Optimal Resiliency against Mobile Faults

(Extended Abstract)

Harry Buhrman* Juan A. Garay† Jaap-Henk Hoepman*

CWI
P.O. Box 94070
1090 GB Amsterdam
The Netherlands

IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598
USA

CWI
P.O. Box 94070
1090 GB Amsterdam
The Netherlands

Abstract

In this paper we consider a model where malicious agents can corrupt hosts and move around in a network of processors. We consider a family of mobile-fault models $\mathtt{MF}(\frac{t}{n-1}, \rho)$. *In* $\mathtt{MF}(\frac{t}{n-1}, \rho)$ *there are a total of n processors, the maximum number of mobile faults is t, and their* roaming pace *is ρ (for example, $\rho = 3$ means that it takes an agent at least 3 rounds to "hop" to the next host). We study in these models the classical testbed problem for fault-tolerant distributed computing: Byzantine agreement.*

It has been shown that if $\rho = 1$, then agreement cannot be reached in the presence of even one fault, unless one of the processors remains uncorrupted for a certain amount of time. Subject to this proviso, we present a protocol for $\mathtt{MF}(\frac{1}{3}, 1)$, *which is optimal. The running time of the protocol is $O(n)$ rounds, also optimal for these models.*

1 Introduction

We consider a model where malicious agents can corrupt hosts and move around in a network of processors. The importance of this issue is rapidly growing with the current trend of universalization of computer networking and open systems, and, specially, with the advent of the so-called "agent technology." Previous works have, in one way or another, addressed the issue of dynamic faults. The following is a review of these results. Perhaps the first article to consider a model with malicious faults covering different fractions of the network at different points in time is due to Reischuk [14]. The author designed a (sub-optimal) Byzantine agreement protocol able to tolerate them as long as they remain stationary for a given interval of time. More recently, Ostrovsky and Yung [12] proposed randomized methods to withstand the attack of mobile viruses. Canetti and Herzberg [3] use cryptographic techniques to achieve secure computation in a system where different sets of servers may be broken into at different times, but not all at once. In [8], Franklin and Yung address the issue of privacy in a system with mobile eavesdroppers. Finally, Garay [9] presents agreement protocols that improve on the work of Reischuk [14].

However, one common characteristic in all the above results is that it is assumed that the faults (viruses, agents) are able to change position from one host to the next independently of when messages are sent in the network. In contrast, in this paper we make the more natural assumption that faults are able to "travel" from one processor to another *only when messages are sent.* This is in indeed the case with network viruses (see, for example, [10]).[1]

For this model, we evaluate the classical testbed problem for fault-tolerant distributed computing: Byzantine agreement (*BA*) [11]. Loosely stated, the problem requires processors in a communication network, some of which may be faulty, and all of which start the computation with an initial value, to decide on the same value. Moreover, it is required that if the initial value among the nonfaulty processors is the same, then that must be the decision value. In fact, a distinct characteristic of the mobile fault environments is that once a global state is reached, then special efforts must be dedicated in order to maintain it. This leads us to consider a variant of the problem (defined in Section 2) that is more adequate for these environments.

In this paper we review impossibility results for this model, and present a protocol that requires the optimal number of processors to achieve (and maintain) consistency in a network, in the presence of the "fastest" mobile faults. (Indeed, as pointed out in [9], the speed with which the faults can move is a fundamental parameters in this model.)

Finally, we emphasize that our model and work nec-

*Partially supported by the Dutch foundation for scientific research (NWO) through NFI Project ALADDIN, under contract number NF 62-376. E-mail: `{burman,jhh}@cwi.nl`.

†Work partially done while the author was visiting CWI. `garay@watson.ibm.com`.

[1] However, one important distinction is that we will assume that the total number of agents in the network is upper-bounded by a parameter t, as opposed to "combinatorial explosion"-type situations.

essarily represent an abstraction of things, in terms of the faults' power of disruption and coordination, as well as of the "clean" synchronous structure and "integral" fault speed assumption we derive our bounds on. The former assumptions we find necessary for the analysis of scenarios where things can really go wrong (e.g., life-critical applications; cryptographic settings; etc.). The latter simplifications we consider important for the understanding of the basic intricacies and issues involved in faulty-agent mobility; once the basic facts are established, similar results can hopefully be extended and derived with lesser efforts for variants and/or weaker versions of the model.

