Perfectly Secure Message Transmission

Danny Dolev[*] Cynthia Dwork[†] Orli Waarts[‡] Moti Yung[§]

Abstract

We study the problem of perfectly secure communication in a general network in which processors and communication lines may be faulty. Lower bounds are obtained on the connectivity required for successful secure communication. Efficient algorithms are obtained that operate with this connectivity and rely on no complexity theoretic assumptions. These are the first algorithms for secure communication in a general network to simultaneously achieve the three goals of perfect secrecy, perfect resiliency, and worst case time linear in the diameter of the network.

1 Introduction

Recent advances in fiber optics make realizable the construction of networks with immense bandwidth. As more and more personal and business communication will take place over these systems, issues of correctness and privacy become increasingly important. In this paper, we solve the problem of perfectly secure message transmission in communication networks, without complexity theoretic assumptions and with perfect correctness, for processor and edge faults alike. Our approach is to abstract away the network entirely and concentrate on solving the Secret Message Transmission problem for a single pair of processors we call *Sender* and *Receiver.* In the *Secret Message Transmission (SMT)* problem two synchronized nonfaulty processors, Sender and Receiver, are connected by some number n of wires. We may think of these wires as a collection of vertex-disjoint paths between Sender and Receiver in the underlying network; each path corresponds to a wire. The Sender has a secret message m, drawn from a finite set Q of values. There are two parameters, σ (for secrecy) and ρ (for resiliency). The problem is for the Sender to convey m to the Receiver while satisfying:

Perfect Secrecy: For all sets L of at most σ wires, no listening adversary $\mathcal{A}_L$, listening to all the wires of L, learns anything about m (secrecy must be information theoretic).

Perfect Resiliency: For all sets D of at most ρ wires (possibly, but not necessarily, disjoint from L), Receiver correctly learns m, regardless of the disrupting adversary $\mathcal{A}_D$ controlling and coordinating the behavior of the wires in D.

Since each wire corresponds to a path in the underlying network, a compromised wire in Secret Message Transmission corresponds to a compromised processor or edge on the corresponding network path. Thus, connectivity bounds for SMT yield connectivity bounds in the network as a function of the number of faulty nodes or edges to be tolerated.

Our protocol for secure transmission in general networks is the first to simultaneously achieve the three goals of perfect secrecy, perfect resiliency, and worst case time linear in the diameter of the network (the constant is at most 3). This contrasts with the similarly fast protocol of Rabin and Ben-Or, based on the idea of "check vectors," which has unconditional secrecy but has small probability of error [12]. Ben-Or, Goldwasser, and Wigderson showed that every function of p inputs can be efficiently computed by a complete network of p processors even in the presence of $t < p/3$ Byzantine faults so that no set of t faulty processors gets any information other than the function value [1]. Using our protocol for secret message transmission we can immediately extend these results to any p processor network of connectivity $2t + 1$, at no cost in secrecy or correctness. The analogous result obtained by Rabin and Ben-Or in [12] for general networks suffers a small probability of error. In this

[*]IBM Almaden Research Center
[†]IBM Almaden Research Center
[‡]Stanford University; Supported in part by contract ONR N00014-88-K-0166.
[§]IBM T.J. Watson Research Center

low probability case, because the entire computation can go awry, the privacy of correct processors is not guaranteed, even though messages sent between correct processors enjoy perfect secrecy. Our solution does not suffer from this weakness, and pays no price in time.

The bounds on connectivity needed for (σ, ρ)-SMT vary according to whether or not the solution must be *1-way*, in that information flows only from Sender to Receiver, or *2-way*, where Sender and Receiver "converse." The bounds are also strongly affected by the extent to which $\mathcal{A}_D$ can communicate to $\mathcal{A}_L$ its view of the communication on the wires in D. One possibility is that the two adversaries share a "back channel" allowing them explicitly to communicate. A more interesting case is when there is no such channel. Here $\mathcal{A}_D$ can communicate information to $\mathcal{A}_L$ if the sets D and L intersect (by placing messages on the shared wires). More subtly, even if the two sets are disjoint, $\mathcal{A}_D$ may be able to transmit information to $\mathcal{A}_L$ by disrupting the protocol so as to elicit certain behavior on the part of Sender or Receiver that $\mathcal{A}_L$ can recognize. One situation in which $\mathcal{A}_D$ clearly cannot communicate useful information to $\mathcal{A}_L$ is, informally, when it disrupts obliviously, independent of the information on the wires in D. As an example, we propose the relatively weak fault model in which communication is disrupted only by random noise.

The case in which $\mathcal{A}_D$ and $\mathcal{A}_L$ are constrained so that $D \subseteq L$ or $L \subseteq D$ is an important one, and in this case we say the ***containment assumption*** holds. In this case there is effectively one adversary. This is the worst-case assumption made in previous papers treating secrecy and resiliency simultaneously [1, 2, 3, 8, 12]. Generally, we assume $\sigma \geq \rho$ and derive our bounds for this case. Our lower and upper bounds match under the containment assumption; moreover, under containment, if $\sigma \geq \rho$ then the bounds are independent of the extent of communication between the adversaries. Under the containment assumption, there is a solution to the 1-way Secret Message Transmission problem if and only if n, the number of wires connecting Sender and Receiver, satisfies $n \geq \sigma + 2\rho + 1$. The solution requires computation and message length polynomial in n. However, if communication is 2-way, in that Sender and Receiver converse, then $n \geq \max\{\sigma + \rho + 1, 2\rho + 1\}$ wires are necessary and sufficient (the latter term arises even in the case $\sigma = 0$, that is, we require correctness but no secrecy). A ***phase*** is a send from Sender to Receiver or ***vice versa***. Surprisingly, the 2-way protocol requires only three phases.

For this value of n we have even obtained a ***2-phase*** protocol (beginning with a transmission from Receiver to Sender), but, unlike our 1-phase and 3-phase solutions, the computation and communication costs of the two-phase solution are not polynomial in n. Our 3-phase solution uses two new techniques. The first is a simple fault detection technique. The second is a method of parallelizing our first technique, permitting us to collapse loop iterations in a $\Theta(\rho)$-phase algorithm to obtain the 3-phase algorithm. Both of these have already been applied to variants of secret sharing [4].

