(\frac{n^l min}{c^l min}\ln n\right)$	$O\left(\frac{n^l min}{c^l min}\right)$	$O(l_{ave})$	$O(l_{ave})$
DC-Net				
•fully connected, $c=(n-1)$	1	1	$O(n)$	$O(\lg n)$
•fully connected, $c<(n-1)$	(see § 6)	(see § 6)	$O(n)$	$O(\lg n)$
•ring connection	$\Theta(n)$	$\Theta(n)$	$O(n)$	$O(\lg n)$

Table II. A summary of the analysis for variations on each of four protocols.

4. SPECIFIC ATTACKS

In this section, we detail specific versions of the generic attack given in the previous section. We provide upper bounds on the time required for the degradation of an initiator's anonymity when faced with such an attack. We bound the time in terms of rounds, as defined above. Table II summarizes our results.

The two resources spent by the attackers are the number of nodes working cooperatively on the attack and the amount of time available to attack; memory and processing resources are generally not significant. Attackers handle no more traffic than normal participants in the protocols. We will explore how these resources can be used to effectively attack and undermine anonymity in systems running these protocols.

Some of the results in this section suggest either very long attacking times or a high proportion of attackers. However, the predecessor attack is passive and would draw no attention to itself — this means that it could continue for long periods of time without interruption and that the proportion of attackers could be very high. We do not suggest that an unsophisticated attacker with very limited resources could learn very much with this attack. It is important, however, to understand the limitations of current protocols and to be aware of the applications for which a given protocol is appropriate.

4.1 Crowds

To attack Crowds, a number of attackers may simply join the crowd and wait for paths to be reformed — a periodic occurrence, usually hourly [Reiter and Rubin 1998]. Each attacker can log its predecessor after each path reformation. Since the initiator I is far more likely than any other node to appear on the path, the attackers will log I much more often than any other node. After a large number of path reformations, it will become clear that the initiator is I. This attack was described by Reiter and Rubin earlier [1998]. They all but stated the number of rounds required to break Crowds; we show the analysis that follows directly from their results.

In terms of our generic proof from Section 3, only one attacker is required, i.e., $A_{min} = 1$. The attacker can appear directly after I and may then easily recover the responder R's address, which is in plain view, and other session-identifying information. Multiple attackers can perform this attack in parallel. They must simply communicate their results between each other, combining them to get larger samples.

It is helpful for attackers in Crowds to determine whether they are the first attacker on the path. This allows an attacker that appears after another attacker in the path to disregard its predecessor, as that predecessor is no more likely to be the initiator as any other node. This may be coordinated by a master attacker that can collect predecessor information from all the other attackers. Another method is for attackers to always submit requests from the session directly to the responder, thereby ending the path. Or the attacker may covertly tag messages before forwarding along the route. In any case, we can assume that only the first attacker on any path will log its predecessor.

Our goal is to give the number of rounds such that the initiator is seen as a predecessor more often than any other node with high probability. To accomplish this, we first calculate a number of rounds T that is sufficient to say that the initiator is seen in at least $f \cdot T$ of the rounds, where f is some significant fraction. We then calculate how many rounds are required such that no other node is seen in as many as $f \cdot T$ of the rounds. If these two conditions are satisfied, the initiator is clearly seen more than any other node, with high probability.

First, we get the number of rounds needed to lower bound how often the initiator is seen. An attacker appears first in the path with probabilty $\frac{c}{n}$. Applying Chernoff bounds [Motawani and Raghavan 1995] to this probability and we see that as long as the number of rounds is at least $T = \frac{8n}{c} \ln n$, the initiator will appear to attackers at least $f \cdot T = \frac{1}{2}\frac{c}{n}T$ times with high probability.

To say that all other nodes are seen fewer times than $f \cdot T$, we need consider the probability of a particular node N being on the path just before the first attacker on the path, if there is one. Let us call this probability σ. We can write σ as the probability that the attacker is on the path not directly after the initiator, and that N appears just before the first such attacker. The latter is $\frac{1}{n-c}$.

For our purposes, we need only know that the former is no more than one, so that $\sigma \leq \frac{1}{n-c}$.

We can now use σ to bound the probability that any non-initiator appears more than $f \cdot T$ times. We choose δ such that $(1+\delta)T\sigma = \frac{1}{2}\frac{c}{n}T$. This yields $\delta = \frac{c}{2n\sigma} - 1$. The number of times that a particular non-initiator is seen, $B(T, \sigma)$, is a binomial random variable that depends on the number of rounds and the probability of seeing it, σ.

We will use a Chernoff bound which holds as long as $\delta > 2e - 1$. Given that $\sigma < \frac{1}{n-c}$, we know that $\delta > \frac{c(n-c)}{2n} - 1$. Thus, as long as $\frac{c(n-c)}{n} > 4e$ (which occurs, for example, when $c = n/8$ and $n > 37e$), then $\delta > 2e - 1$. Applying the Chernoff bound:

$$\begin{aligned} Pr\{B(T,\sigma) \geq (1+\delta)T\sigma\} &< 2^{-(1+\delta)T\sigma} \\ &< 2^{-T\frac{c}{2n}} \end{aligned} \quad (1)$$

we see that if $T \geq 2\frac{2n}{c}\log_2 n$, which we simplify to $T \geq 6\frac{n}{c}\ln n$, then with probability $1/n^2$, we know that a given non-initiator node shows up to the attacker less than $\frac{1}{2}\frac{c}{n}T$ times. And since we have n nodes, there is less than a $\frac{1}{n}$ chance that any node other than the initiator appears to the attackers more than $f \cdot T = \frac{1}{2}\frac{c}{n}T$ times.

Set the number of rounds at $\frac{8n}{c}\ln n$. Then the attackers can use the following algorithm. If exactly one node is seen more than $\frac{1}{2}\frac{c}{n}T$ times, then the attackers believe that node is the initiator. If more than one, or no nodes are over the threshold, the attackers cannot yet determine the initiator. For this algorithm to fail — either by not answering or by answering incorrectly — either the initiator did not appear sufficiently often, or some non-initiator appeared too frequently. Any given non-initiator fails with probability $1/n^2$, and the initiator fails with probability $1/n$, so the total probability of failure is at most $2/n$. The probability that the initiator appeared too few times and some non-initiator appeared too often, leading to an incorrect initiator identification, is at most $1/n^2$.

The results for Crowds also hold for the Hordes protocol [Shields and Levine 2000] — which uses multicast paths from the responder to the initiator — because the attack takes place on the forward path to the responder.

4.2 Onion Routing

The use of layered encryption in Onion Routing results in a substantial advantage: only the last node in the path can recognize a particular data stream. An attacker must compromise the first and last node on the path, and even then must use timing analysis to know that both compromised nodes are on the path. There are several possible scenarios, depending on the ability of the attackers to gain information from timing analysis.

In one scenario, the Onion Routers see very consistent latencies between nodes. This might be possible if packet decryption and encryption dominated the message latency, and if nodes were essentially homogeneous in computing power. Per-hop delay might also be very consistent in some LANs. Timing analysis in this scenario would reveal to the two attacker nodes that they were on the same path and reveal the number of hops between them. Given that the path length is known (see Section 4.3.1 for discussion of this assumption), the attacker will know if the first attacker node follows the initiator directly. In this way, if the attackers compromise both the first and last node on the path, they will immediately identify the initiator.

