On Verification in Secret Sharing

Cynthia Dwork
IBM Almaden Research Center
650 Harry Road
San Jose, CA. 95120
dwork@almaden.ibm.com

Abstract

Verifiable Secret Sharing (VSS) has proven to be a powerful tool in the construction of fault-tolerant distributed algorithms. Previous results show that Unverified Secret Sharing, in which there are no requirements when the dealer is faulty during distribution of the secret, requires the same number of processors as VSS. This is counterintuitive: verification that the secret is well shared out should come at a price. In this paper, by focussing on information leaked to *nonfaulty* processors during verification, we separate a certain strong version of Unverified Secret Sharing (USS) from its VSS analogue in terms of the required number of processors. The proof of the separation theorem yields information about communication needed for the original VSS problem. In order to obtain the separation result we introduce a new definition of secrecy, different from the Shannon definition, capturing the intuition that "information" received from faulty processors may not be informative at all.

1 Introduction

The t-resilient Verifiable Secret Sharing problem (t-VSS) was defined by Chor, Goldreich, Micali, and Awerbuch [4]. A solution to t-VSS is a protocol for a distributed system of n processors allowing a distinguished *dealer* processor to irreversibly commit to a secret value that can be reconstructed with certainty at a later time, even if the dealer attempts to block the reconstruction. Specifically, if the dealer is nonfaulty until reconstruction then no set F of up to t faulty processors can learn anything about the secret before reconstruction, and the value reconstructed will indeed be the secret value of the dealer. Moreover, the members of F cannot prevent reconstruction of the committed secret, even if the dealer is in F.

VSS and its applications have been widely studied in the literature [1, 2, 3, 6, 7, 8, 13]. In particular, Ben-Or, Goldwasser, and Wigderson showed that in a system of $n \geq 3t+1$ processors, where t is an upper bound on the number of faulty processors, any function of n inputs can be computed in such a way that not only can the faulty processors not

J. Feigenbaum (Ed.): Advances in Cryptology - CRYPTO '91, LNCS 576, pp. 114-128, 1992.

disrupt the computation, but they cannot learn any additional information about the inputs of the nonfaulty processors than that implied by their own inputs and the output of the function [2]. A similar, slightly weaker, result was obtained independently by Chaum, Crepeau, and Damgard [3]. In both these constructions, t-resilient VSS is used to share out the inputs to the function. The VSS protocols in [2, 8] differ from other VSS protocols in that they are error-free; secrecy is perfect (information-theoretic) and reconstruction never fails.[1] Dolev, Dwork, Waarts, and Yung [7] showed that the bound of $3t + 1$ processors for error-free t-VSS is tight, even in the presence of a broadcast channel. Moreover, the lower bound holds even for the easier problem of t-Unverified Secret Sharing (t-USS), a relaxation of t-VSS in which there are no requirements if the dealer is faulty during distribution of the secret. For the remainder of this paper we discuss only error-free protocols.

That t-USS and t-VSS should require exactly the same number of processors was puzzling. Error-free verification that the secret is well distributed seemed too powerful to come at no cost. Indeed, elementary arguments show that some communication among shareholders of a secret must take place in order to verify that a secret is well shared out. In this paper we obtain additional information about the structure of this communication. To do this, we define a very strong version of both VSS and its unverified analogue, and prove the separation for those problems in terms of the number of processors. By analyzing the proof of the separation result for the strong versions of the problems we obtain information about communication necessary during verification in any solution of the original t-VSS problem.

An interesting by-product of this work is the introduction of what we call *effectively perfect* secrecy. This new type of secrecy captures the intuition that, loosely speaking, "information" received from a faulty processor may not be information at all, since the faulty processor can "lie." In addition to being useful in obtaining the first separation result, we feel our definition of effectively perfect secrecy is an interesting step in trying to capture an appropriate notion of secrecy when the source of information is Byzantine.

The rest of this abstract is organized as follows. Section 2 contains formal definitions of several variants of secret sharing, of perfect secrecy, and of effectively perfect secrecy. Section 3 discusses some related results and applies one of these to our context. Section 4 contains the separation result and some of its implications for ordinary VSS. Discussion and open problems appear in Section 5.

[1] In some protocols secrecy is imperfect because it relies on public key cryptography; in others, reconstrution may fail because verification is performed using interactive proof systems techniques.