If the two adversaries can communicate explicitly during the execution of the protocol, say, through some auxiliary "back channel," but the containment assumption does not hold, then the (lower and upper) bounds on n increase by exactly ρ for both the 1-way and 2-way cases. This is because when $\mathcal{A}_D$ can communicate with $\mathcal{A}_L$ it is as if there are $\sigma + \rho$ listeners, of which at most ρ are disruptors. This is the containment assumption with secrecy parameter $\sigma + \rho$ and resiliency parameter ρ, and the bounds increase accordingly over the case with only σ listeners.

Even if the two adversaries cannot communicate through a back channel, it may be possible for the disrupting adversary to elicit certain behavior from Sender or Receiver that communicates some extra information to the listening adversary. In fact, at least an additional $\rho - 1$ wires must be added for both the 1-way and 2-way cases. These last results are tight for the 1-way case and leave a gap of a single wire in the 2-way case. However, we also show that in this case any ***3-phase*** algorithm requires exactly ρ additional wires, even if the disruptors and listeners cannot move between phases. This bound is tight, and our algorithm permits these sets to move. All our protocols tolerate adversaries of unlimited computational power.

1-way SMT has an interesting relation to Verifiable Secret Sharing (VSS), a problem first defined by Chor, Goldwasser, Micali, and Awerbuch [3]. VSS plays a central rôle in implementing a global coin [7], as well as in the more general results of [1, 2, 12][1]. Ben-Or, Goldwasser, and Wigderson remark that secure computation is impossible with $2t + 1$ processors, even in the presence of a broadcast channel [1]. We prove that even the more fundamental task of Verifiable Secret Sharing cannot be achieved in this

[1] Roughly speaking, t-VSS permits a (possibly faulty) dealer to commit to a secret in such a way that the secret can later be uniquely reconstructed despite the interference of up to t faulty processors, possibly including the dealer. Moreover, if the dealer is correct then the faulty processors cannot learn any information about the secret until the correct processors execute a reconstruction protocol.

model.[2] Our approach is to reduce a weakened version of 1-way SMT to VSS so that each processor in the VSS protocol corresponds to a wire in the SMT protocol. We then prove a lower bound on connectivity of $3t+1$ for this weakened version of SMT. Interestingly, our lower bound of $3t+1$ also applies to a weak version of VSS called Unverified Secret Sharing (t-USS). This is essentially VSS without Verification. Thus, the cost of Verifiable Secret Sharing comes from the conflicting requirements of secrecy and reconstructability of correctly distributed secrets, rather than from the ability to verify that the secret was correctly dealt out.

Rabin and Lehmann showed that in a distributed environment there exist problems with randomized solutions but with no deterministic solution [11]. There exist error-free and small-error solutions to t-USS requiring $3t+1$ and $2t+1$ processors, respectively [1, 12]. The lower bound of $3t+1$ processors for error-free t-USS yields a new kind of separation result: within the class of problems admitting no deterministic solution, the cost of an error-free solution may necessarily significantly exceed the cost of a solution with even very small probability of error. In a certain model it is therefore possible to separate error-free randomized computation from randomized computation with error.

The full paper also explores the problem of secure communication in graphs of bounded degree. Techniques of Dwork, Peleg, Pippenger, and Upfal [5] for (nonsecret) communication in size n networks of bounded degree can be extended to show that for substantial (as a function of n) σ and ρ, no matter which σ nodes or edges are chosen by $\mathcal{A}_L$, and no matter which ρ nodes or edges are chosen by $\mathcal{A}_D$, there is a large set of nodes that can communicate secretly and correctly among themselves, even though the network is of bounded degree.

The rest of the paper is organized as follows. Section 2 describes our adversary models. Section 3 contains the definition of the Secret Message Transmission problem. Section 4 contains our 3-phase solution to SMT. Additional results for the containment case appear in Section 5. Our results about Verifiable Secret Sharing and the separation result for problems with no deterministic solution appear in Section 6. A complete version of this paper is available as IBM RJ 7496 (5/23/90).

[2]The lower bound for t-VSS with a broadcast channel was obtained independently by Rabin and Ben-Or, and appears without proof in [12]. An informal argument is also given in [2].

2 Adversaries

An *adversary* is an algorithm that takes as input transmissions on certain wires, random bits, and the phase number, and produces a choice of additional wires together with either (faulty) traffic on the chosen wires, in the case of the *disrupting* adversary $\mathcal{A}_D$, or a guess of the message being transmitted, in the case of the *listening* adversary, $\mathcal{A}_L$. A wire tapped or under the control of an adversary is said to be *compromised.*

For our algorithms, our adversaries may be *adaptive*, in the sense that information (communication traffic) obtained from a set of compromised wires can affect the choice of the next wire to be compromised. Our lower bounds hold even if the adversaries are not adaptive.

Our algorithms have a special form: the first phase uses a low quality "secret" channel, while all subsequent phases use a perfect "public" channel. Without going into detail here of how we implement these different types of channels, we point out that the issue of choosing wires to compromise in subsequent phases as a function of traffic in the first phase is moot for these algorithms. Similarly moot is the issue of whether $\mathcal{A}_D$ even exists after the first phase, since by definition it cannot interfere with the perfect public channel. Of course, $\mathcal{A}_L$ may use all the information it has gleaned over the entire execution of the protocol for making its guess as to which message is transmitted. The lower bounds hold even for static adversaries that choose which wires to compromise before execution begins. We therefore assume the sets D and L of disrupting and listening wires are chosen by the end of the first phase.

