Games Computers Play: Game-Theoretic Aspects of Computing

Nathan Linial*

1 Introduction

Computers may interact in great many ways. A parallel computer consists of a group of processors which *cooperate* in order to solve large-scale computational problems. Computers *compete* against each other in chess tournaments and serve investment firms in their battle at the stock-exchange. But much more complex types of interaction do come up - in large computer networks, when some of the components fail, a scenario builds up, which involves both cooperation and conflict. The main focus of this article is on protocols allowing the well-functioning parts of such a large and complex system to carry out their work despite the failure of others. Many deep and interesting results on such problems have been discovered by computer scientists in recent years, the incorporation of which into game theory can greatly enrich this field.

Since we are not aware of previous attempts at a systematic study of the interface between game theory and theoretical computer science, we start with a list of what we consider to be the most outstanding issues and problems of common interest to game theory and theoretical computer science.

1. How does the outcome of a given game depend on the players' computational power?
2. Classify basic game theoretic parameters, such as values, optimal strategies, equilibria, as being easy, hard or even impossible to compute. Wherever possible, develop efficient algorithms to this end.
3. Theories of fault-tolerant computing: Consider a situation where a computational task is to be performed by many computers which have to cooperate for this purpose. When some of the computers malfunction, a conflict builds up. This situation generates a number of new game-theoretic problems.
4. The theories of parallel and distributed computing concern efficient cooperation between computers. However, game theoretic concepts are useful even in sequential computing.

*Institute of Computer Science, Hebrew University Jerusalem 91904 Israel. Work supported in part by the Foundation of Basic Research in the Israel Academy of Sciences. Part of this work was done while the author was visiting IBM Research - Almaden and Stanford University.

For example, in the analysis of algorithms, one often considers an adversary who selects the (worst-case) input. It also often helps to view the same computer at two different instances as two cooperating players.

This article focuses on the third item in this list. We start with some background on fault-tolerant computing and go on to make some brief remarks on the other three items in this list.

1.1 Distributed Processing and Fault Tolerance

A *distributed system* consists of a large number of loosely coupled computing devices (*processors*) which together perform some computation. Processors can communicate by sending messages to each other. Two concrete examples the reader may keep in mind, are large computer networks and neural systems (being thought of as networks consisting of many neurons). The most obvious difficulty is that data is available locally to individual processors, while a solution for the computational problem usually reflects a global condition. But distributed systems need also cope with serious problems of *reliability*. In such a large and complex setting components may be expected to fail: A data link between processors may cease passing data, introduce errors in messages or just operate too slowly or too fast. Processors may malfunction as well, and the reader can certainly imagine the resulting difficulties.

These phenomena create situations where failing components of the network may be viewed as playing against its correctly functioning parts. It is important to notice that the identity of failing components is not necessarily known to the reliable ones, and still the computational process is expected to run its correct course, as long as the number of failing components is not too large. Consequently, and unlike most game-theoretic scenarios, reliable or good players follow a predetermined pattern of behavior (dictated by the computer program they run) while bad players may, in the worst case, deviate from the planned procedure in an arbitrary fashion.

This is, then, the point of departure for most of the present article. How can n parties perform various computational tasks together if faults may occur? We present three different theories that address this problem: Section 2 concerns the Byzantine Agreement problem which is the oldest of the three and calls for processors to consistently agree on a value, rather a simple task. The main issues in this area are already fairly well understood. No previous background is needed to read this section.

A much more ambitious undertaking is surveyed in section 3. Drawing heavily on developments in computational complexity and cryptography, one studies the extent to which computations can be carried out in a distributed environment in a reliable and secure way. We want to guarantee that the computation comes out correct despite sabotage by malfunctioning processors. Moreover, no restricted information should leak to failing processors. Surprisingly satisfactory computational protocols can be obtained, under one of the following plausible assumptions concerning the means available to good players for hiding information from the bad ones. Either one postulates that communication channels are secure and cannot be eavesdropped, or that bad players are computationally restricted, and so cannot decipher the encrypted message sent by good players. This is still an active area of research. All necessary background in computational complexity and cryptography is included in the section.

Can anything be saved, barring any means for hiding information? This is the main issue in section 4. It is no longer possible to warrant absolutely correct and leak-free computation, so only quantitative statements can be made. Namely, bad players are assumed to have some desirable outcome in mind. They can influence the protocol so as to make this outcome more likely, and the question is to find protocols which reduce their influence to a minimum. The notion of influence introduced in this section is, of course, conceptually related to the various measures developed in game theory to gauge the power of coalition in games. Of all three, this theory is still at its earliest stage of development. This section is largely self-contained, except for a short part where (elementary) harmonic analysis is invoked.

Most of the research surveyed here has been developed from a theoretical standpoint, though some of it did already find practical application. A major problem in applying game theory to real life is the notorious difficulty in predicting the behavior of human players. This sharply contrasts with the behavior of a fault-free computer program, which at any given situation is completely predictable. This difference makes much of the foundational or axiomatic complications of game theory irrelevant to the study of interactions between computers. Two caveats are in order here: While it is true that to find out the behavior of a program on a *given* input, all we need to do is run the program on the input, there are numerous natural questions concerning a program's behavior on *unknown, general* inputs which cannot be answered. (Technically, we are referring to the fact that many such questions are *undecidable*). This difficulty is perhaps best illustrated by the well known fact that the *halting problem* is undecidable (e.g. [LP]). Namely, there is no algorithm, which can tell whether a given computer program halts on each and every possible input. (Or whether there is an input causing the program to loop forever). So, the statement that fault-free computer programs behave in a predictable way is to be taken with a grain of salt. Also, numerous models for the behavior of failing components have been considered, some of which are mentioned in the present article, but this aspect of the theory is not yet exhausted.

Most of the work done in this area supposes that the worst set of failures takes place. Game-theorists would probably favor stochastic models of failure. Indeed, there is a great need for interesting theories on fault-tolerant computing where failures occur at random.

1.2 What about the other three main issues?

Of all four issues highlighted in the first paragraph we feel that the first one, i.e., how does a player's computational power affect his performance in various games, is conceptually and intellectually the most intriguing. Some initial work in this direction has already been performed, mostly by game theorists, see surveys by Sorin [So] and Kalai [Ka]. It is important to observe that the scope of this problem is broad enough to include most of modern cryptography. The recent revolution in cryptography, begun with Diffie and Hellman's [DH], starts from the following simple, yet fundamental observation: In the study of secure communication, one should not look for *absolute* measures of security, but rather consider how secure a communication system is *against a given class of code breakers*. The cryptanalyst's only relevant parameter is his computational power, the most interesting case being the one where a randomized polynomial-time algorithm is used in an attempt to break a communication system (all these notions are explained in section 3). Thus we see a situation of conflict whose outcomes can be well understood, given the participants' computational power. A closer look into cryptography will probably provide useful leads to studying the advantage to computationally powerful players in other game playing.

Unfortunately, we have too little to report on the second issue - computational complexity of game-theoretic parameters. Some work in this direction is reviewed in section 5. A systematic study of the computational complexity of game-theoretic parameters is a worthwhile project which will certainly benefit both areas. A historical example may help make the point: Combinatorics has been greatly enriched as a result of the search for efficient algorithms to compute, or approximate classical combinatorial parameters. In the reverse direction, combinatorial techniques are the basis for much of the development in theoretical computer science. It seems safe to predict that a similar happy marriage between game theory and theoretical computer science is possible, too.

Some of the existing theory pertaining to problem (4) is sketched in section 5.4. Results in this direction are still few and isolated. Even the contour lines of this area have not been sketched yet.

1.3 Acknowledgments

This article is by no means a comprehensive survey of all subjects of common interest to game-theory and theoretical computer science, and no claim for completeness is made. I offer my sincere apologies to all those whose work did not receive proper mention. My hope is that this survey will encourage others to write longer, more extensive articles and books where this injustice can be amended. I also had to give up some precision to allow for the coverage of more material. Each section starts with an informal description of the subject-matter and becomes more accurate as discussion develops. In many instances important material is missing and there is no substitute for studying the original papers.

I have received generous help from friends and colleagues, in preparing this paper, which I happily acknowledge: Section 3 could not have been written without many discussions with Shafi Goldwasser. Oded Goldreich's survey article [Gr1] was of great help for that chapter, and should be read by anyone interested in pursuing the subject further. Oded's remarks on an earlier draft were most beneficial. Explanations by Amotz Bar-Noy and the survey by Benny Chor and Cynthia Dwork [CD] were very useful in preparing section 2.

Helpful comments on contents, presentation and references to the literature were made by Yuri Gurevich, Christos Papadimitriou, Moshe Vardi, Yoram Moses, Michael Ben-Or, Daphne Koller, Jeff Rosenschein and Anna Karlin and are gratefully acknowledged.

1.4 Notations

Most of our terminology is rather standard. Theoretical computer scientists are very eloquent in discussing asymptotics, and some of their notations may not be common outside their own circles. As usual, if f, g are real functions on the positive integers, we say that $f = O(g)$, if there is a constant $C > 0$ such that $f(n) < Cg(n)$ for all large enough n. We say that $f = \Omega(g)$ if there is $C > 0$ such that $f(n) > Cg(n)$ for all large enough n. The notation $f = \Theta(g)$ says that f and g have the same growth rate, i.e., both $f = O(g)$ and $f = \Omega(g)$ hold. If for all $\epsilon > 0$ and all large enough n there holds $f(n) < \epsilon g(n)$, we say that $f = o(g)$. The set $\{1, \ldots, n\}$ is sometimes denoted $[n]$.

2 Byzantine Agreement

One of the earliest general problems in the theory of distributed processing, posed in [LSP], was how to establish consensus in a network of processors where faults may occur. The basic question of this type came to be known as the "byzantine agreement" problem. It has many variants, of which we explain one. We start with an informal discussion, then go into more technical details.

An overall description of the byzantine agreement problem which game theorists may find natural is: How to establish *common knowledge* in the absence of a mechanism which can guarantee reliable information passing? In known examples, where a problem is solved by turning some fact into common knowledge, e.g. the betraying wives puzzle (see [MDH] for an amusing discussion of a number of such examples) there is an instance where all parties involved are situated in one location, the pertinent fact is being announced, thus becoming common knowledge. If information passes only through private communication lines, and moreover some participants try to avoid the establishment of common knowledge, is common knowledge achievable?

In a more formal setting, the byzantine agreement problem is defined by two parameters: n, the total number of players, and $t \leq n$, an upper bound on the number of "bad" players. Each player (a computer, or processor) has two special memory cells, marked *input* and *outcome.* As the game starts every input cell contains either zero or one, (not necessarily the same value in all input cells). We look for computer programs to be installed in the processors so that after they are run and communicate with each other, all "good" processors will have the same value written in their outcome cells. To rule out the trivial program which always outputs 1, say, we require that if all input values are the same, then this is the value that has to appear in all outcome cells. The set of programs installed in the processors along with the rules of how they may communicate is a *protocol* in computer science jargon. The difficulty is that up to t of the processors may fail to follow the protocol. Their deviation from the protocol can be as destructive as possible, and in particular, there is no restriction on the messages they send. Bad players' actions are coordinated, and may depend on data available to all other bad players.

The byzantine agreement problem comes in many different flavors, largely depending on which data is available to the bad players at various times. In this survey we consider only the case where the bad players have complete knowledge of each other's information and nothing else. Another factor needed to completely specify the model is timing. Here we assume synchronous action: there is an external clock common to all processors. At each tick of the clock, which we call a *round*, all processors pass their messages. Internal computations are instantaneous. For a comprehensive discussion of other models see [CD].

A protocol *achieves byzantine agreement* (or *consensus*) in this model, if the following conditions hold when all "good" players (i.e., those following the protocol) halt:

Agreement: All good players have the same value written in their outcome cells.

Validity: If all inputs equal v, then all good outcome cells contain v.

A host of questions naturally come up, of which we mention a few:

- Given n, what is the largest t for which byzantine agreement can be achieved?
- When achievable, how costly is byzantine agreement: How many rounds of communication are needed? How long need the messages be? What computational power need the processors have?
- The above description of the problem implicitly assumes that every two processors are linked and can exchange messages. How will the results change under different modes of communication, e.g., if processors can send messages only to their neighbors in some (communication) graph?
- What if simultaneity of action is not guaranteed, and processors communicate via an asynchronous message passing mechanism? Can consensus still be achieved?
- Does randomization help? Are there any advantages to be gained by letting processors flip coins? I.e. can agreement be established more rapidly, with less computation, or against a larger number of bad players? Are there any gains in relaxing the correctness conditions and only asking that they hold with high probability and not with certainty?
- How crucial is the complete freedom granted to bad processors? For example, if failure can only mean that at some point a player stops playing, does that make consensus easier to achieve?

2.1 Deterministic Algorithms for Byzantine Agreement

The earliest papers in this area where many of the problems were posed and some of the basic questions answered are [PSL, LSP]. A large number of papers followed, some of which we now briefly summarize along with review, commentary and a few proofs.

Theorem 2.1:

- *A byzantine agreement (BA) among n players of which t are bad can be achieved iff $n \geq 3t + 1$.*
- *BA can be established in $t + 1$ rounds, but not in fewer.*
- *If G is the communication graph, then BA may be achieved iff G is $(2t + 1)-$connected.*

We start with a presentation of a protocol to establish BA. In the search for such protocols, a plausible idea is to delay all decisions to the very end. That is, investigate protocols which consist of two phases: information dispersal and decision-making. The most obvious approach to information dispersal is to have all players tell everyone else all they know. At round 1 of such protocols, player x is to tell his initial value to all other players. In the second round, x should tell all other players what player y told him on round 1, for each $y \neq x$, and so on. This is repeated a certain number of times, and then all players decide their output bit and halt. It turns out that consensus can indeed be established this way, though such a protocol is excessively wasteful in usage of communication links. We save our brief discussion of establishing BA *efficiently* for later.

Lamport, Pease and Shostak [PSL] gave the first BA protocol. It follows the pattern just sketched: All information is dispersed for $t+1$ rounds and then decisions are made, in a way we soon explain. They also showed that this protocol establishes BA if $n > 3t$. Our presentation of the proof essentially follows [BDDS]. Explanations provided by Yoram Moses on this subject are gratefully acknowledged.

To describe the decision rule, which is, of course, the heart of the protocol, some notation is needed. We will be dealing with sequences or *strings* whose elements are names of players. Λ is the empty sequence, and a string consisting of a single term y is denoted y. The sequence obtained by appending player y to the string σ is denoted σy. By abuse of language we write $s \in \sigma$ to indicate that s is one of the names appearing in σ.

A typical message arriving at a player x during data dispersal has the form: "I, y_{i_k}, say that $y_{i_{k-1}}$ says that $y_{i_{k-2}}$ says that ... that y_{i_1} says that his input is v". Player x encodes such data by means of the function V_x, defined as follows: $V_x(\Lambda)$ is x's input, $V_x(y_5, y_3)$ is the input value of y_5 as reported to x by y_3, and in general, letting σ be the sequence $y_{i_1}, \ldots, y_{i_k}$, the aforementioned message is encoded by setting $V_x(\sigma) = v$. If the above message repeats the name of any player more than once, it is ignored. Consequently, in what follows *only* those σ will be considered where *no player's name is repeated.* No further mention of this assumption is made.

The two properties of V_x that we need are

$$\text{If } x \text{ is a good player, then } input(x) = V_x(\Lambda).$$

$$\text{If } x, y \text{ are good players, then } V_x(\sigma y) = V_y(\sigma).$$

The first statement is part of the definition. The second one says that a good player y correctly reports to (good) x the value of $V_y(\sigma)$ and x keeps that information via his data storage function V_x. If either x or y are bad, nothing can be said about $V_x(\sigma y)$.

The interesting part of the protocol is the way in which each player x determines his outcome bit based on the messages he saw. The goal is roughly to have the outcome bit represent the majority of the input bits to the good players. This simple plan can be easily thwarted by the bad players; e.g., if the input bits to the good players are evenly distributed, then a single bad player who sends conflicting messages to the good ones can fail this strategy. Instead, an approximate version of this idea is employed. Player x associates a bit $W_x(\sigma)$ with every sequence-with-no-repeats σ of *length* $|\sigma| \leq t+1$. The definition is:

$$|\sigma| = t + 1 \Rightarrow W_x(\sigma) = V_x(\sigma)$$

and for shorter σ:

$$W_x(\sigma) = majority\{W_x(\sigma y)\}$$

the majority being over all y not appearing in σ. Finally

$$outcome(x) = W_x(\Lambda).$$

The validity of this protocol is established in two steps:

Proposition 2.2: *If x and y are good players, then*

$$W_x(\sigma y) = V_x(\sigma y) = V_y(\sigma).$$

Proof: The second equality is one of the two basic properties of the function V, quoted above, so it is only the first equality that has to be proved. If $|\sigma y| = t+1$ it follows from the definition of W in that case. For shorter σ we apply decreasing induction on $|\sigma|$: For a good player z the induction hypothesis implies $W_x(\sigma yz) = V_z(\sigma y) = V_y(\sigma)$. But

$$W_x(\sigma y) = majority\{W_x(\sigma ys)\}$$

over all $s \notin \sigma y$ (good or bad). Note that most such s are good players: even if all players in σ are good, the number of bad $s \notin \sigma y$ is $\leq t$ and the number of such good s is $\geq n - |\sigma y| - t$, but

$$n - |\sigma y| - t > t,$$

because $|\sigma y| \leq t$, and by assumption $n \geq 3t+1$ holds. It follows that most terms inside the majority operator equal $V_y(\sigma)$ (s behaves like z) and so $W_x(\sigma y) = V_y(\sigma)$ as claimed. ∎

Observe that this already establishes the **validity** requirement in the definition of BA: If all inputs are ν, apply the lemma with $\sigma = \Lambda$ and conclude that if x, y are good, $W_x(y) = \nu$. In evaluating $outcome(x) = W_x(\Lambda)$ most terms in the majority thus equal ν, so $outcome(x) = \nu$ for all good x, as needed.