The rest of the paper is organized as follows. In Section 2 we define more formally the *Mobile Fault* model `MF`, as well as *MBA*, the version of the agreement problem that is adequate for these new environments. In Section 3 we present the negative results, while in Section 4 we present the protocol that tolerates the optimal number of fast-moving faults.

2 Mobile Fault Model

We are given a network of n processors numbered 1 through n that may communicate only by exchanging messages via a reliable point-to-point channel. In order to be able to measure and quantify more cleanly the issue of mobility, we shall consider the standard model (see, for example, [11]) of synchronous computation. In such a model, the network computation evolves as a series of *rounds*, during which processors can send one message to all other processors, receive the messages, and perform some local computation.

We assume that in the network there are (malicious) agents that can "corrupt" the processors. The semantics of corruption is as follows: In order to understand the worst possible disruption that can be caused in such a setting, we assume that the malicious agents are able to take complete control over the application being run, and even wipe out their memories. Furthermore, the agents are *mobile*, meaning that they can leave their current "hosts," and move on to "infect" new, different processors. Specifically, given the computation's round structure, we make the natural assumption that the agents will "travel" or move with the "Send" or "Broadcast" operations. We will also assume that the total number of agents in the network is bounded by a parameter t.

A consequence of the assumptions above is that, in the worst case, processors which are left by the agents and come "back to life" are unable to contribute in any meaningful way to the on-going computation, and will require the help of the currently correct processors in order to reconstruct the state of the execution (including the code of the protocol being executed!). This model is very powerful in terms of the agents' disruptive capabilities but, as mentioned before, we are interested in understanding what might be required in worst-case scenarios (e.g., life-critical applications, security settings, etc.). These assumptions can be weakened in several ways. For example, one could assume that some form of secure, tamper-proof memory is available.

We shall refer to these agents as *faults*, or simply *agents*. Given an assignment of the faults to the processors at any given round, we will draw from the virus terminology the term *cured* to refer to the processors that were occupied by the agent in the previous round, but no longer. We will refer to the remaining processors not possessed by the agents as *nonfaulty*, or *noninfected*.

We use the formalism of [9] to describe mobile fault environments. A fundamental parameter in such an environment is the *roaming pace* of the faults ρ, $[\rho] = \frac{\#\ rounds}{\#\ nodes}$, with which the agents can travel in the network. For convenience, we will normalize to the number of rounds that it takes to move from one processor/node to another. Thus, $\rho = 3$ indicates that it takes an agent 3 rounds to "hop" to the next nonfaulty processor. Further, for simplicity, we only consider the case of $\rho \in \mathbf{N}$, the set of natural numbers. We will sometimes refer to the case of $\rho = 1$ as "full speed," the case of $\rho = 2$ as "half speed," etc.

We can now more formally define the *Mobile Fault* model as a family `MF` of models indexed by the tuple $(\frac{t}{n-1}, \rho)$. In $\mathtt{MF}(\frac{t}{n-1}, \rho)$ there are a total of n processors, the maximum number of mobile faults is t, and their roaming pace is ρ. (Granted that at first sight, this notation is not the prettiest. However, it does allow us to express in a handy manner that, for example, the number of faults is less than a third—of the size of the network—and that they move at half speed.) Notice that, in contrast with the traditional models of static faults (for example, [13]) in which the only restriction is the bound on the maximum number of faults, there is no guarantee in the mobile-fault model that a processor that is initially correct will remain uninfected forever.

