Anonymous Communication on the Internet

Andy Jones andjones@cs.indiana.edu

September 17, 2004

Abstract

Encryption, or the art of concealing information, dates back thousands of years. Julius Caesar used his "Caesar Cipher" to encrypt messages in government and military communications. The Germans used their Enigma Cipher in World War II to encipher messages. Encryption today is used in bank transactions, online auctions, protecting account information, surfing the web, and many more applications.

However, encryption only hides what is being said. It does not hide who is talking to whom. This information by itself can be used to one's advantage or disadvantage. Consider the case of a company whistleblower who fears the backlash of her superiors or losing her livelihood. A citizen trying to report a crime would fear personal injury if their privacy was compromised to the perpetrators. Computer administrators often keep logs of user activity, either as a matter of security or as instructed by superiors. This allows tracking of online activity, as a result users may fear scrutiny, or betrayal of person information. Compromised computers on the internet are able track all communication passing through it. This again compromises personal information, which may cause personal harm, economic loss, unpopular sentiment, or a combination of these.

This paper is survey of popular anonymous protocols on the internet. What methods are available and do they work? Anonymous communication is discussed starting with the Mix based systems such as, Onion Routing, Crowds, Anonymizer, WebMIXes, and Hordes. Broadcast protocols and DC-Nets are discussed including Herbivore, and P^5. Performance of the protocols is given and classes of vulnerabilities and attacks are discussed.

1 Why Anonymity?

Anonymity is very useful when a person desires to protect their identity from discovery. Anonymity is desired by anyone who fears retaliation, discrediting, unpopular sentiment, or simply desires privacy.

Anonymity finds uses in corporations for whistle blower cases; in law enforcement for anonymous crime tip lines [10]; in universities for course and faculty evaluation; in government for political discussion [1] and voting; in resturaunts for customer feedback; people giving advice [13]; and many other uses.

Perhaps the strongest support for anonymity comes from the 1995 Supreme Court case of McIntyre vs. Ohio Elections Commision [1]. Margaret McIntyre, being against a proposed tax levy by Blendon Middle School in Ohio, created pamphlets stating her opposition, distributing some with her name, other anonymously. The levy eventually passed on its third attempt, however the Ohio Election Commision found her in violation of Ohio Election Code 3599.09:

"no material shall be distributed about an election that may sway its outcome without said material having a person or business responsible for the material."

Mrs. McIntyre was fined $100, at which point she appealed, the case eventually reaching the Supreme Court. The Supreme court ruled in favor of Mrs. McIntyre stating in the majority opinion written by Justice Stevens:

"The decision in favor of anonymity may be motivated by fear of economic or official retaliation, by concern about social ostracism, or merely by a desire

to preserve as much of one's privacy as possible.

... Anonymity is a shield from the tyranny of the majority. It thus exemplifies the purpose behind the Bill of Rights, and of the First Amendment in particular: to protect unpopular individuals from retaliation - and their ideas from suppression - at the hand of an intolerant society."

The internet is critical part of our daily life, used by millions to communicate, however it is a shared medium. Spyware, untrusted routers, packet sniffers, trojans, and wire tappers make any information transmitted over the internet susceptible to capture and analysis. Encryption may hide what is being said, anonymity hides who is saying it.

The ultimate goal in anonymity is to make it appear to an outside observer that no communication happened at all. The internet being a shared medium, we realize this is a reality which may never happen.

In leiu of this, we strive for sender anonymity, receiver anonymity, and sender-receiver unlinkability. From an observer's point of view this is attained if she cannot identify the sender, identify the receiver, or link the sender to the reciever.

When describing anonymous protocols, a participant is any entity in the protocol, an initiator is used interchangably with sender and is the originator of a message, the recipient or receiver is the intended recipient of the initiator's communication. An anonymity group is a set of participants working together as defined in each anonymous protocol.

The layout of the remainder of the paper is as follows. Section 2 provides a survey on the anonymous communication protocols available. A description of each protocol and its inner workings is described. Section 3 provides notes on performance, while section 4 provides vulnerabilities and attacks.

2 What's out there?

The internet, being a shared medium, does allow others to capture and analyze communications. However, by utilizing this shared medium, forming groups, we can hide our identity within the group. Much as a single conversation is difficult to hear in a crowded room, groups of participants form the driving force behind anonymity. Anonymous protocl can be classified into two broad categories, those using Mixes, and those using broadcasts or DC-Nets.