For the second proposition define σ to be *closed* if either (i) its last element is the name of a good player, or (ii) it cannot be extended to a sequence of length $t+1$ using only names of bad players. This condition obviously implies that Λ is closed. Note that if σ is closed and it ends with a bad name, then σs is closed for any $s \notin \sigma$: if s is good, this is implied by (i) and otherwise by (ii).

Proposition 2.3: *If σ is closed and x, y are good, then $W_x(\sigma) = W_y(\sigma)$.*

Proof: We first observe that the previous proposition implies the present one for closed strings which terminate with a good element. For if $\sigma = \mu z$ with z good, then both $W_x(\mu z)$ and $W_y(\mu z)$ equal $V_z(\mu)$. For closed strings ending with a bad player, we again apply decreasing induction on $|\sigma|$. If $|\sigma| = t+1$, and its last element is bad, σ not closed, so assume $|\sigma| \leq t$. Since σ's last element is bad, all its extensions are closed, so by induction, $W_x(\sigma s) = W_y(\sigma s)$ for all s. Therefore

$$W_x(\sigma) = majority\{W_x(\sigma s)\} = majority\{W_y(\sigma s)\} = W_y(\sigma).$$

∎

Since Λ is closed, $outcome(x) = W_x(\Lambda)$ is the same for all good players x, as required by the **agreement** part of BA.

This simple algorithm has a serious drawback from a computational viewpoint, in that it requires a large amount of data to flow ($\Omega(n^t)$ to be more accurate). This difficulty has been recently removed by Garay and Moses [MW], following earlier work in [MW] and [BG]:

Theorem 2.4: *There is an $n-$player BA protocol for $t < \frac{n}{3}$, which runs for $t+1$ rounds and sends a polynomial (in n) number of information bits.*

The proof is based on a much more efficient realization of the previous algorithm, starting from the observation that many of the messages transmitted there are, in fact, redundant. This algorithm is complicated and will not be reviewed here.

A protocol to establish BA on a $(2t+1)-$connected graph is based on the classical theorem of Menger [Me] which states that in a $(2t+1)$-connected graph, there exist $2t+1$ disjoint paths between every pair of vertices. Fix such a system of paths for every pair of vertices x, y in the communication graph and simulate the previous algorithm as follows: Whenever x sends a message to y in that protocol, send a copy of the message along each of the paths. Since no more than t of the paths contain a bad player, most copies will arrive at y intact and the correct message can be thus identified. New and interesting problems come up in trying to establish consensus in networks, especially if the underlying graph has a bounded degree, see [DPPU].

Let us turn now to some of the impossibility claims made in the theorem. That for $n = 3$ and $t = 1$ BA cannot be achieved, means that for any 3-players' protocol, there is a choice of a bad player and a strategy for him that prevents BA. The original proofs of impossibility in this area were geared at supplying such a strategy for each possible protocol and tended to be quite cumbersome. Fischer, Lynch and Merrit [FLM] found a unified way to generate such a strategy, and their method provides short and elegant proofs for many impossibility results in the field, including the fact that $t+1$ rounds are required, and the necessity of high connectivity in general networks.

Here, then, is the [FLM] proof that BA is not achievable for $n = 3, t = 1$. The proof that in general, $n > 3t$ is a necessary condition for achieving BA, follows the same pattern, and will not be described here.

A protocol Q which achieves BA for $n = 3, t = 1$ consists of six computer programs $P_{i,j}$ ($i = 1, 2, 3; j = 0, 1$) where $P_{i,j}$ determines the steps of player i on input j. We are going to develop a strategy for a bad player by observing an "experimental" protocol, $\hat{Q}$, in which all six programs, are connected as in figure 1 and run subject to a slight modification, explained below.

In $\hat{Q}$ all programs run properly and perform all their instructions, but since there are 6 programs involved rather than the usual 3, we should specify how instructions in the program are to be interpreted. The following example should make the conversion rule clear. Suppose that $P_{3,1}$, the program for player 3 on input 1, calls at some point for a certain message M to be sent to player 2. In figure 1, we find $P_{2,1}$ adjacent to our $P_{3,1}$, and so message M is sent to $P_{2,1}$. All this activity takes place in parallel, and so if, say, $P_{3,0}$ happens to be calling at the

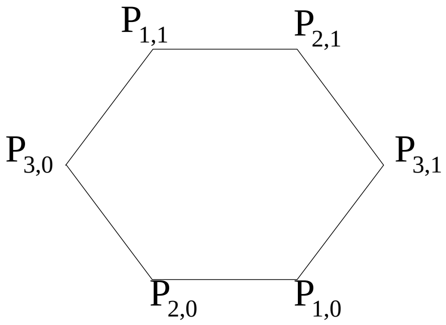

Figure 1: Protocol $\hat{Q}$

same time, for M' to be sent to player 2, then $P_{3,0}$ in $\hat{Q}$ will send M' to $P_{2,0}$, his neighbor in figure 1. Observe that that figure 1 is arranged so that $P_{1,j}$ has one neighbor $P_{2,\alpha}$, and one $P_{3,\beta}$ etc., so this rule can be followed, and our experimental $\hat{Q}$ can be performed. $\hat{Q}$ terminates when all six programs halt. A strategy for a bad player in Q can be devised, by inspecting the messages sent in runs of $\hat{Q}$.

We claim that if indeed, the programs $P_{i,j}$ always establish BA, then by the end of $\hat{Q}$ each $P_{i,j}$ reaches outcome j. E.g., suppose for contradiction, that $P_{3,1}$ does not reach 1. Then, we can exhibit a run of Q which fails to achieve BA, contrary to our assumption. Let Q run with good players 2 and 3 holding input 1 and a faulty player 1. Player 1 makes sure players 2 and 3 follow the same steps as $P_{2,1}$ and $P_{3,1}$ in $\hat{Q}$. He achieves this by sending player 2 the same messages $P_{1,1}$ sends to $P_{2,1}$ in $\hat{Q}$, and to 3 what $P_{1,0}$ sends $P_{3,1}$ there. This puts players $2, 3$ in exactly the same position as are $P_{2,1}, P_{3,1}$ in $\hat{Q}$, since they have the same input and the same exchanges with the neighbors. In particular player 3 fails to reach the outcome 1, and BA is not reached, a contradiction.

But now we can take advantage of $P_{1,0}$ deciding 0 and $P_{3,1}$ deciding 1 in $\hat{Q}$. Run Q with good players 1 and 3 holding inputs 0 and 1 respectively and a bad player 2. Let player 2 send player 1 the messages sent by $P_{2,0}$ to $P_{1,0}$ in $\hat{Q}$ and to player 3 those $P_{2,1}$ sends to $P_{3,1}$ there. This forces player 1 to a 0 outcome and player 3 to a 1 outcome, so no consensus is achieved, a contradiction.

Observe the similarity between the mapping used in this proof and *covering maps* in topology. Indeed, further fascinating connections between topology and the impossibility of various tasks in asynchronous distributed computation have been recently discovered, see [BoG], [SZ] and [HS].

The last remark of this subsection is that the assumed synchronization among processors is crucial for establishing BA. Fischer, Lynch and Patterson [FLP] showed that if message passing is completely asynchronous then consensus cannot be achieved even in the presence of a *single* bad player. In fact, this player need not even be very malicious, and it suffices for him to stop participating in the protocol at some cleverly selected critical point. For many other results in the same spirit see Fischer's survey [Fi].

2.2 Randomization in Byzantine Agreements

As in many other computational problems, randomization may help in establishing byzantine agreements. Compared with the deterministic situation, as summarized in Theorem 2.1, it turns out that a randomized algorithm can establish BA in expected time far below $t+1$. In fact the expected time can be bounded by a constant independent of n and t. It also helps defeating asynchrony, as BA may be achieved by a randomized *asynchronous* algorithm. The requirement that fewer than a third of the players may be bad cannot be relaxed, though, see [GY] for a recent discussion of this issue. Goldreich and Petrank [GP] show that these advantages can be achieved without introducing any probability that the protocol does not terminate, i.e., the best of the deterministic and randomized worlds can be gotten.

We briefly describe the first two algorithms in this area, those of Ben-Or [BO] and Rabin [Ra3] and then explain how these ideas lead to more recent developments. For simplicity, consider Ben-Or's algorithm for small t and in a synchronous set-up: Each processor has a variable called *current value* which is initialized to the input value. At each round every processor is to send its current value to all other participants. Subsequently, each processor has n bits to consider: its own current bit and the $n-1$ values related to him by the rest. Three situations may come up: *high,* where at least $\frac{n}{2}+2t$ of these bits have the same value, say b. In this case the player sets its outcome bit to b, it announces its decision on the coming round and then retires. *Medium:* the more frequent bit occurs between $\frac{n}{2}+t$ and $\frac{n}{2}+2t$ times. This bit now becomes the current value. In the *low* case the current value is set according to a coin flip. The following observations establish the **validity** and run time of this algorithm:

- If all input values are the same, then the algorithm terminates in one round.
- If in a certain round some good processor is in *high,* then every other good processor is either in *high* and commits on the same bit b or it is in *medium*, making b his current value, and so the algorithm terminates successfully in the next step.

Thus the expected run time of this algorithm is determined by how quickly "high" occurs. Now if $t = O(\sqrt{n})$ the expected number of rounds until some good processor goes into *high* is constant. (We have to wait for a random variable distributed according to a binomial law to exceed its expectation by a constant number of standard deviations.) Therefore for such values of t this is already a randomized BA algorithm with a constant expected run time. Is there such an algorithm also for larger t's ? Rabin's algorithm (found at about the same time as Ben-Or's) can be explained by modifying the previous algorithm as follows: If it were true that all the good processors at the *low* situation were flipping one and the same *collective* coin, then there would be no delay at all. This can be brought about by postulating the existence of a trusted referee who generates such global coin flips. Moreover, this scheme is seen to work in constant expected time even for $t = \Omega(n)$. However, the presence of a trusted referee makes the whole BA problem meaningless. Rabin observed that it suffices to sample a random source *before* the protocol is started, and reveal these random bits as needed. Again, revealing should take place without any help from an external trusted party. Such a mechanism for "breaking a secret bit into shares" and then revealing it when needed, is supplied by Shamir's secret sharing protocol [Sh1] to be described in section 3.2.5. An important feature of this protocol is that no

t of the players can gain any information about such a bit, while any $t+1$ can find it out with certainty. Given this protocol, the only problematic part in Rabin's protocol is the assumption that a trusted source of random bits is available prior to its run. This difficulty has been later removed by Feldman and Micali [FM] who show:

Theorem 2.5: *[FM] There is a* randomized *byzantine agreement algorithm with a constant expected run time.*

This algorithm may be perceived as performing Rabin's algorithm without a trusted dealer. Feldman and Micali achieve it by means of an elaborate *election* mechanism, whereby a somewhat-random player is elected by an approximate majority rule. Two important ingredients in this protocol, which show up again in section 3 are a *commitment* mechanism and the *verifiable* secret sharing method of [BGW], which is an improvement of Shamir's secret-sharing mechanism [Sh1].

3 Fault tolerant computation under secure communication

The Byzantine Agreement problem of the previous section is but one instance of the following general problems: How can the action of n parties (players, processors etc.) be coordinated in the absence of a trusted coordinator? How well can they perform various tasks which require cooperation when some of them may fail to perform their roles? An interesting class of examples the reader may want to consider, is that of playing certain games without a trusted referee. If all parties involved are reliable, then no problem arises, since any player may act as a coordinator. Are there any means of coordination when reliability is not granted? This study raises many technical as well as conceptual difficulties, so we begin with a few examples to guide the reader's intuition. Our final goal in this section is to survey the articles [GMW2] and [BGW], however, these papers build on many earlier interesting developments through which we first go. Space limitations prohibit proper mention and credit to the investigators who made significant contributions to the field. One should not fail, however, to mention Yao's contributions, in particular [Y1] as well as [Y2] where the two-player situation was resolved.

Models:

To pose concrete questions in this area, we need to make some assumptions on the behavior of bad players and on the means available to good players for concealing information. Keeping our discussion at an intuitive level, let us say this: bad players are either assumed to be *curious* or *malicious*. Curious players do perform their part in protocol, but try and extract as much side information as possible from exchanges between themselves and the good players, thus raising the difficulty of avoiding *information leaks*. Malicious players will do anything (as do bad players in the Byzantine Agreement problem). This assumption places the focus on *correct* computation in the presence of faults.

There are two models for how good players hide information:

Information-Theoretic: Secure communications lines exist between every two parties. No third party can gain any information by eavesdropping messages sent on such a line.

Alternatively, one postulates:

Bounded computational resources (Or cryptographic set-up): Participants are assumed to have a restricted computational power.

Here, then, are some concrete examples of our general problem.

- Secure Message Passing: Consider a three parties' protocol, where Sender needs to relay a single bit to Receiver without Eavesdropper's gaining any information. This special case of our problem is, clearly, a most fundamental problem of cryptography. It is trivial, of course, if secure channels are provided, but is interesting if the parties have a limited computational power. Stated otherwise, it asks whether bounds on computational power allow simulating a secure communication channel.

- The "and" function: Two players have a single input bit each, and they need to compute the conjunction (logical "and") of these two bits. This extremely simple task brings up some of the care that has to be taken in the formal statement of our definitions: Notice that a player whose input bit is 1 will know at the game's end the other player's input bit, while a player whose input bit is 0, knows that the conjunction of the two bits is 0. This observation should be taken into account in making the formal definition of "no information leaks are allowed". Secure channels do not help in this problem, but this task can be performed in the cryptographic set-up.

- The Millionaires' Problem: Two (curious) millionaires would like to compare their wealth, without revealing any information about how much money they have. In other words: is there a protocol which allows players P_1 and P_2 to find out which of the integers x_1, x_2 is bigger, without P_1 finding out any other information about x_2, and vice versa? Again, this is only interesting in a cryptographic set up and can be done if there is a commonly known upper bound on both x_1, x_2.

- Game playing without a referee: Can noncooperative games be played without a trusted referee? The definition of a correlated equilibrium in a noncooperative game depends on a random choice made by Fortune. Barany and Füredi [BF] show how $n \geq 4$ players $P_1, \ldots, P_n$ may safely play any noncooperative game in the absence of a referee, even if one of the players is trying to cheat. In our terminology the problem may be stated as follows: Let μ be a probability distribution over $A = A_1 \times \ldots \times A_n$ where A_i are finite sets. It is required to sample $a \in A$ according to μ in such a way that player P_i knows only the $i-$th coordinate of a but attains no information at all on a's other coordinates. It is moreover required that a player deviating from the protocol cannot bias the outcome and will be caught cheating with probability 1. As we shall see in this section, the above results can be strengthened, and the same will hold even if as many as $\lfloor \frac{n-1}{3} \rfloor$ players deviate from the rules of the game.

- Secure voting: Consider n voters, each of whom casts a yes/no vote on some issue. We would like to have a protocol which allows every player have exactly one vote and which eventually will inform all players on the outcome, while revealing nothing on votes of any individuals or subsets of players. Alternatively, we may ask that the tally be made known to all players. Or, we could ask that every female voter be told about the outcome among male voters or any other such task. As long as the number of bad players does not exceed a certain fraction of the total n, every such goal can be achieved.

Remark 3.1: Secure communication lines, may at first sight, seem superior to restrictions on faulty players' computational power. After all, what can cryptography buy us other than the freedom to communicate without being eavesdropped? This is, however, incorrect as indicated by the examples involving only two players. ∎

Remark 3.2: Most of our discussion concerns protocols that allow players to safely compute a function, or a set of functions together. The example of noncooperative game playing without a trusted referee, brings up a more general situation where with every list $x_1, \ldots, x_n$ of inputs to the players a probability space is associated, from which the players are to sample together a point z, and have player P_i find out a function $f_i(z)$, and nothing else. (When their goal is to compute a function f this space consists of the single point $f(x_1, \ldots, x_n)$. In the correlated equilibrium problem, $f_i(z)$ is the i-th coordinate of z.) The methods developed here are powerful enough to perform this more general task, in a private and fault-tolerant way. The changes required in the discussion are minimal and the reader can either supply them, or refer to the original papers. ∎

Let us go into more details on the two main models for bad players' behavior:

1. Unreliable participants are assumed to follow the protocol properly, except they store in memory all the messages they (legally) saw during the run of the protocol. Once computation is over, bad players join forces to extract as much information as possible from their records of the run. Bad players behaving in this way are sometimes said to be *curious*. There are certain quantities which they can infer, regardless of the protocol. Consider for example, five players each holding a single bit who compute a majority vote on their bits, and suppose that the input bits are $x_1 = x_4 = 0$, and $x_2 = x_3 = x_5 = 1$. If players 1 and 4 convene after finding out that $majority = 1$, they can deduce that all other x_i are 1. In saying that *no information leaks* in a protocol, it is always meant that only quantities such as this are computable by the unreliable players.

2. A more demanding model assumes nothing with regard to the behavior of the bad players (as in the Byzantine Agreement problem). Such behavior is described as *malicious* or *byzantine*. It is no longer enough to avoid the leak of classified information to the hands of bad players as in (1), for even the *correctness* of the computation is in jeopardy. What, for example, if we try to compute $f(x_1, \ldots, x_n)$ and player i refuses to reveal x_i which is known only to him? Or if, when supposed to compute a quantity depending on x_i and tell the outcome to other players, he intentionally sends an incorrect value, hoping no one will be able to detect it, since x_i is known only to him? If we momentarily relax the requirement for no information leak, then correctness can, at least in principle, be achieved through the following *commitment* mechanism: At the beginning of the protocol each player P_i puts a note in an envelope containing his value x_i. Then, all envelopes are publicly opened, and $f(x_1, \ldots, x_n)$ is evaluated. There are preassigned *default values* $z_1, \ldots, z_n$, and if player P_i fails to write a valid x_i on his note, then x_i is taken to be z_i.

