Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems

Adam Back, Ulf Möller, and Anton Stiglic

Zero-Knowledge Systems Inc.
{adamb,ulf,anton}@zeroknowledge.com

Abstract We discuss problems and trade-offs with systems providing anonymity for web browsing (or more generally any communication system that requires low latency interaction). We focus on two main systems: the Freedom network [12] and PipeNet [8]. Although Freedom is efficient and reasonably secure against denial of service attacks, it is vulnerable to some generic traffic analysis attacks, which we describe. On the other hand, we look at PipeNet, a simple theoretical model which protects against the traffic analysis attacks we point out, but is vulnerable to denial of services attacks and has efficiency problems. In light of these observations, we discuss the trade-offs that one faces when trying to construct an efficient low latency communication system that protects users anonymity.

1 Introduction

For several years, cryptographers have studied ways of providing confidentiality, authentication and integrity to parties that wish to communicate securely amongst each other. Protocols that provide these properties have been thoroughly studied, and we now have efficient, effective, and reasonably well understood protocols. One other desired property that has received much less attention is that of protecting the identity of one's partners in communication. This is an important property, for example the mere fact that two competing companies are exchanging messages might leak valuable information (it could be an indication that the two companies are negotiating a merger). It is also a property desired by Internet users; users do not want to be monitored and have their surfing habits logged and preserved by unauthorized parties, in order to create a dossier that contains extensive information about them and is stored for long periods of time. In this paper we focus on systems providing anonymity for web browsing, or more generally for areas where low latency, interactive communication is needed (e-mail protocols, for example, generally do not require this). These protocols have to be efficient as well as hide the identities of the two communicating parties (what URL a certain user is accessing).

1.1 Overview

We examine the Freedom network [12] and describe traffic analysis attacks against the system, which generalize to many other anonymity providing systems. We work with a simplified threat model that turns out to be very useful. We then take a look at PipeNet

[8], a theoretical model which seems to guard against the traffic analysis attacks we describe. PipeNet is an interesting concept to analyze since it is simple, supports interactive stream-oriented communication and offers good security. It implements synchronicity over an asynchronous network, which allows it to protect against the traffic analysis attacks we know about. However, PipeNet is inefficient and vulnerable to catastrophic denial of services (DoS) attacks which are easy to perpetrate, whereas Freedom seems to withstand DoS attacks better and is efficient. We point out a trade-off that presently exists in anonymity providing systems: one wants to balance traffic analysis resistance, performance, DoS resistance and bandwidth cost.

1.2 Organization

In section 2, we discuss previous theoretical and practical work relating to anonymity providing systems demanding low latency communication. Practical issues regarding the deployment of these systems are explored in section 3. In section 4 we give a high level description of the parts of Freedom that relate to anonymous browsing (ignoring extra functionalities such as cookie management, e-mail and pseudonymity). In section 5 we define anonymity in an interactive setting. We then describe, in section 6, traffic analysis attacks against Freedom under this threat model. Section 7 describes PipeNet and discusses its advantages and disadvantages. Section 8 describes the trade-offs pointed out in earlier sections. Conclusions and open problems can be found in section 9.

2 Related Work

In [6], Chaum describes a way to enable one participant to anonymously broadcast a message (DC-net). If the message is destined to a specific user, it can be encrypted with the user's public key. Since the message is received by all parties, recipient anonymity is trivially maintained. Unfortunately, this method has several serious drawbacks: only one participant can send a message at any given time, throughput is limited to worst case, a participant can deny services to others by constantly sending messages through the DC-net[1], the complexity of communication on most network topologies is very great[2], the number of participants a user needs to share a secret key with can grow to be very large, and active attacks allow malicious users to compute legitimate messages while others gain no information on it. Work has been done to solve the problem of DoS by detecting disrupters, replacing the reliable broadcast assumption and protecting against active attacks ([30], [3], [29]), but the resulting protocols still suffer from efficiency problems – each participant has to send at least as much in the physical sense as all the participants together want to send in the logical sense.

[1] A solution to this problem was presented in [6], but its communication and time complexity is quadratic in the size of the anonymity set, making it infeasible in practice.

