Related Works:
Anonymous Communications Systems

David Molnar <dmolnar (at) fas.harvard.edu>
Michael J. Freedman <mfreed (at) mit.edu>

We earlier described several major implementations of anonymous communications channels. This appendix serves to give a more detailed survey of research and development in the area of anonymous communications. Some of these projects are not implemented; some exist more as a proof-of-concept by their respective designers; and still others repeat design and functionality provided by like systems.

We review three main types of design: proxy-servers, mix-nets, and other anonymous communications channels.

Proxy Services

Proxy services provide one of the most basic forms of anonymity, inserting a third party between the sender and recipient of a given message. Proxy services are characterized as having only one centralized layer of separation between message sender and recipient. The proxy serves as a ``trusted third party,'' responsible for sufficiently stripping headers and other distinguishing information from sender requests.

Proxies only provide unlinkability between sender and receiver, given that the proxy itself remains uncompromised. This unlinkability does not have the quality of perfect forward anonymity, as proxy users often connect from the same IP address. Therefore, any future information used to gain linkability between sender and receiver (i.e., intersection attacks, traffic analysis) can be used against previously recorded communications.

Sender and receiver anonymity is lost to an adversary that may monitor incoming traffic to the proxy. While the actual contents of the message might still be computationally secure via encryption, the adversary can correlate the message to a sender/receiver agent.

This loss of sender/receiver anonymity plagues all systems which include external clients which interact through a separate communications channel - that is, we can define some distinct edge of the channel. If an adversary can monitor this edge link or the first-hop node within the channel, this observer gains agent-message correlation. Obviously, the ability to monitor this link or node depends on the adversary's resources and the number of links and nodes which exist. In a proxy system, this number is small. In a globally-distributed mixnet, this number could be very large. The adversary's ability also depends on her focus: whether she is observing messages and agents at random, or if she is monitored specific senders/receivers on purpose.

Anonymizer.com

The Anonymizer was one of the first examples of a *form-based web proxy* [#!anonymizer!#]. Users point their browsers at the Anonymizer page at `www.anonymizer.com`. Once there, they enter their destination URL into a form displayed on that page. The Anonymizer then acts as an `http` proxy for these users, stripping off all identifying information from `http` requests and forwarding them on to the destination URL.

The functionality is limited. Only `http` requests are proxied, and the Anonymizer does not handle cgi scripts. In addition, unless the user chains several proxies together, he or she may be vulnerable to an adversary which tries to correlate incoming and outgoing `http` requests. Only the data stream is anonymized, not the connection itself. Therefore, the proxy does not prevent traffic analysis attacks like tracking data as it moves through the network.

Lucent's Proxymate

Chaining multiple proxies together by hand is a tedious business, requiring many preliminaries before the first web page is reached. Lucent's Proxymate software automates the process [#!lucent!#]. The software looks like a proxy sitting on the user's computer. By setting software to use the Proxymate proxy, the user causes the software's requests and traffic to go to the software, which then automatically negotiates a chain of proxies for each connection.

Proxomitron

Another piece of software which helps manage many distinct proxies in a transparent manner is Proxomitron[#!proxomitron!#]. In addition to basic listing and chaining of proxies, Proxomitron allows users to write filter scripts. These filters can then be applied to incoming and outgoing traffic to do everything from detecting a request for the user's e-mail address by a web site to automatically changing colors on incoming web pages.

Chaumian Mix-nets

The project of anonymity on the Internet was kicked off by David Chaum in 1981 with a paper in Communications of the ACM describing a system called a ``Mix-net.'' This system uses a very simple technique to provide anonymity: a sender and receiver are linked by a chain of servers called Mixes. Each Mix in the chain strips off the identifying marks on incoming messages and then sends the message to the next Mix, based on routing instructions which encrypted with its public key. Comparatively simple to understand and implement, this Mix-net (or ``mix-net'' or ``mixnet'') design is used in almost all of today's practical anonymous channels.

Chaum's Digital Mix

Chaum's original paper introduced the basic concept of a Mix as a sort of ``permutation box.'' On the incoming side is a list of messages representing the messages which have arrived at the Mix server, each of which is identified with a particular sender. On the outgoing side is a randomly permuted list of messages, which have lost their identification with the sender. The assumption is that if the Mix works correctly, no adversary can do better than guessing to link an incoming message with an outgoing message.