Protocols described in this section perform just that, without using physical envelopes. Given appropriate means for concealing information, as well as an upper bound on the number of faults, it is possible to compute both correctly and without leaking information.

A complete description of a problem in this class consists of:

- A specification of the task which is to be performed together.
- An upper bound t for the number of unreliable players out of a total of n.
- The assumed nature of faulty players: curious or malicious.
- The countermeasures available: Either secure communication lines or a bound on faulty players' computational power.

A secure communication line is a very intuitive concept, though the technical definition is not entirely straightforward. To explain the notion of restricted computational power and how this assumption allows sending encrypted messages, the fundamentals of the theory of computing and modern cryptography have to be reviewed, as we later do. It is a key idea in cryptography that the security of a communication channel should not be quantified in absolute terms but rather expressed in terms of the computational power of the would-be code-breaker. Similar ideas have already been considered in game theory literature (e.g. bounded recall, bounded rationality etc.) but still seem far from being exhausted, see surveys by Kalai [Ka] and Sorin [So].

The main result of this section is that if the number t of unreliable players is properly bounded, then in both modes of failure (curious, malicious) and both safety guarantees (secure lines, restricted computational power) correct and leak-free computation is possible. A more detailed annotated statement of the main results surveyed in this section follows:

1. Given are functions $f; f_1, \ldots, f_n$ in n variables, and n processors $P_1, \ldots, P_n$ which communicate via secure channels and where each P_i holds input x_i known only to it. There is a protocol which, on termination, supplies P_i with the value $f_i(x_1, \ldots, x_n)$. The protocol is leak-free against any proper minority of curious players. I.e., for any $S \subseteq \{1, \ldots, n\}$ with $|S| \leq \lfloor \frac{n-1}{2} \rfloor$, every quantity computable based on the set of messages passed to any $P_j (j \in S)$ can also be computed given only the x_j and $f_j(x_1, \ldots, x_n)$ for $j \in S$.

   Also, the protocol computes correctly in the presence of any coalition S of $\leq \lfloor \frac{n-1}{3} \rfloor$ malicious players. That is, the protocol computes a value $f(y_1, \ldots, y_n)$ so that $y_i = x_i$ for all $i \notin S$ and where the $y_j (j \in S)$ are chosen independent of the values of $x_i (i \notin S)$.

   (Explanation: This definition captures the following intuition: in trying to compute $f(x_1, \ldots, x_n)$ there is no way to guarantee that unreliable players supply their correct input values. The best that can be hoped for is that they be made to *commit* on certain input values, chosen independent of the inputs to good players, and that after such a commitment stage, the computation of f proceeds correctly and without leaking information. In case a bad player refuses to supply an input, his will be substituted for by some default value).

The same results hold if the functions f_j are replaced by probability distributions and instead of computing the functions we need to sample according to these distributions, see remark 3.2. The bounds in the theorem are tight.

2. Assume that one-way trapdoor permutations exist.

   (This is an unproven statement in computational complexity theory which is very plausible, and whose proof is far beyond the power of current methods. Almost nothing can be presently proved in cryptography without this, or similar, unproven hypothesis. We assume that $1:1$ functions exist which are not hard to compute, but whose inverse requires infeasible amount of computation. See section 3.2.2 for more.)

   Modify the situation in (1) as follows: Channels are *not* secure, but processors are probabilistic, polynomial-time Turing Machines (this is theoretical computer scientists' way of saying that they are capable of any *feasible* computation, but nothing beyond). Similar conclusions hold with the bounds $\lfloor\frac{n-1}{2}\rfloor$ and $\lfloor\frac{n-1}{3}\rfloor$ replaced by $n-1$ and $\lfloor\frac{n-1}{2}\rfloor$ respectively. Again the bounds are tight and the results hold also for sampling distributions rather than for evaluation of functions.

Remark 3.3: Certain (but not all) randomized algorithms have the disadvantage that they may fail due to a "statistical accident". As long as the probability of such failure is exponentially small in n, the number of players, *it will henceforth be ignored.* ∎

For the reader's orientation, let us point out that in terms of performing various computational tasks, BA theory is completely subsumed by the present one. For example, the first part of theorem 2.1 is contained in theorem 3.15 below. In terms of efficiency, the present state of the theory leaves much to be desired. For more on efficiency see section 3.3.1.

3.1 Background in Computational Complexity

3.1.1 Turing Machines and the classes P and NP

At this point of our discussion a (very brief) survey of the fundamental concepts of computational complexity theory is in order, see [GJ] or [LP] for a thorough introduction to this area. A reader having some familiarity with computer science may safely skip this subsection. What we need now is a mathematically well-defined notion of a computer. This area is made up of two theories meant an answering some very fundamental questions:

computability theory: What functions, relations etc. can be mechanically computed?

complexity theory: Among computable functions which ones are easy and which are hard to compute?

What is needed is a definition powerful enough, to encompass all possible theoretical and physical model of computers. Also, a computer's performance certainly depends on a variety of physical parameters, such as speed, memory size, its set of basic commands, memory access time etc. Our definitions should be flexible enough so the dichotomy between easy and hard to compute functions does not depend on such technicalities. The most standard definition is that of a Turing Machine which we momentarily review. See remark 3.5 below for why this choice seems justified.

Computability theory flourished 40-60 years ago and most of its big problems have been resolved. Complexity theory, on the other hand, is one of the most active and important areas of present study in computer science. The dichotomy between hard and easy computational tasks is usually made according to the whether or not the problem is solvable in time bounded by a polynomial in the input length (a concept to be elaborated on in the coming paragraphs). As it turns out, this criterion for classifying problems as easy or hard does not depend on the choice of the model, and the classification induced by Turing Machines is the same as given by any other standard model.

Let us review the key features common to any computer the reader may be familiar with. It has a *control unit* which at each time resides in one of a (finite) number of *states*. There would also be *memory* and *input/output* devices. In the formal definition, memory and input/output units are materialized by an unbounded *tape* which initially contains the input to be processed. The tape may also be used for temporary storage, and will finally contain the output given by the machine. What the reader knows as the *computer program* is formally captured by the *transition function* which dictates what the new control state is, given the current state and the current symbol being read off the tape. Note that our definition differs from a regular computer, looking like a *special-purpose* computer, capable of performing only a single function. A computer which uses no software and where all behavior is determined by the hardware (i.e, the transition function). Indeed one of the early conceptual achievements of computability theory was the observation that the definition of a Turing Machine is flexible enough to allow it to be *programmed*. Thus a Turing Machine is a satisfactory mathematical abstraction of a general purpose computer. Let us emphasize that the universal Turing Machine does not require to be separately defined and is just an instance of a Turing Machine, according to the general definition.

We now turn to the formal definition. A Turing Machine M consists of:

- A finite set K of *states*, one of which, s, is called an *initial* state, and an extra *halting* state h, not in K.
- A *transition function* δ from $K \times \{0, 1, \#\}$ to $(K \cup \{h\}) \times \{0, 1, \#, L, R\}$, ($\#$ is to be thought of as the blank symbol.)

The machine M operates on an *input* x, which is a finite string of 0's and 1's, whose *length* is denoted by $|x|$. The input is augmented on the left and on the right by an infinite sequence of $\#$'s. The augmented sequence is called the *tape*, which may be modified during M's run. A *run* formally depicts how a computation of M evolves. It is a sequence, each element of which consists of (M's *current*) state, a tape and a symbol in it (currently *scanned* by M.) The run starts with M in state s and scanning the leftmost bit of the input. If M is in state q, currently scanning a and $\delta(q, a) = (p, b)$, then M's next state is p. Now if $b = R$ (resp. L), the right (left) neighbor of the currently scanned symbol is scanned next, while if b is $0, 1$ or $\#$, the tape is modified, the scanned a is replaced by b, which is the next scanned symbol. The process terminates when M first enters state h, and the number of steps until termination is called the *running time* (of M on x). At termination there is a shortest segment of the tape where 0 and 1 may be found, all other symbols being #. This finite segment is called the *output* of M when run on x. The length of the interval including those positions which are ever scanned during the run is called the *space* (utilized by M when run on x.) Sometimes there is need to talk about machines with more than one input or more than one output, but conceptually the definitions are the same.

A *decision problem* Q is specified by a *language* L i.e., a set of finite strings of $0'$s and $1'$s (the strings are finite, but L itself is usually infinite). The output of M on input x should be either 1 or 0, according to whether x is in L or not. The main concern of computational complexity is the running time of M on inputs x *as a function of their length.* The "dogma" of this area says that Q is feasibly solvable if there is an M which solves the decision problem in time which is bounded above by some polynomial function of the length $|x|$. In such a case Q (or L) is said to be in the class **P**, (it is *solvable in polynomial time*).

One also makes similar definitions for *randomized* Turing Machines. Those make their transition based on coin-flipping (Each current state and current scanned symbol, define a probability distribution on the next step. The new state, whether to move left or right, or to modify the tape are decided at random according to this distribution.) Running time becomes a random variable in this case, and there is room for more than one definition for "random polynomial time". The simplest definition requires the expected running time to be bounded above by a polynomial in the input length, see [J] for more details.

There is another, less obvious, complexity measure for problems, leading to the definition of the class **NP**. Even if Q is not known to be solvable in polynomial time, how hard is checking the validity of a proposed solution to the problem? The difference is analogous to the one between an author of a paper and a referee. Even if you cannot solve a problem, still you may be able to check the correctness of a proposed solution. A language L is said to be in **NP** (the corresponding decision problem is solvable in *nondeterministic polynomial time*) if there is a Turing Machine M with two inputs x and y such that for every $x \in L$ there is a y such that after time polynomial in $|x|$, M outputs 1, but if $x \notin L$, then no such y exists. The string y can be thought of as a "proof" or *witness* as the usual term is, for x's membership in L. Extensive research in classifying natural computational problems yielded a list of a few thousand problems which are in **NP** and are *complete* in it, namely, none of these problems is known to be in **P**, but if any one of them is in **P**, then *all* of them are. This is usually taken as evidence that indeed **P** and **NP** are different. Whether or not **P** and **NP** are equal is generally considered the most important question in theoretical computer science. The standard text on the subject is [GJ].

Example 3.4: Graph 3-Colorability: The following language is **NP**-complete: It consists of all finite graphs G whose set of vertices $V = V(G)$ can be partitioned into three parts $V = \bigcup_{i=1}^{3} V_i$ such that if two vertices x, y belong to the same V_i, then there is no edge between them. ∎

Remark 3.5: We close this subsection by mentioning the *Church-Turing Thesis* which says that all computable functions are computable by Turing Machines. This is not a mathematical statement, of course, but rather a summary of a period of research in the 1930's where many alternative models for computing devices were proposed and studied. Many of these models turned out to be equivalent to Turing Machines, but none were found capable of performing any computations beyond the power of Turing Machines. This point will again come up later in the discussion.

Algebraic circuits are among the other computing devices alluded to above. An algebraic circuit (over a field $\mathbf{F}$) is, by definition, a directed acyclic graph with four types of nodes: input nodes, an output node, constant nodes and gate nodes. Input nodes have only outgoing edges, the output node has only one incoming edge, constant nodes contain fixed field elements, and only have outgoing edges. Gate nodes are either $+$ gates or $\times$ gates. Such a circuit performs algebraic computations over $\mathbf{F}$ in the obvious way. Each input node is assigned a field element and each gate node performs its operation (addition or multiplication) on the values carried by its incoming edges. Finally, the value of the computed function resides in the output node.

Boolean circuits are similar, except they operate on "true" and "false" values and their gates perform logical "and", "or" and "not" operations.

We shall later interpret Church's Thesis as saying that any computable function is computable by some algebraic circuit. Such circuits can clearly simulate Boolean circuits, which are the model of choice in computational complexity. ∎

3.2 Tools of Modern Cryptography

The reader may be aware of the transition that the field of cryptology has been undergoing in recent years, a development that was widely recorded in the general press. It became a legitimate area of open academic study which fruitfully interacts with number theory and computational complexity. Modern cryptography is not restricted to the design of codes for secure data transmission and code-breaking as had been the case since antiquity. It is also concerned with the design of *cryptographic protocols.* The problem in general is how to perform various transactions between mutually untrusting parties, for example how can you safely sign agreements over the telephone, or take a vote in a reliable way over a communication channel or any other similar transactions of commercial or military importance. See [Ri] and and [Gr2] for recent surveys.

Such problems certainly are in the scope of game theory. It is therefore surprising that a formal connection between game theory and cryptography is still lacking.

In the coming paragraphs we briefly review some of the important developments in this area, in particular those which are required for game playing without a trusted referee and reliable computation in the presence of faults. It is crucial to develop this theory in a rigorous way, but lack of space precludes a complete discussion. The explanations concentrate more, therefore, on general ideas than on precise details and the interested reader will have to consult the original papers for full details.

3.2.1 Notions of Security

Any decent theory of cryptography must start by defining what it means for a communication system to be secure. A conceptually important observation is that one should strive for a definition of the security of a cryptosystem *against a given class of code-breakers.* The most reasonable assumption would be that the cryptanalyst can perform any "feasible" computation. The accepted view is that "feasible" computations means those carried by a randomized polynomial-time Turing Machine, and so this is the assumption we adopt.

There is a variety of notions of security that are being considered in the literature, but the general idea is one and the same: *Whatever a randomized polynomial-time Turing Machine can compute, given a ciphertext, should also be computable without it.* In other words, the ciphertext should be indistinguishable from an encryption of purely random bits to such a machine. Among the key notions in attaining this end are those of a one-way function and of hard-core bits which are introduced now.

3.2.2 One-Way Functions and their Applications

Most computer scientists believe that $\mathbf{P}\neq\mathbf{NP}$, or that "the author's job is harder than that of the referee". However, this important problem is still very far from being resolved. If this plausible conjecture fails, then there is no hope for a theory of cryptography in any sense known to us. In fact, to lay the foundations for modern cryptography, researchers had to recourse to even more powerful hypothesis. At the early days of the subject, those were usually assumptions on the intractability of certain computational problems from number theory. For example, factoring numbers, checking whether an integer is a quadratic residue modulo a nonprime integer N and so on.

An ongoing research effort is being made to weaken, or eliminate the unproven assumption required, and in particular to substitute "concrete" unproven hypothesis by more general, abstract counterparts. At present, large parts of known cryptography are based on a single unproven assumption, viz., that *one way functions exist.* This idea originated with Diffie and Hellman [DH] in a paper that marks the beginning of modern cryptography. Its significance for cryptographic protocols was realized shortly after, in [GoM]. Intuitively, a polynomial-time computable function f from binary strings to binary strings, is one-way, if any polynomial time Turing Machine which tries to invert it must have a very small probability of success. (The probability space in which this experiment takes place is defined by x being selected uniformly, and by the random steps taken by M). Specifically, let M be a probabilistic, polynomial-time Turing Machine and let n be an integer. Consider the following experiment where a string x of length n is selected uniformly, at random, and M is run on input $f(x)$ (but see remark 3.8 for some pedantic technicality). This run is considered a success, if the output string y has length n and $f(y) = f(x)$. The function f is said to be *(strongly) one-way* if every M has only a negligible chance of success, i.e., for every $c > 0$, if n is large enough, then the probability of success is smaller than n^{-c}.

- Certain restricted classes of one-way functions are sometimes required. Since we do not develop this theory in full, not all these classes are mentioned here, so when encountered in a theorem, the reader can safely ignore phrases like "non-uniform one-way functions" etc. These should be considered technicalities that are required for the precise statement of the theorem. For example, sometimes we need f to be a permutation, i.e., a $1:1$ mapping between two sets of equal cardinality. Another class of one-way functions which come up sometime, are those having the trapdoor property. Rather than formally introducing this class, let us present the canonical example:

Example 3.6: There is a randomized polynomial-time algorithm ([LeL] and the references therein) to test whether a given number is prime or not. The expected run time of this algorithm on an n digit number is polynomial in n. By the prime number theorem (and in fact, even by older estimates of Tchebyshef, e.g. [HW]), about one out of any n integers with n digits is prime. It is, therefore, possible to generate, in expected time polynomial in n, two primes p, q with n digits each. Let $N = pq$, and for $0 \leq x < N$, let $f(x, N) = (y, N)$ with $0 \leq y < N$ congruent to x^2 modulo N. It is not hard to see that any algorithm to invert f (i.e., take square roots $\bmod N$) can be used to factor N. Now, no polynomial time algorithms are known for factoring integers, and if, indeed, factoring cannot be done in polynomial time, then this f is one-way. Additionally, it is *trapdoor*, in the sense that there is a short piece of information, namely the primes p and q given which, f may be inverted in polynomial time. ∎

Example 3.7: Another example of a function which is generally believed to be one-way is the discrete log. It is a basic fact from algebra that the multiplicative group of a finite field is cyclic. Namely, there is a *generator* g such that every nonzero field element x has a unique representation of the form $x = g^k$ for some integer k. This k is the *discrete log* of x (to base g). While it is easy to evaluate g^k, given g, k, it is believed to be computationally infeasible to find k given x and g. More on computation in finite fields and computational number theory, can be found in [LeL]. ∎

Remark 3.8: Technically we need to assume in the definition of one-way functions, that M is run on $f(x)$ concatenated with a string of n $1'$s, to allow M run in time polynomial in n, because the length $|f(x)|$ may be much smaller than $n = |x|$. ∎

The applications of one-way functions to cryptography are numerous. In the present discussion they are employed for creating "envelopes". The idea is to enable two players A and B to perform the following task in secure message passing: A has a message m to transmit to B. An encrypted version $C(m)$ of this message is to be sent to B, so that the following two conditions are met: Given only $C(m)$, B cannot discover any information about m. Secondly, A is *committed* to the message m, in the following sense: at a later stage, A can prove to B that indeed m is the message sent, but B is able to call A's bluff in case A tries to prove the false claim that the message sent is any $m' \neq m$.