[2] Chaum argues that DC-nets are efficient in a ring topology, which can be found on some local networks, but does not exist in large scale networks such as the Internet.

Secure multi-party computations[3] are a related problem that has received considerable attention ([13], [11], [27], [7]). A multi-party computation protocol can be used to hide participants' communication partners ([24]). But general multi-party computations are inefficient in practice with regards to communication complexity, and most solutions rely on the existence of a synchronous network and are often not secure against dynamic adversaries. Multi-party computations that are secure in an asynchronous network are even more complex (see [4]).

In [5], Chaum introduced the idea of the mix-net. A mix is a node (server) in a network that receives a certain number of messages, modifies them using some cryptographic transformation and outputs them in a random order in such a way that one cannot correlate which output message belongs to which input message, without the aid of the mix node, when several messages are passed simultaneously.

Rackoff and Simon ([24]) define (and provide a proof of security for) a system that uses mix-nodes. Unfortunately, the setting in which they work is not practical either: they assume a synchronous network, use mix-nodes to process at most two messages at a time and put constraints on the routes. Additional work has been done on mix-nets ([22], [17], [16], [18], [21], [10]), but the proposed solutions also rely on a synchronous network, reliable broadcast channels and use public key encryption extensively. In general, mix-nodes introduce some latency because messages are delayed by the mix, which can be acceptable for applications such as e-mail but less so for applications such as web surfing.

On a more practical side, several systems providing fast, anonymous, interactive communication have been implemented. The first one was the Anonymizer ([1]) from Anonymizer.com. It is essentially a server with a web proxy that filters out identifying headers and source addresses from web browsers' requests (instead of seeing the users true identity, a web server sees only the identity of the Anonymizer server). This solution offers rather weak security (no log safeguarding and a single point of vulnerability).

Crowds ([26]) consists of a number of network nodes that are run by the users of the system. Web requests are randomly chained through a number of them before being forwarded to the web server hosting the requested data. The server will see a connection coming from one of the Crowds users, but cannot tell which of them is the original sender. In addition, Crowds uses encryption, so that some protection is provided against attackers who intercept a user's network connection. However, this encryption does not protect against an attacker who cooperates with one of the nodes that the user has selected, since the encryption key is shared between all nodes participating in a connection. Crowds is also vulnerable to passive traffic analysis: since the encrypted messages are forwarded without modification, traffic analysis is trivial if the attacker can observe all network connections. An eavesdropper intercepting only the encrypted messages between the user and the first node in the chain as well as the cleartext messages between the final node and the web server can associate the encrypted data with the plaintext using the data length and the transmission time.

[3] Computations in which participants compute a public function on their private inputs and in which participants learn nothing more than what they can deduce form their own input and the result of the computation of the function

Onion Routing ([14], [28]) is another system that allows anonymous browsing. In this system, a user sends encrypted data to a network of so-called Onion Routers (essentially, these are real-time mixes[4]). A trusted proxy chooses a series of these network nodes and opens a connection by sending a multiply encrypted data structure called an "onion" to the first of them. Each node removes one layer of encryption, which reveals parameters such as session keys, and forwards the encrypted remainder of the onion to the next network node. Once the connection is set up, an application specific proxy forwards HTTP data through the Onion Routing network to a responder proxy which establishes a connection with the web server the user wishes to use. The user's proxy multiply encrypts outgoing packets with the session keys it sent out in the setup phase; each node decrypts and forwards the packets, and encrypts and forwards packets that contain the server's response.

In spite of the similar design, Onion Routing cannot achieve the traffic analysis protection of an ideal mix-net due to the low-latency requirements, as shown in section 6. The same is the case for the Freedom network described in section 4; Freedom however is less similar to mix-nets in that is does not attempt to reorder packets.

