Cryptographic Defense Against Traffic Analysis *

Charles Rackoff †   Daniel R. Simon ‡

"But, after all", he reflected before turning into her room, "what has occurred? Nothing. She had a long conversation with him. Well, what harm is there in that?"....and as soon as he re-entered the dark drawing room a voice would whisper that it was not so, and that if others noticed, that showed there was something to notice.

—Tolstoy, *Anna Karenina*

Abstract

We present a model which allows us formally to define "untraceability" of messages in a network of synchronously communicating processors. We consider several different definitions, based on different assumptions about the strength of the "adversary" attempting to identify the senders and receivers of messages; for example, the adversary may be able to control some of the processors to obtain information, or even disrupt the traffic in the network. We present efficient protocols which are provably secure against each such adversary, using such cryptographic techniques as secure multiparty computation ([GMW]) and non-interactive zero-knowledge proof ([BFM]). One proof of security also relies on an interesting general lemma about the "mixing" achieved by certain kinds of random processes, or "shuffles", performed on a set of items.

*This work was supported in part by NSERC operating grants and ITRC, an Ontario Centre of Excellence.

†Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4; rackoff@cs.toronto.edu

‡Département IRO, Université de Montréal, C.P. 6128, Succursale "A", Montréal, Québec, H3C 3J7; simon@iro.umontreal.ca

1 Introduction

One rarely-discussed aspect of privacy in a network setting is the secrecy of the identities of one's "partners in conversation". Cryptographic treatments of secure communication often assume the identities of the sender, receiver and eavesdropper to be fixed and known; however, in the context of a network of intercommunicating parties, some sets of interlocutors may well wish even the fact of their communication to be secret.

Now, certain types of information cannot reasonably be assumed to be concealed. For instance, an upper bound on the total volume of a party's sent or received communication (of any sort) is obtainable by anyone with the resources to examine all possible physical communication channels available to that party. Moreover, any difference between that upper bound and the "true" amount of (sent or received) information must necessarily be the result of redundancy in the bits physically sent to or by the party in question, in order to give the appearance of extra communication. In other words, any secrecy regarding the total volume of information sent or received by a party is purchased at the cost of the extra "dummy traffic" generated to disguise it.

For example, if each party in a network simply sends (encrypted) messages to *every* party, continuously, as fast as it can (regardless of its actual intended interlocutors), then it will be impossible (assuming that encrypted "dummy" messages are indistinguishable from "legitimate" ones) for any observer to determine which pairs of parties are actually conversing at any given time. Moreover, each time a message is sent and received, the sending party disguises whether as few as one "legitimate" message, or as many as there are parties in the network, is being sent. On the other hand, this "trivial" solution may require each party to divide the maximum rate at which it sends "legitimate" messages (and the network to divide the number of "legitimate" messages it can carry) by the number of parties connected to the network, whenever, for instance, fixed pairs of parties must exchange long sequences of messages. Such an overhead may be prohibitive, and a more efficient solution necessary. We will assume, then, for the sake of simplicity, that all the parties in the network are capable of the same maximum (or maximum affordable) message transmission rate, which they will naturally exploit to the fullest, and that they wish to maximize the rate at which they can (all simultaneously) send "legitimate" messages without revealing the origins or destinations of those messages. In short, they wish to minimize the redundancy in the traffic they generate, while preserving the untraceability of their communication.

Discussion of this problem in a theoretical context is extremely sparse in the literature. In [Ch2], Chaum considers defenses against traffic analysis that depend on previously shared secrets (ie., the ability of parties to share random bits "offline" with perfect security), rather than the complexity-theoretic assumptions associated with public key cryptography. Goldreich ([Go]) considered another traffic-analysis-related problem, that of hiding memory addresses of data in a RAM; his solution was later improved by Ostrovsky ([Os]). In the model of [Go] and [Os], a CPU impervious to examination, and equipped with a secure cryptosystem, must access data in a possibly insecure memory, without revealing any information about its computations through its interaction with the memory. Such security must therefore involve not only encrypting all memory contents, but also concealing the CPU's pattern of memory accesses. This model is different from the multiparty network model primarily in that one node in the communication (the CPU) completely dominates the traffic, and the goal is simply to conceal with which other nodes (ie., memory locations) it is communicating.