2 Definitions

We consider a completely synchronous distributed system of n processors, $\{p_0, \ldots, p_{n-1}\}$, connected by a complete network of perfectly secure channels. Our definition of protocol is the standard one. Let E be a finite field with a primitive nth root of unity ω, such that $|E| > n$.

We assume the secrets are always elements of the finite field E. There is a fixed underlying probability distribution Π on messages in E. *Information-theoretic (perfect) secrecy* says essentially that the best one can do at guessing a secret is to guess an element with maximal probability according to Π. We say a polynomial is of degree ℓ if it is of degree *at most* ℓ.

Given a protocol $\mathcal{P}$, a processor executing $\mathcal{P}$ is a *disruptor* if it does not follow $\mathcal{P}$ correctly. It can misbehave by not choosing random values according to the distribution specified by $\mathcal{P}$, by failing to send specified messages, possibly sending arbitrary messages in their stead, and by making erroneous state transitions. A processor is a *gossip* if it follows $\mathcal{P}$ correctly but in addition sends extraneous messages labelled as gossip messages. Since the messages are labelled as extraneous, they are never confused with protocol messages. A processor is *pure* if it is neither disruptive nor a gossip. A processor is *faulty* if it is not pure.

A solution to the (t, d)-**Verifiable Secret Sharing** problem ((t, d)-VSS) is a pair of protocols $(\mathcal{P}_1, \mathcal{P}_2)$, called the *distribution* and *reconstruction* protocols, respectively. Let p_0 be a dinstinguished *dealer* processor with secret input s. If in $\mathcal{P}_1$ and $\mathcal{P}_2$ combined there are at most t faulty processors (disruptors and gossips), *of which* at most d are disruptors, then the following properties are required.

1. For every adversary strategy, if the adversary allows p_0 to remain pure until the beginning of execution of $\mathcal{P}_2$[2] then until that point no set of up to t faulty processors has any information (in the information-theoretic sense) about the secret s. More precisely, for every adversary $\mathcal{A}$, for all pairs of messages $m, m' \in E$, and for all views $\mathcal{V}$ of the members of F, $\Pr[\mathcal{V} \mid \mathcal{A}, m] = \Pr[\mathcal{V} \mid \mathcal{A}, m']$, where F is the set of processors compromised by $\mathcal{A}$ and the probabilities are over all choices made by the processors and by the adversary.

2. Whether or not p_0 is pure until $\mathcal{P}_2$, the value obtained by executing $\mathcal{P}_2$ is completely

[2] $\mathcal{P}_2$ may begin long after $\mathcal{P}_1$ is completed, so the end of $\mathcal{P}_1$ is not the same as the beginning of $\mathcal{P}_2$. For most applications some secrets will be dealt out (using $\mathcal{P}_1$), some computations will be performed, and at some later time the secrets, or some functions of the secrets such as their sum, will be reconstructed (using $\mathcal{P}_2$).

determined by the end of $\mathcal{P}_1$, and remains unchanged regardless of the behavior of the disruptive processors.

3. If p_0 is nondisruptive throughout execution of $\mathcal{P}_1$ then the value obtained when $\mathcal{P}_2$ is executed is s, the true secret input of p_0.

When $t = d$ this is precisely the usual definition of t-resilient Verifiable Secret Sharing. The change is the addition of the category of "faulty gossips." These only affect secrecy and do not affect reconstructability. This point is subtle, and comes from the fact that we require perfect reconstructability.

Let $Q(\cdot)$ be a predicate, and let $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$ be a VSS protocol. We say a coaliton of processors L *prematurely learns* $Q(s)$ if in some execution ξ of $\mathcal{P}$ in which p_0 has input s and is pure until execution of $\mathcal{P}_2$, the members of L do not have *implicit knowledge*[3] of $Q(s)$ before execution of $\mathcal{P}_1$ but they do have implicit knowledge of $Q(s)$ before $\mathcal{P}_2$.

A protocol solves the (ℓ, t, d)-**Strong Verifiable Secret Sharing** problem ((ℓ, t, d)-SVSS), $\ell \geq t \geq d$, if in addition to solving (t, d)-VSS it satisfies the following conditions[4]:

1. If p_0, with input s, remains pure until the beginning of execution of $\mathcal{P}_2$, then no coalition L of up to ℓ pure processors prematurely learns $Q(s)$ for any predicate $Q(\cdot)$.

2. If p_0, with input s, remains pure until the beginning of $\mathcal{P}_2$, then the protocol messages received by L from the nondisruptive processors contain no information about s.