This attack, as reported by Syverson, et al. has a single-round probability of success of $\frac{c^2}{n^2}$ for path lengths greater than two, and $\frac{c(c-1)}{n^2}$ for path lengths of exactly two [Syverson et al. 2000]. By applying a Chernoff bound, we see that with probability $\frac{n-1}{n}$, the initiator of the communication will be discovered in $T = 2\left(\frac{n}{c}\right)^2 \ln n$ rounds for path lengths greater than two. With path length set to two, the attackers do not need a timing attack and will be sucessful with high probability in $T = 2\frac{n^2}{c(c-1)}\ln n$ rounds.

Another scenario involves networks with varying latencies between nodes, but no mixing. This may be the most likely scenario when nodes are connected by the Internet. In this case, two attackers are able to determine that they are on the same path by a simple timing analysis on the initiator's traffic. If the path length is set to three or less, the attackers will know whether they are in position to see the initiator, and the attack from the above scenario applies. Otherwise, if two attackers are on the same path, with the second attacker in the last position on the path, they log the predecessor to the first attacker. This leads to an attack similar to that used against Crowds. The probability of an attacker being in the last position and an attacker being directly after the initiator is $\frac{c^2}{n^2}$. Using a Chernoff bound, we see that the initiator is logged at least $\frac{1}{2}T\frac{c^2}{n^2}$ times in $T \geq 8\frac{n^2}{c^2}\ln n$ rounds.

As we did with Crowds, let us define σ as the probability that any other node, N, is logged by the attackers. We need not find an expression for σ, but we should know that it can be written as the product of three probabilities: that the first attacker appears after the first proxy but before the second to last one, that an attacker appears last on the path, and that N appears just before the first attacker. The first term requires some explanation, but the second term is just $\frac{c}{n}$, and the third term is $\frac{1}{n-c}$. Thus we know that $\sigma \leq \frac{c}{n(n-c)}$.

Choosing δ such that $(1+\delta)T\sigma = \frac{1}{2}\frac{c^2}{n^2}T$, we get $\delta = \frac{c^2}{2n^2}\frac{1}{\sigma} - 1$. As with Crowds, the number of times that node N is seen, $B(T, \sigma)$, is a binomial random variable that depends on the number of rounds and the probability of seeing it, σ.

The Chernoff bound we want to use requires that $\delta > 2e - 1$. Since we know $\sigma \leq \frac{c}{n(n-c)}$, it follows that $\delta \geq \frac{c}{2} - 1$. So if $c(n-c)/n > 4e$, $\delta > 2e-1$, and we can apply the Chernoff bound as follows:

$$\begin{aligned} Pr\{B(T,\sigma) \geq (1+\delta)T\sigma\} &< 2^{-(1+\delta)T\sigma} \\ &< 2^{-\left(1+(\frac{c^2}{2n^2}\frac{1}{\sigma}-1)\right)T\sigma} \qquad (2) \\ &< 2^{-\frac{c^2}{2n^2}T} \qquad (3) \end{aligned}$$

So by setting $T \geq 2\frac{2n^2}{c^2}\log_2 n$, which we simplify to $6\frac{n^2}{c^2}\ln n$, we get that N is seen $\frac{1}{2}T\frac{c^2}{n^2}$ times with low probability $\frac{1}{n^2}$. This means that no node except for the initiator will be seen $\frac{1}{2}T\frac{c^2}{n^2}$ or more times with high probability $\frac{n-1}{n}$. Given the above number of rounds for seeing the initiator enough times, then the initiator will be seen more than any other node with high probability $\frac{n-2}{n}$ as long as $T \geq 8\frac{n^2}{c^2}\ln n$ rounds.

The final scenario is when timing attacks are not possible against the Onion Routing system. In this case, Onion Routing becomes a Mix-Net and the analysis from the next section applies.

4.3 Mix-Net

When no timing attacks are possible, initiators have an even more substantial advantage against the attack. Specifically, an attacker must compromise every node in the path between the initiator and responder to identify the initiator. In terms of our proof from Section 3, the number of attackers must be equal to the size of the active set, which is the path length. If there is a fixed path length of l for the network, then the probability of the attacker determining the initiator of a particular message is $\frac{c(c-1)^{l-1}}{n^l}$. Again, we use a Chernoff bound and observe that this will happen at least once, with probability $\frac{n-1}{n}$, given $T = 2\frac{n^l}{c(c-1)^{l-1}} \ln n$ rounds.

4.3.1 *Variable Path Lengths.* Attackers know, with certainty, when they have found the initiator in a Mix-Net or Onion Routing system with fixed path lengths given that they own every node on the path. It would seem beneficial, therefore, to vary the path length. Unfortunately, the benefits of this approach are limited.

As we show below, the security of the system against the predecessor attack is only slightly better than the security of a system with the path length fixed to the shortest length value. This means that the proposal in [Syverson et al. 2000] to have Crowds-like path length selection for Onion-Routing, via a probability of forwarding, would only offer the security of a system with only a single possible path length (times a constant factor of slowdown in the attack).

At the same time, the cost of varying path lenghts, in terms of delays in the user experience, can be relatively high. One might think that the costs could be modeled as a weighted average over the costs for each possible path length. However, users may not notice reduced delays for shorter than average path lengths as a significant benefit, while finding longer than average path lengths to be unacceptable. A fixed path length at the average of the variable path length should always have an acceptable performance.

We now show how the security of a Mix-Net with variable path lengths is limited in light of the predecessor attack. Let us suppose that the path length is varied by the initiator, and that the path length is chosen randomly from a range. Let l_s be the shortest path length that is chosen with a probability of at least $p > \frac{1}{n}$. If no such path length exists, then the predecessor attack becomes much slower at the cost of most paths having length $\frac{n}{2}$ or greater. If such a path length does exist, the attackers will see the initiator in $\frac{1}{2}Tp\frac{c(c-1)^{l_s-1}}{n^{l_s}}$ rounds with high probability, as long as $T \geq \frac{16}{p}\frac{n^{l_s}}{c(c-1)^{l_s-1}} \ln n$.