Another related problem, that of *secure multiparty computation*, has received considerably more attention in the literature ([GMW], [B],[MR],[BMR]). A solution to the problem of hiding participants' inputs into a jointly computed function can in fact be used to hide participants' communication partners (as we shall see), and the latter problem can be considered a restriction of the former (one which possibly admits more efficient solutions). The problem of creating "untraceable electronic cash", studied in [Ch3], is also, fundamentally, a subcase of secure multiparty computation; the task is to create a means of computing new "bank balances" for parties following a financial transaction, while hiding information about the size or source of previous balances.

In [Ch1], Chaum introduces the key idea of the *mix*. A mix is simply a node in a network, associated with a particular public encryption key, through which multiple encrypted messages are sent, encrypted first with the receiver's public key, and then using the mix' public key. The mix decrypts each twice-encrypted message to obtain a once-encrypted message which is then relayed to the receiver. When several messages are passed simultaneously through a mix (Chaum informally argues), the association between the sent messages and the received messages is indiscernible to an observer wishing to spot such a correlation without the aid of the mix node. Some refinements to these ideas appear in [P] and [PPW].

One approach to defending against traffic analysis, then, is to pass messages through various mixes in a manner that completely disguises the resulting permutation. How quickly such an effect is achieved can be shown to depend on the rapidity with which certain Markov processes converge on their stable distribution. In particular, we will show that the process in which locations in an n-bit string of weight $n/2$ are randomly paired, and each pair randomly permuted, converges extremely rapidly (in a number of steps polynomial in $\log(n)$) on the uniform distribution on such strings. This result can be used to prove the security of a fairly simple, efficient solution, based on the above "mix" paradigm, to one variation of the traffic analysis problem.

2 The Model

We consider a model consisting of a completely connected network of n processors, each polynomial-time-bounded in n. Of course, the network may in practice have an arbitrary topology, including differences in cost among links. However, the complete network can in principle be implemented on any connected one at some cost related to the network's diameter, connectedness, and so forth (more will be said later about the efficiency of our solutions and its dependence on the underlying topology). We will also assume that the processors in our network are given concise, distinct labels (say, the integers from 1 to n), and send messages back and forth among each other synchronously (not concerning ourselves with how the synchronization is to be effected). Multiple messages sent to a particular processor in the same step are queued in completely arbitrary order.

The goal of each of the parties in the network will be to send a "missive" to one other party in the network. (Henceforth, the term *missive* will represent the information that one party is trying to communicate to another, as opposed to a *message*, which consists of a string of bits sent directly from one processor to another along an edge of the network, whatever its purpose). Hence, each processor receives a special "destination" input on its normal input tape, containing the label of another processor to which it is to send a "missive". It also produces a special "source output" intended to contain the labels of those other processors which it has identified as having sent it a missive, thus demonstrating its having "received" the missive.

We also assume that each processor is provided with a randomly chosen public key-private key pair for a secure public-key cryptosystem (PKC) with security parameter n, as well as the public keys for all other processors. However, if the PKC is too costly to implement, then the costs of encryption alone could well dwarf the costs of even the inefficient "trivial" protocol described above. Hence, we further assume that the PKC is *efficient*—that is, that encryption or decryption of a bit requires time polynomial in $\log(n)$. (It follows that encryptions of a bit are also of length polynomial in $\log(n)$). An alternative informal formulation of this assumption is that our PKC is "exponentially secure", ie., that an adversary without the private key needs exponential time to decrypt a properly encrypted bit with probability better than chance by even an exponentially small amount.

Finally, since many distributed protocols make use of a random assignment of tasks to participating processors to enhance fault-tolerance and security, we will allow that a random string be selected in advance and passed as input to all participating processors. This string can be used in some prespecified way to decide "random" selections of processors for various tasks.

A communication *exchange* will consist of each processor "sending a missive" of some fixed length to the processor entered on its "destination input". (We will refer to the assignment of destination inputs to processors for a particular exchange as its *destination list*.) The content of the missive is not important; for simplicity's sake, we will assume that it consists only of the identity of the sender. What is important is that at the end of the exchange, each processor should be able to output a list of other processors that have sent it a missive.

The exchange is the basic unit of communication by which we will measure the security and efficiency of our protocols. In a realistic setting, of course, we expect traffic to be flowing consistently at some optimal rate; however, it is natural to consider this perennial flow of data to be a resource, and to seek a protocol that optimizes its use. For example, in a truly complete network, where every party has a physical link to every other, it may appear that untraceability can be achieved more or less "for free", simply by having every party send its message to every other party simultaneously whenever it has a message to send. (We will momentarily ignore, for the sake of argument, the cost of processing received messages.) However, the most efficient communication protocols in such a setting would take advantage of the parallel paths available to each node, permitting far greater throughput, in the absence of untraceability concerns, than the redundant method just described. Hence, the cost of the wasted communication sent along most of the links should properly be taken into account.