Since this definition is delicate, we give some intuition. Fix an execution of an (ℓ, t, d)-SVSS protocol, and consider the histories of all pure processors until just before execution of $\mathcal{P}_2$. Then the first additional requirement says that if p_0 is pure at this point then, for every set L of ℓ pure processors, the histories of the members of L do not jointly have enough information to unambiguously determine anything about the secret not implicit in Π. Thus, even if members of L were to "gossip" among themselves (in the colloquial sense), they would not prematurely learn anything about the secret. Since we think about what the members of the set L could figure out if they were to pool their information about the execution, we generally talk about *coalitions* of size ℓ. The second

[3] Implicit knowledge was defined by Halpern and Moses [11]. Roughly speaking, the definition says that at time k in ξ, L has implicit knowledge of a fact ϕ if and only if ϕ holds in all executions of the protocol in which the joint view of the members of L is identical to their joint view at time k of ξ.

[4] The bounds obtained are a function of $\max\{\ell, t\}$, t, and d. Thus, the case in which $\ell < t$ is somewhat degenerate, so we ignore it in this discussion.

additional requirement says that the partial information gained by L necessarily comes from unreliable sources.

Remark 1 *There exists an adversary strategy against the* (t, t)*-VSS protocol in [2] that guarantees that every pure processor other than the dealer prematurely learns the secret. Thus, the protocol in [2] does not solve even* $(1, t, t)$*-SVSS.*

To appreciate the significance of the remark, consider the problem of contract bidding. Let companies X, Y, and Z be such that

1. Company Z can beat any bid made by Company X.
2. Company Y cannot match X's bid.
3. Company Y prefers that Z will win the contract (rather than X).
4. Company Z wishes to behave legally.

Suppose the bids are placed using 1-resilient verifiable secret sharing, and suppose further that "legal" behavior includes using any information gained during the distribution of secrets. If X bids before Z does, then, using the adversarial strategy mentioned in the remark, Y can throw the contract to Z by behaving in such a way that Z learns X's bid (and can therefore underbid X). Of course, Y could always just tell Z its share of X's bid, but Z might have no way of determining whether or not Y is telling the truth. Precisely this point motivates the following discussion and definitions.

Although by definition a secret shared by (ℓ, t, d)-SVSS enjoys perfect secrecy with respect to the faulty processors, and *some* secrecy with respect to small coalitions of pure processors (that is, coalitions of size at most ℓ), it may not have *perfect* secrecy with respect to small coalitions. This is because some probabilistic information may be received from the faulty processors. To see how this might occur, let us say a faulty processor lies "convincingly" about its share of the secret, before reconstruction, if it announces a faulty share but no small coalition of pure processors can detect that the announced share is erroneous[5]. For example, in Shamir's scheme for sharing a secret [14], to ensure secrecy against a coalition of size ℓ the dealer chooses a polynomial P of degree ℓ whose free term is the secret, and gives to the ith processor the "share" $P(i)$, $1 \leq i \leq n$. In this case it is very easy for a processor to lie convincingly to a coalition of ℓ nonfaulty

[5]It may be possible that, while small coalitions can not detect the fact that the announced share is erroneous, large coalitions of pure processors can do so; something like this is necessary so that faulty shares can be discarded during reconstruction of the secret.

procesors about its share: it simply chooses an arbitrary value and claims this is its share. Since every set of $\ell + 1$ points determines a polynomial of degree ℓ the lie is convincing (plausible).

If the probability is low that a faulty processor can lie convincingly to a small coalition of nonfaulty processors about its share of the secret, for example, if shares of the secret are certified with *check vectors* [13], then a plausible share announced by the faulty processor carries some informational content. In this case, continuing the example given above, if the faulty processor were to lie convincingly about its share of a degree ℓ polynomial by fortuitously managing to certify a false value, and if the values held by the coalition members together with this value from the faulty processor interpolate to yield a polynomial whose free term is, say, 5, then the coalition can conclude that almost surely the secret is 5. However, they will not *know*, in the sense of Halpern and Moses, that the secret is 5.

In some scenarios the curious parties have no interests in common with the disruptive parties. In this case, especially if the protocol is such that it is *easy* for the faulty processors to lie convincingly about their shares of the secret, it is not clear what it means to "learn" something from the faulty processors. We therefore define *Effectively Perfect SVSS*, in which we attempt to capture the intuition that information received from faulty processors may be of no use in determining the secret prematurely. As usual, we assume the existence of an adversary that controls the selection and behavior of faulty processors. The intuition to keep in mind is that we want a way of saying that even if the faulty processors truthfully announce their shares of the secret, no coalition of up to ℓ pure processors really learns anything about the secret, since the faulty processors could be lying convincingly about their shares.