2.1 Mixes

David Chaum's seminal 1981 paper defined a method to anonymously send pieces of mail to other participants using "Mixes" [7]. A Mix accepts pieces of encrypted e-mail for delivery from many sources, holds, re-sorts, possibly introduces null mail, rewrites the from address, and possibly sends the mail to another mix for delivery. The end goal is each piece of output mail is equally likely to have come from any original sender. Generically, Mixes need not accept only e-mail, it will become clear from the context what the Mixes in each protocol accept. The following anonymous protocols base their anonymity on his work.

Figure 1 The "Mix" function

2.1.1 Remailers

The aptly named "remailers" are directly based on David Chaum's work and are the most direct implementation of his work.

Cypherpunk [14, 15] was the first remailer available and is commonly referred to as a "Type 1" remailer. Cypherpunk is the simplest of all remailers, they simply strip identifying information and forward the message to a recipient or another remailer in a first-in-first-out basis. If a sender wants to send a message through a chain of remailers it is up to the sender to form this chain.

Protocol	Sender Anon.	Receiver Anon.	Unlinkability	Domain
Type 1 Remailer			x	E-mail
Type 2 Remailer	x		x	E-mail
Type 3 Remailer	x	x	x	E-mail
Anonymizer	x		x	Web Surfing
Crowds	x		x	Web Surfing
WebMIX	x		x	Web Surfing
Hordes	x		x	Internet †
Onion Routing	x	x	x	Internet †
Herbivore	x	x	x	Internet †
P^5	x	x	x	Internet †

Table 1: Protocols discussed, what types of anonymity they offer, and their application.
†Internet here is used to denote the protocol is suitable for any internet application subject, subject to each protocol's implementation

Type 1 remailers have a few problems. They do not address return addresses. To return mail, the recipient would *a priori* need to know the sender's address, or the sender would need to include their address. This jeopordizes the sender's anonymity, all you get is sender-receiver unlinkability. Type 1 remailers also do not provide message padding of fixed lengths. It is important to pad messages to fixed lengths as an adversary who can infer information from the length of a message (i.e. this is an order receipt, or medical record) decreases anonymity. It is also important to store all messages received during time period T and forward them all at once. In this way, any one of the output messages may have originated from any one of the inputs. Enter "Mixmaster" [14, 15, 12], or Type 2 remailers, which implements these features. To address sender anonymity, Mixmaster also uses "reply-blocks". Previously, the only way a receiver could respond to an anonymous message was if she knew the sender. A reply-block is a set of instruction about how to send a piece of e-mail back to the sender, by using a remailer, or a chain of remailers. Through the use of reply blocks, sender anonymity is guaranteed. As of the writing of this paper, the Mixmaster protocol is an internet draft in the RFC series [18].

There is one issue that type 2 remailers do not solve. How does one maintain receiver anonymity? Senders still address their e-mails to the recipient's real e-mail address.

Type 3 remailers address this problem through the use of "Nymserver" [24, 16]. Type 3 remailers mirror the functionality of Type 2 servers, with the important addition of Nymservers that act as stores of virtual pseudonyms. They simply maintain a mapping of pseudonyms to return blocks. The return blocks, as described above, specify how to send a piece of e-mail back to some end point. Nym servers operate in a manner that not even the nym server operator cannot access the recipient address.

Other work has been done on remailers that is orthogonal to the remailers already described. Babel [19] is a Type 2 remailer, implemented as a series of Perl scripts that dates prior to Mixmaster. They were the first to suggest the use of reply-blocks and that reply blocks need not have an e-mail address endpoint. Instead, it is just as reasonable for them to end in a newsgroup to further protect receiver anonymity. This prevents a powerful attacker from sending a message to a pseudonym and watching where it travels in the network to find its destination. When the endpoint is a newsgroup, every member reading the newsgroup becomes the potential receiver. Babel also used fixed length messages, along with a chain of mixes, because it is undesirable to place complete trust in one individual mix. Babel is also the first to suggest the analogy of an onion representing a message traveling through a chain of mixes. A message is encrypted multiple times in such a way that only each mix can only decrypt a part of

the onion, or message. When a chain of mixes is chosen by the initiator, the message is encrypted in the reverse order in which it the Mixes will receive it. In this way, the first mix will be the only one that can decrypt the outermost encryption layer, unwrapping or decrypting that layer will tell the first mix whom to forward it to, along with a smaller onion that only the second mix can unwrap, and so on.

