An Authenticated Byzantine Generals Algorithm with Early Stopping

Kenneth J. Perry
Sam Toueg

TR 84-620
June 1984

Department of Computer Science
Cornell University
Ithaca, New York 14853

*Partial support for this work was provided by the National Science Foundation under grant No. 83-03135.

An Authenticated Byzantine Generals Algorithm with Early Stopping

Kenneth J. Perry
Sam Toueg

Department of Computer Science
Cornell University
Ithaca, New York 14853

ABSTRACT

A protocol that solves the authenticated Byzantine General's Problem is presented. It is proved correct and shown to exhibit early stopping under the condition that fewer than half the processes are faulty.

1. Introduction

Consider a collection of n distributed, potentially faulty processes able to communicate only by signed messages. A protocol which allows all non-faulty processes to agree on a value **m** chosen by a distinguished process *General* is said to solve the Byzantine Generals Problem if it satisfies the constraints:

(Validity):

If *General* does not fail then all non-faulty processes agree on **m**.

(Agreement):

All non-faulty processes agree on a common value.

The Byzantine Generals Problem has been explored extensively, under varying types of failures[DS83], [LF82], [LSP82], [PT84]. In this paper, we allow faulty processes to behave maliciously, that is, not only do faulty processes disobey the protocol, but they may actively work against it. We present a solution to this problem which exhibits *early stopping*[DS82b] under the conditions that messages are authenticated and t, the number of faulty processes, is less than $\frac{n}{2}$.

2. The model

Let P be the set $\{P_1, P_2, \cdots, P_n\}$ of processes participating in the protocol. Assume that a distinguished process $General \in P$ wants to transmit a binary value **m** to all others. We bound the number of faulty processes t at fewer than half the total (i.e. $t < \frac{n}{2}$).

Define a *phase* to be the interval of time in which each process is able to exchange messages with all others processes. Most algorithms for the Byzantine Generals Problem, ours included, are described by the actions taken in a sequence of phases.

Like most other work in this area, we shall assume that the processes communicate through a completely-connected, perfectly reliable message system in which all messages are correctly delivered. Processes are the only source of failure in our model.

With each process $P_i \in P$ we associate a unique *signature function* σ_i which transforms a message μ into *signed* message $\sigma_i(\mu)$. We shall assume that all processes have a means of extracting μ from $\sigma_i(\mu)$ and that no process (other than P_i) may derive σ_i from $\sigma_i(\mu)$. Non-faulty processes keep their signature functions private. Thus, non-faulty process P_i is the only process able to generate messages with signature σ_i. These assumptions are satisfied by a public-key cryptosystem such as the one described in [RSA78].

Finally, we say that process P_i has *decided* as soon as $decision_i \in \{0, 1\}$.

3. The EXCHANGE Protocol

The heart of our agreement algorithm is the two phase EXCHANGE protocol described in Figure 3.1. It will facilitate future elaboration if we view the two phases of EXCHANGE as comprising a single *round*. Round R takes place in phases $2R-1$ and $2R$.

Execution of EXCHANGE by non-faulty process P_i in round R allows P_i to receive a message $RECEIVED_i^R(k)$ from each $P_k \in P$ and to broadcast one of its own. Moreover, the protocol limits the ability of a faulty process to send contradictory values to different processes within a single round as follows. Suppose P_k is a faulty process. Then any two non-faulty processes will either receive the same value from P_k in round R or one will receive positive indication of P_k's fault.

We achieve these properties by forcing all round R messages to pass through intermediary *witnesses*. In order for P_k to broadcast μ in round R, it sends the signed triple $\tau \equiv \sigma_k(\langle Initial, R, \mu \rangle)$ to all processes in phase $2R-1$. Each process receiving τ becomes a witness to the fact that P_k initiated a broadcast of μ in round R. A witness P_j acknowledges its role by embedding τ into its own signed triple $e \equiv \sigma_j(\langle Echo, R, \tau \rangle)$ which it sends to all processes in phase $2R$. We call e an *echo of P_k sending μ in round R*.

At the close of phase $2R$, a process P_i accumulates all the messages received in that phase into a set $ECHOES_i^R$ which is used to determine the value accepted from each $P_k \in P$. P_i *accepts* μ from P_k in round R only if $ECHOES_i^R$ contains no echo of P_k

sending $\mu' \neq \mu$ in round R and at least $(t+1)$ echoes of P_k sending μ in round R. The message (if any) from P_k which is accepted by P_i is assigned to $RECEIVED_i^R(k)$; if no message is accepted, then the special message ϕ is assigned to $RECEIVED_i^R(k)$. Any process P_k, for which $ECHOES_i^R$ contains an echo of P_k sending μ' in round R but from whom P_i does not accept μ', is added to $FAULTY_i^R$, the set of processes whose faults have been observed by P_i through round R.

