An Analysis of the Degradation of Anonymous Protocols

MATTHEW K. WRIGHT, MICAH ADLER, and BRIAN N. LEVINE
University of Massachusetts
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. 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. Additionally, we show how violating an assumption of the attack allows malicious users to setup other participants to falsely appear to be the initiator of a connection.

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

General Terms: Security

Additional Key Words and Phrases: anonymity, anonymous communication, degradation, predecessor attack, onion routing, crowds, mix-nets, mixes, dc-net

1. INTRODUCTION

A variety of different 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 attacks and network performance. 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, any 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 require analysis of the timing or size of packets, 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. Specifically,

This work is supported in part by a grant from the National Science Foundation under award EIA-0080119 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;

This article is an extended version of Wright et al. [2002].

we make the following contributions:

—We define a class of anonymous protocols in which all currently proposed protocols may be placed and prove 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., Web Mixes) protocols, complementing Reiter and Rubin's analysis of Crowds.

—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.

—Finally, we identify *setup attacks* as a new threat that can be used to victimize Crowds, Onion Routing, and Mix-Net users so that they are falsely accused of being initiators of connections.

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 most easily falls subject to 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].

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 4 stating that a generalized attack undermines anonymous protocols. We analyze attacks against specific protocols in Section 5. We introduce a new attack that can falsely identify the initiator of a connection in Section 6. Section 7 discusses the merits of these protocols in light of the attack, and Section 8 concludes.

2. BACKGROUND

Previous work on anonymous communication over the Internet has been extensive [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 also 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 in the next section.

Anonymous protocols route messages from the *initiator* of a connection that wishes to remain anonymous to an overt *responder*, which 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 5.

Onion Routing has generally been implemented with the onion routers being placed in the network outside of the control of the individual users. While it can be argued that this reduces the possibility of corruption of any particular onion router, it requires that the users trust the operators of the onion router to maintain their anonymity. Accordingly, users may instead choose to run their own onion routers locally, and band together cooperatively to forward traffic for each other. This local configuration distributes the trust to many operators, but provides more opportunities for corruption of routing nodes. As described in more detail in Section 3, we primarily analyze the local configuration.

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 5.

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 a high cost in overhead.

3. 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 [Syverson et al. 2000]. They analyzed two configurations: *local-COR configurations* where individuals run their own onion router, and *remote-COR configurations* where individuals first connect to a remote untrusted COR. Their analysis concluded that the attack they described was not successful against the local-COR configuration unless all other routers were compromised. Their analysis also concluded that the attack, complemented with timing analysis, succeeds in the remote-COR configuration with probability $(c/n)^2$, where c is the number of attackers, and n is the total number of nodes.

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.

This paper proves and details how the attack succeeds against the local-COR configuration. 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.

Finally, we are the first to identify *setup attacks* as a threat to anonymous services. Setup attacks allow malicious initiators to make it falsely seem that a third party is the initiator of a connection.

4. 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.

As we discuss in this section, 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; and 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]. 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 ID's, 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. 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.

4.1 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 increase the probability that they can identify the initiator for a given session.

When a single message is transmitted from the initiator, there is some set of packets that are sent between the participants. 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, representing the global order in which 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 active set. Within the total order, there is always some position Π_I where the initiator first sends a message that can be identified as a transmission. In our analysis, we assume that the protocol and the network combine to give A and Π the following property: given that the initiator is in position Π_I the participants in the remainder of the positions are chosen uniformly at random, either with replacement or without replacement. When this is without replacement, the initiator appears only in position Π_I.

Let A_{min} be the minimum over all active sets A and total orders Π that occur with non-zero probability, of the number of attackers 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 5.2), since it is sufficient for an attacker to be the first participant on the path, as well as the last participant on the path. Note that the attackers might not necessarily

Table I. Table of variables.

I	Initiator of a connection.
R	Responder to a connection.
A	The set of of participants used by I to forward I's messages
n	The number of participants.
c	the number of attackers.
T	number of rounds.
Π	Total ordering of the nodes in A.
l	Mix-Net or Onion Routing fixed path length
p_f	Crowds probability of forwarding.

know that they have correctly identified the initiator in this case. For the case of the ring-based implementation of a DC-Net, $A_{min} = 2$, since it is sufficient for the attackers to occupy positions Π_{I-1}, Π_{I+1}.

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 it occurs repeatedly without end in the operation of any protocol. We call the period between resets a *round*. Note that all active sets must be reset at once 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]. Alternately, the protocol cannot make the rounds too long, as users could be without the service until the next reset. Since attackers can likely 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 no protocol that does not operate this way.

4.2 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 4.2.1. *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* $c \geq A_{min}$.