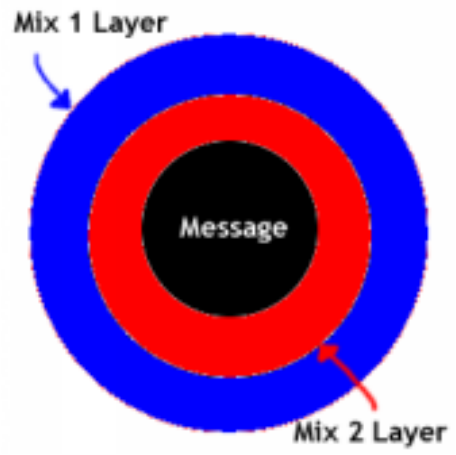

Figure 2 The layers of an onion. Each layer represents one layer of encryption that each Mix must "peel" or decrypt. The last Mix peels off the last layer revealing the message which is delivered to the receiver.

2.1.2 Anonymizer

Sending e-mail anonymously is great, but perhaps we would like to surf the web anonymously? Anonymizer [23] allows anonymous web surfing by acting as an intermidiary between the user and the real world. Anonymizer implements a one-hop HTTP proxy that strips identifying information from a request. In essence it fetches webpages for the user. To the contacted server it appears that one of anonymizer's servers contacted it. Anonymizer provides anonymity of the requester, but not the server returning the webpage.

As only one proxy sits between the user the end server, it does not provide a high degree of anonymity. Anonymizer is vulnerable to many of the attacks as listed in Section 4.

2.1.3 Crowds

Crowds [4] extends the idea of anonymizer by introducing many computers in the communication path between the initiator (web-surfer) and the receiver (the web server). In this manner, no single point of failure compromises the sender's anonymity.

When an initiator would like to make an anonymous request, they form a path of many proxies. They first contact a random Mix, or jondo in the Crowds protocol (pronounced John-Doe). This jondo creates a random path through the network that this and all subsequent requests will make. The first jondo, based on probability p_f decides whether to add another jondo, or forward the request to the end server. Each jondo participanting makes this decision independently. The last jondo on the path contacts the end server, the response from the end server simply follows the reverse path back to the initiator.

Crowds provides sender anonymity, not receiver, as the end server is contacted by the last jondo directly. As long as the sender maintains her anonymity, and does not reveal personal identifying information in a request, sender-receiver unlinkability is provided.

2.1.4 Hordes

Hordes [21] builds on the Crowds work, instead of serving as a proxy for a HTTP connection, the jondos in Hordes serve as proxies for UDP connections. Path creation work analagous to Crowds, however, a multicast return is used instead of the reverse path. When an initiator sends a request, a multicast address is picked on which return responses should be broadcast.

Again Hordes provides sender anonymity, not receiver, as at some point the last jondo makes a connection to the end receiver. Unless the receiver also participates in an anonymous protocol, an attacker may perceive it as receiving data (and possibly responding). Sender-receiver unlinkability is guaranteed if the sender does not betray personal information to the receiver.

2.1.5 WebMIX

WebMIX [20] provide a system similiar to Hordes and Crowds but addresses some of the vulnerabilies inherent in mix based systems. In particular, a system to prevent DoS attacks is presented. In WebMIX a ticket-based authentication system is presented, where participants on the network are issued a ticket to use the network. If the participant desires to communicate anonymously, she redeems her ticket and sends her message. This prevents users from flooding the network with traffic, WebMIX also addresses Mixes that drop messages or break messages by changing their contents. Each Mix will digitally sign each message as a sign that it did not tamper with the message. When a Mix cannot verify the digital signature of a given message it flags a signature-error. Each Mix is then forced to verify that it did not tamper with the message by publishing the encrypted message it received and its decrypted version. If one Mix cannot verify its correctness, it is dropped, if all Mixes verify correctness then the intiating user is dropped. [1]

2.1.6 Onion Routing and Tor

All the previous protocols focus on making e-mail or web surfing anonymous. With the exception of Hordes, the previous protocols focus on making one aspect of the internet anonymous. This moves us toward a generalized mix protocol that can handle any communication on the internet. Onion Routing [5] relies on a chain of Mixes, or Onion Routers, on which a participant created onion can travel. It is suggested that every participant run an Onion Router so that at least one hop of the network is trusted. [2] When a participant desires to send a message, they *a priori* choose a set of Onion Routers, encrypt their message in the public keys of the Onion Routers and send the message, or onion on its way.

Tor [17] is the next generation Onion Router which fixes many of the problems in the Onion Router specification. Tor separates the line between anonymity and "data cleansing". Data cleansing is the process of removing personal information (e.g. cookies, Active X Objects, etc), from message such as requests to web servers (HTTP). Many web servers will tag visitors with cookies to track user activity and discover how often visitors come back. Removal of this information is important to maintaining anonymity, Tor suggests the use of Privoxy [11], when using Tor to communicate anonymously.