Intuitively, in this type of environment, as opposed to the traditional environment of faults that are static, things have more of a continuous, longer execution flavor, since one cannot expect the (currently) noninfected processors to reach meaningful conclusions after a few rounds of communication. This is particularly illustrated by the classical problem of achieving consistency in such an environment. Recall the *Byzantine Agreement* problem [11], which can be defined as follows. At the offset of the computation, each processor p holds an initial value $v_p \in \{0, 1\}$. Regardless of the behavior of the faulty processors, required is a distributed protocol satisfying the following conditions:

- **Decision:** Every nonfaulty processor eventually irreversibly decides on a value $d_p \in \{0, 1\}$.
- **Agreement:** The nonfaulty processors decide on the same value.
- **Validity:** If the initial values v_p of all nonfaulty processors are identical, then $d_p = v$ for all nonfaulty processors p.

However, notice that in `MF`, if there is a point in time when the three conditions are satisfied, the fact that the faults can continue to move along will make this global condition disappear. Thus, in this paper we consider *Mobile-Fault* Byzantine Agreement (*MBA*), a version of the problem that was presented in [9] that is adequate for the mobile-fault environment. *MBA* is defined similarly to the *BA* problem above, together with the additional condition:

- **Consistency Maintenance:** Once Agreement is achieved among the currently noninfected processors, it is preserved among the (possibly different) noninfected processors.

This condition captures the contrast of MF with respect to the traditional models, in that if there is a moment (interval) in time in which a consistency state is achieved among the currently noninfected processors, then special efforts will have to be dedicated to *maintaining* such a state, by having the processors that participated in the decision "pass the token" to the processors that re-join the computation after the agents have left them.

Albeit being more stringent, Consistency Maintenance resembles the correctness condition of *self-stabilizing* protocols (e.g., [4]). In our case, however, it is required that the correctness (consistency) state be maintained once it is reached, even if faults continue to occur. We will sometimes abuse the language and say that *MBA* is "achieved," meaning that consistency is reached, and from that point on, maintained.

3 Impossibility Results

It is well known that in the case of t static (arbitrary) faults, *BA* is achievable whenever the size of the network $n > 3t$ [11]. It turns out that in the case of faults that may move, there is a tight relationship between their speed (pace) and the number of faults that can be tolerated. The following, somewhat curious result from [9], also applies to our model where the agents can only move with the messages:

Proposition 3.1 [9] *In* $\mathtt{MF}(\cdot, 1)$*,* BA *can only be achieved if* $t = 0$.

The proof of this proposition is an extension of the lower bound of Dolev and Strong [6], which shows that $t+1$ rounds of communication are needed in the classical setting of static faults with a maximum of t faults. The basic idea is that by having a single mobile agent "hopping" from one processor to the next, one can construct executions that exhibit *serial faultiness*, in the sense that at each round a new processor misbehaves. Assuming now that *BA* can be achieved in k (not necessarily $< n$) rounds, techniques similar to those of [6] can be used to show that all executions are equivalent—in the sense of the currently noninfected processors deciding on the same value—to that in which the source does not send any value, a contradiction.

The above result is fairly extreme: Can't even tolerate one fault if it moves at full speed! One may wonder if the situation gets any better by slowing the faults down. This is what the following proposition—a generalization of Proposition 3.1—characterizes:

Proposition 3.2 [9] *In* $\mathtt{MF}(\cdot, \rho)$*,* BA *is possible only if* $t < \rho$.

Roughly speaking, the proposition is proved by the t agents implementing serial faultiness through coordination. Note that one can now view the case of $\rho \geq t+1$ as corresponding to the classical setting of static faults, in the sense that if the faults remain immobile for at least that long, then protocols exist that are guaranteed to yield agreement in that much time. Given Propositions 3.1 and 3.2, the following condition constitutes one way of overcoming the above difficulties:

($\mathcal{I}$) At least one processor remains uncorrupted.

In practice, condition $\mathcal{I}$ captures situations where not all places in the network are equally accessible to the faults. We also note that if the identity of the processor that cannot be corrupted by the faults is known beforehand to all processors, then the problem we are trying to solve becomes trivial. We will thus assume that this is not the case. The condition does not specify the length of the period during which the processor cannot be corrupted. In the next section we present a protocol that requires a processor to remain uncorrupted for $O(n)$ rounds of communication.