ISDN Mixes

Chaum's original Digital Mix was described in terms of a series of Mix nodes which passed idealized messages over a network. The first proposal for the practical application of mixes came from Pfitzmann et. al. [#!ISDN-mix!#], who showed how a mix-net could be used with ISDN lines to anonymize a telephone user's real location. Their motivation was to protect the privacy of the user in the face of a telephone network owned by a state telephone monopoly.

Their paper introduced a distinction between *explicit* and *implicit* addresses. An explicit address is something about a message which clearly and unambiguously links it to a recipient and can be read by everyone, such as a To: header. An implicit address is an attribute of a message which links it to a recipient and can only be determined by that recipient. For example, being encrypted with the recipient's public key in a recipient-hiding public key is an implicit address.

Remailers: SMTP Mix-nets

Until the rise of proxy-based and TCP/IP-based systems, the most popular form of anonymous communication was the *anonymous remailer*: a form of mix which works for e-mail sent over SMTP. Remailers are informally divided into three categories, called Type 0, Type 1, and Type 2.

Type 0: anon.penet.fi

One of the first and most popular remailers was `anon.penet.fi`, run by Johan Helsingius. This remailer was very simple to use. A user simply added an extra header to e-mail indicating the final destination, which could be either an e-mail address or a Usenet newsgroup. This e-mail was sent to the `anon.penet.fi` server, which stripped off the return address and forwarded it along. In addition, the server provided for return addresses of the form ``anXXXX@anon.penet.fi''; mail sent to such an address would automatically be forwarded to another e-mail address. These pseudonyms could be set up with a single e-mail to the remailer; the machine simply sent back a reply with the user's new pseudonym.

The `anon.penet.fi` remailer is referred to as a Type 0 remailer for two reasons. First, it was the original ``anonymous remailer.'' More people used `anon.penet.fi` than are known to have used any following type of remailer. Exact statistics are hard to come by, but X number of accounts were registered at `penet.fi`, and only Y are currently registered at `nym.alias.net`.

Second, `anon.penet.fi` did not provide some of the features which motivated the development of ``Type I'' and ``Type II'' remailers. In particular, it provided a single point of failure and the remailer administrator had access to each user's ``real'' e-mail address. In general, any remailer system which consists of a single hop is considered Type 0.

This last feature proved to be the service's undoing. The Church of Scientology, a group founded by the science fiction writer L. Ron Hubbard, sued a `penet.fi` pseudonym for distributing materials reserved for high initiates to a Usenet newsgroup. Scientology claimed that the material was copyrighted ``technology.'' The poster claimed it was a fraud used to extort money from gullible and desperate fools. Scientology won a court judgment requiring the `anon.penet.fi` remailer to give up the true name of the pseudonymous poster, which the operator eventually did. This incident, plus several allegations of traffic in child pornography, eventually convinced Johan Helsingius to close the service in 1995[#!helsingius!#].

Services similar to Type 0 remailers now exist in the form of ``free e-mail'' services such as Hotmail, Hushmail, and ZipLip, which allow anyone to set up an account via a web form. Hushmail and ZipLip even keep e-mail in encrypted form on their server. Unfortunately, these services are not sufficient by themselves, as an eavesdropping adversary can determine which account corresponds to a user simply by watching him or her login.

Type 1: Cypherpunks Remailers

The drawbacks of anon.penet.fi spurred the development of ``cypherpunks'' or ``Type 1'' remailers, so named because their design took place on the cypherpunks mailing list. This generation of remailers addressed the the two major problems with `anon.penet.fi`: first, the single point of failure, and second, the vast amount of information about users of the service collected at that point of failure. Several remailers exist; a current list can be found at the Electronic Frontiers Georgia site [#!mixmaster!#] or on the newsgroup alt.privacy.anon-server.

Each cypherpunk remailer has a public key and uses PGP for encryption. Mail can be sent to each remailer encrypted with its key, preventing an eavesdropper from seeing it in transit. A message sent to a remailer can consist of a request to remail to another remailer and a message encrypted with the second remailer's public key. In this way a chain of remailers can be built, such that the first remailer in the chain knows the sender, the last remailer knows the recipient, and the middle remailers know neither.

Cypherpunk remailers also allow for *reply blocks*. These consist of a series of routing instructions for a chain of remailers which define a route through the remailer net to an address. Reply blocks allow users to create and maintain pseudonyms which receive e-mail. By prepending the reply block to a message and sending the two together to the first remailer in the chain, a message can be sent to a party without knowing his or her real e-mail address.