In this paper, we assume in general that $\mathcal{A}_L$ and $\mathcal{A}_D$ work together to defeat the algorithm. If the adversaries can communicate explicitly during execution of the algorithm, say, through some "back channel," then we simply say that *$\mathcal{A}_L$ and $\mathcal{A}_D$ communicate.* Here, a *back channel* is some channel other than the wires connecting Sender and Receiver. In this case, for example, the adversaries might converse while choosing which wires to compromise. If there is no "back channel" we say *$\mathcal{A}_D$ and $\mathcal{A}_L$ do not communicate.* Even in this case, some communication is possible. For example, if the sets D and L intersect, then $\mathcal{A}_D$ can convey information to $\mathcal{A}_L$ by putting this information onto the shared wire(s) or disrupting communication on a shared wire. Even if D and L are disjoint, the protocol may require Sender and Receiver to send over wires in L information reflecting the choice of D. This too could be meaningful to $\mathcal{A}_L$.

We also consider the special case in which $\mathcal{A}_D$ behaves obliviously, choosing D, communicating with $\mathcal{A}_L$, and disrupting communication along the wires in D, without regard to the information placed along these wires by Sender and Receiver. Such an adversary can model the special case in which disruption is due only to random noise. Clearly, an oblivious $\mathcal{A}_D$ cannot give to $\mathcal{A}_L$ any information about the transmissions of Sender and Receiver not already available to $\mathcal{A}_L$. Not surprisingly, we obtain better upper bounds against this weaker adversary than in the non-oblivious case.

3 Definitions

Sender and Receiver are modeled by communicating probabilstic Turing Machines that communicate through the n wires connecting them. Randomization is modeled by coin flipping (bounded branching).

Throughout, our messages m are drawn from a finite field Q of prime cardinality at least n, where n is always the number of wires between Sender and Receiver in Secret Message Transmission or the number of participants in a secret sharing protocol, whichever is appropriate to the context. We let Π denote the underlying probability distribution on Q.

We use the notation $[k]$ to denote the set of natural numbers less than or equal to k. Note that $0 \notin [k]$. We let $(k) = \{0\} \cup [k]$. For any set S, we let S^i denote the set of j-subsets of S where $0 \leq j \leq i$.

For any alphabet Σ, for any vectors $W, V \in \Sigma^n$, the distance between W and V, denoted $dist(W, V)$, is the number of components on which the two vectors differ.

Fix any secret message transmission protocol, P, and let $\mathcal{A}_L$ be a listening adversary. Intuitively, we require that for all messages m, m' and for all disrupting adversaries $\mathcal{A}_D$, the probability distribution on $\mathcal{A}_L$'s view, given that the message transmitted is m and the disrupting adversary is $\mathcal{A}_D$, is identical to the probability distribution on $\mathcal{A}_L$'s view, given that the message transmitted is m' and the adversary is still $\mathcal{A}_D$. Here, the probability space is the space of all coin tosses of $\mathcal{A}_L$, $\mathcal{A}_D$, Sender, and Receiver, and the view, intuitively, is everything seen by $\mathcal{A}_L$.

More precisely, the *View* of a listening adversary $\mathcal{A}_L$ at any point in the execution of the protocol consists of (1) the algorithms $\mathcal{A}_L$ and $\mathcal{A}_D$, and the protocol P; (2) the random choices that $\mathcal{A}_L$ has made so far; (3) the "back channel" messages received up to this point, if any (and if there is a back channel); (4) for each wire ℓ in the subset of L chosen so far, conversations between Sender and Receiver over ℓ from the time the wire was compromised until this point; (5) for each wire w in the subset of $L \cap D$ chosen so far, the changes by $\mathcal{A}_D$ to conversations over w from the time w was compromised until this point. Sometimes we combine the last two items in the view, calling the combination the ***traffic*** over the wires in L.

None of our lower bound proofs use the assumption that $\mathcal{A}_L$ sees both the original data placed on wires in $L \cap D$ by Sender and Receiver, as well as the changes $\mathcal{A}_D$ makes to these wires. Some proofs use the ability of $\mathcal{A}_L$ to detect that $\mathcal{A}_D$ has cut off communication on a certain wire. However, our algorithms work even if $\mathcal{A}_L$ has access to all the traffic over the wires in L.

For every message $m \in Q$, any pair of adversaries $\mathcal{A}_L$, $\mathcal{A}_D$, and any protocol P for SMT, let $\hat{\Pi}(\mathcal{A}_L, m, \mathcal{A}_D, P)$ denote the probability distribution, on the views of $\mathcal{A}_L$ at the end of the executions of P when the message sent is m and the disrupting adversary is $\mathcal{A}_D$. The probability distribution is taken over the coin tosses of $\mathcal{A}_D$, $\mathcal{A}_L$, Sender, and Receiver.

Definition: (σ, ρ)-Secret Message Transmission ((σ, ρ)-SMT) The Sender begins with a message m drawn from an arbitrary probability distribution Π on Q. For every $\mathcal{A}_L$, $\mathcal{A}_D$, compromising at most σ and ρ wires, respectively, we require:

> **Secrecy**: $\forall m' \in Q \quad \hat{\Pi}(\mathcal{A}_L, m, \mathcal{A}_D, P) = \hat{\Pi}(\mathcal{A}_L, m', \mathcal{A}_D, P)$.
>
> **Resiliency**: Receiver correctly learns m.

In particular, the secrecy requirement implies that at any point in the execution $\mathcal{A}_L$ has absolutely no information about which message is being transmitted. It follows that the choice of L is independent of the message being transmitted, as is the probability distribution on conversations over wires in L.