To create envelopes, one more ingredient is required besides one-way functions, namely, *hard-core bits*. The idea is this: a one-way function is easy to compute, and hard to invert. But while the definition requires that inverting f on x be hard, it does not rule out the possibility that we do gain *some* information on $f^{-1}(x)$. Hard-core bits b map the domain of f into $0, 1$ so that for every z in the range, b is constant on $f^{-1}(z)$. The condition is that given $f(x)$, it is hard to compute $b(x)$. For example, we may be claiming that not only is it hard to find x given $f(x)$, it is not even feasible to tell whether, the seventh bit of x is zero or one. Technically, let f be a function from binary strings to binary strings, (to be thought of as a one-way function). A polynomial-time computable boolean function b which is constant on $f^{-1}(z)$ for all z, (e.g. $b(x)$ may be the seventh bit of x) is said to be a *hard-core bit for* f if for every randomized polynomial-time Turing Machine M, any $c > 0$ and a large enough n, the probability that $M(f(x)) = b(x)$ is smaller than $\frac{1}{2} + n^{-c}$, where $n = |x|$, the probability being over M's random steps. (I.e., if you are asked for the seventh bit of x, given $f(x)$ and your computational power is restricted, you might as well guess by tossing a coin). It would make sense to assume that if f is one-way, there should be certain predicates b on x that cannot be computed in polynomial time with any advantage, given $f(x)$. The following theorem shows almost this. It is shown that any one-way function f can be slightly modified to yield another one-way function $\tilde{f}$ for which there is a hard-core bit. The concatenation of two bit strings $\hat{x}, \hat{y}$ is denoted $\hat{x}\hat{y}$, and if they have the same length, then $< \hat{x}, \hat{y} >$ is their inner product (as vectors over the field of two elements)

Theorem 3.9: *[Goldreich-Levin] Let $f(\cdot)$ be a one-way function and let $\tilde{f}(\hat{x}, \hat{y}) = f(\hat{x})\hat{y}$, then $\tilde{f}$ is one-way and $b(\hat{x}, \hat{y}) =< \hat{x}, \hat{y} >$ is a hard-core bit for it.*

In other words, if I show you $f(x)$ and y, and ask you for the inner product of x and y, you might as well flip a coin to guess an answer. Any significant increase in the probability of success over 1/2 requires super-polynomial computations. We won't say much about the proof, except explain, that given a program which can tell the inner product of x and y, with rate of success bounded away from 1/2, such a program can be utilized to invert f on a nonnegligible fraction of its range. A recent proof of this theorem using Harmonic Analysis (as in the proof of Theorem 4.4) can be found in [KuM].

Given this theorem, envelopes are easily constructed: Suppose f is a one-way function for which b is a hard-core bit. We make the extra simplifying assumption that f is one-to-one (this assumption may be relaxed, see Naor [Na]). To "place a bit β in an envelope" do as follows: Pick a random x in the domain of f for which $b(x) = \beta$, (such an x may not be obtained on first try, but on average two random tries will do) and let the receiver know the value of $f(x)$. When the time to reveal β comes, tell the receiver the value of x. The following properties of the protocol are easy to verify:

- The receiver's chance of finding β before x is revealed, is only $\frac{1}{2} + o(1)$, because b is hard-core.
- The sender cannot make false claims on β's value because f is one-to-one.
- Finally, the process is computationally feasible, since f and b are polynomial-time computable.

3.2.3 Interactive Proofs

Let us return for a moment to the definition of the class **NP**. The situation can be depicted as a two-person game, between a *Prover* P and a *Verifier* V, who wants to solve a decision problem: a language L and a binary string x are specified and the question is whether x belongs to L. The Verifier is computationally bounded, being a probabilistic polynomial-time Turing Machine, and so, possibly incapable of solving the problem on his own. The Prover is computationally unrestricted, and so if x is in L, he can find that out and even discover a witness y for x's membership in L, with $|y|$ polynomial in $|x|$. The Prover then shows y to V, who can verify in polynomial time that indeed $x \in L$.

If V trusts P, and is willing to take any statement from P on faith, then of course P's unbounded computational power is at V's disposal. But what if P is not trustworthy and V demands to see proofs that he can *verify* at least? We know that this can be done for any $L \in \mathbf{NP}$, but can P convince V about the truth of claims whose computational complexity is higher than $\mathbf{NP}$? The immediate answer is negative. For example, permitting V to ask P more queries or even allowing for a series of exchanges would not help, if V is assumed to play deterministically. For then, the all-powerful prover P could figure out all of V's messages ahead of time and supply all his replies at once. Things do change, though, if we deviate from this traditional scenario on two accounts:

- Allow V to flip coins (which P cannot guess in advance).
- Have V settle for something less than a proof, and be content with statistical evidence for the correctness of the claim $x \in L$. Still, V should be able to set the statistical significance of the test arbitrarily high.

That the statistical significance of the test may be as high as V wishes, has two consequences: Only with a preassigned small probability will prover P fail to convince verifier V that x is in L in case it is. At the same time V will almost never be convinced that x is in L if this is not the case, for *any adverserial strategy* the prover may choose. Again, with V trusting P, the interactive proof system becomes as powerful as prover P. It is therefore only interesting to think of a *cheating* Prover who may try and convince V of false claims, but should, however, almost surely fail. This theory has been initiated in [GMR] and [Ba], and some of it is briefly surveyed here.

Technically, one defines the notion of a pair of interacting randomized Turing Machines, which captures the intuition of two players sending messages to each other, by the end of which exchange V outputs either 0 or 1. Two such machines P and V constitute an *interactive proof system* for a language L if:

- P almost surely convinces V that x is in L in case it is: For every $c > 0$ there is a polynomial $p(\cdot)$ such that on input $x \in L$ machine V runs for time at most $p(|x|)$ and (correctly) outputs 1 with probability at least $1 - |x|^{-c}$.
- *No one* has a significant chance of convincing V that x is in L in case it is not in L: For every $c > 0$ there is a polynomial $p(\cdot)$ such that for every Turing Machine P' interacting with V, and every $x \notin L$, V runs for time at most $p(|x|)$ and (correctly) outputs 0 with probability at least $1 - |x|^{-c}$.

(Think of P' as an impostor who tries to convince V that $x \in L$ when indeed $x \notin L$.)

If such P and V exist, we say that L is in **IP** (*there is an interactive proof system for* L.) Are interactive proof systems more powerful than **NP**? That is, are there languages which are in **IP** but not in **NP**? A series of recent papers indicate that apparently this is the case. The complexity class **PSPACE** includes all the languages L for which membership of x can be decided using *space* which is only polynomial in $|x|$. **PSPACE** can be easily shown to contain **NP**, and there is mounting evidence that it is in fact a much bigger class. Note that a Turing Machine which is given space s can run for time *exponential* in s.

In any event, the power of interactive proof is now known, viz.:

Theorem 3.10: *[LFKN],[Sh2]*

$$\mathbf{IP} = \mathbf{PSPACE}.$$

This fact may have some interesting implications for game-playing and the complexity of finding good strategies, since it is known that many computational problems associated with various board games are complete for the class **PSPACE**, see [GJ].

3.2.4 Zero-Knowledge Proofs

In a conversation where Prover convinces Verifier that $x \in L$, what other information does P supply V with? In the cryptographic context it is often desirable for P to convince V of some fact (usually of the form "x is in L" for some $L \in \mathbf{NP}$) while supplying as little side information as possible. Are there languages L for which this can be done with *absolutely no* additional information passing to V other than the mere fact that $x \in L$? The (surprising) answer is that such languages exist, assuming the existence of one-way functions. Such L are said to have a *zero-knowledge* interactive proof system.

A moment's thought will convince the reader that the very definition of this new concept is far from obvious. The intuition is that once V is told that "x is in L" he should be able to reconstruct possible exchanges between himself and P leading to this conclusion. More precisely, given an interactive proof system (P, V) for a language L and given an input x, there is a probability distribution on all exchanges between P and V. Suppose there is a polynomial-time probabilistic Turing Machine M, which on input x and a bit b (to be thought of as a shorthand for the statement "x is in L") randomly generates possible exchanges between P and V. Suppose moreover, that if indeed x is in L, then the distribution generated by M is identical with the one actually taking place between P and V. In this case we say that (P, V) constitute a *zero-knowledge* interactive proof system for L and L is said to be in the class **ZK**.

Since both V and M are probabilistic polynomial-time, we may as well merge them. Thus, given a correct, positive answer to the decision problem "is x in L?", the Verifier can randomly generate conversations between itself and the Prover, and the exchanges are generated according to the genuine distribution. Moreover, V does that *on its own.* This can surely be described as V having learnt only the answer to that decision problem and nothing else.

Zero-knowledge comes in a number of flavors. In the most important versions, one does not postulate that the conversations generated by V are identical with the correct distribution, only that it is *indistinguishable* from it. We do not go into this, only mention that indistiguishability is defined in the spirit of section 3.2.1. [Gr1] contains a more detailed (yet less uptodate) discussion.

We would like to show now that:

Theorem 3.11: *If one-way functions exist, then the language of all 3-colorable graphs is in* **ZK**.

Since 3-colorability is an **NP**-complete problem there follows:

Corollary 3.12: *If one-way functions exist, then*

$$\mathbf{NP} \subseteq \mathbf{ZK}.$$

Together with the method that proved the equality between **IP** and **PSAPCE** (Theorem 3.10) even more can be said:

Corollary 3.13: *If one-way functions exist, then*

$$\mathbf{ZK} = \mathbf{IP} (= \mathbf{PSPACE}).$$

Following is an informal description of the zero-knowledge interactive proof for graphs' 3-colorability. Given a 3-colorable graph G on a set of vertices $X = \{x_1, \ldots, x_n\}$, the prover finds a 3-coloring of G, i.e., a partition (X_1, X_2, X_3) of X so that if x, y belong to the same X_i, there is no edge between them. If $x \in X_t$ this is denoted by $\phi(x) = t$. The prover P also chooses a random permutation π on $\{1, 2, 3\}$ and, using the method of "envelopes" (section 3.2.2) it relays to V the sequence $\pi(\phi(x_i)), i = 1, \ldots, n$. The verifier V then picks an edge $e = [x_i, x_j]$ in the graph, and in return P exposes $\pi(\phi(x_i))$ and $\pi(\phi(x_j))$ which V checks to be distinct. Note that if G is 3-colorable, the above procedure will always succeed. If it is not, then no matter which ϕ is chosen by V, it is almost sure that P, choosing the inspected edge at random, will detect an edge whose two vertices are colored the same after, say, $10n^3$ tries. (There can be at most $\binom{n}{2}$ edges if the graph has n vertices and the previous claim follows immediately from the law of large numbers.)

It is intuitively clear that V gains no extra information from this exchange, other than the fact that G is 3-colorable. Technically, here is how the Zero-Knowledge property of this protocol is established: A Turing Machine which is input the graph G and is given the information that G is 3-colorable can simulate the exchange between P and V as follows: Select an edge in G, say $[x_i, x_j]$ at random (this simulates P's step). Select two random numbers from $\{1, 2, 3\}$ and associate one with x_i and the other with x_j, for V's response. It is easily verified that the resulting exchanges are distributed in exactly the same way as those between the real P and V.

3.2.5 Secret Sharing

This idea was first introduced by A. Shamir [Sh1]. Informally, the simplest form of the question is: One would like to deal pieces, or shares, of a secret among n players so that a coalition of any r of them can together reconstruct it. However, no set of $r-1$ or fewer players can find out anything about the secret. A little more formally, there is a secret s and each player is to receive a piece of information s_i so that s is reconstructable from any set of r of the s_i, but no set of $r-1$ or less of the s_i supplies any information about s. Here is Shamir's solution to this problem.

Fix a finite field $\mathbf{F}$ with $\geq n$ elements and let $s \in \mathbf{F}$ be the secret. Pick a polynomial $f = f(x)$ of degree at most $r-1$, from $\mathbf{F}[x]$, the ring of polynomials in x over $\mathbf{F}$ where f's free term is s and all other coefficients are selected from F, uniformly and independently, at random.

Associate with every player P_i a field element α_i (distinct players are mapped to distinct field elements). Player P_i is dealt the value $f(\alpha_i)$, for $i = 1, \ldots, n$. Note that any r of the elements $f(\alpha_i)$ uniquely define the polynomial f, and so, in particular the secret coefficient s. On the other hand, any proper subset of these field elements yields no information on s which could be any field element with uniform probability distribution.

The performance of Shamir's scheme depends on proper behavior of all parties involved. If the dealer deviates from the protocol or if the players do not show the share of the secret which they actually obtained, then the scheme is deemed useless. Chor, Goldwasser, Micali and Awerbuch [CGMA] proposed a more sophisticated *verifiable* secret sharing wherein each player can prove to the rest that he followed the protocol, again without supplying any extra information. Bivariate, rather than univariate polynomials have to be used for this more sophisticated scheme. The basic idea is for the dealer to choose a random bivariate polynomial $f(x,y)$ subject to the conditions that $f(0,0) = s$, the secret, and that f has degree $\leq r-1$ in both x and y. Again player P_i is associated with field element α_i and its share of the secret is the pair of univariate polynomials $f(\alpha_i, y)$ and $f(x, \alpha_i)$. A key step in verifying the validity of the shares is that players P_i and P_j compare their views of $f(\alpha_i, \alpha_j)$, which is included in both their shares. Mismatches are used in detecting faulty dealers and faulty share holders. The details can be found in [BGW].

3.3 Protocols for Secure Collective Computation

Given a set of n players and an additional trusted party, there are various goals that can be achieved in terms of correct, reliable and leak-free computation. In fact, all they need to do is relay their input values to that party who can compute any functions of these inputs and communicate to every player any data desired. Speaking in broad terms, our goal here is to provide protocol for the n parties to achieve all these tasks in the absence of a trusted party. The two most interesting instances of this general plan are:

Privacy: Consider a protocol for computing n functions $f_1, \ldots, f_n$ of n variables by n parties where originally party i holds x_i, the value of the $i-$th variable and where by the protocol's end it gets to know $f_i(x_1, \ldots, x_n)$ $(1 \leq i \leq n)$. The protocol is $t-$*private* if every quantity which is computable from the information viewed throughout the protocol's run by any coalition of $\leq t$ players, is also computable from their own inputs and outputs.

Fault tolerance: It is $t-$*resilient* if for every coalition S of no more than t parties, the protocol computes a value $f(y_1, \ldots, y_n)$ so that $y_i = x_i$ for all $i \notin S$ and so that $y_j (j \in S)$ are chosen independent on the values of $x_i (i \notin S)$.

Here are the results which hold under the assumption that bad players are computationally restricted.

Theorem 3.14: *[GMW2] Assuming the existence of (non-uniform) trapdoor one-way functions, every function of n variables can be computed by n probabilistic polynomial-time Turing Machines, in an $(n-1)-$private, $\lfloor \frac{n-1}{2} \rfloor -$resilient protocol. The quantities $n-1$ and $\lfloor \frac{n-1}{2} \rfloor -$ are best possible.*

Non-experts can safely ignore the phrase "non-uniform" in the theorem.

In the information-theoretic setting the results are summarized in:

Theorem 3.15: *[BGW](also [CCD]) Every function of n variables can be computed by n processors which communicate via secure channels in an $\lfloor \frac{n-1}{2} \rfloor -$private way. Similarly, a protocol exists which is both $\lfloor \frac{n-1}{3} \rfloor$-private and $\lfloor \frac{n-1}{3} \rfloor$- resilient. These bounds are tight.*

Remark 3.16: The same results hold, even if the players have to jointly sample a probability space determined by their inputs, see remark 3.2. ▮

Having reviewed all the necessary background, we are now ready to explain how the algorithms work. There are four protocols to describe, according to the model (cryptographic or information-theoretic) and whether bad players are assumed curious or malicious. All four protocols follow one general pattern, which is explained below. We review the solution for the (information-theory, curious) case in reasonable detail and then indicate how it is modified to deal with the other three situations.

The problem becomes more structured, when rather than dealing with a general function f the discussion focuses on a *circuit* which computes it (see remark 3.5). This causes no loss of generality, since circuits can simulate Turing Machines. An idea originating in [Y2], and further developed in [GMW2] is for the players to collectively follow the computation carried out by the circuit moving from one gate to the next, but where each of the partial values computed in the circuit (values carried on the circuit's wires) is encoded as a secret shared by the players. To implement this idea one needs to be able to:

- Assign input values to the variables in a shared way.
- Perform the elementary field operations on values which are kept as shared secrets. The outcome should again be kept as a shared secret.

- If, at the computation's end, player P is to possess a certain value computed throughout, then all shares of this secret are to be handed to him by all other players.

The second item is the most interesting and will be our focus here.

In the case under consideration (information-theoretic, curious) this part is carried out as follows: Secrets are shared using Shamir's method, and we need to be able to add and to multiply field elements which are kept as secrets shared by all players. Addition of shared secrets poses no problem: if the secrets s_1, s_2 are encoded via the degree $\leq t$ polynomials f_1, f_2 (i.e., the free term is s_i and all other t coefficients are drawn from a uniform probability distribution over the field) then the secret $s = s_1 + s_2$ may be shared via the polynomial $f = f_1 + f_2$. Note that f constitutes a valid encoding of the secret s. Moreover, player P_i who holds the shares $f_1(\alpha_i)$ and $f_2(\alpha_i)$ of s_1, s_2 respectively, can compute its share in s by adding the above shares. Thus addition may be performed even without any communication among the players. The same holds also for multiplication by a scalar.