Type 2: Cottrell's Mixmaster

While Cypherpunk remailers represented a major advance over anon.penet.fi, they fell short of the anonymity provided by the ideal mix. In 1995, Lance Cottrell outlined some of the problems with ``Type I'' remailers [#!mixmaster!#]:

- **Traffic Analysis:** Cypherpunk remailers tend to send messages as soon as they arrive, or after some specified amount of delay. The first option makes it easy for an adversary to correlate messages across the mix-net. It's not clear how much delay helps protect against this attack.
- **Does Not Hide Length:** The length of messages is not hidden by the encryption used by cypherpunk remailers. This allows an adversary to track a message as it passes through the mixnet by looking for messages of approximately the same length. [*Note that the definitions of semantic security and non-malleability do not seem to imply ``length-hiding'' either.*]

Cottrell wrote the Mixmaster, or ``Type II'', remailer to address these problems. Instead of using PGP, Mixmaster uses its own client software (which is also the server software), which understands a special Mixmaster packet format. All Mixmaster packets are the same length. Every message is encrypted with a separate 3DES key for each mix node in a chain between the sender and receiver; these 3DES keys are in turn encrypted with the RSA public keys of each mix node.

When a message reaches a mix node, it decrypts the header, decrypts the body of the message, and then places the message in a ``message pool.'' Once enough messages have been placed in the pool, the node picks a random message to forward.

As of this writing, Mixmaster is in version 2.9b22[#!mixmaster-code!#]. Discussion of the project can be found on the mix-l mailing list[#!mix-l!#]. A Mixmaster version 3 is planned in which nodes will communicate with each other via TCP/IP connections. All traffic will be encrypted with a key derived by a Diffie-Hellman key exchange and then destroyed immediately after the transaction is ended, thereby providing perfect forward secrecy. Unfortunately, the prototype specification for this protocol is only available in German and is not finished.

Nymservers and nym.alias.net

The reply blocks used by cypherpunks remailers are important for providing for return traffic, but they must be sent to every correspondent individually. In addition, using a reply block requires that a correspondent be familiar with the use of specialized software. This problem is addressed by *nymservers*, which act as holding and processing centers for reply blocks.

To use a nymserver, a user simply registers an e-mail address of the form ``nym@nymserver.net'' and associates a reply block with it. This association can be carried out via anonymous e-mail. Then whenever a message is sent to ``nym@nymserver.net,'' the nymserver automatically prepends the associated reply block, encrypts the aggregate, and sends it off to the appropriate anonymous remailer.

The most popular nymserver may be the one run at nym.alias.net, which is hosted at MIT's Lab for Computer Science. A recent report by Mazieres and Kaashoek details the technical and social details of running the nymserver, including problems of abuse[#!nymserver!#].

Remailer User Interfaces

The major reason for the massive popularity of `anon.penet.fi` was that it was extremely easy to use. Anyone who could type ``Request-Remailing-To:'' at the top of an e-mail message could send anonymous e-mail. With the advent of remailers which required the use of PGP or the Mixmaster software, the difficulty of using remailers increased. This difficulty was aggravated by the fact that for years, both PGP and Mixmaster were only available as command-line applications with a bewildering array of options.

Recent Mix-Net Designs

TAZ / Rewebber

TAZ / Rewebber

Goldberg and Wagner applied Mixes to the task of designing an anonymous publishing network called Rewebber[#!taz-rewebber!#]. Rewebber uses URLs which contain the name of a Rewebber server and a packet of encrypted information. When typed into a web browser, the URL sends the browser to the Rewebber server, whch decrypts the associated packet to find the address of either another Rewebber server or a legitimate web site. In this way, web sites can publish content without revealing their location.

Mapping between intelligible names and Rewebber URLs is performed by a name server called the Temporary Autonomous Zone(TAZ), named after a novel by Hakim Bey. The point of the ``Temporary'' in the name of the nameserver (and the novel) is that static structures are vulnerable to attack. Continually refreshing the Rewebber URL makes it harder for an adversary to gain information about the server to which it refers.

Babel

Contemporary with Cotrell's Mixmaster is an effort by Gulcu and Tsudik called ``Babel''[#!babel!#]. Babel uses a modified version of PGP as its underlying encryption engine. This modified version does not include normal headers, which would include the identity of the receiver, the PGP version number, and other identifying information.