Protocol	Rounds to attack, with high probability	Rounds to attack, Expectation	Work required of participants	Latency from I and R
Crowds (from [Reiter and Rubin 1998])	$O\left(\frac{n}{c}\log n\right)$	$O\left(\frac{n}{c}\right)$	$O\left(\frac{1}{(1-p)^2}\left(1+\frac{1}{n}\right)\right)$	$\left(\frac{p}{1-p}+2\right)$
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, $c<(n-1)$	n/a	n/a	$O(n)$	$O(\lg n)$
fully connected, $c=(n-1)$	1	1	$O(n)$	$O(\lg n)$
ring connection	$O(n)$	$O(n)$	$O(n)$	$O(\lg n)$

Fig. 1. A summary of the analysis for variations on each of four protocols.

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. In any round where the attackers can determine R (which occurs with positive probability) and where the configuration is such that it could be the case that they can identify I, they log the participant who first sent a message that can be identified as a transmission 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.

The key to this attack is that in the case where the initiator can be correctly identified, it is the participant that is logged. This occurs with positive probability. On the other hand, 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 $T \to \infty$, 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 $\epsilon < 1$. □ □

Instead of attempting to completely make up the set A for a particular node, the attackers can attempt to position themselves as the direct receivers of that node's messages. For example, in Mix-Nets, attackers may attempt the easier task of being placed directly after the initiator, and not try also to fill all the other positions on the path. The success of this attack, which is easier in some cases for the attackers than making up all of A, 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. 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.

5. SPECIFIC ATTACKS

Here we describe 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. Figure 1 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.

Before we begin, it might be noted that some of the results in this section suggest either very long attacking times or a high proportion of attackers. While this is true, the fact that the predecessor attack is passive and would draw no attention to itself 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.

5.1 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 see 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 4, only one attacker is required. 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 [Motwani 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 latter 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 $n > 16e$ and $c = n/2$), $\delta > 2e - 1$. Applying the Chernoff bound:

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

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.

5.2 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 5.3.1 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:

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

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.

5.3 Mix-Net

When no timing attacks are possible, users 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 4, 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.

Note that, in Mix-Nets and Onion Routing, because the initiator can directly choose the active set, it may select only trusted nodes it expects to perform honestly. In Crowds, initiators may similarly choose trusted nodes, but are in control of only the first node on their path. This strategy will defeat the attacks we describe, but it has some drawbacks. Because the set is static, an attacker may be able compromise these systems over time, especially if their number is small. Additionally, new paths are easily attributable to nodes that join the protocol late when other nodes use established paths.

5.3.1 *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. It would seem beneficial, therefore, to vary the path length. Unfortunately, the benefits of this approach are limited. The cost, in latency and work, of varying path lengths could be considered as a weighted average over the cost for each possible path length. It is likely, however, that user's experience will be most effected by the highest latency costs, rather than average performance. Good average performance will be forgotten if the system sometimes performs so slowly that it becomes unusable.

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 occasionally provide the worst-case performance of a system with long path lengths and only offer the security of a system with only a single possible path length (times a constant factor of slowdown in the attack).

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.

5.3.2 *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.

5.4 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. A good instantiation of DC-Net would not allow less than all pairs of participants exchanging coin flips. Otherwise, topology reformations provide attackers with opportunities to gain information.

Because data exchange with all participants can become prohibitive, DC-Net as a 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 4 can be applied to DC-net to find the initiator of communications that all nodes can see. We discuss the attack for ring-based DC-Net, but analogous attacks exists for other topologies. Only a fully connected DC-Net 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, $|A_{min}| = 2$.

5.5 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 simply forge that information before joining. 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 $O(n)$.

You can show that it has to be $\Omega(n)$ 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 $\Theta(n)$.

6. SETTING UP THIRD PARTIES

One of the key assumptions that make the above attacks work is that the initiator behaves strictly according to protocol, making random selections uniformly from among all nodes. However, an initiator may select nodes purposefully, possibly with malicious intent. In this section we discuss how malicious selection of the active set by one or more collaborating nodes can be the basis for a new attack that makes it appear that traffic is originating from some particular victim. This attack, which we call the *set-up attack*, can be used to contrive evidence so that anyone using the degradation attack described above will not identify the correct initiator; instead, they will locate the victim of the set-up attack. This attack is present in any protocol where initiators are allowed to non-randomly select any part of the active set and includes Crowds, Onion Routing, and Mix-Nets.

6.1 Set-up Attacks

Suppose the initiator, I, always places a victim node, V, as its direct successor in its routing path. Then, if attackers attempt to determine the initiator's identity, all messages will appear to come from V. If I has malicious intent, it might, for example, then use the link to launch an attack against a responder. Once sufficient evidence existed to point to V, I could stop the stream to ensure that V could not demonstrate its innocence. Alternatively, I could use V to access a responder that carries illegal content, making it appear that V was the one receiving the data.

With multiple attackers all targeting the same node V, the set up attack is very effective. For example, suppose that several nodes were to purposefully route paths through V to a specific responder. Another set of nodes performing the degradation attack from Section 4 would see V as the most likely initiator.