Another consequence of the $\mathcal{I}$ condition is that a direct application of the so-called full information protocols (e.g., [1, 11]) where all correct processors send their views to all other processors, and this for $t+1$ rounds, will not work here. An essential requirement for these protocols to work is that of the "pigeon-hole" principle: A time is needed where a "clean round" (i.e., no new faults) occurs [5], and this cannot be guaranteed given the mobility of the faults. Indeed, as pointed out in [9], in contrast with the classical setting of static faults, solutions to *MBA* necessitate a time proportional to n, the size of the system, as opposed to t. Specifically:

Corollary 3.3 *Every* MBA *protocol requires* n *rounds of communication in its worst-case run.*

Finally, the lower bound on the number of processors, as a function of the number of faults ($n > 3t$ [13]), also holds for MF systems satisfying $\mathcal{I}$. In the next section, we present a protocol for $\mathtt{MF}(\frac{1}{3}, 1)$ (i.e., optimal number of processors and faults af full speed) that relies on $\mathcal{I}$ to achieve *MBA* in $3n$ rounds.

4 Agreement Protocols for Mobile Faults

Given condition $\mathcal{I}$ and the observation from the previous section on the failure in MF of the classical full information protocols, a natural type of solution to look at is the *Phase King* paradigm of Berman and Garay [2], since its correctness relies on the eventual existence of a good processor (the "King"). The execution of a protocol adhering to this paradigm is divided into *phases*, each with a different king. In a phase, the following steps take place:

1. Round(s) of exchange of messages and computation among all processors; and

2. all processors listen to the King.

The purpose of 1 is to eliminate possible discrepancies in the configuration of existing values, and discover the unique value, if such exists. In 2, processors that are not "overwhelmingly convinced" of the existence of a unique value, trust the king of the phase and adopt his value. The reader is referred to [2] for further details.

However, a direct use of the existing Phase King protocols for the static-fault models is not possible, since even if unanimity is achieved by the currently correct processors at a given round, that state may disappear as the agents move along. This brings us to the concept of *reconstruction* of information, the subject of the next section.

4.1 Network Memory

The basic idea is to the use of the network as a collective memory device. Cryptographic settings—where not all parties can be trusted—are the natural places where these techniques originated (e.g., [15]). More recently, the concept of network memory has been utilized by Ostrovsky and Yung [12] and Garay [9]. The concept can be briefly formulated as follows. Let a network *object* $\mathcal{O}$ be defined by the tuple <*round number*, *processor id*, *data*>. Then, given a big enough fraction of noninfected processors at all times, it is possible for a correct processor to store $\mathcal{O}$ in the network by sending copies to all processors, maintain it for a period of time (say, k rounds), and reconstruct it whenever needed. In this section we will be interested in storing objects for just one round. Specifically, we establish the following simple fact for $\mathtt{MF}(\frac{1}{3}, 1)$:

Lemma 4.1 *Let p be a noninfected processor at round r. Then in $\mathtt{MF}(\frac{1}{3}, 1)$, every correct processor (cured or noninfected) can reconstruct at $r+1$ any object stored by p at r.*

Proof: In round r, p is correct, and stores m in the network by sending a copy of m to all processors. This message is received in round r by at least $2t+1$ correct processors: at least $t+1$ that remained correct since last round, plus at least t more, either because they remained correct, or because they became cured. In round $r+1$, all of these processors echo $\mathcal{O}$. The new object becomes:

$$\mathcal{O} = \begin{cases} < r, p, m > & \text{if received } m \text{ from at least} \\ & 2t+1 \text{ processors;} \\ < r, p, \perp > & \text{otherwise.} \end{cases}$$

Since in round $r+1$ at most t of the messages come from processors that were infected in round r, all the correct processors at round $r+1$ receive m with multiplicity at least $2t+1$, and are thus able to reconstruct m. ∎

Note that there is no guarantee that objects stored by infected processors will be univocally reconstructed, but this will be of no consequence to us. Lemma 4.1 can be generalized to reconstruct objects stored, say, k rounds back. Notice though that in MF, per the argument above, it is imperative that the copies of the object be continuously refreshed.