Multiplication of secrets creates more complications: While it is certainly possible to multiply the shares in just the same way to have $g = f_1 f_2$ encode the secret $s_1 s_2$, the polynomial g does not qualify as an encoding. Its degree may rise to $2t$, and also note that to qualify for a shared secret, all non-free terms of the polynomial have to be drawn from a uniform probability distribution. This requirement need not hold here, e.g., because g cannot be irreducible, except in the rare event that one of the f_i is constant. These two problems are resolved in two steps: A re-randomization, and a degree reduction. To re-randomize, every participant chooses a random polynomial q_i of degree $2t$ with a zero free term and deals it among all players. If g is replaced by $g + \sum q_i$, then no set of t players may gain any information on the first t coefficients. (This is a slight modification of the basic argument showing the validity of Shamir's secret sharing method.) This polynomial has the correct free term, all other coefficients are uniformly, randomly distributed, so the only problem left is with the degree being (probably) too high, which thus needs to be reduced. This will be achieved by truncating the high order terms in g.

Let h be the polynomial obtained by deleting all the terms in g of degree exceeding t. If $\vec{a}$ (resp. $\vec{b}$) is the vector whose i-th coordinate is $g(\alpha_i)$ (resp. $h(\alpha_i)$) then there is a matrix C depending only on the α_i such that $\vec{b} = \vec{a}C$. This operation, and thus a degree reduction, may be performed in a shared way, as follows. Each P_j knows $g(\alpha_j)$ and for every i we need to compute $h(\alpha_i) = \sum_j c_{i,j} g(\alpha_j)$ and inform P_i. Now, P_j computes $c_{i,j} g(\alpha_j)$ and deals it as a shared secret among all players. Everyone sums his shares for $c_{i,j} g(\alpha_j)$ over all j, thus obtaining his share of $h(\alpha_i) = \sum c_{i,j} g(\alpha_j)$, which presently becomes a shared secret. (Recall that if s_ν are secrets, and s_ν^μ is the share of s_ν held by player P_μ, then his share of $\sum s_\nu$, is $\sum s_\nu^\mu$.) Each player passes to P_i the share of $h(\alpha_i)$ he is holding, so now P_i can reconstruct the actual $h(\alpha_i)$. Now, the free term of g which is the same as the free term of h, is kept as a shared secret, as needed.

This establishes the $\lfloor \frac{n-1}{2} \rfloor$ privacy part of theorem 3.15. The condition $n > 2t$ was required both at the multiplication step and for re-randomization.

That more curious players cannot be tolerated by any protocol follows from the impossibility of computing the logical "or" function by two players. See also [CK].

We just saw how $\lfloor \frac{n-1}{2} \rfloor$ curious players may be handled, but how can a protocol tolerate $\lfloor \frac{n-1}{3} \rfloor$ *malicious* participants? We consider $n = 3t + 1$ players of which t may fail, and try to maintain the spirit of the previous protocol. This creates the need for a secret sharing mechanism where t shares of a secret reveal nothing of the secret and where even if t out of n shares are either altered or missing, the secret can be recovered.

Notice the following simple fact about polynomials: Let f be a degree d polynomial over a (finite, but large enough) field F, and assume you are being told f's value at $3d + 1$ points. However, this data may be in error for as many as d of the data points. You can still recover f, despite the wrong data, as follows: Let the $3d+1$ points be $x_i \in F$ and the purported values be $y_i \in F$, so that $f(x_i) = y_i$ for at least $2d + 1$ of the indices i. Suppose that g, h are two degree d polynomials, each satisfying at least $2d + 1$ equalities of the form $g(x_i) = y_i$ and $h(x_i) = y_i$, respectively. This implies that for at least $2(2d + 1) - (3d + 1) = d + 1$ indices, there holds $g(x_i) = h(x_i) = y_i$. But g, h have degree $\leq d$, so they must coincide (and be equal to f).

This implies that for $n \geq 3t + 1$, even if as many as t of the shares are corrupted, the secret can be recovered. Consequently, the previous protocol can be performed successfully, if the following step is added: Whenever a player is receiving the n shares of a secret that he is supposed to know, he assumes that as many as t of them might be corrupted or missing. The previous remark implies that he can reconstruct the secret, nevertheless.

Incidently, a clever choice of the field elements α_i allows this recovery to be done in a computationally efficient way. Namely, if the field F has a primitive $n-$th root of unity ω, i.e., $\omega^n = 1$ and $\omega^k \neq 1$ for $1 \leq k < n$, let $\alpha_i = \omega^i (1 \leq i \leq n)$. The efficient recovery mechanism is based on the Discrete Fourier Transform (e.g. [AHU]), and is also related to the theory of Error Correcting Codes [MS].

Besides the corruption of shares, there is also a possibility that bad players who are supposed to share information with others will fail to do so. This difficulty in countered by using a verifiable secret sharing mechanism, briefly described in section 3.2.5.

No more than $\lfloor \frac{n-1}{3} \rfloor$ malicious participants can be handled by any protocol. This follows from the fact (Theorem 2.1) that Byzantine agreement is achievable only if $n \geq 3t + 1$.

Our last two comments are that if players are not restricted to communicate via the secure two-party lines but can also *broadcast* messages then resiliency can be increased from $\lfloor \frac{n-1}{3} \rfloor$ to $\lfloor \frac{n-1}{2} \rfloor$ [RB]. Also, it is not a coincidence that our computations were always over a finite field. The situation when the domain is infinite is discussed in [CGK].

We have reversed the historical order here, since Theorem 3.14, the solution in the cryptographic set-up was obtained (in [GMW]) prior to the information-theoretic Theorem 3.15 ([BGW][CCD]), which, in turn followed [Y2] where the two-player problem was solved. In hindsight, however, having presented the solution for the Information-Theoretic case, there is an easy way to describe the cryptographic solution. Note that in each step of the above protocol one of the players tries to convince another player of a certain statement such as "the value of a polynomial Φ at a point x is so and so". All these statements clearly belong to **NP** and so by Corollary 3.12 can be proved in a Zero Knowledge interactive protocol. Thus if the players are randomized polynomial-time processors, and if one-way functions exist, then the protocol may be run step by step safely. Namely, replace each step in the previous protocol where player

P_i relays a message to P_j, using their secure communication line, by an interaction where P_i proves to P_j in Zero Knowledge, the appropriate statement. This proves theorem 3.14.

3.3.1 Efficiency issues

While our motivating questions are now answered, there is much left to be desired in terms of efficiency. The protocols presented here allow us to transform any given protocol in distributed processing into one which cannot be disrupted by any set of faulty processors, as long as they are not too numerous. However, the transformation from the original protocol to the safe one creates a large overhead in processing time. Even if this transformation must, in general, be costly, could we supply similar, only more efficient results for specific computational tasks? In particular, in distributed computing there are many operations that have to be run in a network for purposes of maintenance (e.g. [AAG]). It would be interesting to find ways in which such tasks can be performed both safely and efficiently.

4 Fault Tolerant Computation - The General Case

Protocols presented in the previous two sections allow a set of players, some of whom may malfunction, to perform various computational tasks without disruption. These protocols require (i) an upper bound on the number of bad players and that (ii) good players can hide information, either through use of secure communication lines, or by exploiting the bad players' restricted computational power. Is fault-tolerant computation at all possible if (ii) is removed? We lower our expectations somewhat: rather than *absolute* results guaranteeing that computations always turn out correctly, *quantitative* statements are sought. It is assumed that the bad coalition favors a certain outcome for the protocol, and faulty players play an optimal strategy to maximize the probability of this outcome. The influence of a coalition towards a particular outcome is the largest increase in this outcome's probability the coalition can achieve. Protocols are sought to minimize all influences. The two tasks to be most extensively considered here are: flipping a coin and electing one of the players. In coin flipping we try to make both outcomes as equally likely as possible, even though the bad players may favor, say, the outcome "heads". In the election game, the goal is to make it most likely for a good player to be elected, despite efforts by the bad players to have one of them elected.

The critical role of *timing* in such problems, is best illustrated, perhaps, through a simple example, viz. the *parity* function. Let all x_i take values 0 or 1, and let $f(x_1, \ldots, x_n) = 0$ or 1 depending on the parity of $\sum x_i$. Suppose that the bad coalition would like f to take the value 1. If all players are good and supply their correct inputs, and if x_i are generated by independent coin flips, then f equals 1 with probability 1/2. Can a bad coalition boost this probability up? That depends critically on the order in which the x_i are announced. If they are supplied simultaneously by all players, then even as many as $n-1$ bad players can exert no bias on f. If, however, the bad players let the good ones move first, then even a *single* bad player can decide f's value with certainty. Simultaneous action cannot be guaranteed in the presence of bad players, and so the relevant assumption is that good players move first.

The n-player computation of parity is an instance of the first class of problems considered in this section: Let f be an $n-$variable boolean function, and let the assignment x_i to f's $i-$th variable be known only to player P_i. Consider a coalition S whose goal is to have f take some desired value. All players not in S disclose the assignments to their variables, after which players in S together select values to be announced as assignments to their variables. Clearly, even after all non-S players announce their input values, f may still be undetermined, a fact which gives S an advantage. The probability of f remaining undetermined is defined as the *influence* of S on f.

The above discussion is a "one shot" protocol for computing f. The most general mechanism for computing a function without hiding information can, as usual, be described in a tree form. Again we consider a coalition trying to make f take on a certain value and define influences accordingly (see section 4.3).

Influences are the main subject of the current chapter, both for boolean functions (which for a game theorist are *simple games*) and in general perfect information games. Many quantities are defined in game theory to measure the power of coalitions in various games and decision processes. Influences can be added to this long list and may be expected to come up in other game-theoretic problems. Specifically, one instance of our notions of influence turns out to coincide with the *Banzhaf-Coleman power index* [Ow].

Here are some of the main results under review. We will freely interchange terminologies between boolean functions and simple games, and correspondingly sets of variables and coalitions. Our first result is stated both ways.

- For every n-variable boolean function there is a set of $o(n)$ variables whose influence is $1-o(1)$. (In every n-player simple game there is a coalition of $o(n)$ players whose influence is $1-o(1)$.)
- There are n-player simple games where coalitions with fewer that $\frac{n}{\log^2 n}$ players have influence $o(1)$.
- In every perfect information, n-player protocol to flip a coin, and for every $n \geq k \geq 1$, there is a k-player coalition with influence $> c\frac{k}{n}$ ($c > 0$ is an absolute constant).
- There is an n-player perfect information coin-flipping game, where the influence of every coalition of $\leq n/4$ players does not exceed 0.9.

4.1 Influence in simple games

We are now ready for formal definitions, after which some concrete examples are considered. Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function and let $S \subseteq \{1,\ldots,n\}$. To define the influence of S on f, assign $0,1$ values to all variables x_i not in S, where the assignments are chosen uniformly and independently at random. The probability that f remains undetermined after this partial assignment of values is the influence of S on f. In fact, notice that this partial assignment falls in one of three categories: either it determines that $f=0$ regardless of the x_i for $i \in S$, or $f=1$ must hold, or f remains undetermined. Let q_0, q_1 and q_2 be the probabilities of these three events. As we said, q_2 is called the *influence of S on f*, and is denoted $I_f(S)$.

There are two other quantities to mention. If $p_0 = p_0(f)$ is the probability that $f = 0$ when *all* its variables are assigned random values and $p_1 = 1 - p_0$, define the *influence of S on f towards zero*

$$I_f^0(S) = q_0 + q_2 - p_0.$$

(Explanation: All x_i with $i \in S$ are assigned after all other x_j are revealed, and are chosen so as to assure f equal 0, if possible. Therefore, the probability that f does evaluate to zero is $q_0 + q_2$. The excess of this probability over p_0 measures the advantage gained towards setting f to 0.) Similarly:

$$I_f^1(S) = q_1 + q_2 - p_1$$

is the *influence of S on f towards one.* It is pleasing to notice that the influence of S on f is the sum of the two

$$I_f(S) = q_2 = I_f^0(S) + I_f^1(S),$$

since $q_0 + q_1 + q_2 = p_0 + p_1 = 1$.

For $S = \{i\}$ a singleton, it is easily checked that $I^0(i) = I^1(i) = \frac{1}{2}I(i)$, and $I(i)$ is the same as the Banzhaf-Coleman power index. The reader should note that the subscript f was omitted and i was written instead of $\{i\}$, for typographic convenience. This shorthand writing will be used throughout.

Let us consider a few examples, all with $p_0 = p_1 = 1/2$.

Example 4.1: Dictatorship: This is the boolean function $f(x_1, \ldots, x_n) = x_1$. It is easily verified that $I(1) = 1, I^0(1) = I^1(1) = 1/2$, and that a set S has zero influence unless $1 \in S$. ∎

Example 4.2: Majority: For convenience, assume $n = 2m + 1$ is odd, and consider the influence of singletons first. The probability for the majority vote to be undetermined when only $i's$ vote is unavailable is $I(i) = 2^{-2m}\binom{2m}{m} = \Theta(n^{-1/2})$. More generally, if S has $2s + 1$ elements, then

$$I(S) = \sum_{m+s \geq j \geq m-s} \binom{2m}{j} / 2^{2m-2s}$$

This expression can be estimated using the normal distribution approximation to the binomial one. In particular, if $|S| = n^{\frac{1}{2}+o(1)}$ the influence of S equals $1 - o(1)$. That is, coalitions of such size have almost sure control over the outcome. This point will be elaborated on in the sequel. ∎

The next example plays a central role in this area:

Example 4.3: Tribes: Variables are partitioned into blocks ("tribes") of size b, to be determined soon and f equals 1 iff there is a tribe in which all variables equal 1. To guarantee $p_0 = p_1 = 1/2$ the following condition must hold:

$$(1 - (1/2)^b)^{\frac{n}{b}} = 1/2,$$

the solution of which is $b = \log n - \log\log n + \Theta(1)$. (In fact, the solution b is not an integer, so we round this number and make a corresponding slight change in f to guarantee $p_0 = p_1 = 1/2$.)

To figure out the influence of a single variable i, note that f is undefined iff all variables in i's tribe are 1 while no other tribe is a unanimous 1. The probability of the conjunction of these events is

$$I(i) = (1/2)^{b-1}(1-(1/2)^b)^{\frac{n}{b}-1} = \Theta(\frac{\log n}{n}).$$

As for larger sets, there are sets of size $O(\log n)$ with influence close to 1/2, e.g. a whole tribe, and no significantly smaller sets have influence bounded away from zero. It is also not hard to check that sets of variables with influence $1 - o(1)$ need to have $\Omega(n/\log n)$ variables and this is achievable. In fact, a set which meets all tribes and contains one complete tribe has influence 1. ∎

We are looking for the best results of the following type: For every n-variable boolean function of expected value $1 > p_1 > 0$, there is set of k variables of influence at least $J(n, k, p_1)$.

Already the case of $k = 1$, the best lower bound on the influence of a single player, turned out to be a challenging problem. An old result of S. Hart [Ha] translated to the present context says, for example, that if $p_1 = 1/2$ then $\sum_{x\in[n]} I_f(x) \geq 1$. The full statement of Hart's theorem supplies the best possible lower bound on the sum of influences for all $0 \leq p_1 \leq 1$. In combinatorial literature (e.g. [Bs]) this result is referred to as the *edge isoperimetric inequality* for the cube. That the bound is tight for $p_1 = 1/2$ is exhibited by the dictatorship game. Consequently, in this case there is always a player with influence at least $1/n$. On the other hand, in all the above mentioned examples there is always a player with influence $\geq \frac{c\log n}{n}$. This indeed is the correct lower bound, as shown by Kahn, Kalai and Linial [KKL].

Theorem 4.4: *For every boolean function f on n variables (an $n-$players simple game) with $p_1(f) = 1/2$, there is a variable (a player) with influence at least $\frac{c\log n}{n}$ where $c > 0$ is an absolute constant. This bound is tight except for the numerical value of c.*

Before sketching the proof let us comment on some of its extensions and corollaries. The vector $(I(x), x \in [n])$ of influences of individual players, may be considered in various L^p norms. The edge isoperimetric inequality yields the optimal lower bound for L^1 while the above theorem answers the same question for L^∞. Can anything be said in other norms? As it turns out, the **tribes** function yields the best (asymptotic) result for all L^p, $(p > 1 + o(1))$. The L^2 case is stated for all $0 < p_1 < 1$ in the following theorem:

Theorem 4.5: *[KKL] For f as above, and $p_1 = p_1(f) \leq \frac{1}{2}$,*

$$\sum_{x\in[n]} I(x)^2 \geq cp_1^2 \frac{\log^2 n}{n}$$

where $c > 0$ is an absolute constant. The inequality is tight, except for the numerical value of c.

By repeated application of this theorem one concludes the interesting result that *there is always a coalition of negligible cardinality and almost absolute influence*:

Corollary 4.6: *For any simple game f with $p_1(f)$ bounded away from zero and one, there is a coalition of $o(n)$ players whose influence is $1 - o(1)$. In fact, for every $\omega = \omega(n)$ which tends to infinity with n there is such a coalition of size $\frac{n}{\omega(n)\log n}$. The result is optimal except for the ω term.*

We have established the existence of small coalitions whose influence is almost 1. Lowering the threshold, it is interesting to establish the existence of (possibly even smaller) coalitions whose influence is only bounded away from zero. While the last two bounds are tight for **tribes**, this game does have coalitions of size $O(\log n)$ with a constant influence. This is a very small cardinality, compared e.g. with the majority function, where cardinality $n^{\frac{1}{2}+o(1)}$ is required to bound the influence away from zero. The situation is not completely clarified at the time of writing, though the gap is not nearly as big. Ajtai and Linial [AL] have an example of a boolean function where sets of $o(\frac{n}{\log^2 n})$ variables have $o(1)$ influence. It is plausible that the truth is closer to $\frac{n}{\log n}$. Following is an outline of this example.