In [23], the mix-net concept is extended to allow for interactive use in the special setting of digital telephony, while retaining most of its security features. So-called mix-channels provide anonymity among the users of a local exchange. A channel establishment message is sent through a fixed sequence of mixes (cascade[5]), which then reserve bandwidth for the channel. If a mix does not receive data in time, it will fill the channel with dummy traffic. Mix-channels would require a large number of connections that are initiated at the same time and have equal length. This problem is solved with the introduction of time-slice channels: Users always maintain a fixed number of active channels and decide at the beginning of each time slice which channels are used for actual communications and which of them generate cover traffic. To signal a connection request, connection requests are broadcast at the callee's local exchange. This results in limiting the anonymity set to about 5000 users.

3 Practical Considerations

For practical systems there are a number of reasons why it is necessary to have a protocol that is implementable on existing Internet routing infrastructure, and implementable with adequate performance in software on existing network hosts which would be likely to participate in the system.

– *infrastructure cost* – replacing Internet infrastructure is prohibitively expensive. This rules out systems relying on communications links and constructs not available on the Internet, such as anonymous broadcast, synchronous connections and reliability.

[4] Real-time mixes, contrary to ordinary Chaum mixes, process messages in real-time, thus can't wait an indefinite amount of time in order to receive an adequate number of messages to mix together.

[5] The advantages of cascades over freely selected routes – especially when a large number of mixes is compromised – are discussed in [2].

- *node hardware cost* – adding hardware acceleration boards to machines acting as nodes is expensive and a barrier to entry. Hardware systems able to perform public key operations per IP packet on high capacity links are currently very expensive. As Internet bandwidth and the bandwidth supported by Internet hosts is growing quickly also, this appears likely to remain the case for the foreseeable future. This rules out systems relying on public key operations per packet, such as mix-net based systems.
- *public auditability* – components in distributed trust security systems should be *publicly auditable*, and performing third party audits of hardware is much harder than for software systems with published source. This makes custom hardware undesirable.
- *convenience* – it must be convenient for potential node operators to participate in the network. Installing custom hardware is not convenient.

In this paper we concern ourselves with systems which are efficient and deployable with respect to the above criteria.

4 Freedom

The Freedom network [12] is composed of a set of nodes called Anonymous Internet Proxies which run on top of the existing Internet infrastructure. To communicate with a web server, the user first selects a series of nodes (a route), and then uses this route to forward IP packets that are stripped of identifying information. (Identifying HTTP headers are also stripped away by a proxy on the clients machine.)

The client uses a route creation protocol to set up a communications channel through the Freedom network. This protocol enables the client to share two secret keys with each node (one for each direction of communication), as well as to tell each node what the previous and next nodes are in the route. During this protocol, each node sets a pair of Anonymous Connection Identifiers (ACIs) which are unique and associate next and previous nodes with a route. Each node ends up knowing only what the next and previous nodes are on a certain route. The client can share keys with these nodes without being identified through the execution of half-certified Diffie-Hellman key agreement (only the node side is certified, the client side is anonymous).

Each node in the route, except for the last, simply forwards the packets it receives to the next node in the route. When the last node receives a packet, it replaces the missing IP source address (that was stripped by the sender) with a special IP address called the wormhole IP address. (Nodes have one or more wormhole IP addresses that are used as exit points for routes in the Freedom network in addition to their regular Internet addresses.[6])

Now, if a user simply sent IP packets in the clear, an observer could easily follow the packets and determine which web server a certain client is communicating with and through which route. To prevent this, the client multiply encrypts each packet it sends. The client first encrypts the whole IP packet with the key it shares with the last node,

[6] Network Address Translation techniques make it possible to support multiple clients using the same wormhole IP address.

the result is then encrypted with the key shared with the penultimate node, and so on, all the way down to the key it shares with the first node. The client concatenates the ACI of the first node to the resulting message, then sends the result to the first node in the route. The node decrypts the first layer, and forwards the packet to the node defined by the ACI, rewriting the ACI for the second node. This is done at each node, in turn, and the IP packet finally exits the wormhole to the web server. The web server sees only a packet with IP headers corresponding to the wormhole. By multiply encrypting the packets, no node (apart from the last one) can view the contents of the packets, nor can any external attacker. To hide the ACIs (which can be used by an attacker to determine which nodes are part of a certain route), all communication between nodes is encrypted using symmetric encryption keys shared by the pairs of nodes in the network. The client also encrypts all communication with the first node. These symmetric keys are obtained by executing an ephemeral Diffie-Hellman key agreement.