A solution to *1-way* (σ, ρ)-SMT runs in exactly one synchronous phase. A solution to *2-way* (σ, ρ)-SMT is a solution to (σ, ρ)-SMT of two or more phases. We adopt the convention that if $\sigma = 0$ then there is no secrecy requirement, and if $\rho = 0$ then there is no resiliency requirement. If $\mathcal{A}_D$ and $\mathcal{A}_L$ are constrained so that $D \subseteq L$ or $L \subseteq D$ then we say the ***containment assumption*** holds. Unless otherwise noted, we assume $\sigma \geq \rho$, and in this case the containment assumption says that $D \subseteq L$. All our results for the containment case are independent of the degree of communication between $\mathcal{A}_L$ and $\mathcal{A}_D$ when $\sigma \geq \rho$. Under the containment assumption the secrecy condition above is equivalent to the following condition:

> **Secrecy Under Containment** $\forall m, m' \in Q$, $\forall \mathcal{A}_D$, $\mathcal{A}_L$, $\forall L \in (n-1)^\sigma$ compromised by $\mathcal{A}_L$,

$\forall D \in L^\rho$ compromised by $\mathcal{A}_D$, for all possible traffic T_L over wires in L, the probability that T_L occurs, given that the message transmitted is m and given $\mathcal{A}_D$ and $\mathcal{A}_L$, is equal to the probability that T_L occurs, given that the message is m' and given $\mathcal{A}_D$ and $\mathcal{A}_L$. The probability space is the set of coin tosses of $\mathcal{A}_D$, $\mathcal{A}_L$, Sender, and Receiver.

Note that for any fixed $\mathcal{A}_L$, $\mathcal{A}_D$ pair, the probability that $\mathcal{A}_L$ will compromise wire 0 is independent of the secret message. Thus, the probability that $\mathcal{A}_L$ chooses any particular L not containing 0 is independent of the message.

As described in the Introduction, we will study a weakened form of 1-way SMT (under containment) in which there is no secrecy requirement if $\mathcal{A}_L$ compromises wire 0. We call this *weakened 1-way SMT.* Specifically, we weaken the above definition to read "$\forall m, m', \mathcal{A}_D, \mathcal{A}_L, L \in (n-1)^\sigma, D \in L^\rho$, if $0 \notin L$, then for all possible traffic T_L ..."

4 The Main Algorithm

In this section we present our 3-phase protocol for 2-way (σ,ρ)-SMT. Let $\tau = \max\{\sigma,\rho\}$. The protocol requires connectivity $n \geq \tau + \rho + 1$ under the containment assumption or in the case $\mathcal{A}_D$ is oblivious. We prove in Theorem 5.2 that this is optimal. Extensions to the non-containment case appear in the full paper. Communication and computation costs are polynomial in σ and ρ.

We will develop the protocol in two stages, beginning with a slow algorithm and modifying this to obtain Algorithm FastSMT. Throughout this section we take the field Q to be Z_q, where q is a prime greater than the connectivity n. Let $T = (t_1, t_2, \ldots t_n)$, where $t_i \in Q$, $1 \leq i \leq n$. If the points (i, t_i) can be interpolated by a polynomial of degree d we say simply that *T can be interpolated by a polynomial of degree d.*

Since Sender and Receiver are $n \geq 2\rho + 1$ connected, they are essentially connected by a fault-free public channel. To send a message x over this public channel, Sender can simply send x on every wire, that is, x is replicated $n \geq 2\rho + 1$ times, and at most ρ of these copies will be destroyed or modified. Thus Receiver can simply see which message appears at least $\rho + 1$ times, and that is the message that Sender sent. Similarly, Receiver can send things to Sender in a fault-free, but public, fashion.

The slow protocol works as follows. Let $m \in Q$ be the secret message Sender wishes to send to Receiver. First, Sender chooses uniformly at random a pad $p \in Q$; p bears no relation to m. Sender *attempts* secret transmission of p. If secret transmission of p is successful, Sender will send $Z = p + m$ to Receiver over the "public channel". In this case Receiver computes $m = Z - p$ (all arithmetic is done in the field Q). If secret transmission of p fails, then Sender and Receiver will use the public channel to detect at least one previously undetected faulty wire, and the entire protocol is repeated but without the detected faulty wires. During the error detection the secrecy of the pad p is lost; however, since p was chosen independently of m this yields no information about m.

There are two drawbacks to this general approach. First, Sender may have to attempt to transmit up to $\rho + 1$ times. Second, the faulty wires D cannot change between phases. Our three phase solution overcomes both of these drawbacks.

To obtain our error-free 3-phase algorithm we first "strengthen" the random pad by sending, in addition to the shares of a random polynomial, some additional "checking" information. This technique has appeared several times in the literature in the context of verifiable secret sharing (see, *e.g.*, [1, 2, 6, 7, 12]). After describing the stronger pads, we show that if sufficiently many ($\rho n + 1$) strong pads are sent then either at least one *succeeds* (is understood by Receiver) or it is possible for Receiver to choose a set of ρn pads to return to Sender such that all the faulty shares of the one retained pad belong to wires whose faultiness will be detected by Sender when it examines the returned pads. Sender informs Receiver of the faulty wires over the perfect public channel. Receiver removes the shares of the retained pad received on the wires identified as faulty, and the remaining shares are interpolated.

To send a random pad to Receiver, Sender chooses a random polynomial $f(x) \in Q(x)$ of degree τ and sets $p = f(0)$. For each i, $1 \leq i \leq n$, we call $f(i)$ a *principal share* of the pad. For each $1 \leq i \leq n$ (recall $n = \tau + \rho + 1$), Sender chooses an additional random degree τ polynomial $h_i(x) \in Q(x)$ satisfying $h_i(0) = f(i)$. Sender sends on wire i the entire polynomial $h_i(\cdot)$ together with a vector $C_i = (c_{1i}, c_{2i}, \ldots, c_{ni})$ of *checking pieces* satisfying, for all $1 \leq i, j \leq n$, $c_{ji} = h_j(i)$. (To send $h_i(\cdot)$ Sender need only send the $\tau + 1$ coefficients of h_i.)