Finally, given a set of messages E, define $Witnesses(E, P_k, \mu, R)$ to be the set of processes from whom there are echoes in E of P_k sending μ in round R. Let $IsEcho(e, P_j, P_k, \mu, R)$ be a predicate which is true if and only if message e is sent by process P_j and e is an echo of P_k sending μ in round R:

$$IsEcho(e, P_j, P_k, \mu, R) \equiv e = \sigma_j(\langle Echo, R, \sigma_k(\langle Initial, R, \mu \rangle)\rangle)$$

Then

$$Witnesses(E, P_k, \mu, R) \equiv \{P_j \in P \mid \exists\, e \in E : IsEcho(e, P_j, P_k, \mu, R)\}$$

3.1. Properties of the EXCHANGE protocol

In this section we summarize the key features of the EXCHANGE protocol. First we prove that for every $P_k \in P$ and round R, each non-faulty process either accepts identical values from P_k or concludes that P_k is faulty.

Process P_i

EXCHANGE μ in round R =

{Physical phase $2R-1$}

```
if  μ ≠ φ → send σ_i(⟨Initial, R, μ⟩) to all P_k ∈ P
  [] μ = φ → skip
fi;
```

receive $received_i^{2R-1}(k)$ from each $P_k \in P$;

{Physical phase $2R$}

send $\{\sigma_i(\langle Echo, R, r\rangle) \mid \exists P_k \in P : r = received_i^{2R-1}(k) \wedge r \neq \phi\}$ to all $P_k \in P$;

receive $received_i^{2R}(k)$ from each $P_k \in P$;

$ECHOES_i^R := \bigcup_{P_k \in P} received_i^{2R}(k)$;

for each $P_k \in P$ do

```
if ∃ μ :( (|Witnesses(ECHOES_i^R, P_k, μ, R)| > t)
          ∧ (∄ μ' ≠ μ : |Witnesses(ECHOES_i^R, P_k, μ', R)| > 0) ) →
              RECEIVED_i^R(k) := μ
  [] ∄ μ :( (|Witnesses(ECHOES_i^R, P_k, μ, R)| > t)
          ∧ (∄ μ' ≠ μ : |Witnesses(ECHOES_i^R, P_k, μ', R)| > 0) ) →
              RECEIVED_i^R(k) := φ
fi;
```

$FAULTY_i :=$

$FAULTY_i \cup \{P_k \in P \mid \exists \mu : (|Witnesses(ECHOES_i^R, P_k, \mu, R)| > 0) \wedge (RECEIVED_i^R(k) \neq \mu)\}$

Figure 3.1

Lemma 3.1

Consider any three processes $P_i, P_j, P_k \in P$. For all rounds R, if $RECEIVED_i^R(k) \neq \phi$ then either $RECEIVED_j^R(k) = RECEIVED_i^R(k)$ or $P_k \in FAULTY_j^R$.

Proof

Suppose $RECEIVED_i^R(k) = r \neq \phi$. Then $|Witnesses(ECHOES_i^R, P_k, r, R)| > t$. Because there are at most t faulty processes, P_j has at least one witness in common with P_i so $|Witnesses(ECHOES_j^R, P_k, r, R)| > 0$. If P_j has a witness to any other $r' \neq r$, then $RECEIVED_j^R(i) = \phi$. Assume there is no such contradictory witness. There are two cases to consider.

($|Witnesses(ECHOES_j^R, P_k, r, R)| > t$):

Then $RECEIVED_j^R(k) = RECEIVED_i^R(k)$.

($0 < |Witnesses(ECHOES_j^R, P_k, r, R)| \leq t$):

By construction of EXCHANGE, $RECEIVED_j^R(k) = \phi$ and $P_k \in FAULTY_j^R$.

□

The broadcast of a non-faulty process is accepted by all non-faulty procesess, a fact stated (without proof) in the next lemma.

Lemma 3.2

Let $P_i, P_j \in P$ be non-faulty processes. If P_i executes the statement

EXCHANGE μ in round R

in round R, then $RECEIVED_j^R(i) = \mu$.