If we consider communication as a resource in this manner, then we can simplify our measure of the efficiency of a constantly-running protocol by isolating a single iteration of it, and calculating its cost in use of available resources. In our setting, the exchange is just such a single iteration, provided that all parties generate messages at a roughly equal rate, and wish to disguise only the destinations of those messages themselves, not the rate itself. Henceforth, then, we will consider the parties in our network simply to be performing repeated exchanges, and trust that the underlying routing (ie., the implementation of the complete network on the physical topology) is making optimal use of the available communication resources. Thus it suffices for us to analyze the security and efficiency of a single exchange.

Our goal is to devise a protocol that will allow parties to send missives efficiently, while preventing some adversary from learning anything about the list of senders for each receiver. Three adversaries are considered: an *external* adversary is given all the data transmitted on all the connections in the network. A *passive internal* adversary is also given access to the computations of some arbitrary fixed subset of the processors, consisting of some constant fraction $c < 1$ of the total; in particular, the private encryption keys, inputs and random bits of these processors are available to the passive internal adversary. (For example, a group of parties that decide to collude after the fact to analyze previous traffic patterns would effectively be passive adversaries.) Finally, an *active internal* adversary is also allowed to alter the computations of that same subset of the processors during the exchange.

To be considered a solution to the problem of defending against traffic analysis by one of these adversaries, a protocol should meet three conditions, *correctness*, *security* and *efficiency*, defined appropriately for each adversary. For external and passive internal adversaries, a protocol is *correct* if all processors always produce correct, complete source outputs, and *secure* if the distribution on information provided to the adversary is computationally indistinguishable for any pair of destination lists, as long as the information "legitimately" available to the adversary for each list is the same. The amount of information legitimately available to an adversary depends on what we assume about the adversary's power. An external adversary, for example, is legitimately entitled to know the number of missives received by each processor, since that information is, as we have pointed out, difficult in any event to conceal (except in the obvious manner, by generating "dummy traffic"). The passive adversary, on the other hand, is also entitled to know the contents of the source inputs and destination outputs of the processors to which it has access *a priori*.

Defining correctness and security against active internal adversaries is more difficult; for example, "maliciously" controlled processors may simply fail *en masse*, disrupting communication completely. While the problem of making protocols resilient against failure or disruption by participants is an interesting one, and has received considerable attention in the literature, it is beyond the scope of our immediate concern here, which is that information about (intended) communication among parties be concealed. Moreover, we are assuming that the participants communicating in our setting are doing so for a reason, and therefore have an interest in the protocol's (normally) functioning correctly. We hence only demand that a correct protocol against active adversaries be correct (in the sense defined earlier for weaker adversaries) whenever the adversary-controlled processors behave as "normal" (non-adversarial) ones would. (More stringent conditions, such as that an honest processor never register reception of an unsent missive from another processor, are achieveable using techniques such as digital signatures and timestamps; such enhancements are, again, beyond the scope of this discussion.) Similarly, we define a protocol to be secure against active adversaries if it is secure in the sense defined for passive adversaries, regardless of the behaviour of the adversary; however, we will assume for this purpose that the network supports efficient, resilient broadcast. The intuition behind this assumption is that if processors can detect deviations from the protocol as they occur, and notify all other processors of the "fault", then they can effectively put a stop to the exchange before significant traffic information is revealed.

We will also demand an *efficiency* condition: that is, that the number of synchronous steps, and the number of computation steps per synchronous step for each processor, be limited. In particular, we will be concerned with the growth in the number of synchronous steps necessary, and the amount of processing required in any one synchronous step, as a function of n. For example, consider the "trivial" solution mentioned previously, in which each processor simply sends a message of some kind to every other processor. This method clearly achieves both correctness and security, even against active adversaries. However, it requires each processor to handle $O(n)$ messages in one synchronous step. If n is very large, then this solution may be impracticable. Similarly, if all processors send their missives via a single "router" processor, then that processor has to process n messages in a single synchronous step. Trying to "spread out" the communication over multiple synchronous steps would increase the required number of synchronous steps to a possibly unacceptable level. We will define an *efficient* solution to be one that requires a(n expected) number of synchronous steps that is at most a fixed polynomial in $\log(n)$, and that requires the (expected) maximum amount of time spent by any processor in each step to be at most a fixed polynomial in $\log(n)$. (This cost must be independent of the destination list, and therefore does not include the "post-exchange" cost of processing received missives and computing source outputs, since for some destination lists that cost will be $O(n)$ for some processors.)