Throughout this discussion, we let h_i, C_i denote the information placed by Sender on wire i, and we let g_i, D_i denote the (possibly corrupted) information received by Receiver on wire i. Consider attempted transmission of a single strong pad. Let T be the received information. If wire i is correct then $g_i = h_i$ and $D_i = C_i$ (this just says that if wire i is correct then what is received on wire i is the same as what is sent on wire i). Thus, if i and j are both correct wires,

```
FastSMT($W, \tau, Q$, private to $S : m$)
PHASES 1 and 2:
Sender: Send $n\rho + 1$ strong pads $P_1, P_2, \ldots, P_{n\rho+1}$.
Receiver: For $1 \le i \le n\rho+1$, let $T_i$ be received in the
    attempted transmission of $P_i$. Ignoring those wires
    thrown out, if any $T_a$ succeeds then compute $P_a$ from
    $T_a$ and publicly send "a,OK" to Sender. Otherwise,
    find an $i$ such that
        $\{\text{conflicts of } T_i\} \subseteq \cup_{j \neq i}\{\text{conflicts of } T_j\}$.
    Publicly send "i" and all $T_j, j \neq i$, back to Sender.
PHASE 3:
Sender: If "a,OK" received over the public channel
    in PHASE 2, then send $Z = P_a + m$ to Receiver
    over the public channel. Else perform error detection
    on all $T_j$ received from Receiver and publicly send
    detected faults and $Z = P_i + m$ to Receiver, where
    "i" was received from Receiver in PHASE 2.
Receiver: if "a, OK" sent to Sender in PHASE 2,
    then compute $m = Z - P_a$. Else correct the retained
    $T_i$ to obtain $P_i$; compute $m = Z - P_i$.
End of FastSMT
```

Figure 1: Algorithm FastSMT

then $d_{ji} = g_j(i)$ (because: i correct implies $d_{ji} = c_{ji}$; j correct implies $g_j = h_j$; and by construction $c_{ji} = h_j(i)$).

If $d_{ji} \neq g_j(i)$ we say the unordered pair (i, j) is a *conflict of* T. Clearly in case of a conflict (i, j) at least one of i and j is faulty.

When Sender attempts to send a strong pad to Receiver, Receiver "throws out" (ignores) all wires j carrying syntactically incorrect messages. In particular, if g_j is not a polynomial of degree τ, then Receiver throws out wire j.

A strong pad is said to *fail* if for all wires i not thrown out, the points $(i, g_i(0))$ cannot be interpolated by a degree τ polynomial. Otherwise it *succeeds*, regardless of conflicts.

Algorithm FastSMT appears in Figure 1. The parameters are the set of wires W over which Sender and Receiver communicate, a secrecy parameter τ, the finite field Q, and the Sender's private input m.

The following lemma is proved by a pigeonhole argument.

Lemma 4.1 *If all $n\rho+1$ strong pads fail, then there exists an i such that*

$$\{\text{conflicts of } T_i\} \subseteq \bigcup_{j \neq i}\{\text{conflicts of } T_j\}.$$

∎

For every conflict (x, y) of the retained T_i, (x, y) is a conflict of some returned T_j, so Receiver learns of the faultiness of at least one of x and y. Of course, both may be faulty. Without loss of generality, suppose the Sender detects that x is faulty, and publicly sends this information to the Receiver. The next Lemma says that even if y is faulty, if Sender did not also identify y as faulty then the principal share of the retained pad reported by y is the correct share of that pad.

Lemma 4.2 *Let P be the retained pad (P_i in the algorithm, but we eliminate the subscripts for ease of discussion). For every wire y that is neither thrown out nor detected faulty by Sender, $g_y = h_y$.*

Proof: Let $x_1, \ldots, x_{\tau+1}$ be nonfaulty wires. Let y be a wire that is not thrown out (so g_y is a polynomial of degree τ). If for some i, $1 \le i \le \tau + 1$, $g_y(x_i) \neq d_{yx_i}$ ($= c_{yx_i} = h_y(x_i)$ because x_i is good), then (y, x_i) is a conflict of T. In this case, by choice of T (T_i in the protocol), (y, x_i) is a conflict of some other strong pad $T' \neq T$, so Sender detects the faultiness of at least one of y, x_i. However, x_i is nonfaulty, so Sender detects the faultiness of y. Thus, if y is not thrown out or detected faulty by Sender, $g_y = h_y$. ∎

That FastSMT satisfies the resiliency condition of (σ, ρ)-SMT follows from the last two lemmas. Secrecy is argued essentially by brute force, and uses the containment assumption (or the fact that $\mathcal{A}_D$ is oblivious). We therefore have

Theorem 4.1 *Let $\tau = \max\{\sigma, \rho\}$. Under the containment assumption, or if $\mathcal{A}_D$ is oblivious, there is a three phase error free protocol for (σ, ρ)-SMT requiring connectivity $\tau + \rho + 1$ and communication polynomial in n.* ∎

5 Tight Bounds for the Containment Case

In this section we continue to restrict attention to the containment case. Generally, we assume $\sigma \ge \rho$. When $\mathcal{A}_D$ is not oblivious, if $\rho > \sigma \ge 1$ and $L \subset D$, then $\mathcal{A}_D$ can always inform $\mathcal{A}_L$ of all the traffic on the wires in D, either by communicating explicitly to $\mathcal{A}_L$ through a back channel or by writing its entire view onto one of the jointly compromised wires. The situation is then as if there were ρ listening wires, all of which could be disruptors, and the lower and upper bounds for (ρ, ρ)-SMT apply.

All the results of this section apply without the containment assumption provided $\mathcal{A}_D$ is oblivious. The upper bounds hold because $\mathcal{A}_D$ cannot communicate to $\mathcal{A}_L$ any information about the conversations on wires not in L. The lower bounds hold because even an $\mathcal{A}_D$ that disrupts completely at random could generate the scenarios leading to erroneous outcomes that will be used in those proofs.

Disruptor-free executions are critical to many of our lower bound proofs. The proofs are by contradiction. We assume the existence of a protocol with a certain amount of connectivity. The protocol must work even against an empty disrupting adversary. We study the protocol with this adversary to learn about its structure and the types of messages Sender and Receiver must send. We then define an $\mathcal{A}_D$ that is chosen accordingly and force an erroneous outcome.

Lemma 5.1 *Let P be any protocol for weakened 1-way (σ, ρ)-SMT. Then the information sent on any $n - 2\rho$ wires completely determines the secret.*

Proof: Let $n = \alpha+\beta+\gamma$, where $1 \leq \alpha \leq \rho$, $n-2\rho \leq \beta < n$ and $0 \leq \gamma \leq \rho$. We say an n-vector V *encodes* a value m if in some execution of P, when Sender begins with message m it places the ith component of V on wire i, $0 \leq i \leq n-1$.