Tor implements a SOCKS proxy to support the idea of a generalized anonymous communication protocol. A SOCKS proxy accepts requests for connections to other computers and decides whether to forward the connection or not. Their original purpose was to allow users to reach resources that may otherwise be unreachable (e.g. a computer sits in between a sender and receiver and blocks connections unless the SOCKS proxy is used). In the case of Tor, the SOCKS proxy serves as an entry point into the network. Any application that can use a SOCKS proxy can communicate anonymously.

Tor also addresses the need for "rendevous points" or the ability to find services anonymously. A rendevous point is an Onion Router that contains instruction about how to contact a receiver anonymously, similiar to a reply-block as described in remailers. Any user who would like to access the receiver can contact the Onion Router introduction point and request a session with the receiver who is identified through means outside of Tor.

Onion Routing tries to make an initiator's request appear equally likely to come from any Onion Router, in this way sender anonymity is provided, as well as sender-receiver unlinkability. Through the use and establishment of rendevous points, receiver anonymity is established.

2.2 DC-Nets and BroadCast Protocols

The following section discusses anonymous broadcast protocols including Herbivore which uses DC-Nets to create anonymity and P^5 which uses pure broadcasts. Herbivore is not a broadcast protocol in the sense that every node broadcasts every piece of information, it suffices to assign one participant the job

[1] WebMIX, Java Anon Proxy (JAP) are all part of the same system and are sometimes used interchangeably

[2] assuming one's own machine is trusted

of collecting broadcasts and sending them back to nodes when all have been collected. DC-Nets can be setup to work in broadcast node, however it is more effecient to run a fully-connected DC-Net in a star topology as shown by [3].

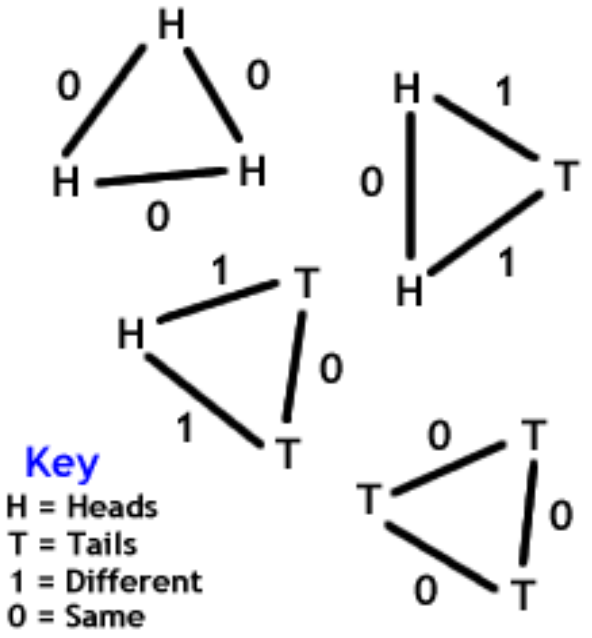

Figure 3 The four possible outcomes, a.k.a. proof by picture. For a more formal treatment, I refer the reader to the Appendix A of Herbivore [3]

Dining Cryptographer Networks [8] (DC-Nets) are a wonderful way to remain anonymous and stem from a problem set forth by David Chaum called the Dining Cryptographer's Problem. The story goes: three cryptographers are having dinner when the maitre'd approaches the trio and annouces that the bill will be paid by one of the cryptographers or the NSA in advance, but does not know which, and to please enjoy the dinner. The three cryptographers look at each other, in only a way cryptographers can look at each other; it is agreed that the three would like to know who is paying, but in an anonymous way. They devise a method to determine the payer, so that if it is one of the cryptographers, they will not know which. The three secretly flip a coin and share the result with the neighbor to their right. They each report the comparison of their coin with the neighbors (to the left) as either same or different. If one cryptographer paid, she should misreport her result. If no one misreports their result, the parity of differences should be even (either zero or two differences), if one of the cryptographer's paid, then the parity would be odd. The three cryptographers were happy with their new found ability and continued to enjoy their dinner.

One important difference between DC-Nets and Mixes is that DC-Nets provide perfect sender and receiver anonymity even under the presence of a local eavesdropper.