The Babel paper defines quantities called the ``guess factor'' and the ``mix factor'' which model the ability of an adversary to match messages passing through the mix with their original senders. Then several attacks are presented, including the trickle and flooding attack, along with some countermeasures. The paper is noteworthy in that it attempts to give an analysis of just how much the practice of batching messages helps the untraceability of a mix-net node.

Stop and Go Mixes

The next step in probabilistic analysis for mixnets comes in the work of Kesdogan, Egner, and Buschkes [#!sg-mix!#], who proposed the ``Stop and Go Mix.'' They divide networks into two kinds: ``closed'' networks, in which the number of users is small, known in advance, and all users can be made distinct, and ``open'' networks like the Internet with extremely large numbers of users. They claim that perfect anonymity cannot be achieved in these open networks, because there is no guarantee that every single client of the mix node is not the same person coming under different names.

Instead, they define and consider a notion of *probabilistic anonymity*: given that the adversary controls some percentage of the clients, some other set of mix servers, and is watching a Mix, can the probability of correlating messages be quantified in terms of some security parameter? They consider queueing theory as an inspiration for a statistical model and manage to prove theorems about the adversary's knowledge in this model.

Variable Implicit Addresses

Later, Kesdogan et. al. applied Mixes to the GSM mobile telephone setting[#!kesdogan-vil!#]. Here, the point is to allow for GSM roaming from cell to cell while still protecting the user's real location from discovery by the phone company or an outside intruder. This is done by the use of *variable implicit addresses*, which work as follows : each roaming area has a publically known and static explicit address. When the client GSM phone comes online or crosses the boundaries of a cell, it queries the surrounding cells and downloads these addresses. Then it creates a new address for itself which combines the addresses of its surrounding cells.

Then, instead of sending the entirety of the new address, the phone sends only some characters, say *log n*, of the address to the network to identify itself. The network then directs traffic intended for the phone to any cell which has those *log n* characters in its address. A refinement process then takes place in which the phone gives out slightly more information to the system to improve performance by sending information to fewer cells, but not so much as to allow its location to be restricted to only one cell.

Jakobsson's Practical Mix

At EUROCRYPT '98, Jakobsson proposed a mixnet which was both practical and could be proved to mix correctly as long as less than 1/2 of the servers were corrupted[#!jakobsson-practical-mix!#]. The crucial idea is to treat the mixing as a secure multiparty computation in which each party is collaborating to make the collective mix look like a ``random enough// permutation on a batch of messages. Then techniques of zero-knowledge proof are used by which each server can prove to all other servers that they are in fact conforming to the mix protocol. Deviating servers cannot produce valid proofs, and so can be caught and excluded from future mixing. Jakobsson's original protocol requires in the neighborhood of 160 modular exponentiations per message per server.

At PODC '99, Jakobsson showed how the use of precomputation could reduce the cost even further[#!jakobsson-flash-mix!#]. This new ``flash mix'' required only around 160 modular *multiplications* per message per server. This level of efficiency makes flash mixing competitive with the encryption used in anonymous remailers, and a serious candidate for low-latency mixing.

At Eurocrypt '00, ``How to Break a practical mix, and fix it.''

Universally Verifiable Mix-nets

With Jakobsson's design, the correctness of a mix-net can only be verified by the mix servers themselves. When more than a threshold of servers is corrupt, the verification fails. Because a user of the mix-net may not be aware of the corruption, this failure may be silent and therefore dangerous. One solution to this problem is a *universally verifiable* mix-net - a mix-net whose correctness can be verified by anyone, regardless of their status as server or user.

as server or user.

The concept was introduced by Killian [#!universal-verifiable-mix-1!#], and recently a design of this type was proposed at EUROCRYPT '98 by Abe [#!abe-mix!#]. This design works along the similar broad lines as the Jakobsson design; each mix server uses zero-knowledge proofs to prove that it is acting in accordance with some protocol to randomly mix messages. The difference here is that these proofs are posted publically by the mix nodes instead of being multicast only to other mix nodes. The novel feature of Abe's design is that the work necessary to verify these proofs grows in a fashion independent of the number of servers. Unfortunately, verifying these proofs requires on the order of 1600 modular exponentiations per message.

Onion Routing

The Onion Routing system designed by Syverson, et. al. creates a mix-net for TCP/IP connections [#!onion-routing-paper!#,#!onion-router!#]. In the Onion Routing system, a mixnet packet, or ``onion'', is created by successively encrypting a packet with the public keys of several mix servers, or ``onion routers.''