A protocol $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$ solves (ℓ, t, d)-**Effectively Perfect SVSS** if in addition to solving (ℓ, t, d)-SVSS it has the following properties.

1. For all adversaries $\mathcal{A}$ there exists an adversary $\mathcal{A}'$ such that, for all coalitions L of at most ℓ pure processors, for every view $\mathcal{V}$ obtained by L in some execution ξ of $\mathcal{P}$ (before the start of $\mathcal{P}_2$) with adversary $\mathcal{A}$,

$$\Pr[\mathcal{V} \mid \mathcal{A}] = \Pr[\mathcal{V} \mid \mathcal{A}']. \tag{1}$$

2. If the dealer is not compromised by $\mathcal{A}'$ before execution of $\mathcal{P}_2$, then for all legal messages m'

$$\Pr[\mathcal{V} \mid \mathcal{A}', m] = \Pr[\mathcal{V} \mid \mathcal{A}', m']. \tag{2}$$

Here the notation $\Pr[\mathcal{V} \mid \mathcal{A}]$ denotes the probability that L has view $\mathcal{V}$ given that the adversary is $\mathcal{A}$, where the probability distribution is over the choice of the secret as well as over the space of all coin tosses of all processors and of the adversary $\mathcal{A}'$. Similarly, the notation $\Pr[\mathcal{V} \mid \mathcal{A}', m]$ denotes the probability that L has view $\mathcal{V}$ given that the adversary is $\mathcal{A}'$ and given that the secret dealt out is m, where the probability distribution is the space of all coin tosses of all processors and of the adversary $\mathcal{A}'$ (but not over choice of m).

The intuition is as follows. If $\Pr[\mathcal{V} \mid \mathcal{A}] = \Pr[\mathcal{V} \mid \mathcal{A}']$ then for all L knows during ξ (before execution of $\mathcal{P}_2$), the adversary is really $\mathcal{A}'$; if the adversary were really $\mathcal{A}'$ then since $\Pr[\mathcal{V} \mid \mathcal{A}', m] = \Pr[\mathcal{V} \mid \mathcal{A}', m']$ the secrecy would be perfect. Thus, if Equations 1 and 2 hold then secrecy is "effectively" perfect (see Remark 2). We quantify m, m' over all legal messages, rather than over all elements of E, because it is possible that not all elements of E are legal inputs to a given protocol.

Remark 2 *The definition of effectively perfect secrecy is most interesting in a situation in which it is impossible to assign a probability distribution to the space of adversaries. In this case the definition implies that L has no advantage in guessing the message over simply guessing according to the underlying probability distribution Π.*

A protocol solves (ℓ, t, d)-**Perfect SVSS** if it solves (ℓ, t, d)-SVSS and in addition secrecy with respect to any coalition of ℓ pure processors is perfect. More formally, for all adversaries $\mathcal{A}$, for all legal messages m, m', for all coalitions L of up to ℓ pure processors, and for every view $\mathcal{V}$ of L before $\mathcal{P}_2$ in an execution in which the dealer remains pure until $\mathcal{P}_2$,

$$\Pr[\mathcal{V} \mid \mathcal{A}, m] = \Pr[\mathcal{V} \mid \mathcal{A}, m'].$$

Remark 3 *An alternative definition for (ℓ, t, d)-Perfect SVSS would be to take the probabilities over the choice of $\mathcal{A}$, and condition only on the secret. However, once again, we can think of no reasonable probability distribution on adversaries.*

For each of these types of Verified Secret Sharing we define a corresponding **Unverified** version by weakening the problem so that there are no requirements if the dealer is faulty during the execution of $\mathcal{P}_1$. In particular, as observed by MacEliece and Sarwate [12], solving (t, d)-Unverified Secret Sharing ((t, d)-USS) is essentially equivalent to constructing d-error-correcting codes with the constraint that no t shares suffice to determine anything about the encoded value. This can be done using the d-error correcting BCH codes of length $t + 2d + 1$.