Other nodes will be seen by attackers that get l_s nodes in a row when the actual path length is longer. The probability, P, of the attackers seeing a non-initiating node this way is at most $\frac{1-p}{n-c}\frac{c(c-1)^{l_s-1}}{n^{l_s}}$. Letting $\delta = \frac{1}{2}\frac{p}{1-p}(n-c) - 1$ and $B(T,P)$ be a binomial random variable representing be the number of times a node is seen, we apply the following Chernoff Bound:

$$\begin{aligned} Pr\left\{B(T,P) \geq \frac{1}{2}Tp\frac{c}{n}\right\} &\leq 2^{-(1+\delta)TP} \\ &\leq 2^{-\left(\frac{1}{2}\frac{p}{1-p}(n-c)\right)T\frac{1-p}{n-c}\frac{c(c-1)^{l_s-1}}{n^{l_s}}} \\ &\leq 2^{-\frac{1}{2}Tp\frac{c(c-1)^{l_s-1}}{n^{l_s}}} \end{aligned} \tag{4}$$

Thus, if $T \geq \frac{4}{p}\frac{n^{l_s}}{c(c-1)^{l_s-1}} \log_2 n$, this node will be seen $\frac{1}{2}Tp\left(\frac{c}{n}\right)$ or more times with probability of only $\frac{1}{n^2}$. The total probability, then, of any such node being seen $\frac{1}{2}Tp\left(\frac{c}{n}\right)$ times is less than $\frac{1}{n}$. So if T is larger than both $\frac{16}{1-p}\frac{n^{l_s}}{c(c-1)^{l_s-1}} \ln n$ and

$\frac{4}{p}\frac{n^{l_s}}{c(c-1)^{l_s-1}}\log_2 n$, the initiator will be seen at least $\frac{1}{2}Tp\left(\frac{c}{n}\right)$ times and will be the only node seen that many times, with probability greater than $\frac{n-2}{n}$.

Note that the number of rounds has the same order of complexity, in terms of n and c, as the attack against the fixed path length of $l = l_s$. Thus, the variable path lengths increase the average and maximum delay but provide approximately the strength of the smallest path length against attackers using this attack.

The primary advantage of variable path lengths is in reducing the certainty of attackers in the result. With a fixed path length, the attackers may determine the initiator's identity with certainty in any single round, including the first round of the attack. However, if there is a non-trivial probability that the path length will be $l + 1$, then a set of attackers that make up a path length of l cannot be certain that they have identified the initiator correctly.

In general, to prevent the predecessor attack, the greatest path length with acceptable performance characteristics should be used. It may, however, be reasonable to select a path length with a good balance of performance and security and then vary path lengths to higher values for greater security against attacker certainty. Of course, against Onion Routing, timing attacks may work independently of path length with the same result [Syverson et al. 2000]. The security of longer paths depends on good general security that leaves attackers without easier attacking options.

4.3.2 *Unknown Path Lengths.* Hiding a fixed path length in Mix-Nets also provides little additional protection. One reason is that for most interactive applications, the typical user can practically stand performance no worse than using 10 or 20 nodes in a path. Only an exceptionally protective user might have a path length outside this range. However, even if the range of possible path lengths is large, the path length can still be determined as quickly as the predecessor attack will work against that path length.

Suppose the initiator uses paths with hidden length l. Given that the path length is fixed, the attackers know that when they comprise the full path length, only the initiator will be seen. They will get paths of length $l - 1$ every $\frac{n^{l-1}}{c(c-1)^{l-2}}$ turns, on expectation. After only two such times, the attackers will have seen two different nodes at the beginning with high probability. With two different nodes, it is clear that the path length is greater than $l - 1$. Getting the same node multiple times at a given path length suggests that the node seen is the initiator.

However, if the attackers wanted strong proof that a node was indeed the initiator, they might wait until the number of turns is high enough to show that a longer path would have been found with high probability. This would require the amount of work necessary to attack a one-step longer path length. Clearly, it is desirable to hide the path length whenever possible, as more information can be useful to attackers. One should not, however, rely on an assumption that the attackers do not have path length information while using such a system.

4.4 DC-Net

In DC-Net, a graph can be constructed by viewing each shared secret as an edge between nodes. To defeat DC-Net and expose the messages of a node N, attackers can surround N by corrupting all nodes that share an edge with N and share their secret coin flips with each other. By doing this, they know all the coin flips that N shared and therefore know what N's bit parity should be and can detect any messages. To determine the initiator in a particular session, the attackers can surround each node in turn until the initiator is found.

Because data exchange with all participants can become prohibitive, DC-Net as a ring (i.e., *DC-Ring*) is described by Chaum [Chaum 1988] and others [Schneier 1996]. In his Ph.D. Thesis, David Martin implemented ring-based DC-Net within the context of a local network [Martin 1999]. In the ring version of DC-Net each participant shares two secret coin flips, one with each of her neighbors.

In this section, we show how the attack detailed in Section 3 can be applied to DC-net to find the initiator of communications that all nodes can see. Only a fully connected DC-Net (i.e., *DC-Clique*), is impervious to attackers because the active set is the entire group, and all nodes are successors to the initiator; in the terms of the proof, $A_{min} = n - 1$. For ring-based DC-net, where the topology can be partitioned with just two attackers, $A_{min} = 2$. In Section 6, we describe how more dense topologies can be used to prevent attackers from gaining much information, while not resorting to the prohibitive expense of DC-Clique.

4.4.1 *Ring-based DC-Net.* The anonymity of a ring-based DC-Net degrades to zero and the initiator's identity can be proven by only two attackers after an average-case of $\Theta(n)$ rounds. A round only requires each attacker to leave the Chaum ring and rejoin it — we assume that joining nodes are placed randomly in the ring. If nodes are placed deterministically based on a piece of information about the nodes, such as a node's IP address, an attacker can forge that information before joining or corrupt nodes in known, static positions. This allows the attacker to effectively choose the best positions in the ring to perform the attack, which then works much faster. We also assume that all nodes hear all outgoing messages. This is a requirement of DC-Net, because the sender must hear the message to know whether it was sent correctly or if a collision occurred. Even with a system to prevent collisions and denial-of-service attacks, such as found in [Waidner and Pfitzmann 1989a], the sender must be able to see its message to know whether a trap was set off.

During a round, two nonadjacent attackers A and B may share their coin flips with each other. This effectively creates a new edge in the DC-Net graph seen only by the attackers. This new edge creates two sub-rings: one new ring consists of the edges from A to B and the new edge; while the other ring consists of the edges from B to A and the new edge. As per Chaum's protocol, the announced parities in the sub-ring without the initiator will sum to zero, and the nodes in that ring may be eliminated as possible initiators. The attackers will be able to identify the initiator immediately if it is the only node present in one of the sub-rings.

Suppose that both attackers leave and join the ring each round. Each round, the two attackers can identify a subset of nodes that are possible initiators. After T rounds, the set of possible initiators is the intersection of the set of possibilities for each round. Thus, the initiator has been determined when this intersection is a set of size one.

The intersection has size one if the following two events have occurred:

1) Attacker 1 is directly to the right of the initiator, and attacker 2 is at most

$\lfloor n/2 \rfloor$ steps to the left of the initiator.

2) Attacker 1 is directly to the left of the initiator, and attacker 2 is at most $\lfloor n/2 \rfloor$ steps to the right of the initiator.

Each of these has a probability of $1/2n$ of occurring. Thus, the expected number of rounds until we find the initiator is $O(n)$.

The expected number of rounds is $\Omega(n)$ as well, since both the position to the left and the position to the right of the initiator must be chosen to isolate him. This gives us an expected $\Theta(n)$ rounds for the attack to reduce the initiator's anonymity to zero. Note that with just a few rounds, an initiator's degree of anonymity will often be substantially reduced. Also, it is possible that the initiator's anonymity may be reduced to zero in any single round.