4. The Byzantine Generals protocol

We now describe the protocol for solving the Byzantine Generals Problem. The concept is quite simple. We bias the algorithm so that only messages containing the value 1 have any effect and decide on 0 only when it can be proven that *General* did not send 1 in round 1. This is achieved by requiring a signed *proof of progress*[DS82a] to accompany all messages containing 1. The proof will be sufficient to conclude that no non-faulty process has decided on 0. Our main task will be to show that once the first non-faulty process P_i (other than *General*) has decided, then each non-faulty P_j eventually arrives at the same decision.

For the case where P_i decides on 1 in round R, we simply require P_i to relay its decision to all processes in the next round. So long as $R < t+1$, this is achievable. One of our obligations will be to show that if P_i decides on 1 in the last round, then so does P_j.

Suppose instead that P_i decides on 0 in round R. To achieve agreement, we must insure that no P_j decides on 1 in R or any subsequent round. We therefore limit the situations in which P_i may decide on 0. Recall that EXCHANGE allows P_i to detect the fault of any process P_k whose message is accepted by P_j but not by P_i. We prohibit P_i

from deciding on 0 in any round in which it observes a *new* faulty process, for this fault may cause P_j to decide on 1. This is still not sufficient to guarantee agreement.

Suppose P_i detects no new fault in round R, decides on 0, and halts. If P_j were unconditionally permitted to believe each accepted message, it might decide on 1 since a faulty process might send 1 in round $R' > R$. We exclude this possibility by imposing a *validity* test upon all accepted messages. A message μ containing $V \in \{0, 1\}$ which is accepted by P_j in round R' is *validated* only if there is *proof* that messages containing V were sent (via EXCHANGE) by distinct processes in all rounds $R \leq R'$ (i.e. a proof of "progress"). No P_i could have decided 0 by round R' because the proof implies that P_i receives echoes from distinct processes in each round up to and including R'. The "proof", transmitted along with V, is the set of *Echo* messages from previous rounds which demonstrate that no non-faulty process could have halted. Messages therefore have some structure, which we now elaborate.

4.1. Message Structure

Define a *simple message* to be an element of {0, 1}. A *compound message* μ is a triple

$$\langle \mu.contents, \mu.originator, \mu.echoes \rangle$$

where $\mu.contents$ is a (simple or compound) message, $\mu.originator \in P$, and $\mu.echoes$ is a set of (simple or compound) messages. Henceforth, use of the term *message* will denote an instance of either simple or compound type.

Let μ be a message. Recursively define the p-fold selection on the fields of μ as follows:

$$\mu.contents^0 = \mu$$
$$\mu.contents^p = (\mu.contents^{p-1}).contents \quad \text{for } p > 0$$
$$\mu.originator^p = (\mu.contents^{p-1}).originator \quad \text{for } p > 0$$
$$\mu.echoes^p = (\mu.contents^{p-1}).echoes \quad \text{for } p > 0$$

The *value in message* μ is defined as:

$$ValueIn(\mu) = \mu \text{ for } \mu \text{ a simple message}$$
$$ValueIn(\mu) = ValueIn(\mu.contents) \text{ for } \mu \text{ a compound message}$$

4.2. Validating messages

The process of validating a message is now described. Validation of message μ is sufficient to insure that all non-faulty processes eventually decide on $ValueIn(\mu)$.

Define predicate VALID^R, which is true only of valid messages, as follows. Let μ be a message, $P_k \in P$ a process, E a set of messages, and O a set of processes.

$$\text{VALID}^1_R(\mu, P_k, \text{E}, \text{O}) = (P_k = General) \wedge (P_k \notin \text{O}) \wedge (\mu = 1) \wedge (|Witnesses(\text{E}, P_k, \mu, 1)| > t)$$
$$\text{VALID}^R(\mu, P_k, \text{E}, \text{O}) =$$
$$((P_k \notin \text{O}) \wedge (|Witnesses(\text{E}, P_k, \mu, R)| > t)$$
$$\wedge \text{VALID}^{R-1}(\mu.contents, \mu.originator, \mu.echoes, \text{O} \cup \{P_k\})) \quad \text{for } R > 1$$

4.2.1. Properties of validation

Essential properties of the validation process are described in this section. First, note that 1 is the only message which is VALID^1. Next, it is shown that if message μ is valid in round R, then the contents of μ are recursively valid in all prior rounds.