Suppose, for the sake of contradiction that the Lemma is false. Without loss of generality, there exist values $m \neq m' \in Q$ and vectors V, V' encoding m and m', respectively, such that $V = XYZ$, where $X \in \Sigma^\alpha$, $Y \in \Sigma^\beta$, and $Z \in \Sigma^\gamma$ and $V' = X'YZ'$, where $X' \in \Sigma^\alpha$ and $Z' \in \Sigma^\gamma$ (Y remains unchanged). The point here is that Y is a subvector of at least $n - 2\rho$ components that does not determine the secret, since Y occurs in an encoding of m and in an encoding of m'. Moreover, since $\alpha \geq 1$, Y does not contain the information sent on wire 0.

Let $W = XYZ'$, where X is as in V, Y is as in both V and V', and Z' is as in V'. Now, $dist(W, V) \leq \rho$, so by the resiliency requirement if Receiver receives W it must output m. However, $dist(W, V') \leq \rho$, so Receiver must output m', a contradiction. ▮

Corollary 5.1 *Weakened 1-way (σ, ρ)-SMT under the containment assumption requires $\sigma < n-2\rho$, i.e., $n \geq \sigma + 2\rho + 1$.* ▮

The issue of containment did not arise in the proof of the lower bound. Intuitively, this is because the Sender does not know in advance which wires $\mathcal{A}_L$ and $\mathcal{A}_D$ will compromise. Thus, it must simultaneously protect against disruption on any ρ wires (in which case we let $L = D$ so $D \subseteq L$) and listening on any σ wires (in which case we let $D = \emptyset$ so again $D \subseteq L$), even if at most one of these adversaries attacks in any single execution.

In the next section we show that any solution to Verifiable Secret Sharing yields an algorithm for 1-way SMT. Thus, combining Corollary 5.1 with results in [1, 10] yields

Theorem 5.1 *Under the containment assumption, connectivity $n = \sigma + 2\rho + 1$ is necessary and sufficient for 1-way (σ, ρ)-SMT.*[3] ▮

We now turn to lower bounds on connectivity for the 2-way case. We begin with a technical lemma that hinges on our assumption that the random choices of Sender and Receiver are made by coin flipping, which yields only bounded branching. An alternative would be to allow unbounded branching at each choice node in the computation tree. While all our results hold in this model as well, the proofs are more difficult.

Lemma 5.2 *Let P be a protocol for 2-way SMT. Then there exists an upper bound B on the number of phases in any disruptor-free execution of P.* ▮

Theorem 5.2 *Let P be any protocol for 2-way (σ, ρ)-SMT. Then P requires connectivity $n \geq \max\{\sigma + \rho + 1, 2\rho + 1\}$, even under the containment assumption.*

Proof: The condition $n \geq 2\rho + 1$ is needed for ρ-resiliency, even if $\sigma = 0$. Intuitively, we see that if $n = 2\rho$ then half the wires can "behave as if" the input to Sender is some value m, and the other half can "behave as if" the input is some $m' \neq m$, and Receiver cannot tell which is the true input.

Assume, for the sake of contradiction, that there exists a protocol P for 2-way $(0, \rho)$-SMT requiring connectivity 2ρ. Let $m \neq m' \in Q$. We will construct two executions E and E' of P that, for every k, are indistinguishable to Receiver after k phases: it has the same coin flip sequence and sees exactly the same messages in each execution. However, in E the secret is m, while in E' the secret is m'. Thus, these executions cannot terminate, violating resiliency. We define the executions in parallel, phase by phase. In the following, the α's, γ's, and x's are always placed on wires $0, 1, \ldots, \rho - 1$, and the β's, δ's, and y's are placed on wires $\rho, \ldots, 2\rho - 1$. In E, for all $0 \leq i$, let $\alpha_{2i+1}\beta_{2i+1}$ be sent by Sender in Phase $2i + 1$, and let $\gamma_{2i+1}\delta_{2i+1}$ be sent by Sender in Phase $2i + 1$ of E'.

[3] If $\rho > \sigma \geq 1$ and $\mathcal{A}_D$ is not oblivious, then, as explained at the beginning of this section, the bound becomes $3\rho + 1$. If $\sigma = 0$ then $2\rho + 1$ wires suffice.

The executions begin

$$\begin{array}{llll}
E: & & E': & \\
\alpha_1 & (\beta_1 \rightarrow \delta_1) & (\gamma_1 \rightarrow \alpha_1) & \delta_1 \\
x_2 & y_2 & x_2 & y_2 \\
\vdots & \vdots & \vdots & \vdots \\
\alpha_{2i+1} & (\beta_{2i+1} \rightarrow \delta_{2i+1}) & (\gamma_{2i+1} \rightarrow \alpha_{2i+1}) & \delta_{2i+1} \\
x_{2(i+1)} & y_{2(i+1)} & x_{2(i+1)} & y_{2(i+1)}
\end{array}$$

where the notation $(x \rightarrow y)$ means that x is placed on the wires in D, but these wires (erroneously) transmit y instead. Clearly, since Receiver cannot distinguish the two executions Sender must continue, and the executions run forever, violating the resiliency requirement.