Two attackers can also determine what communications a particular node N is sending by surrounding it in an average case $\Theta(n)$ rounds. As soon as there is a round where the attackers obtain the position directly to the right of N as well as a round where the attackers obtain the position directly to the left of N, N's communications are exposed. Note that these two events don't have to happen at the same time, and can happen in either order. The probability of either of these events is $\frac{2}{n}$, and so the expected time until the attackers expose N is $\Theta(n)$.

5. DISCUSSION

The amount of work required to establish and maintain anonymity can be very high, and can vary greatly with different protocols. In this section, we discuss picking the best protocol based on network performance requirements as well as security, including resistance to the predecessor attack.

Figure II summarizes the results of two performance metrics. The fourth column shows the upper bounds on the number of active sets in which each participant will appear, which we refer to simply as *work.* The fifth column shows the length of the active set between the initiator and the responder, which directly affects network *latency.* We discuss these results further in this section.

5.1 Crowds, Onion Routing, and Mix-Nets

The network performance of Crowds largely depends on the path length resulting from the chosen probability of forwarding. Larger path lengths lead to a linear increase in delay and result in greater work for participants in the Crowd on behalf of others, as calculated by Reiter and Rubin [Reiter and Rubin 1998]. Unfortunately, an increase in the probability of forwarding does not significantly increase the number of rounds required for a successful attack, as shown by examining the analysis of Section 4. The primary advantage of a longer path in Crowds is to thwart attack by traceback [Yoda and Etoh 2000; Staniford-Chen and Heberlein 1995; Zhang and Paxson 1999]. The intuition for this is that the initiator will be seen by a collaborating attacker in the first path position with probability $\frac{c}{n}$, regardless of the path length.

Crowds and Onion Routing have equivalent work and latency characteristics for equal path lengths (see Figure II). Onion Routing requires more work for encryption and decryption, but has a more consistent performance over time. Crowds, and Onion Routing with Crowds-based variable path lengths, have inconsistent performance which may be a significant problem. Suppose that l^* is the path length at which the latency becomes too high for interactive applications. The path will have length l^* or greater with probability $P^* = p^{l^*}$. Of course, if l^* is large, then P^* will be quite small. In any case, it will occur given enough users and enough rounds. This means that the user will be unable to use the system for the duration of the round. Even if average-case performance is good, this bad case may happen often enough to significantly degrade the user's overall experience.

Mix-Nets face increased work and latency costs due to the costs of effective mixing. Nodes may have to introduce dummy messages, queue received messages while waiting for additional traffic, randomly delay messages, or apply other techniques. All of these introduce a work cost, a latency increase, or both. We note that the work and latency costs should continue to be linear with respect to the path length.

5.2 DC-Net

DC-Net requires substantial work from all participants at all times. First, for every participant with which secret coin flips are shared, a substantial amount of data must be shared. Chaum suggests that large quantities of random data might be shipped on a CD [Chaum 1988]. A more efficient alternative would be to use identical random number generators and share a seed from which future numbers could be generated. However, even in this scenario, when coin flips need not be communicated between the nodes that share the coin, every message sent requires that every node send a message. When n is only moderately large, this linear communication overhead becomes a very high cost.

Latency in DC-Nets can also be high. Nodes must announce the parities of their coin flips. Log-reduction message collection methods could be used to collect parities, followed by a broadcast to let everyone see the results. This introduces a latency of $O(\lg n)$ for each set of bits that is sent. The latency cost is similar to that of Crowds or Onion Routing when $O(\lg n)$ is the same as the expected path length. As n grows, however, DC-Net latency could be much higher than most Crowds or Onion Routing systems. Synchronization also introduces delays in this process, similar to a Mix-Net system.

Another complication of DC-Net is collision resolution. Discussion of this issue and efficient solutions are available elsewhere [Waidner and Pfitzmann 1989b]. Denial of service attacks are the cause of further inefficiencies in DC-Net. Denial of service attacks on other users are simple to perform — attackers need only send a constant stream of bits — and they can be performed under the cloak of nearly unbreakable anonymity. A modification of DC-net exists that isolates nodes launching a denial of service attack [Waidner and Pfitzmann 1989a]; however, it cannot prevent the attack and requires several steps of message exchange prior to a message being sent. It also requires each node to send "traps," rather than messages, some constant fraction of the time.

Despite the resistance of fully-connected DC-net to the attack we described, with all of these difficulties, DC-Net may only be usable between small groups of trusted parties. David Martin has shown that the costs become manageable in a locally-run system [Martin 1999]. The cost involved with using fully-connected DC-Net should be more thoroughly examined, as the ring-based approach has the substantial pitfalls we describe in Section 4.4.1. For large or dynamic groups, the additional work required appears prohibitive for real-time and bandwidth-intensive applications.

As both Crowds and Onion routing do not use the entire set of participants to route messages, they are resilient against denial-of-service attacks, whereas DC-Net is not. Conversely, because fully-connected DC-Nets use all nodes in the active set, they are not subject to degradation of anonymity due to the predecessor attack. Crowds and Onion routing are not.

5.3 The Predecessor Attack on Recent Protocols

Recently, several significant protocols for anonymous communications have been published. In this section, we discuss some of these protocols and their resistance to the predecessor attack.

One of these is P5, by Sherwood, et al. [Sherwood et al. 2002]. This protocol is designed for anonymity between peers connecting to each other, rather than outside responders. It could, however, be adapted to outside communication by using destination peers as the final proxy to the rest of the Internet. P5 uses a tree-based broadcast protocol, where a user's anonymity is based on the sizes of the different broadcast groups in which she is in.

The authors assume that "users do not leave once they join" to prevent a decline in users' anonymity [Sherwood et al. 2002]. Without this assumption, anonymity groups would shrink, leading to degradation of anonymity within the groups. We expect that this assumption does not hold well in today's networks, in which nodes may frequently shut down. When anonymity groups become too small, users must recreate a new communication tree, including a new key. This expensive step keeps the protocol from being vulnerable to the predecessor attack when the number of attackers is less than the size of the user's anonymity group.

The second protocol, Tarzan, by Freedman, et al. [Freedman and Morris 2002], is a peer-to-peer system built at the network layer. From the perspective of our analyses, which assume a peer-to-peer setting, Tarzan may be considered a variant of Onion Routing, as it uses Onion Routing-style layered encryption, but likely remains vulnerable to timing analysis, despite cover traffic.

Tarzan achieves a higher level of practical security in some scenarios by having nodes select relays according to random domain selection. This means that attackers cannot overload the network with malicious nodes from within the same domain, as initiating nodes will not select proxies from that domain with any greater frequency. Attackers can gain an advantage, however, when the number of honest domains represented in the Tarzan group is small. An attacker may be able to operate nodes from domains with no honest Tarzan participants. With attackers in a few such domains, attackers could make it likely to appear on an initiator's path despite only operating a few corrupt nodes.

Another protocol, MorphMix, is also a peer-to-peer path-based protocol [Rennhard and Plattner 2002]. It allows honest participants to find attackers in the system over time. Unfortunately, this comes at a cost of allowing attackers to create paths with only other attackers. Unlike in Tarzan, the peers do not need to know all other peers in the network to operate correctly. This is beneficial for keeping the intersection attack from being as easy, since obtaining a list of all peers currently in the system becomes more difficult.