2.2.1 Herbivore

Herbivore [3] relies on DC-nets (Dining Cryptographer Networks) to create anonymity. In Herbivore, each participant shares a different secret with every other participant. This shared secret is used as a seed in a random number generator that generates only zeroes and ones. To send anonymous bits, each participant uses the random number generator to generate a zero or one key with every other participant. Each participant sums the keys along with a zero or one message that the participant would like to send. The parity of this sum is broadcast to all other participants (zero for even, one for odd). Assuming only one person is communicating an arbitrary message of one or zero and all others are tranmissiting a message of zero, the parity of the sum of all broadcasts will be that of the sender's message. This occurs because each key is summed twice, the sum of all keys will always have even parity ($\equiv 0 \bmod 2$). Without knowing the keys, it is equally likely that any person is the initiator.

However, the DC-net is not enough to run the network. The DC-net does not address who sends when, or joining or leaving the network. Herbivore addresses this by dividing communication into three steps. First, nodes in a group reserve transmission time. If there are n tranmission slots then each node will send n anonymous bits, sending a 1 if she would like to transmit in that slot. Second, actual transmis-

sion occurs. Each node that reserved tramission time transmits their message in their reserved slot. Last, an anonymous vote is taken about whether this is the appropriate time for nodes who wish to leave to do so. It is important for nodes to leave only when a mutual agreement of the group is reached for reasons described in the predecessor attack in section 4. However, nodes in the network can crash and leave the network at any time. This will degrade anonymity a small amount, and this must be tolerated accordingly.

Herbivore uses a star-topology to cut down on the high amount of communication required to maintain anonymity. One nodes acts a center for each round, accepting all communication from every other participant and returning broadcasts to the participants. The center node rotates every round, in a round-robin fashion, so that no one node has an unfair network load.

Sender anonymity is guaranteed by DC-nets. It is equally likely that any member of a clique in Herbivore lied about their broadcast. Herbivore provides a method to address another Herbivore member through the use of broadcasts. Receiver anonymity is insured by these broadcasts. By using these together, sender-receiver unlinkability is also guaranteed.

2.2.2 P^5

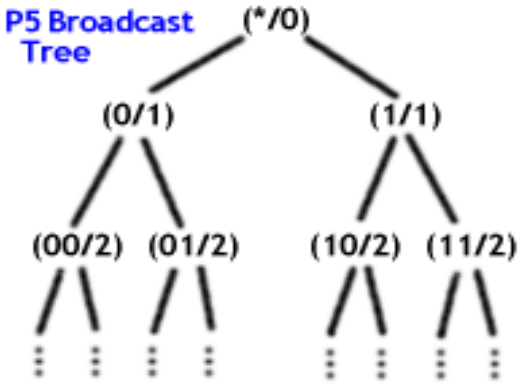

Figure 4 P^5 anonymity tree

A participant in P^5 [9] secretly choose a key which is then hashed to form a P^5 key K. K is used to map a participant to a anonymity group using the following protocol. Each group in P^5 is defined by the by the terminology (b/m) where b is string of zeroes and ones of m the length of that string. Suppose a participant has $K = 011001$, if she would like to join the group $(01/2)$, she would choose $m = 2$, then taking the high bits of her key K, she would be placed in $(01/2)$. By using this method, every participant is then mapped to an anonymity group by their personal choice m, and the value of the high bits of their key K. The choice of m is illuminated in a moment.

As the reader can imagine, this grouping forms a binary tree as in figure 4. When a user would like to send to a particular group, (b/m), the message is sent to that group, as well as every other group having a child/parent relationship with that group. For example, sending to group $(01/2)$ also broadcasts to the parents $(0/1), (*/0)$ and the children

$$\{(010/3), (011/3), (0101/4),$$
$$(0100/4), (0110/4), (0111/4), ...\}$$

The choice of m is determined by the user and should not be revealed. If a participant would like to initiate communication to another participant of P^5 without knowing where they are located in the broadcast tree, they could first try to send to $(*/0)$, perhaps with little success because of the large number of broadcasts that are dropped at this level. The initiator may then try to recursively dig down one of the tree branches, for example say, $(1/1)$, then $(10/2)$, then $(100/3)$, and so forth. When a sender broadcasts close to the receiver's actual choice of m, it is suggested they stop answering communication, as any further digging would effectively shrink the number of anonymous groups in which she can hide. It follows that the choice of m and when to stop answering communication should reflect how anonymous one would like to remain in P^5.

Receiver anonymity is provided by the broadcast nature of P^5. To preserve sender anonymity and sender-receiver unlinkability, each node sends fixed amount of noise to uniformly distributed locations.

In this manner, it is impossible to determine genuine traffic from noise.