Lemma 4.1

Let μ be a message, E a set of echoes, O a set of processes, and $P_j \in P$ a process. If $\text{VALID}^R(\mu, P_j, \text{E}, \text{O})$ then for all rounds $0 < p < R$:

$$\text{VALID}^p(\mu.contents^{R-p}, \mu.originator^{R-p}, \mu.echoes^{R-p}, \text{O} \cup \{P_j\} \cup \bigcup_{R > q > p} \mu.originator^{R-q})$$

Proof

By induction on the definition of VALID^R. □

Next we prove that all processes receive at least one echo of a valid message.

Lemma 4.2

Let μ be a message, E a set of messages, O a set of processes, and $P_i, P_k \in P$ processes such that P_i is non-faulty. If $\text{VALID}^R(\mu, P_k, \text{E}, \text{O})$ then $|Witnesses(ECHOES_i^R, P_k, \mu, R)| > 0$.

Proof

By the definition of VALID^R, $\text{VALID}^R(\mu, P_k, \text{E}, \text{O})$ is true only if $|Witnesses(\text{E}, P_k, \mu, R)| > t$. Since at most t of the witnesses are faulty, $Witnesses(ECHOES_i^R, P_k, \mu, R)$ has at least one member in common with $Witnesses(\text{E}, P_k, \mu, R)$. □

4.3. The Byzantine Generals protocol

The protocol which solves the Byzantine Generals Problem is presented in Figure 4.1 and proved correct in this section.

A key lemma in the proof of our protocol's correctness follows. It shows that the condition under which a non-faulty process decides on 1 is sufficient to prevent any

```
Process P_i
  R := 0;
  if P_i = General → decision_i := m;
                     if m = 1 → NextBroadcast_i := m
                       [] m ≠ 1 → NextBroadcast_i := φ
                     fi
    [] P_i ≠ General → decision_i := '?';  NextBroadcast_i := φ
  fi;

  FAULTY_i := φ;  halted_i := false;

  do (R < t + 1) ∧ ¬halted_i →
    R := R + 1;
    EXCHANGE NextBroadcast_i in round R;

    if decision_i = '?' ∧ ∃ P_j ∈ P : VALID^R(RECEIVED_i^R(j), P_j, ECHOES_i^R) →
         decision_i := RECEIVED_i^R(j);
         NextBroadcast_i := ⟨RECEIVED_i^R(j), P_j, ECHOES_i^R⟩

      [] decision_i = '?' ∧ ∄ P_j ∈ P : VALID^R(RECEIVED_i^R(j), P_j, ECHOES_i^R) →
         NextBroadcast_i := φ;
         if |FAULTY_i| ≥ R → skip
           [] |FAULTY_i| < R → decision_i := 0;
                               halted_i := true
         fi

      [] decision_i ≠ '?' → halted_i := true
    fi

  od
```

Figure 4.1

other non-faulty process from deciding on 0. The essence of the proof involves combining the key properties of the VALID^R predicate. That is, μ is VALID^R implies that $\mu.originator^{R-p}$ is VALID^p for each prior round p, and that all processes receive an

echo of a VALID message. Thus, μ is VALID^R means that all non-faulty processes receive at least one echo in each round. Our proof shows the sources of these echoes to be distinct, therefore no non-faulty process could have decided on 0 up to round R.

Lemma 4.3

Consider the system of processes at the end of round $R < t+1$. Suppose there is an undecided correct process P_i. If there is a process P_j which decides on 1 in round R, then for all rounds p up to and including to R, $|FAULTY_i^p| \geq p$.

Proof

Suppose, at the end of round R, $decision_i \neq 1$ but P_j decides on 1. Because P_j decides on 1, there must be a process P_k such that $RECEIVED_j^R(k) = r$, $\text{VALID}^R(r, P_k, ECHOES_j^R, \phi)$, and $ValueIn(r) = 1$. For each R, we show that $P_k \in FAULTY_i^R$, $decision_i =$ '?', and in all rounds p prior to R

$$r.originator^{R-p} \in FAULTY_i^p$$

($R = 1$): Because *General* is the only process whose message may be VALID^1, $P_k = General$. Since $decision_i \neq 1$, $RECEIVED_i^R(k) \neq r$ (otherwise P_i would validate r as does P_j). By Lemma 3.1, $P_k \in FAULTY_i^R$. Therefore P_i makes no decision in round R so $decision_i =$ '?'.