Additionally, two mutual anonymity protocols have been presented recently [Xiao et al. 2001; Kung et al. 2002]. These protocols, like P5, are designed to hide the identities of both communication parties from each other and from third parties. Both of these protocols are, like Tarzan, variants of Onion Routing, but with anonymity for both the client and server. Since they are designed for file sharing, paths are not maintained. This makes them highly vulnerable to the predecessor attack as new paths create more opportunities for attackers to be on the path. However, repeated connections to the same responder may not be as common for file-sharing as in other applications, and in fact easier to avoid.

6. DIFFERENT TOPOLOGIES IN DC-NET

DC-Net has often been described in the DC-Ring configuration, and it remains secure as long as members do not leave or join the group. In practice though, membership often changes due to computer reboots, network partitions, and the addition of new users into the group. Under this more realistic model, DC-Ring is no longer secure, as described in the analysis of Section 4. However, DC-Ring has low communications overhead compared to the well-known alternative, DC-Clique, which requires $O(n)$ message exchanges for each node. It appears that the topology used represents a trade-off between the resistance to the predecessor attack and the system cost to peers.

In this section, we describe several alternative configurations between these two extremes of a ring and clique. We show how alternative topologies can provide greater security than the DC-Ring configuration at much lower cost than DC-Clique. The basic premise of these configurations is to start with a DC-Ring and gradually increase the number of neighbors which each node communicates with. The amount of work each node must do is proportional to the number of neighbors it has, but the security grows much faster.

6.1 Model

We consider a DC-Net in which there are nodes that must sometimes leave or rejoin the protocol. When a node leaves the network, there is a question of what to do in the protocol – any node that had the absent node for a neighbor is now missing a neighbor. Similarly, nodes that join the protocol cause a problem.

A simple solution might be to add new edges for new nodes and have neighbors of missing nodes all connect to each other. The trouble with this approach is that the attacker can take advantage of this to position itself where it wants. This is seen in the attack on DC-Ring prtocol described in Section 4. When one attacker isn't in a good position to get information, it can leave and rejoin, trying to get a better position each time.

An alternative approach to dealing with network dynamics is to have each node get an identifier, perhaps based on IP address, that determines that node's position relative to the other nodes. In this scheme, however, an attacker that has access to nodes over a wide spectrum of the IP address space will be able to place attackers precisely where they are needed to gain the most information.

We argue that any such scheme that maintains position within the group across memebership changes might face a similar attack. Therefore, instead of incrementally adding and deleting nodes over time, we believe that it is best to occasionally reset the protocol so that each node gets a new random set of neighbors. This prevents attackers from incrementally improving their position to isolate an initiator. Additionally, this comes at little additional cost; each node already needs to share random bits with its neighbors on a regular basis. If the graph is reset at a bit-sharing interval, then the only added cost in forming the new topology.

In the remainder of this section, we assume that the DC-Net is completely reset periodically, with each position in the graph being assigned randomly to a member of the group.

6.2 Configurations

There are a large number of possible configurations for DC-Net. We examine three as representative of the space between **DC-Ring** and **DC-Clique** (illustrated in Figures 1 and 5, respectively):

—**DC-Torus:** each node shares coin flips with four neighbors in an x-by-y lattice. At the edges, the grid wraps around, forming a torus so that none of the nodes have fewer than four neighbors. This is illustrated in Figure 2.

—**DC-Cube:** this is an extension of the torus configuration into a third dimension, so that each node has six neighbors. In other words, we have a cube where all the edges wrap around. This is illustrated in Figure 3.

—**DC-Random:** each node selects, uniformly at random from the graph, a number of other nodes with which it will share coin flips. Each node selects at a minimum i other nodes. Any node may reject a neighbor request if it already has the maximum number of neighbors, j; the requester retries by selecting a new node. This topography is shown in Figure 4.

In the random topology, each node should, in practice, limit the number of requests it takes from other nodes by first favoring its own random choices. Additionally each node should be wary of too many requests from the same nodes over rounds. These precautions should help keep attackers from isolating an node by selecting it as its neighbor. Additionally, there may be ways of identifying attacker nodes if their selection patterns are distinguishable.

We consider a random topology primarily to account for the practical challenges of running a DC-Cube or even DC-Torus, in that the nodes must organize themselves and may not perfectly fit into a grid or cube shape. With DC-Random, each node operates locally, and no global coordination is needed — except to accumulate and sum coin flips, which can be an efficient tree structure. We expect this topography to be less resistant to attack because the lack of regular structure can aid attackers in partitioning or isolating an initiator.

6.3 Attacks

There are two types of attacks against initiators in each topology. The simplest is to attempt to *partition* the network into two groups, one that includes the initiator and one that does not, eliminating the nodes in the non-initiator group as candidates. This attacks occurs over multiple rounds until only one node remains a candidate. The second is to attempt to *isolate* the initiator by occupying all available positions as neighbors; this attacks operates over multiple rounds until the initiator is isolated. Attackers can attempt both attacks in parallel, but isolation of an initiator is the more likely case.

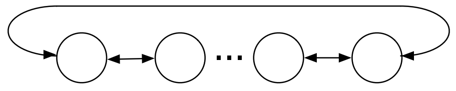


Fig. 1. Ring: 2 neighbors per node.

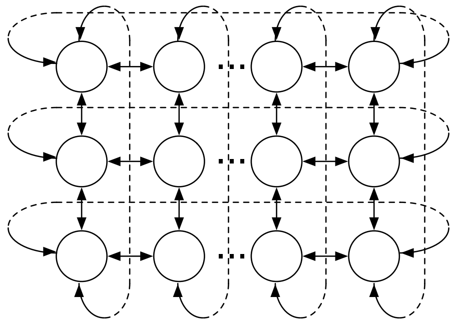


Fig. 2. Torus: 4 neighbors per node.

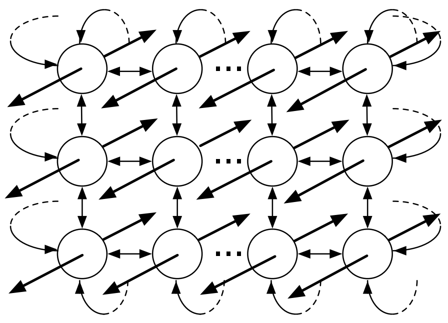


Fig. 3. Cube: 6 neighbors per node.

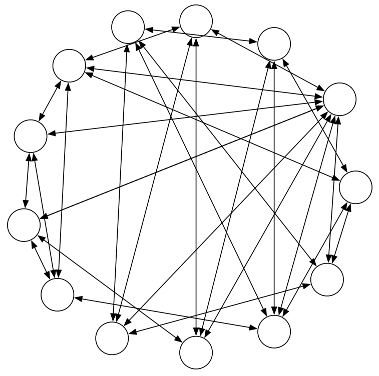


Fig. 4. Random: each node has between i and j neighbors.