This definition of cost also does not take into account the cost introduced by implementing a complete network on some unspecified actual topology. For example, in some topologies, such as the ring (discussed in [Ch2]), routing is in general so inefficient that a natural implementation of the "trivial" solution described above is in fact not significantly less efficient than the normal routing algorithms (note also that a ring can also be superimposed on the spanning tree of any connected such network, using each edge twice). Our discussion of solutions will therefore implicitly assume that routing in our network is in general more efficient than can be achieved using such "ring-style" routing. For example, certain well-connected networks, or those which use special hardware (or external resources —say, those which use a small part of a much larger network, such as the telephone system) to speed routing, might suffer substantial losses in efficiency if "ring-style" routing were to be used for the sake of hiding senders' and receivers' identities.

Finally, we have not yet considered how the efficiency of a protocol "scales up" if parties wish to send long missives (as opposed to merely identifying senders). We will call a solution *scalably efficient* if parties can also output messages of length l from each sending party at the end of the exchange, at the cost of only an extra linear factor (in l) in the (expected) processing time for each party in a synchronous step.

Theorem 2.1 *There exists a correct, scalably efficient protocol, secure against external or passive internal adversaries, for implementing a message exchange, given the network model (and cryptographic assumptions) described above.*

Theorem 2.2 *There exists a correct, efficient protocol, secure against active internal adversaries, for implementing a message exchange, given the above network model and cryptographic assumptions, with the additional assumption of an efficient, resilient broadcast facility.*

3 Solutions

3.1 A Solution for External Adversaries; An Interactive Variation

A simple, scalably efficient solution secure against external adversaries is as follows: each processor is assigned the role of a node (or several nodes) in an n-element sorting network ([K]) of depth polynomial in $\log(n)$. Processors then input their missives into the sorting network, sending them to the appropriate node of the first level of the sorting network (encrypted using the public key of that node). The nodes of the sorting network compare the destinations of the missives they receive, sending them on to the next node in the sorting network, after encrypting them using their respective next nodes' public keys. Once all the missives are sorted, they are sent to their destinations, encrypted using the destinations' respective public keys.

This protocol requires only polynomially many (in $\log(n)$) synchronous steps, with each processor handling only a pair of messages per step (except perhaps at the last step, when multiple messages might be delivered to one destination after sorting). Of course, the traffic pattern will appear identical regardless of which processor sent which missive; the final step of the missive delivery will depend only on the number of missives received by each processor.

Against passive internal adversaries, this solution can be modified by replacing each processor/node with a randomly chosen "committee" of k processors, where $k = O(\log^2(n))$. The sorting process is as before, but each sender first uses a (k, k) secret-sharing scheme ([Sh]) to create k shares of its missive. These shares have the property that when combined, they allow the original missive to be reconstructed, but no subset of them reveals any information about the missive. The sender then "delivers" its missive to the committee representing the appropriate "first" node in the sorting network, by first encrypting each share of the missive using the public key of a different member of that committee, then sending the entire "package" of all the shares, thus encrypted, to each member of that first committee.

The committee then performs a "secure multiparty computation" protocol to compute and send on two similar "packages" of encrypted shares, one for each of the two missives received by the committee. Each missive's package consists of k shares of that missive, each encrypted using the public key of a different member of the committee representing the node in the sorting network to which that particular missive should be sent after the two are sorted. Secure multiparty computation protocols (SMCP's) are protocols by which multiple parties can compute the value of a function without each party learning anything about the inputs to the function (apart from that which is revealed by the value of the function itself). SMCP's were introduced in [GMW]; their definition was refined in [MR] and [B]. An SMCP for any polynomial time-computable function is presented in [GMW] based on the assumption of the existence of a one-way function and an oblivious transfer protocol (shown in [Y] to follow from the existence of a trapdoor permutation). A constant-round SMCP for any polynomial time-computable function is given in [BMR].