5 Simplified Model of Anonymity

We use a model of anonymity that can easily be generalized to describe anonymity in more complex network scenarios. The simplified version is useful when describing attacks, providing a simple context for discussion. We consider two users, Alice and Bob, who are communicating with two web servers, W1 and W2, through a network of anonymizing nodes, which we call a cloud. See figure 1. We have some a priori probability, which models our suspicion about who is communicating with whom. More precisely, the a priori probability that Alice is communicating with W1 is p and the a priori probability that Alice is communicating with W2 is $q = 1 - p$. If we have no a priori information $p = \frac{1}{2}$.

The goal of an attacker is to distinguish the events "Alice is communicating with W1" and "Alice is communicating with W2". If the attacker learns no new information to confirm or deny his suspicions, so that his estimate of the probability that Alice is communicating with W1 is still p after his attack, the system is said to provide anonymity.

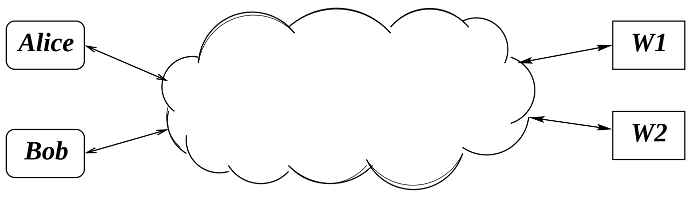

Figure 1. Anonymity in Interactive Setting

6 Traffic Analysis Against Freedom

We describe generic attacks that apply to Freedom, but also to other systems based on similar designs, such as Onion Routing.

6.1 Packet counting attack

One way of discovering with whom Alice is communicating is to find the nodes forming the route that Alice is using. It is easy to discover the first node because all communication from Alice goes to it. This can be accomplished by sniffing packets on Alice's ISP, or any router in the communication path between Alice and her first node. Then, you can count the number of packets entering the first node, originating from Alice, and examine the number of packets leaving it. (Even if the payloads are encrypted, you can still easily count them, as long as you can sniff packets before and after the node. Its ISP, for example, can do this.) You can now determine to which second node, of possibly several, the first node is forwarding Alice's packets. Even if Bob connects to the same first node, you can use a counting method to distinguish between the packets being relayed on behalf of Alice and Bob. One then applies the same method at each node, until arriving at the last.

Constant link padding and traffic shaping One way of defending against such an attack is to use constant link padding. Constant link padding between two nodes has the nodes exchange a constant number of same-sized packets per time unit. But constant link padding leaves the system vulnerable to other types of attacks such as the latency attack described in subsection 6.3. Also, constant link padding is very costly if you are paying for each packet that is being sent over a network. Traffic shaping[7] [15], as implemented in the second generation Onion Routing system, could be a solution to this last problem, but it still leaves the system vulnerable to certain attacks such as Wei Dai's and others described later.

6.2 Wei Dai's attack on Traffic Shaping

In [9], Wei Dai describes a generic attack against systems that allocate bandwidth to the users as connections are established and implement traffic shaping between nodes. Here the attacker creates an anonymous route to himself, through a pair of nodes he suspects to belong to Alice's route. The attacker then increases the traffic through this route until the total traffic between the pair of nodes reaches the bandwidth limit set by the traffic shaping. At this point the nodes no longer send any padding packets to each other, and the real traffic throughput between them can be deduced by subtracting the traffic sent by the attacker from the bandwidth limit.

[7] Traffic shaping in this context refers to the nodes using an algorithm based on a rolling average of real traffic, to let padding decay over some time period based on utilization.