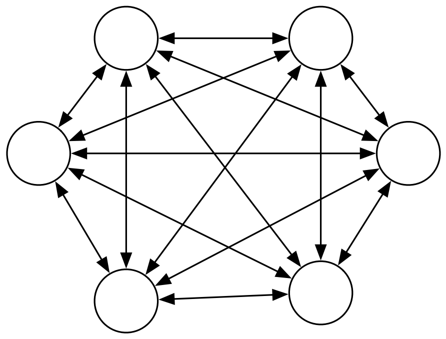


Fig. 5. Clique: a fully-interconnected graph.

Partitioning is easiest in the ring topology. However, the difficulty increases drastically with the number of edges each peer maintains. Only two attackers were needed to partition a DC-Ring, but partitioning the x-by-y torus into two large segments requires $2x$ attackers to fill two entire columns. Other partitions are possible, but they are also difficult to acheive. In DC-Cube, the attackers need fill two *planes* of the graph, which is highly improbable given random node placement.

In a torus, four attackers can isolate one node, or six attackers can isolate two nodes. If the isolated node is the initiator, then the attacker has successfully identified the initiator. If not, then the isolated nodes can be eliminated as possible initiators, thereby reducing the initiator's anonymity set. Given our model of random graph selection, isolation of the initiator occurs with probability $(c!(n-4)!/(c-4)!n!)$ and, using Chernoff bounds, we see that the attacker needs

$O\left((n!(c-4)!/c!(n-4)!)\lg(n)\right)$ rounds for this to occur with high probability. This is difficult to compare directly to the $O(n)$-round upper bound for the DC-Ring with only two attackers, but we will show how significant the difference is in simulation.

In a cube, attackers need six nodes to surround a single node. Isolation of the initiator occurs with probability $(c!(n-6)!/n!(c-6)!)$. Using Chernoff bounds, we see that the attacker needs $O\left((n!(c-6)!/c!(n-6)!)\lg(n)\right)$ rounds for this to occur with high probability.

In each topology, we expect isolation is the much more likely attack, and more easily provides a guide as to why some topologies are more robust. (An exact analysis of the number of rounds required to complete partitioning is beyond the scope of the paper, as the space of possibilities for partitioning is very large.) Our goal is to provide an empirical comparison of the performance of DC-Net over these topologies, each with increasing numbers of neighbors per node. In the simulations we present below, attackers attempt both attacks, as isolation is really just a special case of partitioning.

6.4 Simulations of the Predecessor Attack

To test our hypothesis that more neighbors per node yields greater security against the predecessor attack, we developed simulations of the attack on the DC-Net configurations described above. We compared topologies of 100 and 1000 nodes. We varied the attackers to consist of 10% to 50% of the nodes in the system.

The simulations are simple; we only need to simulate the creation of the DC-Net graph, with the placement of attackers, and determine what the attackers can learn.

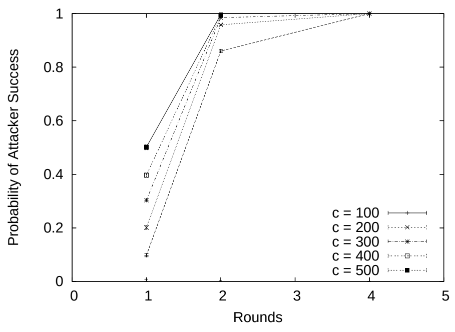
We do not attempt to link the attacks to wallclock (or calendar) time. We instead measure attack times in term of the *rounds*: the breakdown and establishment of the positions of nodes in the topology. For simplicity, we do not change the node membership in each round, only the positions in nodes. Specifically, we measure of the number of changes in the topology that must occur for the attackers to succeed in identifying the initiator.

The simulation uses the same node membership for each round. For that reason, the simulations tend to overestimate the length of the attack as nodes that leave the session could have been eliminated as the initiator. However, we evaluated all protocols with this assumption and we do not expect this significantly affects their performance relative to one another.

All topologies are selected at random in each round. To form the graph, each of the n nodes is given an identifier selected at random from 0 to $n-1$, without replacement. The identifier determines the node's placement in the graph.

For example, the ring is ordered linearly, such that the node that gets identifier i has neighbors $i-1$ and $i+1$, both modulo n. In the DC-Torus and DC-Cube scenarios, each position similarly is defined to have a specific neighbor set.

For the DC-Random setting, each node, by order of identifier, chooses the minimum number of neighbors, each at random, without replacement. If any node that is selected has the maximum number of neighbors, it rejects the connection and a new neighbor is selected. Thus, every node has at least the minimum and no more than the maximum neighbor set. For these simulations, we set the minimum to be five and the maximum to be 10. The average number of neighbors was approximately 6.4, and therefore would have been a bit more expensive to operate than



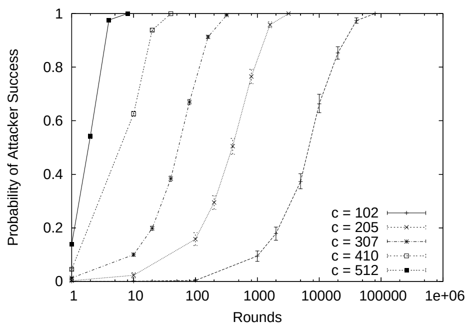


Fig. 6. DC-Ring: 2 edges per node. $N = 1000$. Fig. 7. DC-Torus: 4 edges per node. $N = 1024$.

DC-Cube.

The key to the attack is the random placement of attackers and the knowledge that they might gain. For each round, we can see what the attackers have learned by performing a breadth-first search of the graph, using the initiator as the starting point. The search stops at any attacker node. If less than the entire set of (non-attacker) node is reached at the completion of the BFS, then the attackers have partitioned the graph into two regions. The attackers also know which region contains the initiator and can eliminate nodes in the other partition. There will be no partitions of the graph in some rounds, in which case the attackers learn nothing.

We ran each simulation for a set number of R rounds. After R rounds have passed, the attacker select randomly from the remaining nodes as their best guess at the initiator's identity. If only the initiator remains, then it will be selected. Otherwise, the more nodes that are eliminated, the better the attackers will do, on average. For a given scenario, which is the protocol, the number n of total nodes, and the number c of attackers, and for a given value of R rounds, we run T trials. We then determine the success rate for the attacker by the number of successes for the T trials, with error determined by statistical bootstrapping (shown as error bars on the graphs). The number of trials for a given scenario with R rounds varied from 30 to 10000.

Although somewhat unfortunate for comparison purposes, the simulations for DC-Torus required using $n = k^2$, for integer k. Similarly, DC-Cube required $n = k^3$ for integer k. Although it was possible to do these simulations without exact squares and cubes, it made for the possibility of edge cases, in which the precise configuration makes some nodes easier to isolate. It would be best, perhaps, to consider a range of values for n, in order to cover such edge cases without letting them dominate a set of results. For simplicity, however, we chose to only consider values of n which led to full torus configurations or full three-dimensional toruses. In general, the results for each system are sufficiently different that the variation in n does not hinder comparison.