In the case of the SMCP replacing a node in the sorting network, the inputs and outputs are in the form of encrypted shares, with one share encrypted using the public key of each member of some committee of k parties. Hence, assuming that each committee contains a member to which the adversary has no access (an event that occurs with probability $1 - n^{-\omega(1)}$ if the committees are chosen randomly, and the adversary has access to only a constant fraction of the parties), the passive adversary will be unable to obtain all the shares of any message sent by a committee, and hence will have no information about any of the messages at any point during the sorting. Moreover, the number of bits which must be sent in each (constant-round) SMCP will be some fixed polynomial in the message and committee sizes, which are in turn both bounded by polynomials in $\log(n)$.

It might appear that the same solution should be secure against active adversaries as well; after all, the protocols described in [GMW] and elsewhere include variants which are secure against active adversaries as well as passive ones. However, these variants make extensive use of zero-knowledge interactive proofs ([GMRa]), and in our setting, numerous zero-knowledge interactive proofs (ZKIP's) would have to be performed in parallel, each involving (possibly colluding) adversaries, in the various simultaneously-interacting committees representing the various nodes in the sorting network. Unfortunately, ZKIPs were shown in [GK] not necessarily to preserve their zero-knowledge properties under parallel composition. (Ways to "interleave" parallel ZKIPs so as to preserve their zero-knowledge status are explored in [DDN]; however, whether it suffices to incorporate them into the SMCP protocols in our setting is far from clear.) Hence the security of certain SMCP's against active adversaries cannot simply be exploited in the obvious way as it was in the above case of passive adversaries. This example illustrates one of the subtle effects of viewing communications security in the more realistic setting of multiple simultaneous interactions rather than assuming that individual conversations take place in isolation; parallel, ostensibly independent conversations can conceivably compromise each other's security, particularly if the participants choose to collude to that end.

3.2 Passive Adversaries — A Non-interactive solution

The above protocol, though efficient, is somewhat inelegant in that it makes use of a powerful technique (secure multiparty computation) which is very complex, subtle, and quite difficult to understand, much less implement. Also, it is not scalably efficient, since the communication complexity of the SMCP's described in the literature generally grows superlinearly with the size of the inputs (which in this case are the shares of the messages being routed). Decomposing messages into short segments would solve this problem, but at the cost of expanding the number of rounds, possibly beyond the bounds required by the efficiency condition.

Our scalably efficient solution against passive adversaries is somewhat different in character from the interactive one. In this alternative protocol, each missive is encrypted $\lambda + 1$ times, using the public keys of the destination and each of the processors in a length λ path (specially chosen, as described below), in reverse order. (Each such encryption need require only an additive factor polynomial in $\log(n)$; see [BG].) Hence, if λ is polynomial in $\log(n)$, then the $\lambda + 1$ nested encryptions would only add a length polynomial in $\log(n)$ to each missive.

The crux of the argument for the security of this protocol against passive adversaries will be the fact that as soon as a group of messages passes through a "normal" (non-adversarial) processor (ie., a "mix"), their identification with their previous paths is lost to the adversary, since matching messages with their previous, encrypted versions with probability non-negligibly greater than chance would imply the insecurity of the underlying PKC. Hence the passive adversary cannot determine which message entering a given node "matches up" with a given message leaving that node. From the adversary's point of view, then, each node at which the paths of multiple missives cross can be considered as a "switch", where incoming missives are randomly assigned to outgoing paths. If we can show that the action of these switches over the λ steps is sufficient to disguise the correspondence between senders and receivers of missives, then we have demonstrated that the passive adversary (for whom the "settings" of the switches are effectively random) cannot discern that correspondence.

Ideally, the senders would simply send missives to their receivers along a length-λ path through a randomly chosen sequence of parties. Such length-λ random paths are with overwhelming probability divisible into a sequence of λ' stages (where λ' is still polynomial in $\log(n)$) such that for each stage there exists a matching among the non-adversary-affiliated missives in which each pair of the matching has "shared" a "non-adversarial" party (ie. passed simultaneously through a party not affiliated with the internal adversary) at some point during the stage. The action of a particular stage would therefore be comparable to a "pair-and-swap" operation, in which missives are randomly paired with each other, then the pairs are each exchanged or left unexchanged with probability 1/2. (Since the adversary has no information as to which message from the incoming pair corresponds to which message from the outgoing one, it would be as if these two missives were randomly permuted, from the adversary's point of view.)