3 Performance

Depending on the system, a cost for anonymity is paid in bandwidth, latency, or both. Table 2 shows the number of bits needed to send one anonymous bit. m is the group size in the case of Herbivore and P^5 and n is the number of entities participating in a path helping some initiator I communicate in Remailers, Onion Routing, Anonymizer, Hordes, Crowds, and WebMIX. d is the depth of the broadcast address in P^5 and D is the maximum depth of the broadcast tree.

Protocol	Number of Bits
Anonymizer	2
Remailers	$(n+1)$
Onion Routing	$(n+1)$
Crowds	$(n+1)$
Hordes	$(n+1)$
WebMIX	$(n+1)$
Herbivore	$2(m-1)$
P^5	$m(2^{D-d}+d+1)$

Table 2: Number of bits required to send one anonymous bit

Anonymizer has the smallest requirment, but also provides the least anonymity. P^5 has the largest requirement, however P^5 has message dropping algorithm where messages sent higher in the tree are dropped with increasingly exponential probability. It is not surprising that Remailers, Onion Routing, Hordes, and Crowds all have the same bit requirement as they are all based loosely on Mixes. Herbivore uses twice that of its equivilents that use Mixes, but it is important to note that group sizes typically run larger than the number of entities pariticipating in Onion Routing and others.

In the Crowds paper, a useful taxonomy for classying the level of anonymity is proposed. Levine and Sheilds went on to add a quantitative scale to the Crowd's taxonomy. The following taxonomy can be thought of as a continuum between 0 and 1, with zero being no anonymity and one being complete privacy. $Pr_e(x)$ is the probability that x is the initiator of some communication from the viewpoint of adversary e. For some anonymous group S, it holds that $\Sigma_{x \in S} Pr_e(x) = 1$. From [21], the anonymity provided for participant x from adversary e denoted by $d_{e,x}(A)$ using anonymous protocol A is:

$$d_{e,x}(A) = \sum_{y \in S \neq x} Pr_e(y)$$

- Absolute Privacy - An attacker cannot discern any communication occured. $d_{x,e}(A) = 1$.
- Beyond Suspicion - Signs of communication are visible to an attacker, the initiator appears no more likely than other in the protocol $d_{x,e}(A) \geq (1 - 1/|S|)$ and $d_{y,e}(A) \leq d_{x,e}(A)$ for all $y \neq x \in S$.
- Probable Innocence - An entity is no more likely to be the initiator than not, although an attacker suspects one entity is more likely the initiator $1/2 \leq d_{x,e}(A) \leq d_{y,e}(A)$ for all $y \neq x \in S$.
- Exposed - There still exists some probability the attacker cannot identify the initiator, although this probability is decreasingly small, $0 < d_{x,e}(A) < 1/2$.
- Provably Exposed - An attacker can prove the identity of the initiator to others, $d_{x,e}(A) = 0$.

4 Vulnerabilities and Security

The end goal of all anonymous system described is to attain probable innocence, or greater. Absoluate privacy may never be acheivable, as the internet is a shared-medium, a powerful adversary may have access to all traffic crossing the internet. Below I list many of the generic attacks an attacker may degrade the anonymity of a participant from beyond suspicion to exposed or even worse, provably exposed.

4.1 Predecessor Attack

The predecessor attack works through the combination of a malicious entity participating correctly in

the protocol and time. It applies to all protocols except for a fully connected DC-Net (like Herbivore). In [6], Wright shows, given enough time and collaborators, the probability of identifying an initiator I with a responder R approaches arbitrarily close to 1 as time approaches infinity. The proof assumes I is the only one communicating with R. Malicious nodes sit on the network and record the likely initiators of a communication when R is involved as the receiver. Over time, the probability of I being the initiator will approach 1 as I will appear in the possible initiator set more often than any other initiator.

In other words, by using the receiver as a reference, we record all possible initiators when that receiver is contacted. We discover the initiator who appears the largest number of times in the intersection of all sets of possible initiators. This attack works because anonymity groups must be reformed every so often as people leave and join the anonymity group, and initiators will often contact the same receiver (e.g. an initiator reading the news at http://www.slashdot.org/ every day). This demonstrates a need for participants to participate in the protocol as long as possible, even if they are not actively using it. This keeps paths from being reformed, which would change the set of possible initiators. This goal often conflicts with new participants joining the network, when they join, they force everyone to create new paths. If path reformation does not happen when a participant joins, then any new path could be attributed to that participant.

By using the predecessor attack, an adversary can degrade a participant's anonymity in any protocol to exposed ($0 < d_{x,e}(A) < 1/2$), but never provably exposed, as there is always the tinest chance the initiator appears most often by coincidence.

