

A Framework for Local Anonymity in the Internet*

David M. Martin Jr.
Department of Computer Science
111 Cummington Street
Boston, MA 02215
dm@cs.bu.edu

21st February 1998

Abstract

We describe and evaluate options for providing anonymous IP service, argue for the further investigation of local anonymity, and sketch a framework for the implementation of locally anonymous networks.

1 Introduction

Anonymity is not the same as confidentiality. Existing systems such as IPSEC can ably protect the contents of IP-based communication [Atk95, Pos81b], making it impractical for network eavesdroppers to deduce the purpose of observed traffic. However, the fact of the traffic itself is not hidden; the sender and receiver of each packet is clearly visible in the packet headers even when the packet is end-to-end encrypted. Such information should sometimes be protected as well.

Suppose that Alice, the CEO of Apricot, decides to buy out the publicly-traded firm Banananon. When other Apricot officers learn of the plans, the subsequent flurry of connections from the executive suite to the Banananon's web site could betray Alice's intent. If Alice's network administrator Eve reacts to the network connection logs by purchasing Banananon stock, then Alice and others could end up in serious trouble. Anonymity schemes can prevent Eve from identifying the executive suite as the source of the traffic, even though the traffic flows through Eve's equipment.

The literature defines three kinds of anonymity. We are interested in all three as applied to IP traffic. *Sender anonymity* refers to message formats and transmission techniques that do not identify the sender. Sender anonymity would protect Alice from Eve's snooping in

the above example. *Receiver anonymity* means that the intended recipient(s) cannot be inferred from the message; this would prevent Eve from seeing that the web traffic concerned Banananon at all. *Unlinkability* means that independently observed messages seem uncorrelated even when they comprise a single logical traffic stream.

Unlinkability only makes sense when the semantics of the message flows do not require a priori correlations, so unlinkability must always be considered with respect a certain granularity. For example, TCP [Pos81c] streams require recipients to acknowledge sent data. Therefore, if an eavesdropper knows that a certain outbound flow comprises a TCP stream, then they will rightly expect to observe (at least) an inbound flow of acknowledgments. Even without knowing which node is the communication endpoint, the eavesdropper can infer that outbound and inbound packets concern the same TCP stream. This implies that providing packet-level unlinkability of TCP streams requires generating a large amount of dummy traffic just to obscure the acknowledgment packets. It probably makes more sense for a security designer to instead take entire TCP streams as the primitive units of unlinkability.

Often the granularity is even larger, as in the case of most web traffic, where the first TCP connection delivers the HTML source and separate TCP connections are created to fetch embedded images and other secondary structures [BLC95, FGM⁺97]. An eavesdropper aware that an HTML transfer is in progress would have good reason to suspect that two back-to-back TCP streams originate from the same node even if their source labels appear unrelated. In general, unlinkability stands at odds with efficiency in communication. The anonymity designer wants to dissociate current communications from future ones, while the network designer wants to minimize unnecessary delays.

Computer systems offering network anonymity fall into two broad categories: the *dispersed* or *global* schemes

*This research was supported in part by NSF grant CCR-9706685.

based on large-scale multi-domain infrastructure support, and the *end-to-end* or *local* schemes whose security follows from purely local cooperation. The first category includes trusted-host rewriting schemes such as those used in `anon.penet.fi` and the Web Anonymizer. Mix-derived [Cha81] systems such as Cypherpunk, Mixmaster, and Babel remailers; Onion Routing; Crowds; and the Non-Disclosure Method for mobile IP stations also fall into the family of dispersed architectures. Almost as well represented in the literature, but largely unimplemented, are the local schemes invariably based on DC networks [Cha88]. The following sections address both kinds of schemes in turn.

1.1 Dispersed Anonymity Architecture

Generally speaking, the dispersed designs provide comprehensive untraceability in the following manner. The promoters of a dispersed anonymous service first deposit forwarding agents throughout the Internet. When a client requests an anonymous connection, it aims its encrypted data streams at or through several such agents, each of which is entrusted to redirect the stream appropriately. By the time the stream appears at its desired endpoint, the receiver can only trace the nearest hop in the route. In a Mix system, the data is additionally encoded so that *each* forwarding agent, not just the ultimate receiver, can only trace the stream one hop in either direction along a stream's route. Therefore, even informers close to the source or destination of the stream can see only that the stream is being sent on an anonymous channel, not where the channel begins or ends. (Naturally, details and subtle attacks make an actual implementation more complex than this in practice.) Widely deployed, such a system can be visualized as a cotton ball with threads—possible routes—going every which way. The perspective one has while sitting on a thread makes it very difficult to find its endpoints.

1.2 Local Anonymity Architecture

Local anonymity schemes can support stronger untraceability guarantees than global ones, but they exact severe performance penalties and have therefore attracted little attention among implementors. In a DC network, the best example of local anonymity, each node in a coalition on a local area network continually exchanges encrypted messages at a constant rate, and only a fraction of that bandwidth is used for actual data transfer between nodes. The untraceability of such data then depends directly on the quality of random numbers used. By using one-time pads,

the source of any piece of data in the coalition can be utterly hidden.

Such a design is probably unusable on any wide area network with the poor and inconsistent latency of the Internet for applications requiring even human interaction speeds. **Instead, we propose gatewaying a locally untraceable stream over a standard high-bandwidth channel to a remote destination.** This design offers *end-to-end anonymity*; although the remote site can easily see that the data came from a particular coalition of nodes, it is unable to identify any particular node as the data's creator. The picture for such a scheme might be a Q-tip: a well-defined backbone with easily locatable extremes, but chaos when you look at the endpoints closely.

After reviewing features of the current art in Section 2, we argue for the deployment of local anonymity in Section 3, present the outline of an flexible encapsulating framework for local schemes in Section 4, describe two possible anonymity-realizing protocols in Section 5, and conclude in Section 6.

2 Related Work

Almost every implemented anonymous network service currently described in the literature—production *or* experimental—is based on a dispersed architecture. In turn, most of these are Mix-derived. However, two low-security simple rewriting schemes have completely dominated the Internet anonymity field: the Penet mail rewriter and the (Web) Anonymizer.

Mix networks and DC networks were both invented by David Chaum; see [Cha81] and [Cha88] for his original papers on the topic. Mix networks have been analyzed, extended, and implemented in [PPW89, Pfi90, PPW91, PP90]. DC networks are usually considered a more elegant but less practical anonymity scheme than Mix networks. Accordingly, the literature describing DC networks is mostly theoretical [Wai90, WP90, Pfi90, Böt90, DO97].