4.2 A Protocol with Optimal Resiliency

We are now ready to introduce MOPT, a protocol that is able to cope with mobile faults, for model $\mathtt{MF}(\frac{1}{3}, 1)$. The protocol is shown in Figure 1. It consists of *phases*, each consisting of three rounds of message exchange. There are three values that get transmitted at any given time: $\perp$ (for "undecided"), 0, and 1; we assume that $\perp < 0$. In "universal exchange 1," all the currently correct processors send their values V to all processors, and store the received values in vector MV. (We assume that the cured processors have the ability to receive messages. These would also include the protocol itself and the global state of the execution—e.g., round number—which we omit for clarity.) Each processor (both noninfected and cured) then checks if there exists a value $\in \{0,1\}$ that is very popular among the received values. If not, the processor stays undecided. The structure of "universal exchange 2" is similar to that of the first exchange, except that its purpose is the discovery of a unique value, if such exists. Again, it could be that in this exchange up to t agents decide to move to new hosts. The cured processors get up to speed by computing the values of $D[\cdot]$ based on the received MV's. In the "reconstruction and king's broadcast" exchange, processors re-send their MV's from the previous round, which they store in table *ECHO*. Processors that became cured in this round use that introspective ability to execute procedure RECONSTRUCT, shown in Figure 2. Finally, if a correct (cured or noninfected) is either still undecided, or not overwhelmingly convinced about the existence of a possibly unique value, it "listens" to processor k, the king of the phase. Since all processors are supposed to send their "view" (i.e., MV) from the last round, every processor is able to reproduce the king's computation (i.e., V_k). . Note that in MOPT the reconstruction procedure is only used in the last exchange.

We now address the correctness of the protocol. We first define the following potential function:

$$\Phi_v(r) \stackrel{\text{def}}{=} \text{the number of correct processors } p \text{ for which } V_p = v \text{ at the end of phase } r.$$

Observe that the Consistency Maintenance condition of *MBA* enforces on functions Φ the property that whenever they reach the "ceiling" of $n-t$, they should irrevocably remain there. The next lemma shows that this is indeed the case for MOPT. For simplicity we assume $n = 3t+1$.

Lemma 4.2 (Consistency Maintenance) *In $\mathtt{MF}(\frac{1}{3}, 1)$, if in phase $f \geq 0$ of* MOPT *$\Phi_v(f) \geq 2t+1$ holds for some $v \in \{0,1\}$, then $\Phi_v(r) \geq 2t+1$ holds at all phases $r > f$.*

Proof: Assume $\Phi_v(f) \geq 2t+1$ holds for some $v \in \{0,1\}$. In universal exchange 1 of phase $f+1$ at least $2t+1$ v's get sent, and since for each fault that moves there is a cured processor, each correct (cured or noninfected) processor has $C[v] = 2t+1 \geq n-t$ and $C[\overline{v}] \leq t < n-t$. Thus, all of these processors assign v to V.

protocol MOPT;
  $V := v_p$;
  for $r := 0$ **to** ∞ **do begin**
      (* universal exchange 1 *)
    broadcast(V);
    receive$(MV[\cdot])$;
    for $j := 0$ **to** 1 **do** $C[j] := \#$ of j's in MV;

$$V := \begin{cases} 0 & \text{if } C[0] \geq n-t \\ 1 & \text{if } C[1] \geq n-t \\ \perp & \text{otherwise} \end{cases};$$

      (* universal exchange 2 *)
    broadcast(V);
    receive$(MV[\cdot])$;
    for $j := \perp$ **to** 1 **do** $D[j] := \#$ of j's in MV;

$$V := \begin{cases} 0 & \text{if } D[0] > t \\ 1 & \text{if } D[1] > t \\ \perp & \text{otherwise} \end{cases};$$

      (* reconstruction and king's broadcast *)
    broadcast$(MV[\cdot])$;
    for each i **do**
      $ECHO[i,\cdot] := MV[\cdot]$ received from i;
    if *cured* **then**
      RECONSTRUCT;
    $k := (r \bmod n) + 1$; (* k is the phase's king *)
    if $(V = \perp$ **or** $D[V] < n-t)$ **then**
      $V := \max(0, V_k)$; (* V_k is value received from k *)
  end;

Figure 1: A protocol for $\mathtt{MF}(\frac{1}{3}, 1)$; code for proc. p.