If (as seems likely) the action of a(n appropriately short) sequence of such "pair-and-swap" operations results in an approximately random permutation, then the adversary would be unable to distinguish the resulting traffic pattern from that of one which randomly permutes the parties' missives, and the resulting protocol would therefore be secure against passive adversaries. However, such a "rapid mixing" property of the "pair-and-swap" process remains unproven. Instead, we prove a similar but slightly weaker theorem about the "rapid merging" effect of the "pair-and-swap" process on *bit strings* in which half the bits are 0 and half are 1. That is, we show that polylogarithmically many applications of the "pair-and-swap" process on such a bit string results, with overwhelming probability, in an approximately random bit string of that type.

We can exploit this weaker result using a slightly more complicated protocol, as follows: first, we divide the n parties randomly into two equal-sized groups, then we randomly divide those groups in two equally, and so on, so that there are $n/2^i$ ith-level groups, each containing 2^i members, for i ranging from 1 to $\log(n)$. (We assume here that n is a power of 2.) Since the groups are chosen randomly, each pair of groups will with overwhelming probability contain an approximately equal number of parties not affiliated with the adversary (the ratio being no worse than, say, $1/O(\log^2(n))$). For convenience, we will assume that the non-adversary-affiliated populations of the two groups are *exactly* equal; a technical argument takes care of the fact that the equality is only approximate.

In the protocol, then, each sender chooses a path for its missive which is divided into segments corresponding to the levels of division of the parties into groups. In the first segment, the missive is sent through a random length-λ_1 sequence of parties each of which is one of the two in its first-level group; the next segment consists of a path corresponding to a random length-λ_2 sequence of parties in its next-higher-level group. The third segment then corresponds to a length-λ_3 sequence of parties chosen randomly from those in its third-level group, and so on, up to the top-level group. (Note that each party in the sequence at each level for a given missive is chosen randomly and independently; hence missives may well pass through the same party several times at one level.) The λ_i's will all be polynomial in $\log(n)$ and i; and since i is at most $\log(n)$, the resulting path lengths will still be polynomial in $\log(n)$.

After the first "level" of this protocol, the order of each pair of (two paired non-adversary-affiliated) missives will be indistinguishable from a random order, from the adversary's perspective, since they will with overwhelming probability pass through the same mix at some point. At each subsequent level in the process, moreover, a sequence of matchings of the (non-adversary-affiliated) missives can be found which, from the adversary's perspective, produces the effect of a sequence of "pair-and-swap" operations on the missives in the two paired groups. Hence, due to the "rapid merging" property of the "pair-and-swap" operation, each pair of equal-size groups of (non-adversary-affiliated) missives (each group already being in an order indistinguishable to the adversary from a random one, due to the actions of previous levels of the protocol) will at this "level" merge into a single group of missives, whose order the adversary cannot distinguish from a random one. Thus, after $\log(n)$ levels of mergings, the adversary will be unable to distinguish the permutation on the (non-adversary-affiliated) missives from a random one.

3.3 Proving Rapid Mixing

We can restate the required "pair and swap" theorem as follows: given a bit string containing half zeroes and half ones, we perform a sequence of "pair-and-swap" operations which consists of randomly pairing bit positions, and then exchanging each pair independently with probability 1/2. We would like to show that after a number of such operations which is polynomial in $\log(n)$, the probability is overwhelming that, *even given the sequence of pairings* (to which the passive adversary in our setting would have at least partial access, knowing roughly which messages are being paired with which), the resulting bit string is statistically close to a random one (ie., the sum of the distances of the bit strings' probabilities from their probabilities under a uniform distribution is $n^{-\omega(1)}$).

Theorem 3.1 *Let B be the set of n-bit strings containing exactly $n/2$ ones. There is some polynomial q such that given $b \in B$, a sequence of $q(\log(n))$ "pair and swap" operations on b (in which all the bit positions in b are randomly paired, and the bits in each pair exchanged with probability $1/2$) produce, with probability $1 - n^{-\omega(1)}$ (over choices of pairings in the "pair and swap"), a statistically random element of B (ie., an element of B chosen accoding to a distribution whose statistical distance from the uniform distribution, given by $\sum_{b \in B} |Pr[b] - (1/|B|)|$, is $n^{-\omega(1)}$).*

The proof of this theorem is quite involved. First, we must show that given an initial distribution D on B, the distribution D' on bit strings generated by the "pair-and-swap" process performed on a bit-string chosen according to D is related to one generated by taking the exclusive-or of that string with the output of a particular "slightly random source" ([SV]). This allows us to use a strengthened version of a theorem due to Vazirani and Vazirani ([VV]) to show that polynomially many steps (in $\log(n)$) suffice to reduce the distance of any such D from the uniform to a polynomial factor (in n).