4.2 Denial of Server (DoS) Attacks

Denial of Service attacks are a concern for anonymous systems as they are for all systems connected to a shared-medium network. Since the communication is anonymous, a malicious user conducting a denail of service attack is also anonymous, and undetectable. Ironically, the fully connect DC-nets which are resilient to the predecessor attack are most vulernable to anonymous DoS attacks. These attacks can work in many modes, the malicious node(s) can flood the network with random traffic, follow the protocol but use all the transmission time, act slowly in protocols such as Herbivore where all broadcasts must be collected before continuing, or simply refuse to pass along broadcasts and drop requests all together.

All protocols are vulnerable to DoS attacks, some protocols handle DoS attacks more gracefully than others. Tor suggests robustness of the network to mitigate DoS attacks, while Herbivore suggests if a group is running too slowly, to simply join another group. Herbivore is vulnerable to DoS attacks through reserving too many transmission slots, or by creating a collision during the transmission phase. Herbivore forces users to join a random anonymity group and provides a rate-limiting scheme to prevent too many malicious nodes from joining a group at once. As a result, it is hard to DoS a specific group in Herbivore. [3]

Crowds, Hordes, and Remailers make no effort to defend against DoS attacks [4, 21]. Attacks against the network results in increased communication times and possibly communication, or e-mail being dropped. A novel way to defend against DoS attacks is introduce a ticket-based system. Every client is issued so many tickets that they redeem for time on the network, as in WebMIX. However, it does not address a powerful attacker who has the ability to create an unlimited number of virtual identities, and collect a large number of tickets.

4.3 Sybil Attacks

A powerful attacker who can create a large number of virtual identities segues into a larger vulnerability called a "Sybil Attack" [22]. The Sybil Attack occurs when large number of malicious nodes (or forged nodes) join the network in a short time interval in order to use the protocols method for joining to create a DoS on itself, or flood out a legitimate user joining within the same time period, thus compromising the lone legitimate user's anonymity.

[3] It is still possible to DoS a user if their IP address is known, the Herbivore protocol makes it difficult to DoS a specific user anonymously

Herbivore mitigates this problem without identifying the malicious user by limiting the rate at which new participants can join a given group. This prevents honest nodes from being crowded and forces the attacker to spend an enormous amount of resources if she would like to crowd out an honest node.

P5 allows a Sybil attack as each participant may choose which broadcast group they join through a dictionary attack on the keys. Onion Routing allows the pariticipant to choose the path their onion will follow through the network. Through creating many identities or controlling many computers, an adversary can control large portions of a Hordes, WebMIX, Crowds, or remailer network.

A successful Sybil Attack will either degrade the protocol's preformance, or degrade a participant's anonymity to exposed, or worse provably exposed in the case where malicious user(s) are able to crowd out all but one legitimate user.

4.4 Local Eavesdropper

Local eavesdropper are a problem in protocols where every participant is not sending and receiving uniformly. For example, when an initiator in the Onion Protocols desires to communicate, a first outbound connection must be made. A local eavesdropper can detect this, although in the Onion Protocol, encryption is used to hide the communication contents, so the attacker cannot tell what is being said. Even though receiver anonymity, and sender-receiver unlinkability has been maintained, the senders anonymity is compromised. If the attacker is powerful enough to access other network traffic, the attacker can use the timing of the sender's message to determine a probable list of recipients.

Onion Routing, Hordes, Crowds, Hordes, and WebMIX are especially vulnerable to this as try to maintain real-time communication. Remailers are vulnerable to a lesser degree since it is not critical that e-mail be delivered in real-time. Remailers can take hours or even days to deliver pieces of mail which lowers the correlation between inputs and output.

Herbivore and P5 do not fall into this category as the protocols require participants to broadcast zero or null messages when they have nothing to send. Both use broadcasts to receive messages, every participant becomes a potential receiver.

5 Conclusion and Discussion

This survey of anonymous protocol raises many questions about anonymous protocols, their use, and future which is now discussed.

Do common users know how to use anonymous protocols, why they would need them, or where to go if they did need anonymity?

How do we prevent malicious participants from degrading the protocol when they are doing so anonymously? Is it possible to prevent nefarious activity when participants remain anonymous? Despite the many good uses listed in Section 1, anonymity and the protocols that create it may find nefarious use in planning terrorist plots, distributing copyrighted material, or launching an attack on hosts on the internet. This behavior existed before anonymous protocols and will exist with or without anonymous protocols. It does raise the questions, can we detect people who are misbehaving anonymously, or even should we?