In the following sections, we briefly sketch the implementations and designs relevant to our pursuit.

2.1 Anon.penet.fi

The Internet host `anon.penet.fi`, maintained by Johan Helsingius, ran as a cleartext email forwarding agent from 1993 until it closed in 1996. It offered anonymity by maintaining a private database associating real and assigned anonymous email addresses and rewriting addresses appropriately when mail was directed through it. Subpoenas demanding the secret mapping for specific

email addresses were an obviously major factor in the service's demise; see [Hel95, New97] for details. At its peak, the service claimed a mapping for 500,000 users and was indisputably the most important service of its kind.

2.2 Cypherpunk Remailers

The Cypherpunk remailers remove the persistent database from the `anon.penet.fi` approach, add optional encryption, and allow a sender to route an email through several remailers before hitting its destination. Without the persistent database, no anonymous ID is available to enable return mail (although the service `nym.alias.net` [Nym97] helps with this problem). Since there is no single point of presence, and since the individual remailers are very easy to set up, legal action or network attacks are unlikely to threaten the system as a whole. While court action or service provider protest could easily shut down a remailer, there is no database to raid. Meanwhile, another remailer would appear elsewhere. Cypherpunk remailers are widely available; see [Lev].

2.3 The (Web) Anonymizer

The Anonymizer <http://www.anonymizer.com>, provided by Anonymizer, Inc., can be thought of as a remailer for HTTP [BLFN96, FGM⁺97] traffic. Since HTTP transactions are completely self-contained, no long-term mapping for return addresses is required. The Anonymizer just removes revealing HTTP headers and relays the results. As with Cypherpunk remailers, subpoenas are intrinsically insufficient to recover the identity of a past user: once an HTTP transaction completes (typically in seconds), the Anonymizer can forget the mapping used, and so be truly unable to comply with such demands.

However, Anonymizer, Inc. has made some interesting policy decisions regarding the use of their service, most of which seem fully informed by the legal reality of doing business in California and the United States. In the Terms and Conditions as of September 24, 1997 [Ano97b], they write

8.3 Usage logs are usually kept for fifteen (15) days for maintenance purposes, monitoring Spamming and monitoring abuses of netiquette. Any relevant portion(s) of such logs may be kept for as long as needed to stop the abuses.

In addition, their FAQ clearly states their intention to use these logs in order to cooperate with appropriate authorities. So the service closely parallels `anon.penet.fi`

in terms of confidentiality even though the same technical guarantees of untraceability offered by Cypherpunk remailers are possible. Unlike `anon.penet.fi`, Anonymizer, Inc. does not intend to shut down if it is merely required to reveal a user's identity.

2.4 A Pedagogical Mix Network

An implementation of the Mix design was first described in Andreas Ort's Diplomarbeit [Ort91]. This paper describes an Appletalk-based Mix network at the Karlsruhe Institut für Rechnerentwurf und Fehlertoleranz and a series of lab-style experiments designed for interns. At the time, the Macintosh OS did not support multiprogramming, so nodes in the experiment were either purely Mix forwarders or specialized application endpoints. The system supported one-way anonymous messages, not bidirectional streams.

2.5 The Mixmaster and Babel Remailers

The Mixmaster for email by Lance Cottrell¹ [Cot] is probably the most perfectly suited implementation of Mix technology to date. It includes facets of the Mix design such as replay detection, uniform sizing, buffering, and reordering, all of which contribute to the untraceability of messages. The technology is almost as easy to deploy as Cypherpunk remailers, the only difference being that one needs to use a special client for sending mail with the Mixmaster. The Mixmaster is currently available within the U.S. at [Cot96].

Gene Tsudik and Ceki Gülcü at IBM Zurich developed a similar remailer called Babel, described in detail in [GT96], that improves on the design and exposition of the Mixmaster. Babel extends the possibility of anonymous reply to receivers of anonymous email with reply "onions", a feature envisioned by Chaum [Cha81] and implemented both in the Non-Disclosure Method described below and the remailer reply service `nym.alias.net` [Nym97] (which is not described here). The Babel implementation is not currently available to the public.

Generally speaking, the Mix design works well with email because email tends to be insensitive to network latency. One Mix option imposes random latencies on all traffic in order to foil attempts to correlate the input and output packets on a single mix node; this would obviously not work well with latency-sensitive applications. Discounting distributed and coordinated network observers allows one to dispense with the random latencies, but mul-

¹Also President of Anonymizer, Inc.

multiple Mix hops (and the attendant latencies) are still recommended to hide identities.

2.6 Onion Routing

Michael Reed, Paul Syverson, and David Goldschlag at the U.S. Naval Research Laboratory describe an implementation based on the Mix design called Onion Routing [GRS96, SGR97]. Their implementation defines protocols for bidirectional streams, currently rlogin, HTTP, and SMTP (email). The forwarding nodes called *onion routers* maintain permanent encrypted TCP connections to each other, and the Mix structure is built on top of that virtual network. Client applications encounter the anonymous network through the *application proxy* module of a fixed and trusted onion router. The proxy then operates on the client's behalf.

When opening an anonymous connection stream (e.g., an HTTP GET from a Web browser), a proxy chooses a random path of routers through the onion network to its destination and constructs an onion corresponding to that path in the classical Mix form: each layer of the onion both *represents* a router in the path, and when "peeled off", recursively *contains* the remaining path encrypted with the router's public key. As a performance optimization, the proxy also chooses and deposits a symmetric key and a stream identifier into each layer to be used in future communication on the route. Having constructed the onion structure, the proxy injects it into the network via the router named by the outermost layer of the onion. This router peels off the outside layer (which only it can do, since it alone possesses the correct private key), extracts and stores its symmetric key and stream identifier, and passes on the remaining encrypted layers to the next router named in the onion. By the time the onion reaches its destination, all of the layers have been peeled off and the proxy's chosen route is represented in the onion network by the states of the incident routers, each of which knows the stream identifier and corresponding symmetric key. At this point the anonymous route is complete, and the stream identifiers and symmetric keys between nodes are used to transfer data on that connection at high speed.

Several experimental onion networks have been established, and a next-generation system is in progress. See [Pri] for information on the current availability of Onion Routing.