6.3 Latency attack

The latency attack is probably the most difficult to protect against. It is based on the fact that the latency on different routes will differ, and these latencies can be computed by the attacker. To compute the latency in a communication path going from the user through nodes A, B and C to a server W1, an attacker simply needs to use the system to create a route through those nodes to communicate to W1 and compute the latency (e.g. using ping times) of communication and subtract the latency from the communication path between the attacker and node A. The closer the attacker is to the first node, the more precise his timings will be (communication won't be greatly re-routed by the underlying network). The attacker can then compute the latency between Alice and the first node (this is trivial if he controls the first node). Once the attacker has computed a set of timings, there are several things the attacker can do, depending on the timings he gathered. If some routes clearly differ by their latency timings, it is easy to determine which route Alice was using. Statistical methods can be used to remove noise in order to obtain extra precision, similarly to the methods proposed in [19] (in a different context). If the attacker notices spikes on a graph of latency versus time for Alice's route, he can match those with spikes on the graphs of routes whose latency he has been measuring.

This attack reveals what seems to be a fallacy in theoretical definitions of security. For example, in [28], the authors state that if links are padded or bandwidth is limited to a constant rate, one can ignore passive eavesdroppers[8]. This is technically correct if a passive eavesdropper is defined as someone who cannot access the network as a regular user and compute timings on the network (which is implied by the definition used in most theoretical work). However this attack model is not very interesting and definitely misleading. The latency attack pointed out above and the next attack we present demonstrate that if an attacker can simply compute timings (which is as passive as one can expect an attacker to be in practice), or use the system, link padding or bandwidth limiting links to a constant rate does not protect the system against easy traffic analysis attacks.

6.4 Clogging attack

In a simpler timing attack, an attacker observes the communication between a certain last node C and W1. He then creates a route through a chosen set of nodes and clogs the route with many requests. If he observes a decrease in throughput from C to W1, he can deduce that one of the nodes in the route he created belongs to a route containing C. The attacker can use a binary style search to find all the nodes belonging to a certain route. Once the route to W1 is known, the attacker knows the users first node. He can then use similar techniques to identify the individual user of the possible users of that node. This attack is plausibly deniable as Internet traffic is often bursty.

A variant of the clogging attack is to exploit some IP protocol or implementation flaw to temporarily delay packet delivery at an intermediate router (not necessarily a node) on a targeted route.

[8] Section 4, Assumption 2 of [28]

7 PipeNet

PipeNet [8] is a synchronous network implemented on top of an asynchronous network. Routes are created through the network from entry to exit node with hops chosen uniformly at random. The route creation requests are mixed – a certain number of route creation requests are collected by a node, shuffled and then acted upon. The user establishes a shared key with each node on its route as part of the route creation process, using a key negotiation algorithm. The routes are padded end to end for their duration. End-to-end padding means that the originator creates all of the padding and the recipient (or exit node) strips the padding, each of the intermediate nodes is unable to distinguish padding from normal traffic, and just processes it as normal.

Each node uses the scheduling algorithm that consists of waiting for a packet on each link before forwarding any packets, when all packets have arrived the packets are shuffled and forwarded to their respective next hops. (Route destroy requests are also mixed.) The network is synchronous in the sense that one packet per link is sent, however there may be more than one link between pairs of nodes; the number of links between a pair of nodes corresponds on the number of routes currently active between that pair of nodes.

It is presumed that the topology would likely be fully connected due to the randomized route selection process. In any case, the topology is considered public knowledge, as an observer can watch the increase in traffic per neighbor pair after route creations. So for example, if there was three routes using node A, and two new route creations are processed in a batch, after that time unit that node will send five packets per time unit.

The attacker can observe the effect of route creations as described above, and so has a map of candidate exit nodes corresponding to a given user. However, because the route creations are batched and mixed and all traffic for the duration of the route is fully padded and synchronous, he can not distinguish between these based on passive attacks. In addition, as the routes are end-to-end padded, compromised nodes obtain no information. The exit node is able to observe padding, since it must remove it.

PipeNet is also invulnerable to active attacks based on selective DoS because the scheduling algorithm ensures that the network *reacts* to selective DoS attacks by shutting down. No information other than the already public topology is leaked by this process. However, this exposes PipeNet to an easy and catastrophic DoS attack: any user can forever shut down the entire network by creating a route and sending no packets through it. Performance suffers for similar reasons: the scheduling algorithm means that performance is lowered to the worst-case latency of the links between each pair of nodes in the route (in the fully connected case, the worst-case latency in the entire network). In addition, the system is not robust, even in absence of intentional attackers – PipeNet would amplify the unreliability of the Internet; a temporary outage on any link would make the entire network unavailable.