When a user places a message into the system, an ``onion proxy'' determines a route through the anonymous network and onion encrypts the message accordingly. Each onion router which receives the message peels the topmost layer, as normal, then adds some key seed material to be used to generate keys for the anonymous communication. As usual, the changing nature of the onion - the ``peeling'' process - stops message coding attacks. Onions are numbered and have expire times, to stop replay attacks. Onion routers maintain network topology by communicating with neighbors, using this information to initially build routes when messages are funneled into the system. By this process, routers also establish shared DES keys for link encryption.

The routing is performed on the application layer of onion proxies, the path between proxies dependent upon the underlying IP network. Therefore, this type of system is comparable to loose source routing.

Onion Routing is mainly used for sender-anonymous communications with non-anonymous receivers. Users may wish to Web browse, send email, or use applications such as `rlogin`. In most of these real-time applications, the user supplies the destination hostname/port or IP address/port. Therefore, this system only provides receiver-anonymity from a third-party, not from the sender.

Furthermore, Onion Routing makes no attempt to stop timing attacks using traffic analysis at the network endpoints. They assume that the routing infrastructure is uniformly busy, thus making passive intra-network timing difficult. However, the network might not be statistically uniformly busy, and attackers can tell if two parties are communicating via increased traffic at their respective endpoints. This endpoint-linkable timing attack remains a difficulty for all low-latency networks.

Zero Knowledge Systems

Recently, the Canadian company Zero Knowledge Systems has begun the process of building the first mix-net operated for profit, known as Freedom [#!zks!#]. They have deployed two major systems, one for e-mail and another for TCP/IP. The e-mail system is broadly similar to Mixmaster, and the TCP/IP system similar to Onion Routing.

ZKS's ``Freedom 1.0'' application is designed to allow users to use a nym to anonymously access web pages, use IRC, etc. The anonymity comes from two aspects: first of all, ZKS maintains what it calls the Freedom Network, which is a series of nodes which route traffic amongst themselves in order to hide the origin and destination of packets, using the normal layered encryption mixnet mechanism. All packets are of the same size. The second aspect of anonymity comes from the fact that clients purchase ``tokens'' from ZKS, and exchange these token for nyms - supposedly even ZKS isn't able to correlate identities with their use of their nyms.

The Freedom Network looks like it does a good job of actually demonstrating an anonymous mixnet that functions in real-time. The system differs from Onion Routing in several ways.

First of all, the system maintains Network Information Query and Status Servers, which are databases which provide network topology, status, and ratings information. Nodes also query the key servers every hour to maintain fresh public keys for other nodes, then undergo authenticated Diffie-Hellman key exchange to allow link encryption. This system differs from online inter-node querying that occurs with Onion Routing. Combined with centralized nym servers, time synchronization, and key update/query servers, the Freedom Network is not fully decentralized [#!freedom-architecture!#].

Second, the system does not assume uniform traffic distribution, but instead uses a basic ``heartbeat'' function that limits the amount of inter-node communication. Link padding, cover traffic, and a more robust traffic-shaping algorithm have been planned and discussed, but are currently disabled due to engineering difficulty and load on the servers. ZKS recognizes that statistical traffic analysis is possible [#!freedom-security!#].

Third, Freedom loses anonymity for the primary reason that it is a commercial network operated for profit. Users must purchase the nyms used in pseudonymous communications. Purchasing is performed out-of-band via an online Web store, through credit-card or cash payments. ZKS uses a protocol of issuing serial numbers, which are reclaimed for nym tokens, which in turn are used to anonymously purchase nyms. However, this system relies on ``trusted third party'' security: the user must trust that ZKS is not logging IP information or recording serial-token exchanges that would allow them to correlate nyms to users [#!freedom-nyms!#]. The future adoption of anonymous ecash purchasing should remove this weakness, and allow truely anonymous nym issuing.

Web Mixes

Another more recent effort to apply a Mix network to web browsing is due to Federrath et. al [#!web-mix!#] who call their browsing is due to Federrath et. al.[#!web-mix!#] who call their system, appropriately enough, ``Web Mixes.'' From Chaum's mix model, similar to other real-time systems, they use: layered public-key encryption, prevention of replay, constant message length within a certain time period, and reordering outgoing messages.