2.7 Crowds

Michael Reiter and Aviel Rubin of AT&T Research have developed a system called Crowds [RR97] with the same purview as Onion Routing but only loosely related to the

Mix design. Crowds falls nonetheless squarely into the family of dispersed anonymity architectures. As in Onion Routing, anonymous traffic visits a secret sequence of forwarding nodes in the network represented by a sequence of path identifiers and protected by symmetric keys, and client applications use proxies to access the anonymous network. However, an originating proxy does not choose its own path. Instead, at the Crowds network boot time, each forwarding node creates and sends a special path construction packet on a random walk through the network: on receiving such a packet, a node flips a coin to decide whether to lengthen the path with a randomly chosen successor node or to conclude the path at the current node. (The expectation and variance of the path length can thus be controlled by adjusting the coin's bias.)

After this phase of random walks, the Crowds network state encodes a set of secret paths: each Crowds node has exactly one secret path beginning at that node and ending at another node (possibly the same one). Furthermore, the chosen secret path is used for all subsequent traffic originating at that Crowds node. That is, when a client wishes to perform an anonymous transaction (such as a Web browser issuing an HTTP GET), it contacts the proxy side of its trusted Crowds node, which then routes the client's stream along the single Crowds path originating at that node.

Crowds has some advantages over Onion Routing. Firstly, it requires much less encryption: a *single* symmetric key is shared by every node on a path and used to protect the traffic, and no asymmetric cryptography is required at all. Secondly, the designers have built a mechanism that allows both users and nodes to join and exit the network easily. In contrast, most of the other anonymous network implementations do not easily adapt to topology changes. And thirdly, the ability of each node to see the final destination of a packet allows it to choose an alternate path in case of failure. This is impossible in a classical Mix network because each Mix node can only see its immediate neighbors on a path. However, the advertisement of final destinations to every encountered node may also be seen as a disadvantage, since it stands at odds with receiver anonymity.

The Crowds system is currently in alpha testing phase. See [Cro] for more information.

2.8 The Non-Disclosure Method

The Mix-based Non-Disclosure Method (NDM) of Andreas Fasbender, Dogan Kesdogan, and Olaf Kubitz [FKK96b] hides the source and destination addresses of an IP packet from every node except the packet's destination. Their primary application is in mobile IP, where a

mobile computer’s current location (i.e., its transient foreign agent or “care-of” IP address) should not be made public. NDM ensures this privacy by tunneling packets from the mobile station’s permanent home network through a Mix-like network to the foreign agent responsible for final LAN delivery.

As in Onion Routing and Crowds, packets are neither delayed nor reordered at the internal forwarding nodes. Since the packet’s destination node (e.g., the mobile IP station) receives the packet’s source address, it knows where to direct its reply—no reply onions are required. This allows every packet to be routed using an independently chosen Mix path. In addition, each packet creates a transient virtual circuit on the way to its destination. If the destination turns out to be unreachable, the intervening nodes can generate and transmit the appropriate ICMP [Pos81a] error packet to the source.

In [FKK96a], the same authors compute a probability distribution predicting the latency added by NDM using equipment assumptions accurate in 1995. For a route through three forwarding nodes equipped with RSA hardware, they estimate an additional delay of less than 100 milliseconds, which makes the system quite practical for interactive use.

2.9 Summary

Let us summarize and make some observations about the current art. Since our goal is to consider anonymity for IP services in general, including those with response time requirements, we concentrate on the properties of Onion Routing, Crowds, and NDM. Each of these dispersed systems achieve excellent untraceability properties by crossing multiple administrative domains; only the most determined, powerful, and coordinated attackers could defeat them. In each scheme, a connection is built between a client requesting attention from a server. Both client and server are identified by IP addresses. Table 1 shows which of those addresses protected from participants in a packet transmission in the three schemes.

	Addresses Hidden	Onion Routing	Crowds	NDM
Forwarding Agents	Client	x	x	x
	Server	x		x
Internet Observers	Client	x	x	x
	Server	x	x	x
Server	Client	x	x	

Table 1: IP addresses hidden from participants in the low-latency dispersed anonymity schemes.

3 The Case for Local Anonymity

Good as they are, the above schemes are not the end of the story. In this section, we argue that local anonymity has important advantages over dispersed anonymity including *sustainability* in terms of *performance* and *fault tolerance*, and *responsibility*. Meanwhile, we must recognize its primary disadvantage: locally anonymous packets can easily be traced to a (probably small) coalition of likely suspects.

3.1 Sustainability

Once a packet is locally anonymized and emitted, it is treated the same as non-anonymous packets: it comprises the same statistics and is subject to precisely the same routing rules as ordinary ones. Therefore, when communicating with a remote party, a locally anonymous client’s packets will have service (e.g., latency and throughput) characteristics no worse than those of the best non-anonymous packets emerging from the same LAN. The equipment involved in anonymization is determined locally, so an anonymity administrator² can take local performance and management requirements into consideration when deploying, improving, or eliminating the service. There is usually considerably better network service available between neighboring nodes than between nodes several hops removed—often by at least one order of magnitude—and this abundance of local network resources permits the use of high security and otherwise impractical anonymity-preserving schemes, such as DC networks.

In short, the support of anonymity does not in itself make the administrator beholden to external agents. Since the policy and implementation of the local scheme falls entirely within the administrator’s realm, it is indefinitely sustainable, requiring attention only when the local situation changes.

A similar line of reasoning has long been used in the networking community to inform decisions about network architecture in general; it is called the *end-to-end argument*, and is particularly taken to heart in today’s Internet. One way to phrase the argument is as follows: “The more you rely on your neighbor to provide network services, the more you are at his mercy.” Stateless IP routers and TCP timeouts, retransmission, and window control are all realizations of this principle.

Compare this to the method of dispersed anonymity schemes. Each one superposes a meta-network onto the real network for the sake of anonymity alone. In exchange

²The anonymity administrator need not be the general network administrator.

for making the identification of an endpoint or even the neighborhood of an endpoint difficult, the administrators have to cooperatively manage a logically distinct network and its topology, performance, and failure modes. Such an undertaking is fraught with real-world concerns. For instance, how does a new client node join the network? Is it required to supply a forwarding agent as well? Where can new clients and forwarding agents tap into the existing network, and with whom do they exchange keys? Why should established forwarding nodes trust new ones? Do such topology changes require manual intervention?