The condition $n \geq \sigma + \rho + 1$ is needed for σ-secrecy, even if $\rho = 0$. Specifically, an argument similar to (but more straightforward than) the one just presented shows that in a disruptor-free execution any $n - \rho$ wires must contain enough information to completely determine the secret. It follows that if $\sigma \geq n - \rho$, then listening to any σ wires yields the secret. Thus, $\sigma < n - \rho$, whence $n \geq \sigma + \rho + 1$. ∎

6 Perfect and Imperfect Secret Sharing: A Separation Result

We now turn briefly to definitions of secret sharing (cf. [3, 10, 13]). Due to lack of space, these definitions are merely sketched. Secret sharing is actually a class of problems, all having a similar form. The model is a distributed system in which certain processors may be disruptors and certain others may be listeners. As above, the disruptors are controlled by a disrupting adversary $\mathcal{A}_D$, while the messages and other inputs received by the listeners are available to a listening adversary $\mathcal{A}_L$. In this model, there is no way to prevent the disruptors from sending messages to the listeners, and hence $\mathcal{A}_D$ can communicate with $\mathcal{A}_L$. Thus either we work under the containment assumption or we assume $\mathcal{A}_D$ is oblivious. In keeping with the literature on secret sharing, the results in this section are for the case in which the containment assumption holds. We state our results for the case $\sigma \geq \rho$. The general results can be obtained by replacing every occurence of "σ" by "$\max\{\sigma, \rho\}$."

Definition: (σ, ρ)-Secret Sharing A protocol for a (σ, ρ)-Secret Sharing problem is a pair of n-processor protocols $(\mathcal{P}_1, \mathcal{P}_2)$, run in sequence, and designed to tolerate up to ρ faults in any execution of the pair. One special processor, p_0, is called the *dealer*. The dealer has a private input m. During $\mathcal{P}_1$ the dealer distributes shares of m in such a way that no set of σ processors not including the dealer, learns any information about the secret during execution of $\mathcal{P}_1$. $\mathcal{P}_2$ is a protocol for reconstructing the secret m from the shares distributed during $\mathcal{P}_1$. Finally, if the dealer p_0 remains nonfaulty throughout $\mathcal{P}_1$, then the value obtained by applying $\mathcal{P}_2$ is in fact the initial value (input) of p_0, provided at most ρ processors fail in total.

In analogy to the definition of SMT, we assume $\mathcal{A}_L$ compromises a set L of listening processors, and $\mathcal{A}_D$ compromises a set D of disrupting processors. In this case, the view of $\mathcal{A}_L$ is the complete history of every processor in L, from the moment it is compromised by $\mathcal{A}_L$ until the beginning of the execution of $\mathcal{P}_2$.

Definition: (σ, ρ)-Unverified Secret Sharing For every $m \in Q$, if p_0 has input m and remains nonfaulty throughout execution of $\mathcal{P}_1$ we require that for all $\mathcal{A}_L$ and $\mathcal{A}_D$ compromising sets $L \in (n-1)^\sigma$ and $D \in L^\rho$, respectively,

> **Secrecy:** $\forall m' \in Q$, if $p_0 \notin L$ then the probability distribution on the views of $\mathcal{A}_L$, given $\mathcal{A}_L$, $\mathcal{A}_D$, and given that p_0 has input m, is identical to the probability distribution on the views of $\mathcal{A}_L$ given $\mathcal{A}_L$, $\mathcal{A}_D$, and given that p_0 has input m'.
>
> **Resiliency:** At the end of $\mathcal{P}_2$, every processor not in D outputs m, regardless of the behavior of the members of D.

Note that execution of $\mathcal{P}_2$ need not immediately follow execution of $\mathcal{P}_1$, but may be delayed, so even if the dealer is correct throughout execution of $\mathcal{P}_1$ it may fail before execution of $\mathcal{P}_2$. When the secrecy and resiliency parameters are the same, say, t, we simply write *t-Unverified Secret Sharing*.

t-Verifiable Secret Sharing [3] is t-Unverified Secret Sharing with additional correctness constraints for the case in which the dealer is faulty during execution of $\mathcal{P}_1$. Specifically, even if the dealer is faulty during execution of $\mathcal{P}_1$, VSS requires that the outcome of $\mathcal{P}_2$ is uniquely determined by the states of any subset of $n - t$ processors correct at the end of $\mathcal{P}_1$, provided at most t processors fail in total during execution of the two protocols. That is, once $\mathcal{P}_1$ is completed, the dealer is committed to the secret dealt out.

In this section we show that (σ, ρ)-Unverified Secret Sharing requires $\sigma + 2\rho + 1$ processors.[4] This bound

[4] We assume $\sigma > 0$, since secret sharing makes no sense if there is no secrecy requirement.

can be achieved [1, 10]. Rabin and Ben-Or show that, for any k, t-USS can be achieved with $2t+1$ processors with probability at most 2^{-k} of error [12]. This generalizes to $\sigma+\rho+1$ processors for (σ,ρ)-USS with small probability of error. Secrecy considerations imply that (σ,ρ)-SMT cannot be solved deterministically. It follows that, within the class of problems that have no deterministic solution, error-free computation comes at a price (in this case, an extra ρ processors). It is therefore possible to separate error-free randomized computation from small-error randomized computation.

Our $\sigma+2\rho+1$-processor lower bound for (σ,ρ)-USS holds even in the model with a broadcast channel. It follows that t-Verifiable Secret Sharing requires at least $3t+1$ processors, even in the presence of a broadcast channel. This result has been claimed elsewhere [2, 12]. Because our lower bound applies to the weaker problem of t-USS, our result is stronger. Moreover, it follows from our result that the cost of Verifiable Secret Sharing has nothing to do with verification, but rather comes from the conflicting requirements of secrecy and resiliency of correctly distributed secrets.

The following theorem describes the relationship between (σ,ρ)-USS and 1-way (σ,ρ)-SMT under containment.