In universal exchange 2, at least $2t+1$ processors send v, and again for at least $2t+1$ cured and noninfected processors $D[v] \geq n-t$ and $D[w] \leq t$, for $w \in \{\perp, \overline{v}\}$, assigning v to V.

During the reconstruction and king's broadcast exchange, of the $2t+1$ correct processors from the previous round, at least $t+1$ processors remain uninfected, and therefore ignore the king's broadcast. Of the remaining t correct processors from the previous round, again for each new infected processor there is a cured processor. Each cured processor executes procedure RECONSTRUCT, and it follows from Lemma 4.1 that each cured processor computes $D[v] = 2t+1 \geq n-t$, and $V = v$, thus also ignoring the king's broadcast. This yields at least $2t+1$ correct processors $V = v$ at the end of phase $f+1$, and thus $\Phi_v(f+1) \geq 2t+1$. Now the situation repeats itself, and we are done. ∎

Theorem 4.3 *In model* $\mathtt{MF}(\frac{1}{3}, 1)$ *satisfying condition* $\mathcal{I}$*, protocol* MOPT *achieves Mobile Byzantine Agreement in* $3n$ *rounds of communication.*

Proof: We first consider Validity. The configuration is unanimous at the start of phase 0 with at least $2t+1$ noninfected processors holding value $v \in \{0,1\}$. Similar arguments to those of Lemma 4.2 allows us to

procedure RECONSTRUCT;
  for each i **do**
    if $\exists\, v \in \{\perp, 0, 1\}$ s.t. $ECHO[\cdot, i] = v$ at least $n-t$ times **then**
      $MV[i] = v$
    else
      $MV[i] =$ ' ';
  for $j := \perp$ **to** 1 **do** $D[j] := \#$ of j's in MV;

$$V := \begin{cases} 0 & \text{if } D[0] > t \\ 1 & \text{if } D[1] > t \\ \perp & \text{otherwise} \end{cases};$$

  end;

Figure 2: Procedure RECONSTRUCT for $\mathtt{MF}(\frac{1}{3}, 1)$.

establish that $\Phi_v(0) \geq 2t+1$, and using Lemma 4.2 for the following phases yields the theorem.

We now address Agreement. Let I be the phase corresponding to the processor that cannot be corrupted by the faults. In particular, I will be sending the same MV in the "reconstruction and king's broadcast" exchange as it did in "universal exchange 2." Two cases are possible after the last exchange of phase I:

1. $D_p[V_p] < n-t$ or $V_p = \perp$ for every correct (cured or noninfected) p. Then all processors execute the assignment to V. As I is correct by assumption, all correct processors receive the same MV from I, and compute the same V_I. If $V_I = \perp$, they all assign 0 to V.

2. $D_p[V] \geq n-t$ and $V_p \neq \perp$ for at least some correct (cured or noninfected) p. Lemma 4.1 and the fact that there are no more than t corrupted processors guarantee that $|C_q[v] - C_r[v]| \leq t$ and $|D_q[v] - D_r[v]| \leq t$, for every non-faulty q, r and $v \in \{\perp, 0, 1\}$. Therefore, if a correct processor p computes $D_p[V_p] \geq n-t$ after the second universal exchange (or reconstructs and computes $D_p[V_p] \geq n-t$ during the reconstruction and king's broadcast), then $D_q[V_p] > t$ and $D_q[\overline{V_p}] \leq t$ for all other correct processors q, including the king I. As a result, each correct processor q has $V_q = V_I$ at the end of the phase, either because it ignores I's value, or because of accepting it.