Example 4.7: The example is obtained via a probabilistic argument, (for a reference on this method in general see [AS].) First introduce the following modification of **tribes:** Reduce tribe sizes from $\log n - \log\log n + c_1$, to $\log n - 2\log\log n + c$, thus changing $p_0(f)$ from $1/2$ to $\Theta(\frac{1}{n})$ (c, c_1 are absolute constants). Also introduce, for each variable i, a bit $\beta_i \in \{0,1\}$ and define f to be 1 iff there is a tribe B such that for every $i \in B$ the corresponding input bit x_i equals β_i. This is essentially the same example as **tribes**, except for the slight change of parameters and the fact that some coordinates (where $\beta_i = 0$) are switched.

Now generate at random $m = \Theta(n)$ partitions $\pi^{(j)} (j = 1 \dots m)$ of the variables into blocks of the above size, as well as m corresponding binary vectors $\beta^{(j)} (j = 1 \dots m)$. Partition $\pi^{(j)}$ and vector $\beta^{(j)}$ define a function $f^{(j)}$ as above. The function considered is $f = \bigwedge_j f^{(j)}$. It can be shown that with positive probability $p_0(f)$ is very close to $1/2$ and that no set of size $o(\frac{n}{\log^2 n})$ can have influence bounded away from zero. Some small modification of this f gives the desired example. ∎

Let us turn to a sketch of the proof of theorem 4.4. The reader is assumed to have some familiarity with the basics of harmonic analysis as can be found e.g. in [DM]. Identify $\{0,1\}^n$ with the abelian group $\mathbf{Z_2^n}$, so it makes sense to consider the Fourier transform of boolean functions. It is also convenient to identify $\{0,1\}^n$ with the collection of all subsets of $[n]$. The 2^n characters of this group are parameterized by all $S \subseteq [n]$ as follows:

$$u_S(T) = (-1)^{|S \bigcap T|}$$

and f is expressed as:

$$f(T) = \sum_S \alpha_S u_S(T).$$

(α_S is usually denoted $\hat{f}_S$, the Fourier Transform of f at S.) The easiest way to see the connection between influences and Fourier coefficients is to notice that if f is a monotone boolean function (i.e., a monotone simple game), then the influence of player j is given by:

$$I_f(j) = \alpha_{\{j\}}$$

and there is no loss of generality in considering only the monotone case because of the following:

Proposition 4.8: *[BL] For every boolean function g there is a monotone function h such that $p_1(h) = p_1(g)$ and so that for every $S \subseteq [n]$ and $\beta \in \{0,1\}$, there holds*

$$I_h^\beta(S) \leq I_g^\beta(S).$$

So to show the existence of an influential variable one needs to prove lower bounds on certain Fourier coefficients of f. Monotonicity is not used in the present proof, but see Theorem 4.12. First one considers "directional derivatives of f :"

$$f^j(T) = f(T) - f(T \oplus \{j\})$$

where $\oplus$ is mod 2 addition. Note that $I(j)$ is proportional to the fraction of T's for which $f^j(T) \neq 0$.

Since f^j takes only the values $-1, 0, 1$ the square of its L^2 norm is proportional to $I(j)$. Now Parseval's identity enables us to express the L^2 norm of a function in terms of its Fourier coefficients. At the same time there is a simple relationship between the coefficients of f^j and those of f. As a consequence an expression is obtained for $I(j)$ in terms of f's Fourier expansion:

$$I(j) = 4 \sum_{j \in S} \alpha_S^2,$$

and summing over all j,

$$\sum I(j) = 4 \sum_{|S|} |S| \alpha_S^2.$$

But Parseval's identity also gives:

$$\sum \alpha_S^2 = \|f\|_2^2 = p_1(f)$$

and so the $I(j)$ can be small only if most of $\sum \alpha_S^2$ comes from sets S of bounded cardinality. The main part of the proof is to show that this cannot happen. In fact a similar statement is proved with regards to the functions f^j. The only two properties available about these functions is that their range is $\{-1, 0, 1\}$ and (arguing by contradiction) their supports are small. The problem is how to express the first property in analytical terms and the answer is that having $\{-1, 0, 1\}$ for range means that all L^p norms of such a function are easy to express. The issue then becomes to show that under some restrictions on L^p norms, it is impossible for almost all of the L^2 norm to come from coefficients α_S for S of bounded size.

In his work on the sharpest form of some classical inequalities in harmonic analysis Beckner [Be] considers the *two point space* $X = \{-1, 1\}$ and operators from $L^p(X)$ to $L^q(X)$. This is a two-parameter class of operators and Beckner finds tight bounds on their norms. To state his results, note that any function on the two point space is linear.

Lemma 4.9: *Consider the operator $T_1 : L^p(X) \to L^2(X)$ which maps the function $a + bx$ to $a + \epsilon bx$. If $p \leq 1 + \epsilon^2$ then the norm of T_1 is 1.*

Another lemma of Beckner deals with similar problems concerning X^n. The space X^n is naturally identified with the $n-$dimensional cube and the operator $T = T_1^n$ is uniquely specified via knowledge of $T(u_S)$ for all S, since the characters span the whole space of real functions on the cube. As it turns out

$$T(u_S) = \epsilon^{|S|} u_S,$$

and the lemma is:

Lemma 4.10: *If* $p \leq 1 + \epsilon^2$ *then* $\|T\| = 1$.

The conclusion that is useful for our proof is given in the following lemma:

Lemma 4.11: *Let* $g : \{0,1\}^n \to \{-1,0,1\}$ *be a function whose Fourier expansion is* $g = \sum \gamma_S u_S$ *and let* g *be nonzero on exactly* $t \cdot 2^n$ *points* $(0 \leq t \leq 1)$. *Then for every* $0 \leq \delta \leq 1$

$$t^{\frac{2}{1+\delta}} \geq \sum \delta^{|S|} \gamma_S^2.$$

This last lemma is applied to $g = f^j$ and the resulting inequalities can be combined to derive the theorem. ∎

4.1.1 The non-boolean case

Even one-shot protocols for collective coin flipping need not be confined to a boolean function, since players' input to the process might well contain more than a single bit. So, we may consider a function $f : X^n \to \{-1,1\}$ where X is any probability space. Each player P_i samples a point $x_i \in X$, and the collective coin flip is $f(x_1, \ldots, x_n)$. (In the previous discussion we had $X = \{0,1\}$, both points having probability 1/2). The phenomena established in Theorems 4.4 and 4.5 have been recently extended to this more general situation, as we now explain. The definition of influence is easy to extend: let $S \subseteq [n]$ and assign all $x_i (i \notin S)$ values independently and according to the probability distribution on X. The probability that this partial assignment leaves f undetermined is $I_f(S)$, *the influence of* S *on* f. Though the same phenomena come up in the more general situation, the previous proof techniques breaks down, so new ideas have to be introduced. It is interesting to comment that this time proposition 4.8 is being employed.

Theorem 4.12: *[Bourgain, Kahn, Kalai, Katznelson and Linial] Let* X *be a probability space, and let* $f : X^n \to \{0,1\}$ *have expectation* $E(f) = 1/2$. *Then there is a variable with influence at least* $\frac{c \log n}{n}$ *where* $c > 0$ *is an absolute constant. Also for every* $\omega = \omega(n)$ *which tends to infinity with* n *there is a set of* $\frac{n}{\omega(n) \log n}$ *variables whose influence is* $1 - o(1)$. *The results are optimal except for the value of* c *and the* ω *term respectively.*

The old theorem of Loomis and Whitney [LW] plays here the same role as does Hart's result in the case $X = \{0,1\}$, providing the best L^1 estimate for the vector of influences. In this case the extremal example consists of $X = [0,1]$, the unit interval with the function $f(x_1, \ldots, x_n)$ which equals 1 iff at least one of the x_i exceeds $2^{-\frac{1}{n}}$.

4.2 Symmetric simple games

A simple game f is said to be *symmetric* if $f(\bar{x}) = \overline{f(x)}$ for every input x where bar stands for boolean negation. In other words, reversal of each individual vote reverses the overall decision. Clearly $p_1(f) = 1/2$ holds for symmetric games. Evaluation of boolean functions can be thought of as a voting scheme, and from this perspective, it is natural to require f to be symmetric. The same questions posed about general simple games may be asked in this context, though the results are more fragmentary.

The known results are summarized in the following theorems:

Theorem 4.13: *[Feldman, Dubiner; unpublished] There is a symmetric $n-$player simple game where each player x has influence $I_f(x) = O(\frac{\log n}{n})$.*

Theorem 4.14: *[BL] There is a symmetric $n-$player game where a coalition having influence ϵ must have cardinality $\geq \epsilon n^{\alpha}$, where $\alpha = \frac{\log 2}{\log 3} = 0.63\ldots$*

These theorems are derived from the following two examples:

Example 4.15: The circle game (Feldman, Dubiner): Assume n to be odd, and imagine the variables being arranged around a circle. Consider any assignment for the variables of f and let $\lambda_1 \geq \lambda_2 \geq \ldots$ (resp. $\mu_1 \geq \mu_2 \ldots$) be the lengths of consecutive strings of zeros (resp. ones) in the input. The value of f is determined by a lexicographic comparison of these two sequences, i.e, let j be the smallest index for which $\lambda_j \neq \mu_j$. If $\lambda_j > \mu_j$, then $f = 0$, otherwise $f = 1$. Since n is odd, such a j exists and f is well-defined.

The analysis which shows that individual influences are $O(\log n/n)$ is not hard and is omitted. The reader looking to reconstruct a proof should notice that almost surely λ_1, μ_1, the longest consecutive blocks are equal $\Theta(\log n)$.

Another remark concerning this game is that while it is asymptotically optimal with respect to minimizing individual influences it is not particularly efficient in bounding influences of larger coalitions. A coalition of size $O(\sqrt{n})$ with influence 1 can be constructed as follows: it has $\sqrt{n}$ members equally spaced on the circle, plus a consecutive segment of length $\sqrt{n}$. In this respect this example is not even as good as majority voting. ∎

In the next example individual players have influence $O(n^{-\alpha})$, which is inferior to the circle game. However, it yields interesting bounds for the influences of larger coalitions.

Example 4.16: Iterated majority of three: In this example it is convenient to assume that n is a power of 3. The set of variable is partitioned into three "committees" of equal size. Each committee is divided into three "sub-committees" which are themselves partitioned into three etc. until the level of individual variables is reached. The overall decision is taken by a simple majority vote among the three committees. Within each committee, decision is made by majority vote of subcommittees etc. Analysis of this game ([BL]) shows that a coalition of size $k \leq n^{\alpha}$ has influence $O(\frac{k}{n^{\alpha}})$ where $\alpha = \frac{\log 2}{\log 3} = 0.63\ldots$ ∎

It is not clear whether symmetric games exist where individual influences are at only $O(\frac{\log n}{n})$ and at the same time, size much larger that $\sqrt{n}$ is needed to achieve constant influence. Also, does this class have examples where constant influence can be achieved only by coalitions of size $\Omega(n^{1-\epsilon})$ for all $\epsilon > 0$? Not much is known at the moment.

4.3 General perfect-information coin flipping games

It is not hard to think of more elaborate mechanisms for generating a random bit other than the one-shot games considered so far. Would it not be possible to find perfect information games which are less amenable to bias than one-shot games? Indeed this is possible, but even in the most general perfect-information games there are influential coalitions. An exact description of the games under consideration has to come in first. This definition is rather cumbersome, but luckily, each such game can be approximated arbitrarily well by much more structured games. A *standard* coin flipping game G is a binary tree each of whose internal nodes in *controlled* by one of the players and where the leaves are marked by zero or one. The game is started at the root. At each step the player which controls the present node has to flip a coin and accordingly the game proceeds to either the right or left son. When a leaf is reached the game terminates and the outcome is that leaf's label. The interested reader can find the general definition as well as the (easy) proof that there is no loss of generality in dealing with standard games in [BL]. Most of the subsequent discussion concerns standard games and the reader should have no difficulty adapting it where necessary to, say, trees which are not binary or other slight modifications.

With the experience gained on one-shot games the definition of influence should be easy to figure out: Let $p_0 = p_0(G)$ be the probability that the outcome of G is zero (assuming all players abide by the rules) and let $p_1 = 1 - p_0$. Now suppose that a coalition $S \subseteq [n]$ tries to bias the outcome and maximize the probability of a zero outcome. So when a node controlled by a player from S is reached, it is not by a coin flip that the next step is determined, but rather according to an optimal, mutually coordinated strategy. The new probability for zero outcome is denoted $p_0(G) + I_G^0(S)$ and this additional term is called *the influence of S on G towards zero.* The corresponding term with zero replaced by one is also defined. In keeping with simple games we define the *influence of S on G* as $I_G(S) = I_G^0(S) + I_G^1(S)$.

Now that the necessary technicalities have been dealt with, we can come down to business and state the results.

Theorem 4.17: *[BL],[AN],[BN] Let G be an $n-$player coin flipping game with $p_1(G)$ bounded away from zero and one and let $1 \leq k \leq n$, then*

$$Average(I_G^1(S)) > c\frac{k}{n}$$

where the average is over all coalitions S of k players and where c is an absolute constant.

For every $0 < \gamma < 1/2$ and every large enough integer n, there is a game G such that if $|S| < \gamma n$ then

$$I_G(S) = O(|S|/n).$$

The lower bound on the average influence was shown in [BL] and the existence of a hard-to-bias protocol is from Alon and Naor [AN]. The validity of this protocol for all $\gamma < 1/2$ (which is essentially best possible) is from [BN].

Proof: (Sketch for the lower bound) Consider the case $k = 1$ first. With every node ν of the tree is associated a vector $(q_0^{(\nu)}, q_1^{(\nu)} \ldots q_n^{(\nu)})$ where $q_0^{(\nu)}$ is the fraction of zero leaves in the subtree rooted at ν and $q_i^{(\nu)}$ is the probability of a zero outcome assuming i plays an optimal strategy to this end and all other players play at random. It turns out that the following inequality holds at every node ν of the tree:

$$\prod_{i \neq 0} q_i^{(\nu)} \geq (q_0^{(\nu)})^{n-1}.$$

This inequality is proved by induction on the height of ν in the tree. Let us skip this part and show only how to derive the theorem from it, and then proceed with a few extra remarks. When ν is the root, $q_0 = p_0 = 1/2$ and so

$$\frac{1}{n} \sum q_j \geq (\prod q_j)^{1/n} \geq (1/2)^{\frac{n-1}{n}} = \frac{1}{2} + \frac{c}{n} + O(n^{-2})$$

which proves the first part of the theorem for $k = 1$. The general case in proved similarly.

A curious fact about the inequality $\prod_{i \neq 0} q_i^{(\nu)} \geq (q_0^{(\nu)})^{n-1}$ is that it is inhomogeneous (comparing degrees on both its sides). It can be shown, however, to completely characterize the attainable q-vectors. Also note an analogous inequality appearing in the previously mentioned paper [LW] on measures of projection sets. ∎

Before we explain how the upper bound is proved, let us look at an example (which, incidently, is not in standard form).

Example 4.18: Baton-passing game: This game seems to have been suggested by a number of people. Its first analysis appears in [S] and further results were obtained in [AL]. The game starts with player 1 "holding the baton". At each round, the player holding the baton has to randomly select a player among those who had not previously held it and pass it on to that player. The last player to hold the baton is said to be *elected*. The elected player flips a coin which is the outcome of the game. Consider now a situation where this game is played by $n = s + t$ players where s abide by the rules and the complementary coalition B of t try to bring about some desired outcome. Clearly, B can bias the outcome only if one of its members is elected. It is easily verified that the best strategy towards this end is for members of B to always pass the baton to a player outside B.

Let $f(s,t)$ denote the probability for a member of B being elected, given that the first player to hold the baton is not in B. It is easily seen that f is defined by the recurrence:

$$f(s,t) = \frac{s}{s+t} f(s-1,t) + \frac{t}{s+t} f(s-1,t-1)$$

and the initial conditions $f(s, s+1) = 0$, and $f(s,1) = \frac{1}{s+1}$. The analysis of this game in [AL] shows the second part of theorem 4.17 for coalitions of size $O(\frac{n}{\log n})$. ∎

We now turn to some brief remarks on the protocol of Alon and Naor which is also an election game. If the elected player belongs to the bad coalition B we say that B *won*. If B's probability of winning (when B plays as well as possible and all others play at random) is smaller than some $c < 1$, the game is said to be *immune* against B. The [AN] protocol is immune against any coalition of $n/3$ players or less. Consider a probability space $\mathcal{T} = \mathcal{T}_{\rm d}$ consisting of all depth d complete binary trees, whose nodes are labeled by players' names, selected uniformly at random. Each such tree corresponds to an election protocol, where the label of an internal node is the name of the player who controls it. If the game terminates at a certain leaf, the player whose name marks that leaf is elected.