8 Tradeoffs, Hybrid Version

The traffic analysis problem can be considered to be a four-way optimization problem, with the following optimization criteria:

– traffic analysis resistance
– performance
– resistance to catastrophic DoS
– bandwidth cost

In addition, the security of anonymity systems is affected by the size of the user base. The fact that users are using the system is not hidden, so the anonymity of a given action is only protected to the extent that the identity is known to be one of the set of people who was online for the duration over which the activity took place. In anonymity systems usability, efficiency, reliability and cost become *security* objectives because they affect the size of user base which in turn affects the degree of anonymity it is possible to achieve.

Interestingly, the two networks which provide good theoretical security – PipeNet and DC-net – are both vulnerable to catastrophic DoS attacks (presuming that one must adopt a PipeNet like scheduling algorithm to implement a DC-net on the Internet), and both have scheduling algorithms that adversely affect performance. The bandwidth consumption is high in both, but worst in DC-nets.

Freedom is bandwidth efficient, has reasonable performance, is resistant to catastrophic DoS, but only provides traffic analysis resistance in a weaker threat model.

It remains an open question whether there exist hybrid or alternate protocols which have resistance to catastrophic DoS, reasonable bandwidth cost, reasonable performance and provide traffic analysis resistance against a more aggressive threat model than Freedom does.

9 Conclusion

Traffic Analysis is an area of cryptography that is not well represented in the open literature. We have examined theoretical as well as practical network designs and compared their characteristics in the four-way optimization model. We pose the question as to whether other interesting protocols exist, with better trade-offs, that would be practical to implement and deploy.

10 Acknowledgements

We thank Wei Dai for discussions on PipeNet, and Adam Shostack for general discussions on traffic analysis. We also thank Andreas Pfitzmann and the anonymous referees for valuable comments and Bill Heelan for advice on the presentation.

References

[1] ANONYMIZER.COM. The anonymizer.
[2] BERTHOLD, O., PFITZMANN, A., AND STANDTKE, R. The disadvantages of free mix routes and how to overcome them. In *Proc. Workshop on Design Issues in Anonymity and Unobservability* (25–26 July 2000), ICSI TR-00-011, pp. 27–42.

[3] Bos, J., and Boer, B. D. Detection of disrupters in the DC protocol. In *Advances in Cryptology – EUROCRYPT '89* (1989), pp. 320–327.

[4] Canetti, R. *Studies in Secure Multiparty Computation and Applications*. PhD thesis, Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, June 1995. revised version.

[5] Chaum, D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the Association for Computing Machinery 24*, 2 (Feb. 1981), 84–88.

[6] Chaum, D. The Dining Cryptographers Problem: Unconditional sender and recipient untraceability. *Journal of Cryptology 1*, 1 (1988), 65–75.

[7] Cramer, R., Damgård, I., Dziembowski, S., Hirt, M., and Rabin, T. Efficient multiparty computations with dishonest minority. In *Advances in Cryptology—EUROCRYPT 99* (March 1999), vol. 1561 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 311–326.

[8] Dai, W. Pipenet 1.1. http://www.eskimo.com/˜weidai/pipenet.txt, 1998.

[9] Dai, W. Two attacks against freedom. http://www.eskimo.com/˜weidai/freedom-attacks.txt, 2000.

[10] Desmedt, Y., and Kurosawa, K. How to break a practical mix and design a new one. In *Advances in Cryptology – EUROCRYPT '2000* (2000), Lecture Notes in Computer Science, International Association for Cryptologic Research, Springer-Verlag, Berlin Heidelberg, pp. 557–572.

[11] Gennaro, R., Rabin, M. O., and Rabin, T. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *PODC: 17th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing* (1998).