Definition 3.1 *([SV]) A slightly random source is a source sequentially producing random bits, each with a bias (towards 0 or 1) chosen by an arbitrary "adversary" whose only restriction is that the bias never fall outside the range $(\delta, 1-\delta)$ for some fixed constant δ.*

Lemma 3.1 *(based on [VV]) Let D be a distribution on $\{0,1\}^n$, with $Pr_{b' \leftarrow D}[b' = b] = p_b$ for n-bit strings b; let s be an n-bit string produced by a slightly random source S whose biases are bounded away from 0 and 1 by the constant δ. Then $E_{s \leftarrow S}(\sum_b p_b p_{b \oplus s}) \leq (2^n|D|)^{\delta/(\delta-1)}|D|$, where $|D|$, the squared 2-norm of D, is given by $\sum_b p_b^2$.*

We thus obtain

Lemma 3.2 *For any distribution D assigning probability p_b to each member of $\{b \in \{0,1\}^n$ with exactly $n/2$ ones$\}$, and $\Pi = \{$ pairings of bit positions of $b\}$, let D' be the distribution generated by selecting pairing $\pi \in_R \Pi$ and bit string b with probability p_b (ie. according to D), and exchanging each pair of bits of b (as defined by π) with probability $1/2$. Then $E_{\pi \in_R \Pi}(|D'|) \leq O(\sqrt{n})((2^n|D|)^{\delta/(\delta-1)}|D| + 2^{-\epsilon n}|D|)$, for some constants ϵ, δ (where $|\cdot|$ is the square of the 2-norm, $0 < \delta < 1$, $\epsilon > 0$).*

Lemma 3.2 guarantees that polynomially many (in $\log(n)$) "pair and swap" operations produce a distribution whose statistical distance from the uniform is at most a polynomial factor (in n). To achieve true statistical closeness to a uniform distribution, we use use a "canonical-path" argument similar to those used by Sinclair and Jerrum ([SJ]) to show that the underlying graph representing the "pair and swap" process is an expander. This fact, together with a theorem due to Mihail ([Mi]) concerning the rapidity of convergence of random walks on expanders, allows us to show that the aforementioned polynomial factor is reduced to one which is $O(n^{-\omega(1)})$, after a number of steps which is polynomial in $\log(n)$.

3.4 Active Adversaries

To achieve security against active adversaries, we can combine the idea of "committees" with the above non-interactive solution for passive adversaries in the following manner: random committees of $O(\log^2(n))$ processors are formed to represent each of the original nodes in the network. One member of each committee is designated the "effective node"; the rest of the committee only checks the work of this processor. When a node receives a message from another node, each member of the committee representing the receiving node receives a copy of the message from each member of the committee representing the sending node, broadcasting a "fault detected" message if there are any inconsistencies among received versions. The effective node processor then decides what new messages are to be sent on behalf of the node, and sends copies of each new message to the rest of the committee members, along with a non-interactive zero-knowledge proof that the new message was generated in accordance with the protocol.

Committee members check the non-interactive zero-knowledge proofs, and if they are valid, relay a copy of the outgoing messages to each member of the receiving node's committee; otherwise, they broadcast a "fault detected" message. (Similarly, they can output a "fault detected" message if an incoming message is not followed by an appropriately validated outgoing message.) Clearly, as long as each committee has at least one member not allied with the adversary, there will (with overwhelming probability) be at least one "fault detected" whenever a processor breaks with the protocol, since the properties of non-interactive zero-knowledge proofs guarantee that processors will be incapable of producing non-interactive zero-knowledge proofs of the correctness of their behaviour (with non-negligible probability) unless they are in fact behaving correctly.

4 Conclusion

A generalization of theorem 3.1 to permutations and/or to more general processes (for example, some broader class of "comparator shuffles", in which the primitive operation is the random permutation of pairs of objects) would provide simple, efficient parallel/distributed algorithms for generating random permutations, and might also lead to proofs of rapid mixing for a number of natural Markov processes, with applications to approximate counting and uniform generation algorithms in a variety of contexts (as well as greatly simplifying the protocols presented here for untraceable communication).

In a cryptographic context, a scalably efficient solution against active adversaries has yet to be found, a key obstacle being the lack of efficient enough constructions for such primitives as non-interactive zero-knowledge proof and secure multiparty computation. One possible approach may be to use non-malleable cryptosystems ([DDN]), a suitably efficient construction of which can be found in [Si]. The broader problem remains of integrating these solutions into a coherent model that incorporates the many disparate notions of security that have been explored in the literature.