If 50% of the forwarding agents are unreachable, should the reachable agents automatically pick another route or consider it an active attack? What about 20%? Is it ever appropriate to choose “random” paths through the meta-network while favoring nodes with the fastest service, or does that threaten anonymity unacceptably? Do inconsistent views of the network topology threaten anonymity or stability or performance?

If Alice notices that Bob seems to be reachable by Internet standards but is not responding adequately to forwarding requests, should she contact him or route around him? Can she contact his network administrator? Is it her responsibility to report it at all? Can she or Bob hope to diagnose the fault without keeping some form of traffic log? Does every node in the network have to agree on these questions, and if so, how could an agreement possibly be enforced?

Policies can be set, guidelines issued, and questions such as these answered. Indeed, many of these questions must be addressed in locally anonymous networks too. Nevertheless, crossing administrative domains—the mantra of the dispersed architectures—is not a maintenance-free proposition. In a dispersed scheme, there have to be meetings. When something goes wrong at Bob’s site, it will affect Alice too. In a locally anonymous scheme, the administrator can set policy and just let it be.

3.2 Responsibility

One argument for network anonymity goes like this: “Anonymity is the default mode. When you walk into a store, the clerks do not usually know who you are, and they have no way of finding out. Therefore, we should demand the same from computer networks.” There is only some truth in this analysis. The problem becomes clear when you consider that masked people behave differently than unmasked people, even among complete strangers.

Abuse has been a difficult problem in anonymous systems. The `anon.penet.fi` remailer was ultimately deactivated because of abuse; after a copyright was vio-

lated by an anonymous post, the authorities took action against the remailer owner [New97]. This case is sometimes misunderstood because it has many strange facets: the notion that religious scripture can be copyrighted at all, the bewildering aggressiveness with which the copyright owner pursued its case, the seismic culture clash between lawyers and netizens, and the international scope and several jurisdictions involved. Still, in the final analysis, one party took offense at anonymous messages and attacked the anonymity infrastructure in response.

This is not an isolated reaction; other remailers have closed because of abuse. Raph Levien, who keeps statistics on remailer availability, estimated in 1994 that individual remailers last only about six months on average before being shut down by offended users and tired system administrators [NYT94, Lev]. Anonymizer, Inc.’s policy of keeping verbose logs is clearly intended to control abuse. At a destination site’s request, they will even program the Anonymizer to block client requests to anonymously contact that site [Ano97a]. This allows sites actively interested in tracking their clientele to easily opt-out of the anonymity mechanism altogether.

As spam, harassment, and unsolicited commercial email become more prevalent, the administrators of existing dispersed anonymity schemes spend more and more time defending themselves, their Internet Service Providers, and their users against charges of abuse stemming from their networks. We predict that unless the administrators find dramatic new ways of controlling abuse, they will find their traffic blocked from networks whose administrators do not share their zeal for or even tolerance of anonymous expression. The most straightforward option available to an administrator weary of a particular strain of abuse is to block the abuser’s anonymous service entirely—along with all of its non-abusive members. (Some have proposed charging for anonymous service, where the payers can pay with anonymous electronic cash. While this may control the flow of idle, unfinanced spam, we do not see it rescuing the services from abuse. Wealth and the propensity to abuse seem to run along orthogonal axes.)

Much has been written about the promise and peril of anonymous messaging [Lee96, Sie95, Mos95, SM96] assuming the use of a scheme in which even a sender’s neighborhood is difficult to identify. These dispersed anonymity schemes consider all traffic equally worthy of protection. As anonymity administrators concede that some people do use anonymity as an untraceable offensive weapon, they react by weakening the schemes to a level where users are explicitly threatened to behave responsibly, for instance, by keeping incriminating logs of system use. The anonymity providers then have no

choice but to become arbiters of acceptable and unacceptable demeanor—hardly the role they or their anonymous clients expected them to don.

In a locally anonymous network, the landscape is fundamentally different. Users join in a coalition because they essentially support each other even though they are uninterested in their respective day-to-day details. The fact that they share an address formalizes this disinterest, saying “I will defend my coalition members’ activities as my own.” To the outside world, they appear as one entity. If the unit becomes abusive, there is only one address to block and other anonymous groups are unaffected. Even to accuse the group of misbehaving requires addressing all of the members, which gives them an opportunity to revisit the terms of their cooperation and disband if necessary. To insiders, they appear as a tightly-knit group willing to share responsibility for their network use. The proper analogy is no longer a masked shopper but rather the impossibility of shopping alone.

We believe that local anonymity encourages responsible network use. It lends a certain amount of formal deniability to coalition members, not immunity, and users may hesitate before implicating their colleagues with truly heinous conduct. Those who do not hesitate will soon find themselves without a functional coalition.

3.3 Applications of Local Anonymity

- A company could use local anonymity to protect itself from eavesdropping by a subcontractor providing on-site network infrastructure.
- Individuals in a networked apartment building or dormitory could use local anonymity to hide individual interests and habits from their ISP and each other.
- Members of a political body such as the U.S. House of Representatives could use local anonymity to provide shelter from the media, who already use sophisticated technologies to scrutinize their public and private behavior [RIS].
- Similarly, a celebrity and her staff could use local anonymity to prevent tabloids and other rumor mongers from distinguishing the celebrity’s shocking interests from her staff’s shocking interests.
- Dolev and Ostrovsky [DO97] suggest a scenario where a network connects many possible military command sites, only one of which is ever truly active, and the local anonymity is used to prevent the enemy from learning the present location of the commanders.

- Laptop users can remain locally anonymous without requiring the cooperation of any authorities. (Picture a coffeehouse with ethernet drops.)
- Weak local anonymity can be deployed simply in order to hide rich local network structures. In the case where local traffic is not methodically hidden from local eavesdroppers, this can be achieved with firewalls employing proxy servers and network address translators, both of which are already widely deployed [CB94, CZ95].

Local anonymity can also be used in situations where a coalition provides a service, i.e., listens for inbound connections:

- Radically different Web services could be offered at a single site in order to directly protect its clients. As an example, the U.S. Congress could host a secure (encrypted) web site on a locally anonymous network along with the secure web site of a foreign political party supported by Congress but unpopular or even illegal abroad. Then remote observers would be unable to distinguish accesses to the Congress site from accesses to the illegal site by examining endpoint addresses alone. Furthermore, this disguises accesses *without requiring* the clients to install special software or hop through general-purpose anonymizing sites. In effect, local anonymity allows the powerful to lend their protection to the vulnerable when they deem it necessary.