($R > 1$): By Lemma 4.1, we know that for all rounds p prior to R:

$$\text{VALID}^p(r.contents^{R-p}, r.originator^{R-p}, r.echoes^{R-p}_{P_k \cup \bigcup^{R>q>p}} r.originator^{R-q})$$

Lemma 4.2 tells us that in round p, P_i gets at least one echo of $r.originator^{R-p}$ sending $r.contents^{R-p}$. But the validity or r together with the fact that P_i does not decide on 1 in round p implies

$RECEIVED_i^p(r.originator^{R-p}) \neq r.contents^{R-p}$. Therefore $r.originator^{R-p} \in FAULTY_i^p$ by the definition of $FAULTY_i^p$. Validity implies that all the originators of r are unique so in round $p < R$ there are at least p members of $FAULTY_i^p$. Thus P_i does not decide 0 in round p so $decision_i$ = '?' prior to round R.

Because P_i does not decide on 1 in round R, $RECEIVED_i^R(k) \neq r$ (otherwise P_i would validate r as does P_j). Hence $P_k \in FAULTY_i^R$. Since r is validated by P_j, P_k is distinct from all its prior originators. Therefore $|FAULTY_i^R| \geq R$.

□

Our protocol causes a non-faulty process P_i which decides on 1 in round $R < t+1$ to relay its decision in the next round. We have just shown that no non-faulty process could have decided on 0 up to round R. Thus, all non-faulty processes will accept P_i's message in $R+1$, validate it, and decide on 1. Therefore agreement is acheived, providing that the first non-faulty process decides on 1 by round $R < t+1$. We now prove that agreement is achieved if the first non-faulty process decides on 0.

Lemma 4.4

Suppose non-faulty process P_i decides on 0 at the end of round R. Then no non-faulty process ever decides on 1.

Proof

For the sake of contradiction, suppose P_j decides on 1 in round R'. Then P_j receives a message r in round R' from some process P_k such that $\text{VALID}^{R'}(r, P_k, ECHOES_j^{R'}, \phi)$ and $ValueIn(r) = 1$. There are three cases to consider.

$(0 < R' < R)$: Since P_j is non-faulty, in round $(R' + 1) \leq R$, P_i will receive $\langle r, P_k, ECHOES_j^{R'} \rangle$ from P_j which it will be able to validate. Thus P_i must decide on 1, a contradiction.

$(R' = R)$: Contradicts Lemma 4.3.

$(R < R' \leq t + 1)$: In the proof of Lemma 4.3 we showed that $r.originator^{R-p}$ is included in $FAULTY_i^p$ for all rounds p prior to R'. But then $|FAULTY_i^R| \geq R$, a contradiction since we've assumed that P_i decides on 0 in round R.

□

We are now able to prove that our protocol is a correct solution to the Byzantine Generals Problem.

Theorem 4.5

Consider a collection of n processes of whom at most $t < \frac{n}{2}$ are faulty. The algorithm of Figure 4.1 correctly solves the Byzantine Generals Problem. Furthermore, if only $f < t$ processes are faulty, then all non-faulty processes will halt by the end of round $f + 2$.

Proof

(Validity): Assume *General* is non-faulty.

If $\mathbf{m} = 1$, then by the end of the first round all processes receive and validate 1. In this case $f \geq 0$, all non-faulty processes decide on 1 in round 1 and halt after round 2.

If $\mathbf{m} \neq 1$, no non-faulty process may ever decide on 1 since no message containing 1 and signed by *General* is ever sent. Each $P_j \in P$ will halt by round $f+1$ since no non-faulty process is ever included in $FAULTY_j^R$.

(Agreement): Let P_i be the first non-faulty process (other than *General*) to decide. Assume P_i reaches its decision in round $R > 0$.

Suppose $decision_i = 0$. Then by Lemma 4.4, all non-faulty processes decide 0. As was stated above, only faulty processes are included in $FAULTY_j^R$ for all non-faulty $P_j \in P$. So if there are f faulty processes, P_j decides on 0 and halts in round $f+1$.

On the other hand, suppose $decision_i \neq 0$. By definition, 1 is the only message which may be VALID^1. Therefore all processes which do not decide on 0 must decide on 1.

Consider the following cases:

($R < t+1$): Then all non-faulty P_j will receive a message μ with $ValueIn(\mu) = 1$ from P_i in round $(R+1) \leq t+1$. P_j will validate μ, reach the same decision as P_i and halt within one more round. By Lemma

4.3, if P_j is undecided in round R, $f \geq R$. Thus, all processes will have halted by round $f + 2$.

($R = t + 1$): P_j must also decide in round R, otherwise by applying Lemma 4.3 we conclude that there are greater than t faulty processes, contradicting the bound on the number of faults.