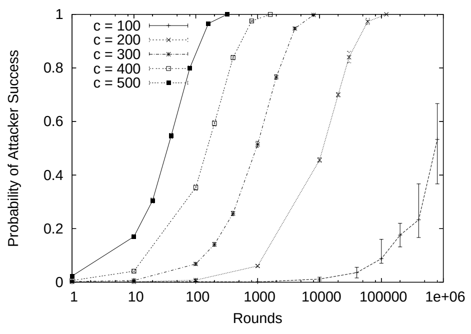


Fig. 8. DC-Cube: 6 edges per node. $N = 1000$.

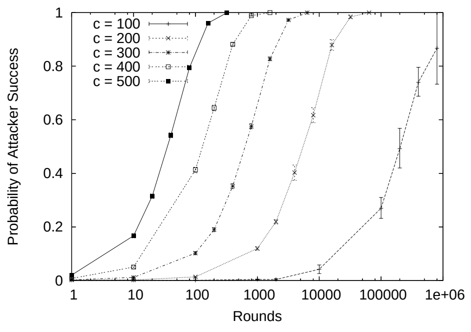


Fig. 9. DC-Random: 5–10 edges per node. $N = 1000$.

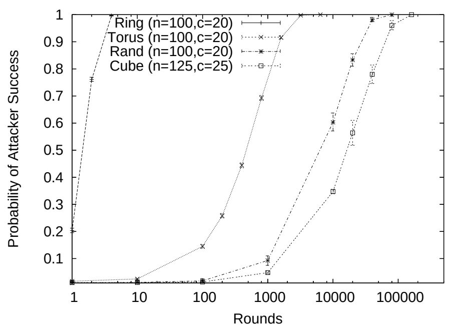


Fig. 10. Comparison of all protocols. $N = 100, c = 20\%$.

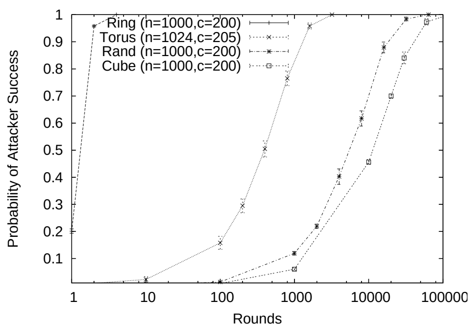


Fig. 11. Comparison of all protocols. $N = 1000, c = 20\%$.

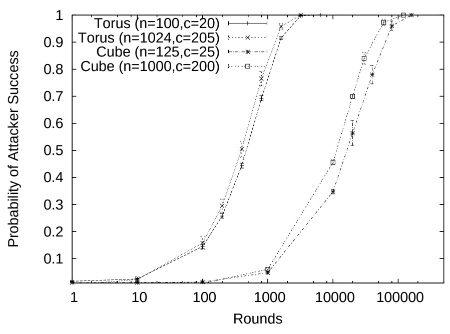


Fig. 12. DC-Torus and DC-Cube: $C \approx 0.2N$

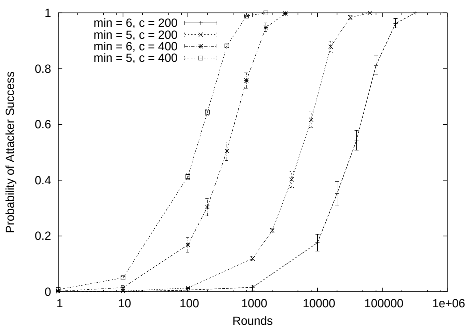


Fig. 13. DC-Random: Comparing with min edges per node. $N = 1000$.

6.5 Results

These results confirm our hypothesis that increasing the number of links does mean greater security against the predecessor attack, and that it is possible to gain greater security without the overhead of DC-Cliques. Note for all topologies, the amount of work required of each node is directly porportional to the number of neighbors (i.e., edges) it has in the graph.

We can see this by first looking at Figure 6, which shows that attackers against DC-Ring need a only four rounds to identify the initiator with a high probability of success, even in a very large network. Figure 7 shows that attackers against DC-Torus required hundreds of rounds or more when there are 30% or fewer attackers (note that the rightmost line represents the fewest attackers). The addition of two neighbors per node helps tremendously. Figure 8 shows that the DC-Cube does even better. With control over half of the nodes, the attackers still need around 100 rounds to be successful with high probability. When fewer attacker nodes are present, the time it takes for the attacker to succeed extends into the thousands of rounds and beyond. The two additional neighbors per node greatly hamper the attack.

We can see the effects more directly in Figures 11 and 10, where the protocols are compared, and which include the DC-Random topography. While a node in DC-Random has more links on average than one in DC-Cube, nodes have a lower minimum number of links and there is less structure to organize and maintain in the system. DC-Random therefore does not hold up quite as well as DC-Cube, though it still manages to defend against the attack for a long time. Note, however, that DC-Random is not limited to the number of edges we chose. As we see in Figure 13, increasing the minimum number of edges from five to six, with an increase in the maximum number of edges from 10 to 12, makes DC-Random last much longer against the attack. It also increases the work, however, with an average number of links of 7.6.

As we would expect, the more attackers there are the faster the attack works in all scenarios. The effect is very clear. Looking at Figure 8, we see that as the number of attackers goes down by 100, the length of the attack appears to increase exponentially. This suggests that finding ways to keep attackers from joining the system is at least as important as the number of edges per neighbor.

The effects that we see do not change much with the total number of nodes n. From Figure 12, we see that with fewer total nodes, the attacker's job is only slightly harder.

7. CONCLUSION

Anonymity continues to be an elusive and challenging problem. We have presented several results that show the inability of protocols to maintain high degrees of anonymity with low overhead in the face of persistent attackers.

We provided upper bounds for Onion Routing and Mix-Nets on the time required for attackers to degrade the anonymity of a particular initiator with high probability. Figure II summarizes our results. Crowds' degradation is bounded by $O\left(\frac{n}{c}\lg n\right)$ rounds [Reiter and Rubin 1998]. In Onion Routing, the number of rounds is bounded by $O\left(\left(\frac{n}{c}\right)^2\lg n\right)$. With Mix-Nets, the number of rounds required depends on the path length, l and is bounded by $O\left(\frac{n^l}{c^l}\lg n\right)$.

We proved that as long as attackers are selected uniformly at random to be a part of active set and sessions can be identified across path reformations, the degree of anonymity of any sender will degrade under attack. This allows us to understand why some proposed and exisiting protocols have a better defense against the predecessor attack than others. For example, because the data is encrypted into layers so that only the final node on the path can determine to which stream the packet belongs, Onion Routing holds its defense against attackers longer than Crowds. Since Mix-Nets thwart timing analysis, they further increase the defense. With DC-Net, only when all pairs of nodes shared coin-flips does the attacker require unreasonable resources to succeed; however, this result does not hold for other topologies. We also discussed, however, some weaknesses in the protocol that might prevent it from being usable in practice.

We have shown by simulation that the upper bounds obtained in our prior work were fairly loose, and that the attack can potentially succeed much more quickly than previously described. We have also shown, through study of real network traffic, that users may follow the necessary communication patterns for the attacks to be successful over time.

Defenses against these attacks are possible. We identified a set of defenses based on static path selection in Onion Routing and demonstrated that they are effective against the predecessor attack in a static setting. However, these defenses do not hold up in a peer-to-peer setting, when group membership is dynamic. Finding other defenses, especially in the peer-to-peer setting, is an important unresolved issue.