3 Related Work

As mentioned in the Introduction, secret sharing has been very widely studied. Simple secret sharing, resilient only to curious coalitions of t gossips but to no disruptors, was defined by Shamir [14]. t-Verified Secret Sharing (what we are calling (t,t)-VSS) was defined by Chor, Goldwasser, Micali, and Awerbuch [4], whose original solution required many processors (as a function of t) and was not error-free. The introduction of zero-knowledge techniques [10], particularly in [9], yielded conceptually simple, elegant, and highly resilient small-error soutions to t-VSS requiring only $3t+1$ processors (see also [5]). The $3t+1$-processor solution due to Chaum, Crepeau, and Damgard [3] enjoys perfect secrecy and requires no cryptographic assumptions, but has small probability of error. The first error-free solution to t-VSS was obtained by Ben-Or, Goldwasser, and Wigderson [2]. Their solution required only $3t+1$ processors, which is optimal, even in the presence of a broadcast channel [7]. More precisely, the results in [2, 7] show that error-free (t,d)-VSS can be achieved with $t+2d+1$ processors, and this number is necessary even for the *unverified* version of the problem, in which there are no requirements if the dealer is faulty during the distribution protocol. Given a broadcast channel, Rabin and Ben-Or [13] achieved small-error t-VSS using only $2t+1$ processors. Given *both* a broadcast channel and the ability to achieve *oblivious transfer*, Beaver and Goldwasser solve the problem in the presence of a faulty majority [1].

(ℓ,t,d)-Strong Verifiable Secret Sharing was defined in [6]. The solution in that paper requires only $\max\{\ell,t\}+2d+1$ processors. In particular, when ℓ, the size of the coalition of pure processors, equals t, the total number of faulty processors, strong verification incurs no additional cost in processors over (t,d)-*unverified* secret sharing. On the other hand, the paper also showed that if the secrecy with respect to coalitions of pure processors is required to be perfect then $\ell+t+2d+1$ processors are necessary and sufficient. We briefly review those arguments.

Theorem 3.1 [6] *Any protocol for (ℓ,t,d)-Perfect Strong Unverified Secret Sharing requires $\ell+t+2d+1$ processors. Moreover, this number of processors suffices even for (ℓ,t,d)-Perfect Strong Verified Secret Sharing.*

Proof: By definition, any protocol for $(\ell+t,d)$-VSS guarantees perfect secrecy with respect to any set of $\ell+t$ players, even if d of these are disruptive. Thus, such a protocol also solves (ℓ,t,d)-SVSS in which the members of the curious coalition and the faulty processors number at most $\ell+d$ in total. The protocol in [7] for $(\ell+t,d)$-VSS[6] requires exactly $\ell+t+2d+1$ processors.

[6] based on the protocol for $(\ell+t,\ell+t)$-VSS in [2]

For necessity, suppose $n = \ell + t + 2d$ processors suffice for (ℓ, t, d)-SVSS, and consider an adversary $\mathcal{A}$ that always compromises the t processors $p_1 \ldots p_t$ (recall, the dealer is p_0), causing the compromised processors always to truthfully announce their shares in gossip messages. It is shown in [7] that any $n - 2d$ shares must be enough to reconstruct the secret. Thus, if $n = \ell + t + 2d$ then $\ell + t$ shares suffice. Clearly, for any two legal messages m and m', with this adversary the view of any coalition of ℓ pure processors when the secret message is m does not occur with the same probability that they do when the message is m', since the coalition has access to $\ell + t$ shares: its own ℓ shares plus the t announced shares. This violates perfect secrecy. ∎

4 Separating Unverified Secret Sharing from Verifiable Secret Sharing

4.1 Separating the Strong Versions of the Problems

In this subsection we separate effectively perfect Strong Unverified Secret Sharing, in which there are no requirements if the dealer is faulty during the distribution protocol, from effectively perfect Strong Verifiable Secret Sharing. For simplicity, we take all parameters to be t. The full paper contains results for the general case. We abbreviate (t, t, t)-SVSS (respectively, (t, t, t)-SUSS) by t-SVSS (respectively, t-SUSS).

Theorem 4.1 *Let $n = 4t$ and let E be a field of at least $n + 1$ elements with a primitive nth root of unity ω. Let Π be the uniform probability distribution on E. There is an n-processor protocol for t-effectively perfect Strong Unverified Secret Sharing with message space E and probability distribution Π.*

Proof: The protocol is trivial: given input v, the dealer chooses a random polynomial $f(\cdot) \in E(\cdot)$ of degree $2t - 1$ with free term v, and sends to each processor p_i, $0 \leq i < n$, (including itself) the value $f(\omega^i)$. The vector of shares is a codeword in the t-error correcting BCH code of length $4t$. To reconstruct the secret, agreement is run on the shares of all processors and the error-correction procedure for BCH codes is applied to retrieve f.