The Web Mixes system incorporates several new concepts. First, they use an adaptive ``chop-and-slice'' algorithm that adjusts the length used for all messages between time periods according to the amount of network traffic. Second, dummy messages are sent from user clients as long as the clients are connected to the Mix network. This cover traffic makes it harder for an adversary to perform traffic analysis and determine when a user sends an anonymous message, although the adversary can still tell when a client is connected to the mixnet. Third, Web Mixes attempt to restrict insider and outsider flooding attacks by limited either available bandwidth or the number of used time slices for each user. To do this, users are issued a set number of blind signature tickets for each time slice, which are spent to send anonymous messages. Lastly, this effort includes an attempt to build a statistical model which characterizes the knowledge of an adversary attempting to perform traffic analysis.

Other Anonymous Channels

The Dining Cryptographers

The Dining Cryptographers protocol was introduced by David Chaum[#!chaum-dc!#] and later improved by Pfitzmann and Waidner as a means of guaranteeing untraceability for the sender and receiver of a message, even against a computationally all-powerful adversary. The protocol converts any broadcast channel into an anonymous broadcast channel. In the context of Free Haven, however, we have a problem : the participants in the protocol are identified, even though the sender and receiver of any given message is not. If the only long-term participants in the protocol are likely to be Free Haven servnet nodes, then we do not achieve the server-anonymity we desire. Less serious, but still important, problems are the efficiency of the protocol and the difficulty of correct implementation.

Therefore we have not seriously considered using the dining cryptographers protocol to provide Free Haven's anonymous channel. If we were to do so, we might consider running a dining cryptographer protocol using Mixes to hide the legal identity of each participant. In that case, while a failure of the Mix would reveal a participant's identity, the anonymous broadcast would prevent him or her from being linked to any particular message.

Crowds

The Crowds system was proposed and implemented by AT&T Research, named for collections of users that are used to achieve partial anonymity for Web browsing [#!crowds!#]. A user initially joins some crowd and her system begins acting as a node, or anonymous *jondo*, within that crowd. In order to instantiate communications, the user creates some path through the crowd by a random-walk of *jondos*, in which each *jondo* has some small probability of sending the actual `http` request to the end server. Once established, this path remains static as long as the user remains a member of that crowd. The Crowds system does not use dynamic path creation so that colluding crowd eavesdroppers are not able to probabilistically determine the initiator (i.e., the actual sender) of requests, given repeated requests through a crowd. The *jondos* in a given path also share a secret *path key*, such that local listeners, not part of the path, only see an encrypted end server address until the request is finally sent off. The Crowds system also includes some optimizations to handle timing attacks against repeated requests, as certain HTML tags cause browsers to automatically issue re-requests.

Similar to other real-time anonymous communication channels (Onion Routing, the Freedom Network, Web Mixes), Crowds is used for senders to communicate with a known destination. The system attempts to achieve sender-anonymity from the receiver and a third-party adversary. Receiver-anonymity is only meant to be kept from adversaries, not from the sender herself.

The Crowds system serves primarily to achieve sender and receiver anonymity from an attacker, not provide unlinkability between the two agents. Due to high availibility of data - real-time access is faster that mix-nets as Crowds does not use public key encryption - an adversary can more easily use traffic analysis or timing attacks. However, Crowds differs from all other systems we have discussed, as users are *members* of the communications channel, rather than merely communicating *through* it. Sender-anonymity is still lost to a local eavesdropper that can observe all communications to and from a node. However, other colluding *jondos* along the sender's path - even the first-hop - cannot expose the sender as originated the message. Reiter and Rubin show that as the number of crowd members goes to infinity, the probable innocence of the last-hop being the sender approaches one.

Ostrovsky's Anonymous Broadcast via XOR-Trees

In CRYPTO '97, Ostrovsky considered a slightly different model of anonymous broadcast. In this model, there are n servers broadcasting into a shared broadcast channel. One of the servers is a special ``Command and Control'' server; the rest are broadcasting dummy traffic. Then there is an adversary who has control of some of the servers and wants to know which server is the ``Command and Control.'' Ostrovsky shows how to use correlated pseudo-random number generators whose output reveals a certain message when XORed together to create a protocol which prevents the adversary from discovering which server is the correct one, even if he can eavesdrop on all communications and corrupt up to k servers, where k is a security parameter which affects the efficiency of the protocol.

About this document ...

This document was generated using the **LaTeX**2`HTML` translator Version 98.1 release (February 19th, 1998)