The above applications all suggest Internet use (i.e., packets routed outside of the LAN and onto the Internet backbone). But local anonymity allows arbitrarily good intranet protection as well; encoding a packet with IP does not require it to leave the site.

4 A Framework for Locally Anonymous Systems

Local anonymity means that several nodes somehow share a single identity; the details of that arrangement can vary depending on the local network topology, trusted devices, and desired deniability. At one extreme, all internal equipment is trusted, and traffic is anonymous only in that an external observer is unable to tell which internal agent is the peer in a cross-gateway message exchange. At another extreme, local nodes suspect active and/or passive attacks from their own network infrastructure and implement a DC net to protect purely local traffic in addition to cross-network traffic.

Common elements across the spectrum include gateways bridging internal and external traffic, connection naming strategies, encapsulation, performance monitoring, diagnostics, and so on. In the following sections we treat some such issues independently from specific anonymity-preserving techniques. However we assume that coalition members are at least somewhat suspicious of each other or their networking fabric, otherwise a simple firewalled proxying scheme could be used.

4.1 Definitions

We call a locally anonymous network a *lanon* (think LANon). A lanon will usually consist of a possibly proper subset of nodes in a LAN. However, since we construct anonymous IP from ordinary IP, this constraint is enforced not by policy but by the level of performance that can be tolerated. *Internal* entities have access to at least some of the lanon's packets, while *external* entities have access only to packets that emerge from a lanon gateway. Correspondingly, *outbound* packets originate in the lanon and cross a gateway into the Internet and *inbound* packets take a reverse path. Let the source IP address of an outbound lanon packet be called its *anonymous IP address*.

A connection to an external site initiated by a lanon client is an *outbound connection*. For example, a Web browser in a lanon would create many outbound TCP connections, each of would carry both outbound and inbound packets. *Inbound connections* are defined conversely. *Purely local connections* have source and destination within the lanon. TCP is the most important and common kind of IP connection, but UDP and other IP "connections" are possible even though those protocols are officially connectionless. We will concentrate on the TCP case.

Finally, the underlying *anonymity policy* defines the entities (nodes, links, administrators) involved, their capabilities, and their trustworthiness. For each policy, the designer chooses an *anonymity protocol* that preserves the required anonymity. Each protocol is realized within the framework set out below.

4.2 Uniformity

Clearly, a 1460-byte IP packet is unlikely to come from the client side of a telnet connection. To impede monitoring attacks, the packets exchanged in a lanon should be fixed-size (after encryption).

Each anonymity protocol has to deal with gateway contention, i.e., multiple nodes attempting to transmit across the gateway simultaneously. If a contending packet

must be dropped, the originating node should be (anonymously) informed. Without that knowledge, the originating node could not tell if the packet had been discarded inside the lanon, where an immediate retransmission is appropriate, or remotely, where standard IP techniques apply (such as TCP timeouts). The simplest mechanism is for the gateway to *reflect* transmitted packets back onto the lanon so that each node knows when its packet emerged. (If the anonymity policy requires protection against active attacks, then even this becomes involved; see [Wai90].)

4.3 Anonymous IP Address Sets

Here we consider methods to determine the set of anonymous addresses that can be used in a lanon. The set is a relatively stable (see §4.5) global parameter shared by lanon members. When constructing actual packets, they choose addresses from this set using techniques from §4.4.

Let us consider options for assigning IP addresses to anonymous endpoints.

1. One or many IP addresses could be allocated from the ordinary network's pool of unused addresses.
2. One stable IP address of one of the lanon nodes could be reclaimed as the common anonymous address.
3. Many or all of the lanon nodes' IP addresses could be reclaimed as anonymous addresses; multiple new addresses could be allocated as well.

Option 1 has the superficially attractive property that anonymous traffic can be distinguished from nonanonymous traffic on the lanon-carrying LAN. However, we caution against providing standard nonanonymous service in all but the friendliest firewalled lanons. For instance, an anonymized node should obviously reject connections to the TCP finger port (used to display current user statistics). Likewise, the anonymous node should yield no additional information when accessed by its nonanonymous address.

Option 2 naturally suggests giving one of the lanon nodes gateway responsibility in addition to its usual functions. The member nodes' capabilities, network topology, and link protocol could strongly argue for this arrangement in some cases. For instance, the center of a star network is the obvious choice for a gateway to the Internet. On a broadcast or switched network, it would make sense to designate the fastest node as a gateway.

Option 3 may seem superfluous: if the point is to make the lanon seem like a single node, why use multiple addresses? One justification concerns return addressing.

When two processes on a single ordinary node simultaneously construct TCP connections to the same destination (e.g., `http://www.netscape.com`), the node chooses two unique port numbers to identify the distinct connections. These port numbers appear in every packet associated with the connection.

In a lanon with a single anonymous IP address, the same thing must happen when two lanon nodes open TCP connections to the same destination, even though we are now talking about *two distinct* computers. Since the 16-bit port space (10 bit for certain applications) must be shared among all lanon nodes, port numbers qualify as a scarce resource in a lanon. But the port space available for disambiguating connections increases linearly with the number of available anonymous IP addresses, so assigning multiple addresses provides some relief. (For purely local connections, the port space increases quadratically.)

The LAN technology in place may require the use of gateways anyhow. For example, in an Ethernet setting, the router connecting the lanon to the larger network will associate each anonymous IP address with a *single* Ethernet address. It may be difficult or impossible to convince the router to use an Ethernet multicast or broadcast address for this purpose, so inbound packets will usually be addressed to a particular lanon gateway node.

4.4 Indirect Connection Addressing

Indirect addressing is straightforward but expensive. Let A_{IP} be the set of anonymous IP addresses in the lanon, $PORT = \{0, 1, \dots, 2^{16} - 1\}$ be the set of possible port numbers, and $A_{TCP} = A_{IP} \times PORT$ be the set of all possible TCP endpoint connection identifiers. Each such identifier is called an *anonymous TCP address*. A lanon client building an outbound TCP connection should select its source address/port pair from A_{TCP} at random subject to lanon uniqueness and application-specific constraints.

Randomly choosing the source label hides the node's identity from external (and internal) observers. Later, when inbound packets on the same connection arrive at the gateway, the gateway must forward it to the appropriate physical node. However, the anonymity policy may not allow the gateway to possess the mapping between anonymous addresses and node identities. In this case the gateway must simply broadcast the packet; the intended receiver will then process it and every other node will ignore it. This technique preserves receiver anonymity within the lanon.