Theorem 6.1 *Any n processor solution $\mathcal{P} = (\mathcal{P}_1,\mathcal{P}_2)$ to (σ,ρ)-USS with or without a broadcast channel, yields a connectivity n solution to weakened 1-way (σ,ρ)-SMT under containment.*

Proof: (Sketch): A separate argument (omitted) shows that $\mathcal{P}$ requires $n \geq 2\rho+1$ processors, even in the presence of a broadcast channel. Let the wires be labelled $0,1,\ldots,n-1$. Let the processors be $p_0,\ldots,p_{n-1}$. To send a message m, the Sender first simulates a disruptor-free execution E of $\mathcal{P}_1$ in which p_0's input is m. Letting v_i denote the complete history of p_i in E, for $1 \leq i \leq n-1$, Sender places v_i on wire i. Let $V = (v_0,\ldots,v_{n-1})$. Let $W = (w_0,\ldots w_{n-1})$ denote the vector of histories received by the Receiver. By assumption, $dist(V,W) \leq \rho$. To compute the message encoded by the vector W, Receiver simulates that execution of $\mathcal{P}_2$ in which each processor p_i begins in the state given by w_i and no further disruption occurs. This results in a set of outputs, one for each p_i. Receiver outputs that value which is output by a majority of processors in the simulation. Secrecy and resiliency are "inherited" from $\mathcal{P}$. ▮

Combining this with Corollary 5.1 yields

Corollary 6.1 *(σ,ρ)-Unverified Secret Sharing requires at least $\sigma+2\rho+1$ processors, with or without a broadcast channel.* ▮

Corollary 6.2 *t-Verifiable Secret Sharing requires $3t+1$ processors, even in the presence of a broadcast channel.* ▮

Theorem 6.2 *Within the class of problems having no determistic solution, the cost of an error-free solution can provably exceed the cost of a solution with arbitrarily small probability of error.* ▮

7 Beyond Containment

In this section, mostly omitted from these Proceedings, we study how the bounds obtained in Section 5 change when the containment assumption is removed, provided $\mathcal{A}_D$ is not oblivious. To obtain upper bounds in this case is simple: any algorithm for $(\sigma+\rho,\rho)$-SMT, under the containment assumption completely solves the general (σ,ρ)-SMT problem, even if the adversaries are actually allowed to communicate during execution of the protocol. This yields an increase of ρ wires in both the 1-way and 2-way cases. We therefore have the following upper bounds.

Theorem 7.1 *Connectivity $\sigma+3\rho+1$ is sufficient for 1-way (σ,ρ)-SMT, and connectivity $\sigma+2\rho+1$ is sufficient for 2-way (σ,ρ)-SMT, even without the containment assumption.* ▮

Our results for the model in which the adversaries cooperate but do not communicate are summarized in the next two theorems.

Theorem 7.2 *In the model in which $\mathcal{A}_L$ and $\mathcal{A}_D$ cooperate but do not communicate, if $\sigma \geq 1$ then (σ,ρ)-SMT requires $\sigma+3\rho$ wires in the 1-way case and $\sigma+2\rho$ wires in the 2-way case.*[5] ▮

Theorem 7.3 *Let P be any protocol for 2-way (σ,ρ)-SMT in the model in which $\mathcal{A}_L$ and $\mathcal{A}_D$ do not communicate and $\mathcal{A}_D$ is not oblivious. If every execution of P lasts exactly 3 phases, beginning with a transmission from Sender to Receiver, then P requires $n \geq \sigma+2\rho+1$ wires.* ▮

The proof of this last theorem involves ten scenarios. We see no way of generalizing it to protocols requiring more than three phases.

[5] If $\sigma=0$ then connectivity $2\rho+1$ is necessary and sufficient for both the 1-way and 2-way cases.

8 Additional Remarks

The concept of two distinct adversaries, $\mathcal{A}_L$ and $\mathcal{A}_D$, is an intriguing one. Generally, we have assumed in this paper that the adversaries cooperate with the goal of defeating the algorithm. However, it may be the case that the adversaries do not cooperate. Essentially, this is the case when $\mathcal{A}_D$ simply disrupts at random. As we have seen, without the containment assumption the upper bounds are better in this case than when the adversaries cooperate. Are there other models and problems in which it makes sense to consider non-cooperating adversaries?

Acknowledgement: Many people have helped us by listening to arguments, reading early drafts, and making suggestions. In particular, we thank Hagit Attiya, Shafi Goldwasser, Silvio Micali, Stephen Ponzio, Ruediger Reischuk, and Eva Tardos. Most of all, we thank Larry Stockmeyer for many hours of invaluable discussion.

References

[1] M. Ben-Or, S. Goldwasser, and A. Wigderson, Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation, *Proc. 20th Symp. on Theory of Computing*, pp. 1-10, 1988.

[2] D. Chaum, C. Crepeau, and I. Damgard, Multiparty Unconditionally Secure Protocols, *Proc. 20th Symp. on Theory of Computing*, 11-19, 1988.

[3] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults, *Proc. 26 Symp. on Foundations of Computing*, pp. 383-395, 1985.

[4] C. Dwork, Strong Verifiable Secret Sharing, *Proceedings, 4th International Workshop on Distributed Algorithms (WDAG-4)*, Bari, Italy, September, 1990.

[5] C. Dwork, D. Peleg, N. Pippenger, E. Upfal, Fault Tolerance in Networks of Bounded Degree, *SiComp*, 1989.

[6] P. Feldman, Optimal Algorithms for Byzantine Agreement, *PhD Thesis*, Department of Mathematics, MIT, 1988.

[7] P. Feldman, and S. Micali, Optimal Algorithms for Byzantine Agreement, *Proc. 20th Symp. on Theory of Computing*, pp. 148-161, 1988.

[8] Z. Galil, S. Haber, and M. Yung, Primitives for Designing Multi-Party Protocols from Specifications, *manuscript*, 1989.

[9] O. Goldreich, S. Micali, and A. Wigderson, How to Play Any Mental Game, sl Proc. 19th Symp. on Theory of Computing, pp. 218-229, 1987.

[10] R. McEliece and D. Sarwate, On Sharing Secrets and Reed-Solomon Codes, *CACM 24*(9), pp. 583-584, 1981.

[11] M. Rabin and D. Lehmann, On the Advantages of Free Choice: A Symmetric and Fully Distributed Solution to the Dining Philosophers Problem, *Proc. Symposium on Principles of Programming Languages*, pp.133-138, 1981.

[12] T. Rabin, and M. Ben-Or, Verifiable Secret Sharing and Multiparty Protocols with Honest Majority, *Proc. 21st Symp. on Theory of Computing*, pp. 73-85, 1989.

[13] A. Shamir, How to Share a Secret, *CACM 22*, pp. 612-613, 1979.