They show that a tree sampled from this space has a positive probability to be immune against all coalition of $n/3$ players or less. The line of reasoning is this: Fix a bad coalition B, and consider a random variable $X = X_B^d$ which on every labeled tree of depth d evaluates to B's winning probability. If we can get tight enough bounds on X's expectation and variance, then we can conclude that with probability $> 1 - \delta$, a tree drawn from $\mathcal{T}$ is immune against B, for some small $\delta > 0$. If, moreover $\delta < \frac{1}{\binom{n}{n/3}}$, then there is positive probability for a randomly drawn tree in $\mathcal{T}$ to be immune against *all* coalitions of $n/3$ members, i.e. this establishes the existence of the desired protocol. To estimate X_B^d's expectation and variance, consider what happens upon reaching a node v in the tree. If v is controlled by a good player, B's chance of winning is the average over the left and right subtrees rooted at v. If v is controlled by a bad player, this probability is the maximum over the two subtrees. However, the probabilities of winning for either subtree are themselves given by random variables distributed as X_B^r, for some $r < d$. Thus the key step in the proof is provided by the following lemma in probability:

Proposition 4.19: *Let Y and Z be independent random variables with equal expectations $E = E(Y) = E(Z)$ and equal variances $\sigma^2 = Var(Y) = Var(Z)$. Let X be the random variable which equals $\frac{Y+Z}{2}$ with probability $1 - \epsilon$ and $max\{Y, Z\}$ with probability ϵ for some $0 \leq \epsilon \leq 1$. Then $E(X) \leq E + \frac{\epsilon\sigma}{\sqrt{2}}$ and $Var(X) \leq (1 + 3\epsilon)\sigma^2/2$.*

This proof does not, however, tell us how to concretely construct such a game. This is a problem encountered often in trying to convert probabilistic arguments into algorithms. Alon and Naor do supply a somewhat weaker explicit construction for such game trees, based on the baton passing game and Bracha's [Br] scheme from the theory of Byzantine agreement, as well as some other steps of common use in derandomization arguments. This more complex protocol will not be reviewed here. The analysis in [BN] is based on examining higher moments of the same random variables.

4.4 Quo vadis?

4.4.1 Fault tolerant computation with full information?

We return to the following fundamental question: What are the limitations of fault tolerant computing in the absence of secure communication and barring any restrictions on the computational power of bad players? The only attempt at the problem in this generality has so far been made by Goldreich Goldwasser and Linial [GGL]. They introduce the following example:

Example 4.20: Equality game: Two players draw a random number in the range $\{1, \ldots, n\}$. They compute a function f which equals 1, if the two numbers agree and is 0 otherwise. The expectation of f is, clearly $1/n$. However, if they compute f by revealing their inputs in turn, then the player who moves second can determine the outcome at will, by pretending to have an appropriate input. Consider, in contrast, the following protocol: player 1 is to report the remainder of his input $\bmod \sqrt{n}$, then player 2 is to report his input, and then 1 tells all his input. Here, even if one of the players cheats, trying to increase the expectation of the outcome, he can only boost it to $1/\sqrt{n}$. ∎

A slight modification of the lower bound in Theorem 4.17 shows this protocol to be best possible. In [GGL] the problem of fault tolerant computing by two players is solved. The previous example turns out to be typical, in the following sense: Consider two parties trying to compute any function of two variables. By alternately revealing appropriate parts of their inputs, the function can be computed so that even if one of the players is trying to bias the outcome, his advantage would only very slightly exceed the above mentioned lower bound.

Computations by more than two players, bring about new complications. Already simple examples show that the satisfactory answer obtained for two players is incorrect for 3 players or more. In fact, there is not even a conjecture as to what the answer in general is. On the positive side [GGL] show a protocol for computing functions of n variables, which for almost all functions reduces the bias to the lower bound in Theorem 4.17. Similar results are shown also for some specific functions. Most notably they show how n parties can sample from any probability distribution with minimum bias.

4.4.2 Efficient coin flipping

Alon and Naor's coin flipping protocol reviewed above, takes time proportional to n, the number of players (and even slightly more in the protocol's explicit version). This raises the question whether, perhaps, a faster coin-flipping, or election is possible in a full-information game, even if a constant fraction of the players are bad. Cooper and Linial [CL] have recently shown that indeed these tasks can be performed in time only $O(\log^c n)$ for some constant c. Their current best protocol works with $c = 17$.

There is no reason known to us why the same performance will not be achievable in constant time. Such a protocol will greatly improve on Feldman and Micali's [FM] protocol, and would constitute a critical step in constructing protocols for fault-tolerant and efficient computation without secure communication.

5 More Points of Interest

This section contains a brief description of various problems and results where consideration from both theoretical computer science and game theory come to bear. In some instances the interaction between the two areas is very clear, in others it is more obscure, and the discussion should be taken as an invitation for researchers to try and find out whether a connection can be made. We tried to provide references wherever we are aware of a more complete discussion.

5.1 Efficient Computation of Game-Theoretic Parameters

Many of the quantities and parameters studied in game theory can, at least in principle, be computed or approximated. Are there efficient algorithms to compute these quantities? Obvious examples are optimal or near-optimal strategies for various games, states of equilibrium, values and other related quantities. Some recent work on this subject can be found in [KM1], [KM2] and the references therein. They consider the computational complexity of finding max-min strategies for two-person zero-sum game. Most instances of this problem are shown **NP**-complete, the most notable exception being that they provide an algorithm for a game given in extensive form, whose run time is polynomial in the size of the tree. See also Futia [Fu] and Deng Papadimitriou [DP]. The case may be made to include much of the work on linear programming under this heading, because of the well-known connection between LP and zero-sum games (e.g. [Ow]). Significant progress has been made in the design and analysis of algorithms for LP in the past few years (e.g. Megiddo [Mg1]). Among the important advances are the development of efficient internal-point algorithms, better understanding of the average-case behavior of simplex-type algorithms, and the introduction of randomized algorithms to this area. These developments may help in designing better algorithms for a variety of computational problems from game theory.

A particularly fascinating problem in this area is that of computing the value of a stochastic game [Co2]. It is quite easy to show that this problem is in $\mathbf{NP} \bigcap co-\mathbf{NP}$, that is, both upper and lower bounds for the value can be established using short proofs. However, no polynomial time algorithm is known, and the computational complexity of this problem is still unknown. See also section 5.3.

5.2 Games and Logic in Computer Science

This subject has its roots in the connection between games, descriptive set theory and logic. [Mos] and [Ho] are two texts in this general area. Similar consideration turn out to be useful in some areas of theoretical computer science. For example Rabin's work [Ra1] on the decidability of the second order theory of the infinite binary tree was shown by Gurevich and Harrington [GH] to be best viewed from a game theoretic point of view. The method of Fraisse-Ehrenfeucht games [Eh] from mathematical logic turns out to be useful in delineating the expressive power of languages used in computer science. An interesting recent example for this approach is [AF] where a variation of a Fraisse-Ehrenfeucht game is used to show that connectivity problems for directed graphs are harder than for graphs, in a certain technical sense. Games of similar character were applied also in setting up the connection between classes of computational complexity and logical theories as in Papadimitriou's [Pa] "games against nature", which can also be viewed as a forrunner to the theory of interactive proof system (see section 3.2.3). A systematic discussion of this connection can be found in Condon's thesis [Co1].

The notion of *Common Knowledge* which seems to be of interest to economists and game theorists received a good deal of attention in computer science circles as well. Formal theories were set up to capture this notion and decision procedures for the various logical theories which emerge were developed. For survey see the forthcoming book [FHMV] as well as chapter by Geanakopolos [Ge] in this handbook.

5.3 The Complexity of Specific Games

In the mid-70's a large amount of research was dedicated to studying the computational complexity of playing specific board games. For many combinatorial games (e.g. go, chess) it was shown that it is **NP**-hard or even **PSPACE**-hard to find an optimal strategy (technically, one defines a generalized version of chess played on an $n \times n$ board and the complexity is evaluated in terms of n.) While complexity of optimal play is already fairly well-understood, there is still much room for results on the complexity of finding *near*-optimal strategies. The standard reference on **NP**-completeness is [GJ], which lists many of the known results; [BCG] is an encyclopedia of combinatorial games where complexity issues are considered as well. See also [SS] in this handbook.

Much attention was also paid to specific games as actually played by computers, traditional games such as chess, checkers and go as well as more modern ones. Games played by computers motivated a sizable amount of research in Artificial Intelligence (AI), with machines being constructed to play specific games and some methodologies developed to evaluate general game trees. Most notable among those is Knuth and Moore's α-β pruning method [KnM] for efficient evaluation of game trees. This algorithm had many offsprings and ramifications, see e.g., Pearl's book [Pe]. Tarsi [Ta] and Saks Wigderson [SW] consider efficient randomized algorithms for evaluating game trees.

Having mentioned AI, let us briefly point out some interesting connections between this discipline and game theory, both in situations where a number of "intelligent" agents interact as well as in the design of a single agent. In the multi-agent situation, most of the work to date has focused on analyzing protocols for interactions among agents, and the performance of these protocols in various cooperative and competitive scenarios (e.g., [RG], [ZR], [KsW]). In the single agent case as well, AI researchers have begun to use the tools of decision theory and game theory. Traditionally, AI theorists modeled objects such as an agent's goals and plans using mathematical logic, and attempted to define a prescriptive theory of action using the same tool. An alternative approach, placing more emphasis on decision-theoretic considerations (so as to ensure reasoned choice among alternative strategies) is represented, e.g., in [FS], [Hz], [Do], [DB], [De], and the book [RW].

Computers playing classical games are already a field in itself with special-purpose computers and specialized periodicals. Two nontraditional games that may be worth mentioning are Diplomacy, for which a program was developed [KEL] which negotiates with the other players, besides taking the regular moves. Axelrod's experiments with computer programs playing the *prisoner's dilemma* are documented in [Ax]; for a more recent survey see [AD]. In the context of prisoner's dilemma and repeated games, the idea of using bounded rationality to resolve the ensuing paradoxes was raised and considered extensively, see [Ka] and [So] for surveys. This approach ties naturally with methods of computer science where bounds on computational resources are a main concern.

More on this subject can be found in the survey [SS].

5.4 Game-Theoretic Consideration in Computational Complexity

Interesting and largely unexplored connections with game theory arise in complexity theory. Various measures for the computational complexity of functions are considered in this area some of which are defined in a terms of certain cooperative games. For example, let $f = f(x, y)$ be a function from n-bit strings to $\{0, 1\}$. The *communication complexity* of f is defined [Y1] as follows: Player 1 holds x and player 2 holds y and they want to compute $f(x, y)$ by sending messages to each other over a communication channel. This can be done e.g. by player 1 sending x over to player 2 who calculates $f(x, y)$ and sends the outcome over, for a total cost of $n + 1$ bits transmitted. For certain functions f, more efficient algorithms may be devised. For example, if $f(x, y)$ is the parity of $x + y$, it is enough for the first player to transmit a single bit indicating the parity of x, whence the second player can compute f. The smallest upper bound that can be achieved by the best algorithm is called f's communication complexity. Randomized and nondeterministic variations are being considered too. This is an important concept which comes up in many studies of complexity, but no concrete connection with game theory has been set up yet. A comprehensive survey article by Lovasz [Lo] is highly recommended. The *depth* of a boolean circuit (see remark 3.5) is the length of a longest directed path from an input gate to an output gate in the underlying directed graph. (See [BS] for an updated survey of circuit complexity). Given a function g, it is of interest to find the least possible depth of a circuit computing that function. It turns out that the least possible depth can be expressed in terms, of a cooperative two-person game, somewhat like the communication complexity game. This point of view enabled [KW] to solve some interesting problems in circuit complexity.

It should be mentioned that in many search problems, one establishes a lower bound for time complexity by assuming that answers to the queries the algorithm makes, are supplied by a computationally unbounded adversary. The only restriction the adversary obeys is to never give contradictory answers. This standard point of view appears e.g., throughout Knuth's books [Kn].

5.5 Random Number Generation as Games

In recent years randomization has been playing a central role in the theory of efficient algorithms. It was Rabin's [Ra2] insight that randomized computation is a methodology in its own right, and this is by now widely realized. As observed by Megiddo [Mg2] the relationship between randomized and deterministic algorithms resembles that between mixed and pure strategies, though some interesting differences are pointed out as well. An easy but important result of Yao [Yao] is based on viewing the run time of a randomized algorithm as the value of a game between the algorithm and an adversary designing a hard instance for it to run on. The min-max theorem implies that it is possible to replace the analysis of a randomized algorithm running on a hard instance by a discussion where firstly one side selects a probability distribution on problem instances and then the other devises a deterministic algorithm to have a low expected run time on inputs sampled from this distribution. This easy but very useful observation can sometimes simplify the analysis of randomized algorithms.

Randomized computing requires random bit generators, a need which creates new problems. The theory of pseudo-random bit generators was revolutionized by the recent advents in cryptography. Knuth [Kn] explains the classical approach to pseudo random number generation. Goldreich's review [Gr1] explains the new point of view, see also section 3.2 and [Gr2], [Ri]. The property required from a random number generator is that no randomized polynomial-time Turing Machine can distinguish the generator's output from truly random numbers. A fundamental problem in this area has been recently resolved, by Impagliazzo, Levin and Luby [ILL] and Hastad [Hd] who showed:

Theorem 5.1: *[ILL][Hd] If one-way functions exist, then pseudo-random number generators exist.*

A traditional approach to random bit generation is to sample a physical process which "seems random enough". There are no theorems, of course, guaranteeing a truly random behavior of such sources. Consequently, the view developed of an imperfect source of randomness as an *adversary* who tries to supply the sampling program with "bad" random bits. This adversary cannot behave in a completely deterministic way, though, and the program should be smart enough to take advantage of the random aspects of the the source's behavior and extract the randomness needed for its proper operation. U. Vazirani's thesis [V] contains some of the fundamental results along this line of research. Further developments can be found in [CG] and [LLS] where possible connections with control theory are pointed out. Some typical results are: It is impossible to "improve the quality" of individual bits [SV] sampled from such an adversary source. On the other hand the amount of randomness inherent in the source suffices for running randomized algorithms [VV]. A survey of this area may be found in [BLS]. The most recent progress in the area of "amplifying randomness" can be found in [Zu] and the references therein.

5.6 Amortized Cost and the Quality of On-Line Decisions

A common problem which arises in many situations in economics is that decisions made at present have both an immediate impact, as well as consequences for the future. For example, the choice between saving plans of varying yields and time spans affects our situation at future points when interest rates may change. The economists' standard answer to problems such as this is to devise a statistical model for the future and invoke methods of Markov decision processes.

There is a class of examples from data structure theory, which prompted a different, more conservative, approach to similar problems in Computer Science. The aim is to develop decision-making strategies which are *guaranteed* to perform well, regardless of the upcoming sequence of events, and avoiding any statistical assumptions. See section 1.1 where a similar difference in approach between computer scientists and game theorists manifests itself. Analysis of some problems in economics along the same line of thought have been performed e.g. in [Cv], [FV] and the survey [Mh].

Let us present one of the canonical examples in this area: Consider a filing cabinet where many documents are kept. They are kept in no particular order, so when the need for a certain document arises, it has to be searched for sequentially starting at the top of the list. The cost of searching is proportional to the item's ordinal number in the cabinet at that time. To anticipate future requests, items may be permuted at any time, for a unit cost per any swap performed. The *move to front* strategy always locates the requested item and then moves it to the front of the cabinet. Given a sequence of requests, it is possible to find the optimal sequence of permutations which minimizes the overall cost (of searching and permuting). It is surprising, perhaps, that the cost incurred by the "move to front" strategy is at most twice this optimum. Notice that the optimum is achieved based on a complete knowledge of the sequence of requests, while the heuristic operates only on local information, never anticipating the future, and no statistical assumptions are made on the stream of requests.

Sleator and Tarjan's [ST] may be the earliest paper where this paradigm is presented, and [MSr] is a collection of some recent articles in this area. In [BLS] a general framework for dealing with questions like this is laid out. The role of randomization in this area is elucidated in [BBK+], using game theoretic considerations. Answering a question of [MMS], recently Fiat, Rabani and Ravid [FRR] proved the $k-$*server* theorem: A player controls k *servers* which he can move around a metric space (X, d). At the beginning of the game all servers are located at a fixed origin point in X. At each step, an adversary makes a *request*, i.e., he selects a point p of X. The player has to move one of its servers to p, and it is up to him which of the k servers to move. (One may think of some kind of a service company operating k service vehicles and responding to calls of customers around the country.) The cost incurred by the player at this step is the distance between that server's old location and p. [FRR] provide a strategy for the player which guarantees that his cost would be at most $f(k)$ times the least cost to an algorithm who knows all requests in advance. The function f depends only on the number of servers k but not on X or the sequence of requests. It is conjectured that the same holds with $f(k) = k$, but this is still open.

El-Yaniv and Karp [EK] study strategies for refinancing mortgages based on this methodology. Obviously, there is much room here for further research.

References

[AAG] Y. Afek, B. Awerbuch and E. Gafni, Applying Static Network Protocols to Dynamic Networks, *Proc. 28th IEEE Symp. on the Foundations of Computer Science* (1987) 358-370.

[AHU] A. V. Aho, J. E. Hopcroft and J. D. Ullman, **The Design and Analysis of Computer Algorithms** Addison-Wesley, 1974.

[AF] M. Ajtai and R. Fagin, Reachability is harder for directed than for undirected finite graphs, *J. Symbolic Logic* 55(1990) 113 - 150.

[AL] M. Ajtai and N. Linial, The influence of large coalitions, *Combinatorica*, 13(1993) 129 - 145.

[AN] N. Alon and M. Naor, Coin-flipping games immune against linear-sized coalitions, *SIAM J. Computing*, to appear. Also, *Proc. 31st IEEE Symp. on the Foundations of Computer Science* (1990) 46 - 54.

[AS] N. Alon and J. Spencer, **The Probabilistic Method**, Wiley Interscience, 1992.

[Ax] R. Axelrod, **The Evolution of Cooperation**, Basic Books, New York, 1984.

[AD] R. Axelrod and D. Dion, The further development of cooperation, *Science* 242(1988) 1385 - 1390.