References

[B] D. Beaver, *Foundations of Secure Interactive Computing*, Advances in Cryptology: Proc. CRYPTO '91.

[BFM] M. Blum, P. Feldman, and S. Micali, *Non-Interactive Zero Knowledge and its Applications*, Proc. 20th ACM Symposium on Theory of Computing (1988), pp. 103–112.

[BG] M. Blum and S. Goldwasser, *An Efficient Probabilistic Public-Key Encryption Scheme Which Hides All Partial Information*, Advances in Cryptology: Proc. CRYPTO '84, pp. 289–302.

[BMR] D. Beaver, S. Micali and P. Rogaway, *The Round Complexity of Secure Protocols*, Proc. 22nd ACM Symposium on Theory of Computing (1990), pp. 503–513.

[Ch1] D. Chaum, *Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms*, CACM vol. 24, no. 2 (1981), p. 84-88.

[Ch2] D. Chaum, *The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability*, Journal of Cryptology vol. 1, no. 1 (1988), pp. 65-75.

[Ch3] D. Chaum, *Achieving Electronic Privacy*, Scientific American, vol. 267, no. 2 (1992), pp. 96–101.

[DDN] D. Dolev, C. Dwork and M. Naor, *Non-Malleable Cryptography*, proc. 23rd ACM Symposium on Theory of Computing (1991), pp. 542-552.

[Go] O. Goldreich, *Towards a Theory of Software Protection and Simulation by Oblivious RAMs*, proc. 19th ACM Symposium on Theory of Computing 1987, pp. 182-194.

[GK] O. Goldreich and H. Krawczyk, *On the Composition of Zero-Knowledge Proof Systems*, Proc. ICALP '90.

[GMRa] S. Goldwasser, S. Micali and C. Rackoff, *The Knowledge Complexity of Interactive Proof Systems*, Proc. 17th ACM Symp. on Theory of Computing (1985), pp. 291–304.

[GMW] O. Goldreich, S. Micali and A Wigderson, *How To Play Any Mental Game*, proc. 19th ACM Symposium on Theory of Computing (1987), pp. 218-229.

[K] D. Knuth, *The Art of Computer Programming*, 2nd ed., Addison-Wesley Pub. Co., Reading, Mass., 1975.

[Mi] M. Mihail, *Conductance and Convergence of Markov Chains: A Combinatorial Treatment of Expanders*, Proc. 30th IEEE Symposium on Foundations of Computer Science (1989), pp. 526-531.

[MR] S. Micali and P. Rogaway, *Secure Computation*, Advances in Cryptology: Proc. CRYPTO '91.

[Os] R. Ostrovsky, *Efficient Computation on Oblivious RAMs*, proc. 31st IEEE Symp. on Foundations of Computer Science (1990), pp. 514-523.

[P] A. Pfitzmann, *How to Implement ISDNs Without User Observability—Some Remarks*, TR 14/85, Fakultät für Informatik, Universität Karlsruhe, 1985.

[PPW] A. Pfitzmann, B. Pfitzmann and M. Waidner, *ISDN-MIXes—Untraceable Communication with Very Small Bandwidth Overhead*, Proc. Kommunikation in verteilten Systemen (1991), pp. 451–463.

[Sh] A. Shamir, *How to Share a Secret*, CACM, vol. 22, no. 11 (1979), pp. 612-613.

[Si] D. Simon, *On Defining and Achieving Cryptographic Security in a Multiparty Network*, PhD thesis, University of Toronto, 1992.

[SJ] A. Sinclair and M.R. Jerrum, *Approximate Counting, Uniform Generation, and Rapidly Mixing Markov Chains*, Information and Computation, vo. 82, no. 1 (1990), pp. 93-133.

[SV] M. Santha and U.V. Vazirani, *Generating Quasi-Random Sequences from Slightly-Random Sources*, Proc. 25th ACM Symp. on Theory of Computing (1984), pp. 434–440.

[VV] U.V. Vazirani and V.V. Vazirani, *Random Polynomial Time is Equal to Slightly-Random Polynomial Time*, Proc. 26th IEEE Symp. on Foundations of Computer Science (1985), pp. 417–428.

[Y] A.C. Yao, *How to Generate and Exchange Secrets*, Proc. 27th IEEE Symp. on Foundations of Computer Science (1986), pp. 162–167.