In either case, all correct processors have the same value in V at the end of phase I, and Lemma 4.2 applies.

Regarding the round complexity, observe that I's turn becomes at phase n at the latest. ∎

5 Final Remarks

In this paper we have studied the problem of t malicious agents moving around in a network of n processors. In contrast with previous works, the agents in our model can only move when messages are sent in the network. Subject to the condition that one of the processors remains uncorrupted for a certain amount of time, we have presented a protocol to reach and maintain agreement among the uncorrupted processors that tolerates the maximal number of bad agents (namely, $t < \frac{n}{3}$), in $3n$ communication rounds. Notice the gap between the running time of our protocol and that of Corollary 3.3. We also point out that if randomization is allowed, then the above condition can be dropped, and techniques similar to those of [7] can be used in a straightforward manner to reach agreement in $O(1)$ expected time, but only tolerating up to $t < \frac{n}{6}$ mobile agents. Can this be improved?

Acknowledgements

The authors are thankful to Mark Moir for useful comments and suggestions. This work was partly carried out while J. Garay was visiting CWI. The author thanks the Center for its ambiance and hospitality.

References

[1] A. Bar-Noy, D. Dolev, C. Dwork and H.R. Strong, "Shifting gears: changing algorithms on the fly to expedite Byzantine Agreement," *Proc. 6th PODC*, pp. 42-51, August 1987.

[2] P. Berman and J.A. Garay, "Asymptotically Optimal Distributed Consensus," *Proc. ICALP 89*, LNCS Vol. 372, pp. 80-94, July 1989.

[3] R. Canetti and A. Herzberg, "Maintaining Security in the Presence of Transient Faults," *Proc. Advances in Cryptology—Crypto '94*, pp. 425-438, LNCS (839), Springer Verlag, August 1994.

[4] E.W. Dijkstra, "Self-stabilizing systems in spite of distributed control," *Communications of the ACM*, 17(11), pp. 643-644, 1974.

[5] C. Dwork and Y. Moses, "Knowledge and Common Knowledge in a in Byzantine Environment: Crash Failures," *Information and Computation*, Vol. 88, No. 2 (1990), pp. 156–186.

[6] D. Dolev and H.R. Strong, "Polynomial Algorithms for Multiple Processor Agreement," *Proc. 14th STOC*, pp. 401-407, May 1982.

[7] P. Feldman and S. Micali, "Optimal Algorithms for Byzantine Agreement," *Proc. 20th STOC*, pp. 148-161, May 1988.

[8] M. Franklin and M. Yung, "Eavesdropping Games: A Graph-Theoretic Approach to Privacy in Distributed Systems," *Proc. 34th FOCS*, pp. 670-679, November 1993.

[9] J.A. Garay, "Reaching (and Maintaining) Agreement in the Presence of Mobile Faults," *Proc. 8th International Workshop on Distributed Algorithms*, LNCS (857), Springer-Verlag, pp. 253-264, Terschelling (NL), September/October 1994.

[10] J. Kephart and S. White, "Directed-graph epidemiological models of computer viruses," *Proc. 1991 IEEE Computer Society Symp. on Research in Security and Privacy*, pp. 343-359, Oakland, CA, May 1991.

[11] L. Lamport, R.E. Shostak and M. Pease, "The Byzantine Generals Problem," *ACM ToPLaS*, Vol. 4, No. 3 (1982), pp. 382-401.

[12] R. Ostrovsky and M. Yung, "How to withstand mobile virus attacks," *Proc. 10th PODC*, pp. 51-59, 1991.

[13] M. Pease, R. Shostak and L. Lamport, "Reaching Agreement in the Presence of Faults," *JACM*, Vol. 27, No. 2 (1980), pp. 121-169.

[14] R. Reischuk, "A New Solution for the Byzantine Generals Problem," *Information and Control*, Vol. 64 (1985), pp. 23-42.

[15] A. Shamir, "How to share a secret," *CACM*, 22, pp. 612-613, 1979.