Not only must the anonymous TCP addresses of outbound connections be chosen in this manner, but clients set up to listen for inbound connections must follow the procedure as well. Otherwise, applications such as FTP

that rely on inbound connections will either fail or trivially betray the client's identity. This implies that simply binding a name to a socket in the Internet domain requires lanon uniqueness negotiation, as the application may then begin to advertise the name returned. Unfortunately, client applications usually do not expect `bind()` calls to take long, and so this could result in unusual application behavior.

4.5 Lanon Membership

A lanon's reaction to an observed or requested lanon membership change depends intimately on the anonymity policy in effect. In general, adding new nodes to the coalition is easy and relatively risk-free. However, removing nodes can directly threaten the anonymity of the group: the fewer members in the group, the easier it is to identify a node. Once a lanon becomes too small, it must wholly disband. In this circumstance, it is appropriate to break all known persistent (TCP) network connections, since there is no hope of recovering them.

On the other hand, if a node crashes or otherwise resigns from the lanon, it should not break its network connections, nor should any other node do so on its behalf. A rapid sequence of TCP resets emitting from the lanon would suggest that each named connection originated on the same node, thus betraying the relationship between previously unlinked streams. Suppressing closes and resets is unfortunately not very network-friendly; hopefully, sudden resignations will be rare. Alternatively, the lanon could agree to periodically (perhaps once an hour) reset only some of the connections for which inbound packets have unexpectedly generated no response.

A nonresponsive or withdrawn gateway node is even more difficult. In addition to the anonymity threats described above, all of the connections using the gateway address are in jeopardy unless the address can be resurrected in the lanon. In an Ethernet, another gateway could attempt to use promiscuous ARP [CMQ87] to assume its identity and carry its traffic. Whether this would work or not depends on the LAN's border router.

The anonymity policy may require that each lanon node transmit locally before a gateway emits an outbound packet so that no conclusions can be drawn about the packet's origin by observing internal sequencing. What if a lanon node instead transmits nothing in one round? Perhaps it is a transient fault. If none of the other lanon clients requested a packet emission either, then the gateway could temporarily forgive the lapse. Similarly, if the gateway has recently been continuously emitting packets, it could emit one in this round in spite of the nonreporting node on the grounds that the packet could have been

queued. (Such laxity is possible only in some anonymity protocols. In a DC network, none of the lanon nodes’ transmissions are even decryptable until all of them arrive.) After too many rounds of nonresponse, the fault must be treated as a resignation.

5 Anonymity Protocols

We now briefly sketch two lanon anonymity protocols: trusted gateways and DC networks. In each of these, inbound packets are broadcast in the lanon when they arrive at the appropriate gateway. Outbound packets require considerably more work.

5.1 Trusted Gateways

If one node is trusted by all of the other participating nodes, then it can be designated a lanon gateway. To produce one outbound packet, the other nodes contact the gateway in round-robin fashion using authenticated secret-key cryptography until every node has submitted a packet. The gateway then chooses a single packet among them and emits it. Inbound packets are broadcast to the lanon after being authenticated by the gateway.

This arrangement hides source and destination addresses from all non-gateway participants, including the network infrastructure. In addition, multiple trusted gateways can cooperate to provide better scalability. However, the resulting lanon is nowhere near as secure or compelling as one based on DC networks.

5.2 DC networks

It is possible to distribute the gateways’ knowledge among all of the participant nodes, protected by a kind of secret-sharing scheme:

Consider an undirected connected graph (V, E) with n lanon member nodes at the vertices and each edge $\{i, j\} \in E$ representing a shared but secret random packet stream $r(\{i, j\}, t)$ between the two nodes. This *key-sharing graph* is fixed by an anonymity administrator according to the desired attack resistance and performance requirements. In communication round t , each lanon node i chooses the packet $p(i, t)$ it wants to emit through a gateway and broadcasts the sum

$$b(i, t) := p(i, t) + \sum_{j:\{i,j\} \in E} \text{sign}(i - j) r(\{i, j\}, t)$$

onto the lanon. (Note that “sharing” a random key stream might mean that the related nodes negotiate a new key

on the network at each round.) Each node computes the grand total

$$g(t) := \sum_{j \in V} b(j, t)$$

of all of the broadcast sums. Since the graph is undirected, each random packet is added and subtracted once, and therefore

$$g(t) = \sum_{j \in V} p(j, t).$$

If exactly one of the nodes i wanted to send a nonzero packet $p(i, t)$, then $g(t) = p(i, t)$. (Otherwise, a collision resolution procedure must follow, see [Cha88, Mar88, BdB90, Pfi90, Wai90] for options.) Each gateway in the system checks to see if the packet $g(t)$ bears its address in its source field, and if so, that gateway assumes responsibility and transmits the packet out of the lanon.

If the packet streams $r(\{i, j\}, t)$ are truly random, then senders are perfectly untraceable in the following sense. For each observed partial sum and each suspicion regarding the identity of the sender, there exist equally many random packet streams $\hat{r}(\{i, j\}, t)$ that are consistent with the suspicion. In other words, all suspicions are equally plausible and therefore equally worthless. As long as the packet streams $r(\{i, j\}, t)$ are secret and random, no information about the true senders is leaked.

This protocol and the original untraceability proof is due to Chaum [Cha88]. It has been extended in [Pfi90, Wai90, WP90], where it is known as *superposed sending*. The above formulation is from [Wai90, WP90].

5.3 Minimizing Latency

Trusted gateways and DC nets require every node to transmit at least once before a single packet emerges from the lanon. This means that an n -node lanon based on a physical broadcast LAN (such as ethernet) can transmit across its gateways at only $O(1/n)$ of the LAN’s natural capacity.

On the other hand, a lanon built on a switched LAN is well-suited to a simple parallel divide-and-conquer communication reduction. (Switched networks allow any partition of their nodes into disjoint pairs to communicate simultaneously at the LAN’s full natural speed. Examples of such networks include ATM and switched ethernet.)