Let $\mathcal{A}$ be a disrupting adversary. Let ξ be an execution of the dealing protocol with disrupting adversary $\mathcal{B}$ defined as follows. $\mathcal{B}$ simulates $\mathcal{A}$ precisely until $\mathcal{A}$ has compromised t processors. If $\mathcal{A}$ compromises fewer than t processors, or if $\mathcal{A}$ compromises the dealer, then $\mathcal{B}$ just simulates $\mathcal{A}$. Otherwise, let D be the processors compromised

by $\mathcal{B}$ and let q be compromised by $\mathcal{B}$ no later than any other processor in D. Let $\mathcal{B}$ choose uniformly at random an element $s_q \in_R E$. $\mathcal{B}$ then simulates $\mathcal{A}$ in an execution in which all processors in $D - \{q\}$ have the shares they actually received in ξ, but q has value s_q.

The proof of effectively perfect secrecy rests on two claims, whose proofs are omitted for lack of space.

Claim 4.1 *Let ξ be an execution of the t-SUSS protocol with adversary $\mathcal{B}$. Let L be a set of t processors pure in ξ and not containing the dealer. Let $\mathcal{V}$ be L's view during ξ. Then* $\Pr[\mathcal{V} \mid \mathcal{B}] = \Pr[\mathcal{V} \mid \mathcal{A}]$, *where the probability space is over choice of secret as well as the random choices of all processors and of the adversary.*

Claim 4.2 *Let ξ be an execution of the t-SUSS protocol with adversary $\mathcal{B}$, and let L be a set of t processors pure in ξ and not containing the dealer. Let $\mathcal{V}$ be L's view during ξ. Then for all messages m, m', if the dealer is not disrupted by $\mathcal{B}$ then* $\Pr[\mathcal{V} \mid \mathcal{B}, m] = \Pr[\mathcal{V} \mid \mathcal{B}, m']$, *where the probability space is over all random choices of the processors and of $\mathcal{B}$.*

Thus, the conditions of Equations 1 and 2 are satisfied. This completes the proof of the theorem. ∎

Theorem 4.2 *t-Effectively Perfect Strong Verifiable Secret Sharing requires $4t + 1$ processors, for all $t \geq 2$.*

Proof: Suppose for the sake of contradiction that the theorem is false. Let $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$ be a protocol requiring only $n \leq 4t$ processors. Let adversary $\mathcal{A}$ be as follows. In $\mathcal{P}_1$, $\mathcal{A}$ compromises a set D of t processors, chosen uniformly at random from the set of all $n - 1$ processors not containing the dealer. $\mathcal{A}$ allows the execution to be failure-free; the point of compromising the processors is to give $\mathcal{A}$ access to their message histories and random choices. Before execution of $\mathcal{P}_2$ each processor in D broadcasts its entire history during the execution of $\mathcal{P}_1$. That is, each processor broadcasts its complete message history (sent and received), and all random choices made during the execution of $\mathcal{P}_1$. For every coalition L of at most t pure processors, for all $m \neq m'$ and for all views $\mathcal{V}$ of the members of L, $\Pr[\mathcal{V} \mid \mathcal{A}, m] \neq \Pr[\mathcal{V} \mid \mathcal{A}, m']$, since, together with the (truthful) history announced by D, the members of L have $n - 2t$ shares, which by a result in [7] suffice to reconstruct the secret.

Now, let $\mathcal{A}'$ be any disruptive adversary other than $\mathcal{A}$. To satisfy the constraint that for all coalitions L of up to t pure processors, not containing the dealer, and for all views $\mathcal{V}$ of L, $\Pr[\mathcal{V} \mid \mathcal{A}'] = \Pr[\mathcal{V} \mid \mathcal{A}]$ (Equation 1), adversary $\mathcal{A}'$ must choose the set D with the same probability distribution as $\mathcal{A}$, must never disrupt execution of $\mathcal{P}_1$ in a way detectable by any coalition of t pure processors not containing the dealer, and must announce a supposed history of D before execution of $\mathcal{P}_2$. Moreover, the members of D must send the same message to every processor other than the dealer and other members of D, just as they would do under control of $\mathcal{A}$; for if some member of D were to send different messages to, say, p and q, then the set $L = \{p, q\}$ would be able to distinguish $\mathcal{A}'$ from A. This is where we use the condition $t \geq 2$.