We also showed how there are many topologies that range between a ring and a clique for DC-Net. Adding a few edges to each node in the graph greatly increases the work required for a successful attack. We showed through simulation how even random topologies in DC-Net survive orders of magnitude longer than DC-Ring.

The churn in group membership of anonymous protocols also provides another means for attackers to degrade initiator anonymity. Specifically, we have shown that both Tarzan and Crowds are particularly vulnerable to the intersection attack, since lists of current users are readily available. We note that other peer-to-peer protocols, such as MorphMix, may not be as vulnerable to this attack, since awareness of all other peers is not needed for operation of the protocol. Since MorphMix has other vulnerabilities, developing strong peer-to-peer anonymity protocols that do not require knowledge of all peers remains an open problem.

These results are important for both the designers and users of anonymous protocols. It appears that designing a long-lived anonymous protocol is very difficult, and that users of current protocols need to be cautious in how often and how long they attempt to communicate anonymously.

REFERENCES

Berthold, O., Federrath, H., and Kohntopp, M. 2000 . Project anonymity and unobservability in the internet. In *Computers Freedom and Privacy Conference 2000 (CFP 2000) Workshop on Freedom and Privacy by Design* (April 2000).

Bertsekas, D. and Gallager, R. 1987 . *Data Networks.* Prentice-Hall.

CHAUM, D. 1981 . Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM 24*, 2 (February), 84–88.

CHAUM, D. 1988 . The Dining Cryptographers Problem: Unconditional Sender and Receipient Untraceability. *Journal of Cryptography 1*, 1, 65–75.

FREEDMAN, M. J. AND MORRIS, R. 2002 . Tarzan: A peer-to-peer anonymizing network layer. In *in Proc. ACM Conference on Computer and Communications Security (CCS 2002)* (November 2002).

GOLDBERG, I. AND WAGNER, D. 1998 . Taz servers and the rewebber network: Enabling anonymous publishing on the world wide web. *First Monday*.

HARMON, A. 1998 . Exploration of the world wide web tilts from eclectic to mundane. *New York Times*, (August 26). National Desk.

KESDOGAN, D., EGNER, J., AND BUSCHKES, R. 1998 . Stop-and-go-mixes providing probablilistic anonymity in an open system. In *Information Hiding* (April 1998).

KUNG, H. T., BRADNER, S., AND TAN, K.-S. 2002 . An ip-layer anonymizing infrastructure. In *Proc. MILCOM: Military Communications Conference* (October 2002).

LEVINE, B. N., REITER, M., WANG, C., AND WRIGHT, M. 2004 . Stopping timing attacks in low-latency mix-based systems. In *Proc. Financial Cryptography* (February 2004).

MARTIN, D. 1999 . *Local Anonymity in the Internet.* Boston, MA. Ph.D Thesis.

MOTAWANI, R. AND RAGHAVAN, P. 1995 . *Randomized Algorithms.* Chapter 4. Cambridge University Press.

PFITZMANN, A., PFITZMANN, B., AND WAIDNER, M. 1991 . Isdnmixes: Untraceable communication with very small bandwidth overhead. In *"GI/ITG Conference: Communication in Distributed Systems"* (February 1991).

RAYMOND, J.-F. 2001 . Traffic analysis: Protocols, attacks, design issues and open problems. In *Designing Privacy Enhancing Technologies: Proceedings of International Workshop on Design Issues in Anonymity and Unobservability*, H. Federrath, Ed., *LNCS* vol. 2009 (2001). Springer-Verlag, 10–29.

REED, M., SYVERSON, P., AND GOLDSCHLAG, D. 1998 . Anonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communication Special Issue on Copyright and Privacy Protection.*

REITER, M. K. AND RUBIN, A. D. 1998 . Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security 1*, 1 (November), 66–92.

RENNHARD, M. AND PLATTNER, B. 2002 . Introducing morphmix: Peer-to-peer based anonymous internet usage with collusion detection. In *Proc. 2002 ACM Workshop on Privacy in the Electronic Society (WPES)* (November 2002).

SCARLATTA, V., LEVINE, B., AND SHIELDS, C. 2001 . Responder anonymity and anonymous peer-to-peer file sharing. In *Proc. IEEE International Conference on Network Protocols (ICNP)* (November 2001).

SCHNEIER, B. 1996 . *Applied Cryptography.* J. Wiley and Sons.

SHERWOOD, R., BHATTACHARJEE, B., AND SRINIVASAN, A. 2002 . P5: A protocol for scalable anonymous communication. In *Proc. 2002 IEEE Symposium on Security and Privacy* (May 2002).

SHIELDS, C. AND LEVINE, B. 2000 . A Protocol for Anonymous Communication Over the Internet. In *Proc. 7th ACM Conference on Computer and Communication Security (ACM CCS 2000)* (November 2000).

SHMATIKOV, V. 2002 . Probabilistic analysis of anonymity. In *IEEE Computer Security Foundations Workshop (CSFW)* (2002). 119–128.

STANIFORD-CHEN, S. AND HEBERLEIN, L. 1995 . Holding Intruders Accountable on the Internet. In *Proc. of the 1995 IEEE Symposium on Security and Privacy* (Oakland, CA, May 1995). 39–49.

SYVERSON, P. AND STUBBLEBINE, S. 1999 . Group Principals and the Formalization of Anonymity. In *FM'99—Formal Methods, Volume I*, J. Wing, J. Woodcock, and J. Davies, Eds., *Lecture Notes in Computer Science* vol. 1708 (1999). Springer, 814–833.

Syverson, P., Tsudik, G., Reed, M., and Landwehr, C. 2000 . Towards an analysis of onion routing security. In *Workshop on Design Issues in Anonymity and Unobservability* (July 2000).

Waidner, M. and Pfitzmann, B. 1989a . The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability. In *Eurocrypt '89* (1989).

Waidner, M. and Pfitzmann, B. 1989b . Unconditional Sender and Recipient Untraceability in spite of Active Attacks – Some Remarks. Technical report, Fakultat fur Informatik, Universitat Karlsruhe.

Wright, M., Adler, M., Levine, B., and Shields, C. 2002 . An analysis of the degradation of anonymous protocols. In *ISOC Symposium on Network and Distributed System Security* (February 2002).

Wright, M., Adler, M., Levine, B., and Shields, C. 2003 . Defending anonymous communications against passive logging attacks. In *IEEE Symposium on Security and Privacy* (May 2003).

Xiao, L., Xu, Z., and Zhang, X. 2001 . Low-cost and reliable mutual anonymity protocols in peer-to-peer networks. Technical Report HPL-2001-204 (August), Hewlett Packard Laboratories.

Yoda, K. and Etoh, H. 2000 . Finding a Connection Chain for Tracing Intruders. In *Proceedings of the 6th European Symposium on Research in Computer Security* (2000). ESORICS.

Zhang, Y. and Paxson, V. 1999 . Stepping Stone Detection. Presentation at SIGCOMM'99, New Areas of Research.

Received April 2002; revised September 2002; accepted Later