□

5. An optimization

As presented, the bit complexity of our protocol is quite bad, in fact, it is exponential in f, the number of faults that actually occur. The reason is clear: when a non-faulty process decides, it sends a round R message containing $t+1$ round $R-1$ echoes. Thus, the echoes of round R are at least $t+1$ times as long as those of the previous round. Fortunately, the bit complexity may be reduced by a more intelligent means of transmitting the "proof of progress" (i.e. the set of echoes sent along with all messages containing 1).

Suppose we used the EXCHANGE protocol *only* to broadcast simple messages containing 1 and *directly* sent the proofs of progress to all processes (i.e. without witnesses). Represent a proof of progress as a $(t+1)$ by $(t+1)$ matrix whose elements are echoes of processes sending the simple message 1 in some round. Hence, each entry requires $O(\log n)$ bits. A simple message 1, broadcast by P_j, is validated by process P_i in round R if P_i accepts it (so there are $t+1$ echoes of P_j sending 1 in round R) and the proof of progress sent by P_j meets all of the following requirements. For each round $R' < R$,

row R' of the proof contains $t+1$ *different* echoes. The elements of row R' must all be echoes of the *same* P_k sending simple message 1 in round R'; call P_k the *relayer of round* R'. Furthermore, round R' must be the only one in which P_k is the relayer and *General* must be the relayer of round 1. Once validated, P_i adds P_j's round R echoes as the R^{th} row of the proof and sends it (directly) to all processes.

The reader may verify that the new matrix representation contains a proof of progress which satisfies all the conditions imposed upon the original. By not sending echoes of echoes, we reduce the bit complexity of a single proof of progress (ignoring empty entries) to $O(f \cdot t \cdot \log n)$. Each non-faulty process which decides on 1 may send one proof to each of the n processes, hence the total bit complexity of all proofs amounts to $O(n^2 \cdot f \cdot t \cdot \log n)$. The EXCHANGE initiated by each non-faulty process causes $O((n+n^2) \cdot \log n)$ bits to be sent for a total of $O(n^3 \cdot \log n)$ bits over all processes. Because f and t are $O(n)$, the total bit complexity of the optimized protocol is $O(n^4 \cdot \log n)$ at worst.

6. Conclusions

We presented an early-stopping solution to the Byzantine Generals Problem assuming authentication of messages and an upper bound t on faults equal to one half the number of processes. Our protocol terminates within $\min(f+2, t+1)$ rounds (or $\min(2f+4, 2t+2)$ phases) when f faults actually occur. Thus, the cost (in phases) is proportional to the resiliency actually needed.

The number of phases is slightly less than that used by the early stopping protocol of [DRS82] which does not require authentication. However, the main advantage of authentication is being able to withstand a greater number of faults than is achievable otherwise (i.e. $t < \frac{n}{3}$).

7. Postscript

The authors are currently working on improved early-stopping algorithms. We hope to report our results shortly.

8. References

[DRS82] Dolev, D., Reischuk, R., and Strong H.R. "'Eventual' Is Earlier Than 'Immediate'". *Proc. 23rd Symposium on Foundations of Computer Science*, Chicago, Illinois, 1982, pp. 196-203.

[DS82a] Dolev, D. and Strong, H.R. "Polynomial Algorithms for Multiple Processor Agreement". *Proc. 14th ACM Symposium on Theory of Computing*, 1982, pp. 401-407.

[DS82b] Dolev, D. and Strong, H.R. "Requirements for Agreement in a Distributed System". *Proc., the Second International Symposium on Distributed Data Bases*, Berlin, 1982.

[DS83] Dolev, D., and Strong, H.R. "Authenticated Algorithms for Byzantine Agreement". *SIAM Journal of Computing*, vol. 12, no. 4 (November 1983), pp. 656-665.

[LF82] Lamport, L., Fischer, M. *Byzantine Generals and Transaction Commit Protocols*. Opus 62, SRI International, April 1982.

[LSP82] Lamport, L., Shostak, R., and Pease, M. "The Byzantine Generals Problem". *ACM Trans. on Programming Languages and Systems*, vol. 4, no. 3 (July 1982), 382-401.

[PT84] Perry, K., Toueg, S. *The Byzantine Monkeys Problem*. Department of Computer Science Technical Report TR 84-610, Cornell University, May 1984.

[RSA78] Rivest, R., Shamir, A., and Adleman, L. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". *Comm. of the ACM*, vol. 21, no. 2 (Feb. 1978), pp. 120-126.