Consider a particular execution ξ of $\mathcal{P}_1$ with adversary $\mathcal{A}'$. Let the processors be divided into four disjoint groups R, S, T, W, each of size at least 1 and at most t, where the dealer is in T and the processors in S have been compromised by $\mathcal{A}'$. Let $\alpha, \beta, \tau, \delta$ be the histories, respectively, of R, S, T, W at the end of ξ. We say ξ *results in* $\alpha\beta\tau\delta$. Let γ be the history announced by the members of S. Note that since $\mathcal{A}$ always compromises exactly t processors, $\mathcal{A}'$ must do so as well, so $|S| = t$.

For $\mathcal{A}'$ not to be distinguished from $\mathcal{A}$ by R, for some vectors of histories a_1, a_2 there must exist a failure-free execution E_R of $\mathcal{P}_1$ resulting in $\alpha\gamma a_1 a_2$. Similarly, for some c_1, c_2 there exists a failure-free execution E_W resulting in $c_1\gamma c_2\delta$.

Since R has history α in both ξ and E_R, the messages exchanged between R and W are the same in ξ as in E_R. Similarly, since W has history δ in both ξ and E_W, the messages exchanged between R and W are the same in ξ as in E_W. Thus the messages exchanged between R and W are the same in executions E_R and E_W. Since S has history γ in E_W and E_R, the messages exchanged between S and W are the same in executions E_R and E_W. For the same reason, messages exchanged between S and R are the same in executions E_R and E_W.

Consider an execution of $\mathcal{P}_1$ with the following adversary $\mathcal{B}$. The processors in T are faulty (including the dealer). They act toward the members of R and S as in execution E_R, while acting toward the members of W as in execution E_W. At the end of $\mathcal{P}_1$, with the adversary following this strategy, the vector of histories is $\alpha\gamma * \delta$, where we have omitted the history of T because these processors are faulty.

For histories r, s, t, w, let $\mathcal{P}_2(rstw)$ denote the outcome of $\mathcal{P}_2$ when the members of R, S, T, W begin $\mathcal{P}_2$ in the states they would be in had they finished $\mathcal{P}_1$ with histories r, s, t, w, repectively, and were there to be no other faulty behavior in $\mathcal{P}_2$.

Let $v = \mathcal{P}_2(\alpha\gamma a_1 a_2)$. Since E_R is failure-free and results in $\alpha\gamma a_1 a_2$, it must be the case that $\mathcal{P}_2(\alpha\gamma a_1 \delta) = v$, since otherwise, by choosing between a_2 and δ, the members of W could control the outcome of $\mathcal{P}_2$ after execution E_R of $\mathcal{P}_1$. This implies that $\mathcal{P}_2(\alpha\gamma\tau\delta) = v$, since otherwise, after arranging an outcome $\alpha\gamma * \delta$ of $\mathcal{P}_1$, adversary $\mathcal{B}$ could control the outcome of $\mathcal{P}_2$ by choosing between τ and a_1 to determine its starting state in $\mathcal{P}_2$.

We now argue that $\mathcal{P}_2(\alpha\beta\tau\delta) = \mathcal{P}_2(\alpha\gamma\tau\delta)$ $(= v)$. This is because otherwise, after execution ξ (which results in $\alpha\beta\tau\delta$ and in which only the members of S are faulty), the members of S could control the outcome of $\mathcal{P}_2$ by choosing between β and γ. Thus, if $\mathcal{V}$ is the view of the members of R after ξ and after the members of S have announced γ, then for all $v' \neq v$,

$$\Pr[\mathcal{V} \mid \mathcal{A}', v] \neq \Pr[\mathcal{V} \mid \mathcal{A}', v'].$$

This violates Equation 2 in the definition of effectively perfect secrecy. Thus, no adversary $\mathcal{A}'$ satisfying Equation 1 can also satisfy Equation 2. ▮

4.2 Implications for t-VSS

As we have seen, the distribution phase for t-USS requires only a single round of message exchange (the dealer sends a share to every player other than itself). An elementary hybrid argument shows that additional communication is required for the distribution phase of t-VSS. The proof of Theorem 4.2 yields information about the structure of this additional communication.

Let $n = 3t+1$ and consider any n-processor protocol $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$ for t-VSS (what we have been calling (t,t)-VSS). Let us emphasize certain points in the proof of Theorem 4.2.