Further optimizations are available through Mix-Nets, in which the whole path is chosen by the initiator. If the malicious nodes conducting the set-up attack were to know which attackers were attempting the degradation attack for particular sessions, they could set the entire path after V to be these attackers, thereby ensuring V gets discovered. Unless V joins the attackers, it will appear to be the initiator and will bear any consequences that arise from I's communications.

There is some risk to I in conducting this attack, as I might either accidently choose one of the degradation attack members as V, or the attackers might give V the chance to join them and show that I is indeed the initiator. V might also monitor its connections and decide not to carry any links sent from I once it appears that it is being setup.

6.2 Self-Protection

Ironically, the set-up attack can be seen as a method of self-defense against the degradation attack. Careful selection of the nodes that will forward messages to the responder also allows an initiator to protect itself from degradation of anonymity. In fact, placing even a single trusted node in Onion Routing on the path every time ensures that the path cannot consist entirely of attackers. By placing this trusted node at the end of the path, the responder's identity and other session-identifying information is hidden from the attackers in every round. Knowing who to trust in a random group is difficult if not impossible, but nodes could arrange to join a group with several other trusted nodes, and share paths among each other.

Protection against the set-up attack described in this section for both Crowds and Onion Routing is not a simple task. We consider it a fundamental problem with protocols that allow nodes to select the active set. Detecting set-up attacks and developing methods of countering them are among our areas of future examination.

7. 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 1 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.

7.1 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 5. 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 2000]. 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 1). 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.

7.2 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. Even in this scenario, however, a node that joins the DC-Net would need to exchange messages with every node with which it will share coin flips; the attack described in Section 4 is successful for any DC-Net in which fewer than all participants are neighbors to each other.

Once nodes have joined, they must make their announcements of the parity 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 will be much higher than most Crowds or Onion Routing systems.

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 which isolates nodes launching the attack exists [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 5.5. 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. Finally, because all variants of DC-Net do not allow nodes to pick neighbors, they are not subject to set-up attacks, as can occur in Crowds and Onion Routing.

8. 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 1 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 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.

Finally, we identified set-up attacks as a new threat to protocols that allow participants to choose part or all of the active set. Detecting when a node is the victim of a setup attack is difficult and is a fundamental problem for Crowds, Onion Routing and Mix-Nets.

The predecessor attack applies to existing responder anonymous protocols [Goldberg and Wagner 1998; Scarlatta et al. 2001], which are intended to provide anonymous web sites. Such responder anonymous protocols are meant to support servers that stay up indefinitely, though the predecessor attack insures only a finite time is possible. Our related work [Scarlatta et al. 2001] uses the special properties of peer-to-peer networks to thwart the predecessor attack we have described here.

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.*

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 (Feb.), 84–88.

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

Goldberg, I. and Wagner, D. 1998. TAZ Servers and the Rewebber Network: Enabling Anonymous Publishing on the World Wide Web. *First Monday 3,* 4.

Harmon, A. 1998. Exploration of the World Wide Web Tilts from Eclectic to Mundane. New York Times.

Kesdogan, D., Egner, J., and Buschkes, R. 1998. Stop-and-Go-MIXes Providing Probablilistic Anonymity in an Open System. In *Information Hiding.*

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

Motwani, R. and Raghavan, P. 1995. *Randomized Algorithms.* Cambridge University Press, Chapter 4.

Pfitzmann, A., Pfitzmann, B., and Waidner, M. 1991. ISDNMixes: Untraceable Communication with Very Small Bandwidth Overhead. In *"GI/ITG Conference: Communication in Distributed Systems".*

Reed, M. G., Syverson, P. F., and Goldschlag, D. M. 1998. Anonymous Connections and Onion Routing. *IEEE Journal on Special Areas in Communications 16,* 4, 482–494.

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

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).*

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

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).*

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, 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. 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.*

WAIDNER, M. AND PFITZMANN, B. 1989a. The Dining Cryptographers in the Disco: Unconditional Sender and Recipient Untraceability with Computationally Secure Serviceability. In *Eurocrypt '89.*

WAIDNER, M. AND PFITZMANN, B. 1989b. Unconditional Sender and Recipient Untraceability in spite of Active Attacks – Some Remarks. Tech. rep., Fakultat fur Informatik, Universitat Karlsruhe.

WRIGHT, M., ADLER, M., LEVINE, B., AND SHIELDS, C. 2002. An Analysis of the Degradation of Anonymous Protocols. In *Proc. ISOC Network and Distributed System Security Symposium (NDSS 2002).*

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.* ESORICS.

ZHANG, Y. AND PAXSON, V. 2000. Detecting Stepping Stones. In *Proceedings of the 9th USENIX Security Symposium.* Denver, CO.

Received April 2001; revised September 2001; accepted Later