Figure 1 shows a reduction schedule on 16 nodes in 4 stages. In stage 1, the top row of *children* nodes transmit to their *parents* in the second row, while the third row of children simultaneously transmits to the parents in the fourth row. (To avoid clutter in the diagram, only the top-left-most transmission is labeled with the stage number.) In the second stage, the second row transmits to the fourth

row. At the end of the fourth stage, all nodes have had a chance to influence the lower-right gateway node. In general, an n -node lanon based on a switched LAN can transmit across its gateways at $O(1/\log n)$ of the LAN's natural capacity.

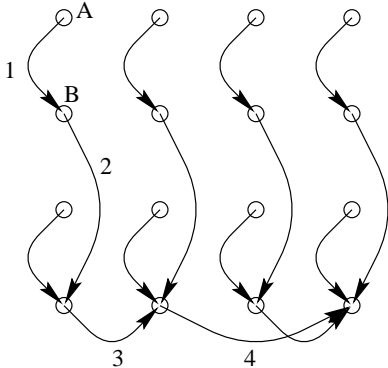


Figure 1: A reduction on 16 nodes. Each node contributes to the system output, which is produced after only $\log 16 = 4$ rounds.

5.3.1 Trusted Gateways

One must be careful when attempting to minimize latency on trusted gateways; a naïve implementation of this reduction could well result in *worse* latency than the n -node round-robin schedule. Since each node encrypts using a secret key shared only with the gateway, no interior node in the schedule can tell whether it is being asked to forward random data—indicating that the child had no packet to send—or a real packet. There are few choices. If interior nodes forward both packets to their parents, then the last transmission to the gateway alone would take almost as long as the entire round-robin schedule would have taken (discounting overhead). If nodes choose between the two packets at random (with appropriate biases so as not to starve the children at greater depth), then each node would have to wait n rounds on average to address the gateway in addition to the logarithmic latency imposed by the reduction schedule. Worse yet, this expected wait is unconditional, i.e., a node would endure the wait even if it were the only node in the system wishing to transmit.

Consider this refinement: suppose that interior nodes always forward their child's packet unless they also have their own packet to send, in which case they choose between them by flipping an appropriately biased coin. This assures each node equal access to the gateway, and furthermore, the gateway is certain to emit a packet if any lanon node wishes to send. But it also changes the

anonymity properties of the system. By measuring the proportion of their packets that emerge from the gateway, children can detect whether parents are transmitting and so zero in on suspected communication endpoints. While tradeoffs such as these are possible, their consequences must be kept in mind.

5.3.2 DC Networks

In a DC net, each node's output is the sum of its inputs and the shared secret keys it holds with its neighbors. By adopting Figure 1 as the communication schedule of a DC net, the gateway utilization improves to $O(1/\log n)$ —the packets do not grow, and no choice between packets is necessary. However, as a key-sharing graph, it provides poor sender anonymity: node B always knows whether A has a packet to send or not, since B knows all of the random key streams that A does (viz., the one associated with the edge labeled 1 in Figure 1.) A good key-sharing graph should have no such isolated points of failure.

Formally, an undirected graph G is k -connected if, after removing any $k - 1$ of its edges, it still contains a path between each pair of its vertices. Thus a 1-connected graph is connected in the usual sense of the word, and a 2-connected graph remains connected even after the elimination of any single edge. A good key-sharing graph must be at least 2-connected, otherwise (as in the example above) it will contain nodes that always know exactly what their neighbors transmit. In general, at least k lanon nodes must collude in order to compromise a single sender's anonymity in a k -connected key-sharing graph.

For anonymity purposes, the key-sharing graph and the communication schedule do not interact. But the keys of the key-sharing graph do need to be generated and exchanged, and the frequency of this exchange affects the system's susceptibility to cryptanalytic attacks. In other words, the successful cryptanalysis of keys is as good as intentional lanon node collusion to a passive attacker.

Therefore, to increase a DC net's resistance to attack, one should increase its key-graph connectivity and the frequency of its key generation and exchange. But the connectivity of the key-sharing graph affects the communication complexity of the key exchange. In a frequent exchange situation, then, it is important to choose a key-sharing graph having a communication schedule that also minimizes latency (perhaps by maximizing parallelism) on the network topology at hand.

6 Conclusion

Local anonymity complements the privacy offered by existing dispersed anonymity schemes. When endpoint neighborhood identification is not confidential, lanons can take advantage of high-performance local networks to provide strong privacy guarantees without incurring any additional dependence on foreign agents' policies, performance, or reliability. Since tracing a packet to a lanon (but no further) is trivial, lanon users will tend to behave responsibly, thus easing the burden on faraway anonymity administrators and the network community at large.

In this paper we have sketched a framework for local anonymity that allows nodes' local anonymity to be preserved even when their packets leave the lanon for service in the wider Internet. The recommended techniques include IP address reuse, gatewaying, indirect addressing with distributed uniqueness, and trusted gateways or DC networks. Ongoing research focuses more directly on encapsulation issues and the underlying anonymity-preserving protocols.

References

- [Ano97a] Anonymizer FAQ, 1997. See <http://www.anonymizer.com/faq.html>.
- [Ano97b] Terms and conditions of the Anonymizer service, 1997. See <http://www.anonymizer.com/terms.html>.
- [Atk95] R. Atkinson. Security architecture for the internet protocol. RFC 1825, 1995.
- [BdB90] J. Bos and B. den Boer. Detection of disrupters in the DC protocol. In *Eurocrypt '89*, volume Lecture Notes in Computer Science of 434. Springer-Verlag, 1990.
- [BLC95] T. Berners-Lee and D. Connolly. Hypertext markup language — 2.0. RFC 1866, 1995.
- [BLFN96] T. Berners-Lee, R. Fielding, and H. Nielsen. Hypertext transfer protocol — HTTP/1.0. RFC 1945, 1996.
- [Böt90] Manfred Böttger. Realisierung des DC-Netz-Versuchs und einer einheitlichen Praktikums-Netzchnittstelle (Realization of a DC-Network and Uniform Network Interface Including Lab Experiments). Diplomarbeit, Universität Karlsruhe, Institut für Rechnerentwurf und Fehlertoleranz, 1990. In German.
- [CB94] W. Cheswick and S. Bellovin. *Firewalls and Internet Security*. Addison-Wesley, 1994.
- [Cha81] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2), February 1981.
- [Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.
- [CMQ87] S. Carl-Mitchell and J. Quarterman. Using ARP to implement transparent subnet gateways. RFC 1027, 1987.
- [Cot] Lance Cottrell. Mixmaster and remailer attacks. See <http://obscura.com/~loki/remailer-essay.html>.
- [Cot96] Lance Cottrell. Mixmaster FAQ. See <http://www.obscura.com/~loki/remailer/mixmaster-faq.html>, 1996.
- [Cro] Crowds home page. See <http://www.research.att.com/projects/crowds>.
- [CZ95] D. B. Chapman and E. Zwicky. *Building Internet Firewalls*. O'Reilly & Associates Inc., 1995.
- [DO97] Shlomi Dolev and Rafail Ostrovsky. Efficient anonymous multicast and reception. In *Advances in Cryptology: Proceedings of CRYPTO '97*, 1997. to appear.
- [FGM⁺97] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext transfer protocol — HTTP/1.1. RFC 2068, 1997.
- [FKK96a] Andreas Fasbender, Dogan Kesdogan, and Olaf Kubitz. Analysis of security and privacy in mobile IP. In *4th International Conference on Telecommunication Systems Modeling & Analysis*, Nashville, USA, March 1996.
- [FKK96b] Andreas Fasbender, Dogan Kesdogan, and Olaf Kubitz. Variable and scalable security: Protection of location information in mobile IP. In *Mobile IP, 46th IEEE Vehicular Technology Society Conference*, Atlanta, March 1996.