1. α and γ "fit," that is, there exists a failure-free execution of $\mathcal{P}_1$ that results in $\alpha\gamma a_1 a_2$, for some $a_1 a_2$. We defined v to be $\mathcal{P}_2(\alpha\gamma a_1 a_2)$.
2. γ and δ "fit."
3. $|S|, |T|, |W| \leq t$.
4. From these three points alone we concluded that the secret dealt out during ξ was also v. For example, we did not need that γ and τ fit, and we did not need to know τ or δ.

Since we did not need that γ and τ fit, the members of S could be lying in γ about their communication with the members of T. In particular, they could be lying about what

they received from the dealer. However, since γ fits δ, the members of S are not lying about their communication with W.

We obtained points (1) and (2) above from the definition of effectively perfect secrecy. Suppose we are given a failure-free execution of a distribution protocol in which the members of R have history α. Given α, if we had any way of knowing that a history γ, fitting with α, also fits the history of W, whatever that history may be, then we could determine the secret, provided the sizes of the sets S, T, W satisfy condition (3). Thus, the proof of Theorem 4.2 shows that the secret is completely determined by α and the communication between the members of S and W during the distribution protocol. Since $|R| \leq t$, α itself cannot hold any information about the secret. Thus there must exist some communication between S and W. If we assume processors only communicate with explicit messages (and not via timeouts) this tells us a fair amount about the communication needed even in failure-free executions of the dealing protocol.

For example, if we choose R, S, and T each to have size exactly t, then W has size 1. The above discussion implies there must be communication between S and W. Varying our choice of W over all processors in T other than the dealer we conclude S must communicate with every processor not in $R \cup S$, except possibly the dealer. (Communication with the dealer can be argued separately.)

5 Discussion and Open Questions

This research was motivated by a desire to understand on an intuitive level precisely what must occur during verification that a secret is well distributed. By focussing on secrecy with respect to small coalitions of pure processors we were able to separate a strong version of Verified Secret Sharing from its Unverified analogue. To do this we introduced the notion of effectively perfect secrecy. As we saw in Section 4.2, examination of the proof of Theorem 4.2 yields information about the structure of communication during the distribution phase of any protocol for the original t-VSS problem. An example of the type of structure discovered was given in that section. More complete analysis remians to be performed.

We do not have a lower bound on processors needed for effectively perfect (ℓ, t, d)-SUSS, nor even for effectively perfect t-SUSS (the case in which all three parameters are the same). The proof of Theorem 4.1 relies on the fact that every vector of $2t$ shares is in a sense equally probable, so the t faulty players can always make up a vector of t shares that fits with those shares held by the coalition of pure processors, *whoever may be in the coalition.* Interestingly, it is possible to show that if the faulty processors only wish to lie

convincingly to a *particular* coaltion of pure gossips then it is possible to use polynomials of degree t (rather than of degree $2t-1$, as done in the proof of the Theorem), in which case $3t+1$ processors suffice.

Although our bound on effectively perfect t-SVSS is tight, in the general case there is a gap between the lower bound of $\ell+3d+1$ processors for effectively perfect (ℓ, t, d)-SVSS and the upper bound of $\ell+t+2d+1$.

Acknowledgements I am grateful to Stephen Ponzio for many hours of discussion. My original proof of effectively perfect secrecy for the protocol in Theorem 4.1 held only for static adversaries. Ponzio suggested the modification of $\mathcal{B}$ used to prove the result for dynamic adversaries. Discussions with Joe Kilian and Larry Stockmeyer were also helpful; I thank them both.

References

[1] D. Beaver, and S. Goldwasser, Multiparty Computation with Faulty Majority, *Proc. 30th Symp. on Foundations of Comp. Science*, pp. 468-473, 1989.

[2] 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.

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

[4] 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.

[5] B. Chor, and M. Rabin, Achieving Independence in Logarithmic Number of Rounds, *Proc. 6th Annual ACM Symp. on Principles of Distributed Computing*, pp. 260-268 (1987).

[6] C. Dwork, Strong Verifiable Secret Sharing, *to appear, Proc. 4th International Workshop on Distributed Algorithms (1990)*, Springer Verlag.

[7] D. Dolev, C. Dwork, O. Waarts, and M. Yung, Perfectly Secure Message Transmission, *Proc. 31st Annual Symposium on Foundations of Computer Science*, pp. 36-45 (1990).

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

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

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

[11] J. Halpern and Y. Moses, Knowledge and Common Knowledge in a Distributed Environment, *JACM 37*(3), pp. 549-587, 1990.

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

[13] 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.

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