From a technical standpoint, is it possible to prevent the predecessor attack on long lived or reoccuring transactions? In Mix based systems, does creating the path on the fly provide like in Crowds create better anonymity that creating it *a priori* as in Onion Routing? Is there a quantitative trade off between anonymity, bandwidth, and latency? Is it possible in an anonymous protocol to make other participants seem probably innocent so that the true initiator seems beyond suspicion?

Despite the many open questions with anonymous protocols, good progress is being made. As noted, The Mixmaster protocol is on track to become a RFC, and has public interfaces available to send anonymous e-mail. A public implementation of the Onion Router exists for anyone who would like to run one. As of the time of this writing there are 36 Onion Routers operating on the internet [2]. Hopefully the future will see raised public awareness, and greater access to public interfaces, where we will need not fear censorship and privacy will as free and open as the internet itself.

Acknowledgements

I would like to thank Kay Connelly for getting me started on this paper and for her many helpful comments. I would also like to thank Scott Jones and Heather Teed for their comments and encouragment, I am more grateful than perhaps they know.

References

[1] Supreme Court Case McIntyre vs. Ohio Elections Commision 514 U.S. 334 (1995) http://supct.law.cornell.edu/supct/html/93-986.ZO.html Retrived July 8, 2004

[2] http://www.noreply.org/tor-running-routers/ Retrieved September 16, 2004

[3] Goel, S., Robson, M., Polte, M., Gun Sirer, E. Herbivore: A Scalable and Effecient Protocol for Anonymous Communication (2003) *Cornell University technical report*

[4] Reiter, M., Rubin, A. (1998) Crowds: Anonymity for Web Transactions *In ACM Transactions on Information and System Security* 1(1)

[5] Syverson, P., Goldschlag, D., Reed, M. Onion Routing Access Configurations (2000) *In the DARPA Information Survivability Conference and Exposition* 34-40

[6] Wright, M., Adler, M., Levine, B., Shields, C. An Analysis of the Degradation of Anonymous Protocols (2002) *In the Proceedings of the Network and Distributed Security Symposium*

[7] Chaum, David (1981) Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms *In Communications of the ACM* 4(2)

[8] Chaum, David (1988) The Dining Cryptographer Problem: Unconditional Sender and Recipient Untraceability *In Journal of Cryptology 1* 65-75

[9] Sherwood, R., Bhattacharjee, B., Srinivasan, A. (2000) P^5: A Protocol for Scalable Anonymous Communication *In the Proceedings of the 2002 IEEE Symposium on Security and Privacy*

[10] WeTip Crime Reporting http://www.wetip.com/ Retrieved August 12, 2004

[11] Privoxy http://www.privoxy.org/ Retrieved September 16, 2004

[12] Mixmaster Type II Remailer http://mixmaster.sourceforge.net/ Retrieved August 15, 2004

[13] AdviceBox http://www.advicebox.com/ Retrieved August 15, 2004

[14] WikiPedia: Anonymous Remailer http://www.wowarea.com/english/help/remailer.htm Retrieved August 15, 2004

[15] StayInvisible.com http://www.stayinvisible.com/index.pl/e_privacy_remailers Retrieved August 16, 2004

[16] Wikipedia: Nym Servers http://en.wikipedia.org/wiki/Nym_server Retrieved August 16, 2004

[17] Dingledine, R., Mathewson, N., Syverson, P. (2004) Tor: The Second-Generation Onion Router Roger Dingledine, Nick Mathewson, Paul Syverson *In the Proceedings of the 13th USENIX Security Symposium*

[18] Mixmaster Protocol Version 2 http://www.ietf.org/internet-drafts/draft-sassaman-mixmaster-02.txt Retrieved August 16, 2004

[19] Gulcu, C., Tsudik, G. (1996) Mixing E-mail with BABEL *Proceedings of Network and Distributed Security Symposium* 2-16

[20] Berthold, O., Federrath, H., Kospell, H. Web MIXes: A System for Anonymous and Unobservable Internet Access (2001) *International workshop on Designing privacy enhancing technologies: design issues in anonymity and unobservability* 115-129

[21] Shields, C., Levine, B. A Protocol for Anonymous Communication Over the Internet (2000)

[22] Douceur, J. The Sybil Attack (2002) *In the Proceedings of the 1st International Peer To Peer Systems Workshop*

[23] Anonymizer http://www.anonymizer.com/ Retrieved August 12, 2004

[24] Danezis, G., Dingledine, R., Mathewson, N. Mixmion: Design of a Type III Anonymous Remailer Protocol (2003) *Proceedings of the 2003 IEEE Symposium on Security and Privacy*