[12] Goldberg, I., and Shostack, A. Freedom network 1.0 architecture and protocols. htt://www.freedom.net/info/freedompapers/index.html, 1999.

[13] Goldreich, O., Micali, S., and Wigderson, A. How to play any mental game — A completeness theorem for protocols with honest majority. In *Proceedings of the nineteenth annual ACM Symposium on Theory of Computing, New York City, May 25–27, 1987* (New York, NY 10036, USA, 1987), ACM, Ed., ACM Press, pp. 218–229.

[14] Goldschlag, D., Reed, M., and Syverson, P. Onion routing for anonymous and private internet connections. *Communications of the ACM (USA) 42*, 2 (Feb. 1999), 39–41.

[15] Green, L. Traffic shaping argument. Article on cypherpunks list, 1993.

[16] Jakobsson. Flash mixing. In *PODC: 18th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing* (1999).

[17] Jakobsson, M. A practical mix. *Lecture Notes in Computer Science 1403* (1998), 448–??

[18] Jakobsson, M., and Juels, A. Millimix: Mixing in small batches. Tech. Rep. 99-33, DIMACS, June 10 1999. Thu, 22 Jul 1999 23:50:00 GMT.

[19] Kocher, P. C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology – CRYPTO ' 96* (1996), N. Koblitz, Ed., Lecture Notes in Computer Science, International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, pp. 104–113.

[20] Menezes, A. J., van Oorschot, P. C., and Vanstone, S. A. *Handbook of applied cryptography*. The CRC Press series on discrete mathematics and its applications. CRC Press, 2000 N.W. Corporate Blvd., Boca Raton, FL 33431-9868, USA, 1997.

[21] Ohkubo, M., and Abe, M. A length-invariant hybrid mix. In *Advances in Cryptology – ASIACRYPT '2000* (2000), Lecture Notes in Computer Science, International Association for Cryptologic Research, Springer-Verlag, Berlin Heidelberg, pp. 178–191.

[22] Park, C., Itoh, K., and Kurosawa, K. Efficient anonymous channel and all/nothing election scheme. *Lecture Notes in Computer Science 765* (1994), 248–??

[23] PFITZMANN, A., PFITZMANN, B., AND WAIDNER, M. ISDN-MIXes: untraceable communication with very small bandwidth overhead. In *Information Security, Proc. IFIP/Sec '91* (1991), pp. 245–258.

[24] RACKOFF, C., AND SIMON, D. R. Cryptographic defense against traffic analysis. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing* (San Diego, California, 16–18 May 1993), pp. 672–681.

[25] RAYMOND, J.-F. Traffic analysis: Protocols, attacks, design issues and open problems. In *Proc. Workshop on Design Issues in Anonymity and Unobservability* (25–26 July 2000), ICSI TR-00-011, pp. 7–26.

[26] REITER, M. K., AND RUBIN, A. D. Anonymous Web transactions with crowds. *Communications of the ACM 42*, 2 (Feb. 1999), 32–48.

[27] SMITH, A., AND STIGLIC, A. Multiparty computation unconditionally secure against Π^2 adversary structures. Cryptology SOCS-98.2, School of Computer Science, McGill University, Montreal, Canada, 1998.

[28] SYVERSON, P. F., TSUDIK, G., REED, M. G., AND LANDWEHR, C. E. Towards an analysis of onion routing security. In *Proc. Workshop on Design Issues in Anonymity and Unobservability* (25–26 July 2000), ICSI RR-00-011, pp. 83–100.

[29] WAIDNER, M. Unconditional sender and recipient untraceability in spite of active attacks. In *Advances in Cryptology – EUROCRYPT ' 89* (1990), J.-J. Quisquater and J. Vandewalle, Eds., Lecture Notes in Computer Science, International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, pp. 302–319.

[30] WAIDNER, M., AND PFITZMANN, B. The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability. In *Advances in Cryptology—EUROCRYPT 89* (10–13 Apr. 1989), J.-J. Quisquater and J. Vandewalle, Eds., vol. 434 of *Lecture Notes in Computer Science*, Springer-Verlag, 1990, p. 690.