- [GRS96] David Goldschlag, Michael Reed, and Paul Syverson. Hiding routing information. Workshop on Information Hiding, May 1996. Cambridge, UK.
- [GT96] Ceki Gülcü and Gene Tsudik. Mixing email with Babel. In *Proceedings of the 1996 Internet Society Symposium on Network and Distributed System Security*, February 1996.
- [Hel95] Johan Helsingus, February 20 1995. Press Release.
- [Lee96] Gia Lee. Addressing anonymous messages in cyberspace. *Journal of Computer Mediated Communication*, 2(1), 1996.
- [Lev] Raph Levien. Remailer list. See <http://www.cs.berkeley.edu/~raph/remailer-list.html>.
- [Mar88] Eckhard Marchel. Leistungsbewertung von überlagendem Empfangen bei Mehrfachzugriffsverfahren mittels Kollisionsauflösung (Performance Evaluation of Superposed Receiving with Multiple-Access Channel Collision Resolution). Diplomarbeit, Universität Karlsruhe, Institut für Rechnerentwurf und Fehlertoleranz, 1988. In German.
- [Mos95] Walter S. Mossberg. Accountability is key to democracy in the on-line world. *The Wall Street Journal*, January 26 1995.
- [New97] Ron Newman. The Church of Scientology vs. the Net, 1997. See <http://www2.thecia.net/users/rnewman/scientology/home.html>.
- [Nym97] Instructions for nym.alias.net, 1997. See <http://www.cs.berkeley.edu/~raph/n.a.n.html>.
- [NYT94] Computer jokes and threats ignite debate on anonymity. *The New York Times*, December 31 1994. pp. 1,53.
- [Ort91] Andreas Ort. Implementierung eines MIX-Netzes sowie Konzeption und Realisierung des MIX-Netz-Versuchs (Implementation of a MIX-Network as well as Design and Realization of MIX-Network Lab Experiments). Diplomarbeit, Universität Karlsruhe, Institut für Rechnerentwurf und Fehlertoleranz, 1991. In German.
- [Pfi90] Andreas Pfitzmann. *Diensteintegrierende Kommunikationsnetze mit teilnehmerüberprüfbarem Datenschutz (ISDN Networks with Member-Testable Data Privacy)*, volume 234 of *Informatik-Fachberichte*. Springer-Verlag, 1990. Also Ph.D. Thesis.
- [Pos81a] J. Postel. Internet control message protocol. RFC 792, 1981.
- [Pos81b] J. Postel. Internet protocol. RFC 791, 1981.
- [Pos81c] J. Postel. Transmission control protocol. RFC 793, 1981.
- [PP90] Birgit Pfitzmann and Andreas Pfitzmann. How to break the direct RSA-implementation of MIXes. In *Eurocrypt '89*, volume 434 of *Lecture Notes in Computer Science*, pages 373–381. Springer-Verlag, 1990. See also [SIR].
- [PPW89] Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. Garantierter Datenschutz für zwei 64-kbit/s-Duplexkanäle über den (2*64+16)-kbit/s-Teilnehmeranschluß durch Telefon-MIXe. In *Tagungsband 3 der 4. SAVE-Tagung*, pages 1417–1447, Köln, April 1989. North-Holland. See also [SIR].
- [PPW91] Andreas Pfitzmann, Birgit Pfitzmann, and Michael Waidner. ISDN-MIXes—untraceable communication with very small bandwidth overhead. In D. T. Lindsay and W. L. Price, editors, *Information Security, Proc. IFIP/Sec '91*, pages 245–258, Amsterdam, May 1991. North-Holland. See also [SIR].
- [Pri] Privacy and anonymity on the internet. See <http://www.itd.nrl.navy.mil/ITD/5540/projects/onion-routing>.
- [RIS] RISKS Digest 19.39. See <http://www.CSL.sri.com/risksinfo.html>.
- [RR97] Michael Reiter and Aviel Rubin. Crowds: Anonymity for web transactions. Technical report, DIMACS, 1997. See also [Cro].
- [SGR97] Paul Syverson, David Goldschlag, and Michael Reed. Anonymous connections and onion routing. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, May 1997.

- [Sie95] Martha S. Siegel. Anarchy, chaos on the internet must end. The San Francisco Chronicle, January 2 1995. Editorial.
- [SIR] Sirene publications links. See <http://www.informatik.uni-hildesheim.de/FB4/Projekte/sirene/lit/sirene.lit.html>.
- [SM96] Paul A. Strassmann and William Marlow. Risk-free access into the global information infrastructure via anonymous re-mailers. In *Symposium on the Global Information Infrastructure*, Cambridge, MA, January 28–30 1996. See <http://www.strassmann.com/pubs/anon-remail.html>.
- [Wai90] Michael Waidner. Unconditional sender and recipient untraceability in spite of active attacks. In *Eurocrypt '89*, volume Lecture Notes in Computer Science of 434, pages 302–319. Springer-Verlag, 1990. See also [SIR].
- [WP90] Michael Waidner and Birgit Pfitzmann. The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability. In *Eurocrypt '89*, volume Lecture Notes in Computer Science of 434. Springer-Verlag, 1990. See also [SIR].