[Ba] L. Babai, Trading group theory for randomness, *Proc. 27th IEEE Symp. on the Foundations of Computer Science* (1986)368 - 379.

[BF] I. Barany and Z. Furedi, Mental poker with three or more players, *Information and Control* 59(1983), 84 - 93.

[BDDS] A. Bar-Noy, D. Dolev, C. Dwork, and R. Strong, Shifting Gears: Changing Algorithms on the Fly to Expedite Byzantine Agreement, *Proc. 6th Annual ACM Symp. Principles Distributed Computing* 1987, 42 - 51.

[Be] W. Beckner, Inequalities in Fourier Analysis, *Annals of Mathematics* 102(1975), 159 - 182.

[BBK+] S. Ben-David, A. Borodin, R. Karp, G. Tardos and A. Wigderson, On the power of randomization in online algorithms, *Proc. 22nd Annual ACM Symp. on Theory of Computing* (1990) 379 - 386.

[BO] M. Ben-Or, Another advantage of free choice: Completely asynchronous agreement protocols, *Proc. 2nd Annual ACM Symp. Principles Distributed Computing* (1983) 27-30.

[BGW] M. Ben-Or, S. Godwasser and A. Wigderson, Completeness theorems for non-cryptographic fault-tolerant distributed computation, *Proc. 20th Annual ACM Symp. on Theory of Computing* 1988, 1 - 10.

[BL] M. Ben-Or and N. Linial, Collective coin flipping, in **Randomness and Computation**, (S. Micali, ed.) Academic Press, 1989, 91 - 115. Early version: M. Ben-Or and N. Linial, Collective coin flipping, robust voting games and minima of Banzhaf values, *Proc. 26th IEEE Symp. on the Foundations of Computer Science* Portland, 1985, 408 - 416.

[BLS] M. Ben-Or, N. Linial and M. Saks, Collective coin flipping and other models of imperfect information, in **Colloq. Math Soc. Janos Bolyai no. 52, Combinatorics** Eger 1987, 75-112.

[BCG] E. R. Berlekamp, J. H. Conway and R. K. Guy, **Winning Ways for your Mathematical Play** (2 vols.), Academic Press, 1982.

[BG] P. Berman and J. Garay, Efficient Distributed Consensus with $n = (3 + \varepsilon)t$ Processors, *Proc. 5th International Workshop on Dist. Algorithms,* LNCS (Springer-Verlag), October 1991.

[Bs] B. Bollobas, **Combinatorics**, Cambridge University Press, 1986.

[BN1] . B. Boppana and B. O. Narayanan, The biased coin problem, *Proc. 25th ACM Symp. on Theory of Computing* (1993) 252-257.

[BN2] . B. Boppana and B. O. Narayanan, Collective coin flipping and leader election with optimal immunity, manuscript, May 1993.

[BS] R. B. Boppana and M. Sipser, The Complexity of Finite Functions, in **Handbook of Theoretical Computer Science, vol. A**, (J. van Leeuwen ed.), The MIT Press/Elsevier, (1990) 757 - 804.

[BoLS] A. Borodin, N. Linial and M. Saks, Online task systems, *SIAM J. on Computing* 18 (1989) 268 - 277.

[BoG] E. Borowsky and E. Gafni, Generalized FLP impossibility results for t-resilient asynchronous computations, *Proc. 25th ACM Symp. on Theory of Computing* (1993) 91 - 100.

[BKK+] J. Bourgain, J. Kahn, G. Kalai, Y. Katznelson and N. Linial, The influence of variables in product spaces, *Israel Jour. Math.*, 77(1992) 55-64.

[Br] G. Bracha, An $O(\log n)$ expected rounds randomized Byzantine generals protocols *J. ACM* 32(1985) 824 - 840.

[CCD] D. Chaum, C. Crepeau and I. Damgaard, Multiparty unconditionally secure protocols, *Proc. 20th Annual ACM Symp. on Theory of Computing* (1988) 11 - 19.

[CD] B. Chor and C. Dwork, Randomization in Byzantine Agreement, in **Randomness and Computation**, (S. Micali, ed.) Academic Press, 1989, 443 - 497.

[CG] B. Chor and M. Gerèb-Grauss, On the Influence of Single Participants in Coin Flipping Schemes, *SIAM Jour. Disc. Math.*, 1 (1988), 411 - 415.

[CGK] B. Chor, M. Geréb-Graus, and E. Kushilevitz, Private Computations Over the Integers, *SIAM Jour. Comput*, in press. Also *Proc. 30th IEEE Conf. Foundation of Computer Science* (1990), 335 - 344.

[CGo] B. Chor and O. Goldreich, Unbiased bits from sources of weak randomness and probabilistic communication complexity, *SIAM J. Computing* 17(1988) 230 - 261.

[CGMA] B. Chor, S. Goldwasser, S. Micali and B. Awerbuch, Verifiable secret sharing and achieving simultaniety in the presence of faults, in *Proc. 26th IEEE Conf. Foundation of Computer Science* (1985) 383 - 395.

[CK] B. Chor and E. Kushilevitz, A Zero-One Law for Boolean Privacy, *SIAM Jour. Disc. Math.*, 4 (1991) 36 - 47.

[Co1] A. Condon, **Computational Models of Games**, *ACM Distinguished Dissertation Series*, The MIT Press, 1988.

[Co2] A. Condon, The complexity of stochastic games, *Information and Computation*, 96 (1992) 203 - 224.

[CL] J. Cooper and N. Linial, Fast perfect-information leader election protocol with linear immunity, *25th Symposium on the Theory of Computing* (1993) 662 - 671.

[Cv] T. M. Cover, Universal Portfolio's, Technical report no. 66, July 1988, Dept. of Statistics, Stanford University.

[DB] T. Dean and M. Boddy, An Analysis of Time-Dependent Planning, in *Proceedings of the Seventh National Conference on Artificial Intelligence* (1988) 49 - 54.

[DP] X. Deng and C.H. Papadimitriou, "On the complexity of cooperative solution concepts", unpublished manuscript, December 1987.

[DH] W. Diffie and M. E. Hellman, New Directions in Cryptography, *IEEE Trans. Information Theory 22(1976) 644 - 654.*

[Do] J. Doyle, Rationality and its Roles in Reasoning (Extended Abstract), in *Proceedings of the Eighth National Conference on Artificial Intelligence* (1990) 1093 - 1100.

[Du] M. Dubiner, private communication, 1986.

[DPPU] C. Dwork, D. Peleg, N. Pippenger and E. Upfal, Fault tolerance in networks of bounded degree, *SIAM J. Computing*, 17(1988) 975 - 988.

[DM] H. Dym, and H. P. McKean, **Fourier Series and Integrals**, Academic Press, 1972.

[Eh] A. Ehrenfeucht, An application of games to the completeness problem for formalized theories *Fund. Math.* 49(1961), 129 - 141.

[EK] R. El-Yaniv and R. M. Karp, The mortgage problem, *Proc. 2nd Israel Symp. on Theory and Computing Systems* (1993) 304 - 312.

[FHMV] R. Fagin, J. Y. Halpern, Y. Moses and M. Y. Vardi, **Reasoning About Knowledge** to appear 1994.

[Fe] P. Feldman, private communication, 1986.

[FS] J. A. Feldman and R. F. Sproull, Decision Theory and Artificial Intelligence II: The Hungry Monkey, *Cognitive Science* 1 (1977) 158 - 192.

[FM] P. Feldman and S. Micali, Optimal algorithms for Byzantine Agreement, *Proc. 20th Annual ACM Symp. on Theory of Computing* (1988) 148 - 161.

[FRR] A. Fiat, Y. Rabani and Y. Ravid, Competitive k-server algorithms, *31st Symposium on the Foundations of Computer Science*, (1990) 454 - 463.

[Fi] M. J. Fischer, The concensus problem in unreliable distributed systems (a brief survey), Technical report, Dept. of Computer Science, Yale University, 1983.

[FLM] M. J. Fischer, N. A. Lynch and M. Merritt, Easy impossibility proofs for distributed consensus problems, *Distributed Computing* 1(1986), 26-39.

[FLP] M. J. Fischer, N. A. Lynch and M. Patterson, Impossibility of distributed concensus with one faulty processor, *J. ACM* 32 (1985) 374 - 382.

[FV] D.P. Foster and R. V. Vorha, On Combining Forecasts, Technical report, September '87, Ohio State university.

[Fu] C. Futia, "The complexity of economic decision rules", *J. of Mathematical Economics* 4, 1977.

[GM] J. A. Garay and Y. Moses, Fully polynomial byzantine agreement in $t+1$ rounds, *Proc. 25th ACM Symp. on Theory of Computing* (1993) 31 - 41.

[GJ] M. R. Garey and D. S. Johnson, **Computers and Intractability - a Guide to the Theory of NP-Completeness** Freeman, 1979.

[Ge] J. Geanakopolos, Common Knowledge, this handbook.

[GR] M. R. Genesereth, and J. S. Rosenschein, Cooperation Without Communication, in **Readings in Distributed Artificial Intelligence**, (A. H. Bond and L. Gasser, eds.), Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988, 227 - 234.

[GZ] I. Gilboa and E. Zemel, Nash and correlated equilibria: Some complexity considerations, *Games and Economic Behavior* 1 (1989).

[Gr1] O. Goldreich, Randomness, Interactive Proofs and Zero-Knowledge – A survey, in **The Universal Turing Machine, A Half Century Survey** (R. Herken ed.) Kammerer and Unverzagt, Hamburg - Berlin, 1988, 377 - 405.

[Gr2] O. Goldreich, **Foundations of Cryptography**, Class Notes, Computer Science Department, Technion, Haifa, Israel, 1989.

[GGL] O. Goldreich, S. Goldwasser and N. Linial, Fault-tolerant computation in the full information model, *32nd Symposium on the Foundations of Computer Science*, 1991, 447 - 457.

[GL] O. Goldreich and L. A. Levin, A hard-core predicate for all one-way functions, *Proc. 21st ACM Symp. on Theory of Computing* (1989) 25 - 32.

[GMW1] O. Goldreich, S. Micali and A. Wigderson, Proofs that yield nothing but their validity, or all languages in **NP** have zero-knowledge proofs, *J. ACM* 38 (1991) 691-729. Also, *27th Symposium on the Foundations of Computer Science*, (1986) 174 - 187.

[GMW2] O. Goldreich, S. Micali and A. Wigderson, How to play any mental game, or, a completeness theorem for protocols with honest majority, *Proc. 19th ACM Symp. on Theory of Computing* (1987) 218 - 229.

[GP] O. Goldreich and E. Petrank, The best of both worlds: guaranteeing termination in fast randomized byzantine agreement protocols, *Information Processing Letters* 36(1990) 45 - 49.

[GoM] S. Goldwasser and S. Micali, Probabilistic encryption, *Jour. Comput. System Sci.* 28 (1984) 270 - 299.

[GMR] S. Goldwasser, S. Micali and C. Rackoff, The knowledge complexity of interactive proof systems, *SIAM J. Computing*, 18(1988) 186 - 208.

[GY] R. L. Graham and A. C. Yao, On the improbability of reaching Byzantine Agreements, *Proc. 21st ACM Symp. on Theory of Computing* (1989) 467 - 478.

[GH] Y. Gurevich and L. Harrington, Trees, automata and games, *Proc. 14th ACM Symp. on Theory of Computing* 1982, 60-65.

[HM] J. Y. Halpern and Y. Moses, Knowledge and Common Knowledge in a Distributed Environment, *J. ACM* 37(1990), 549 - 587.

[HW] G. H. Hardy and E. M. Wright, **An Introduction to the Theory of Numbers** Oxfor University Press, 5th ed., 1979.

[Ha] S. Hart, A note on the edges of the n-cube, *Discrete Mathematics* 14(1976) 157 - 163.

[Hd] J. Hastad, Pseudo-random generators under uniform assumptions, *Proc. 22nd ACM Symp. on Theory of Computing* (1990) 395 - 404.

[HS] M. Herlihy and N. Shavit, The asynchronous computability theorem for t-resilient tasks, *Proc. 25th ACM Symp. on Theory of Computing* (1993) 111 - 120.

[Ho] W. Hodges, **Building Models by Games**, Cambridge University Press, 1985.

[Hz] E. J. Horvitz, Reasoning Under Varying and Uncertain Resource Constraints, in *Proceedings of the Seventh National Conference on Artificial Intelligence* (1988) 111 - 116.

[ILL] R. Impagliazzo, L. A. Levin and M. Luby, Pseudo-random generation from one-way functions, *Proc. 21st ACM Symp. on Theory of Computing* (1989) 12 - 24.

[J] D. S. Johnson, A catalog of complexity classes, in **Handbook of Theoretical Computer Science, vol. A**, (J. van Leeuwen ed.), The MIT Press/Elsevier, (1990) 67 - 161.

[KKL] J. Kahn, G. Kalai and N. Linial, The influence of variables on boolean functions, *29th Symposium on the Foundations of Computer Science*, 1988, 68 - 80.

[Ka] E. Kalai, Bounded rationality and strategic complexity in repeated games, in **Game Theory and Applications** (T. Ichiishi, A. Neyman and Y. Tauman eds.) Academic Press 1990, 131 - 157.

[KW] M. Karchmer and A. Wigderson, Monotone circuits for connectivity require super-logarithmic depth, *SIAM J. Discrete Math.* 3(1990) 255 - 265.

[Kn] D. Knuth, **The Art of Computer Programming** (3 vols.), Addison - Wesley, 1973.

[KnM] D. E. Knuth and R. N. Moore, An analysis of the alpha-beta pruning, *Artificial Intelligence* 6(1975) 293 - 322.

[KM1] D. Koller and N. Megiddo, The complexity of two-person zero-sum games in extensive form, *Games and Economic Behavior* 4 (1992) 528–552.

[KM2] D. Koller and N. Megiddo, Finding mixed strategies with small supports in extensive form games, Research Report RJ 8380, IBM Almaden Research Center, 1991. International Journal of Game Theory, to appear.

[KEL] S. Kraus, E. Ephrati and D. Lehmann, Negotiation in a Non-cooperative Environment, *Journal of Experimental and Theoretical Artificial Intelligence,* 4(1991) 255 - 282.

[KsW] S. Kraus and J. Wilkenfeld, Negotiations over Time in a Multi Agent Environment: Preliminary Report, in *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, (1991) 56 - 61.

[KuM] E. Kushilevitz and Y. Mansour, Learning decision trees using the Fourier spectrum, *Proc. 23rd Annual ACM Symp. on Theory of Computing* (1991) 455 - 464.

[LSP] L. Lamport, R. Shostak, and M. Pease, The Byzantine generals problem, *ACM TOPLAS* 4 (1982) 382 - 401.

[LeL] A. K. Lenstra and H. W. Lenstra Jr., Algorithms in Number Theory, in **Handbook of Theoretical Computer Science, vol. A**, (J. van Leeuwen ed.), The MIT Press/Elsevier, (1990) 673 - 715.

[LP] H. R. Lewis and C. H. Papadimitriou, **Elements of the theory of computation**, Prentice Hall, 1981.

[LLS] D. Lichtenstein, N.Linial and M. Saks, Some extremal problems arising from discrete control processes, *Combinatorica*, 9 (1989) 269-287.

[LW] L. Loomis, and H. Whitney, An inequality related to the isoperimetric inequality, *Bull. Amer. Math. Soc.* 55(1949) 961-962.

[Lo] L. Lovasz, Communication complexity: A survey, in **Paths, Flows and VLSI Layout**, (B. Korte, L. Lovasz, H. J. Prömel and A. Schrijver eds.) Springer, 1990, 235 - 265.

[LFKN] C. Lund, L. Fortnow, H. Karloff and N. Nisan, Algebraic methods for interactive proof systems, *J. ACM*, 39 (4), 1992, pp. 859-868. Also, *Proc. 31st Annual IEEE Symposium on Foundations of Computer Science* (1990) 16 - 25.

[MS] J. MacWilliams, and N. Sloane, **The Theory of Error Correcting Codes** North-Holland, 1977.

[MSr] L. A. McGeoch and D. D. Sleator (editors), *Workshop on On-Line Algorithms,* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 7, American Mathematical Society, Providence, 1991.

[Mh] E. Mahmond, Accuracy in Forecasting: A survey, *J. Forecasting* 3 (1984), 139-159.

[MMS] M. S. Manasse, L. A. McGeoch and D. D. Sleator, Competitive algorithms for server problems, *J. Algorithms* 11(1990) 208 - 230.

[Mg1] N. Megiddo, On the complexity of linear programming, in **Advances in economic theory Fifth world congress**, T. Bewley, ed., Cambridge University Press, Cambridge, 1987, 225 - 268.

[Mg2] N. Megiddo, On probabilistic machines, bounded rationality, and average-case complexity, Research Report RJ 7039, IBM Almaden Research Center, San Jose, California, 1989.

[MgW] N. Megiddo and A. Wigderson, On play by means of computing machines, in *Theoretical Aspects of Reasoning about Knowledge*, J. Y. Halpern, ed., Morgan Kaufmann Publishers, Inc., Los Altos, California, 1986, 259 - 274.

[Me] K. Menger, Zur allgemeinen Kurventheorie, *Fund. Math.* 10(1927) 95 - 115.

[MDH] Y. Moses, D. Dolev and J. Y. Halpern, Cheating husbands and other stories: a case study of knowledge, action, and communication, *Distributed Computing* 1(1986) 167 - 176.

[MW] Y. Moses and O. Waartz, Coordinated tranversal: $(t+1)$-round Byzantine Agreement in polynomial time, *Proc. 29th Annual IEEE Symposium on Foundations of Computer Science* (1988) 246 - 255.

[Mos] Y. N. Mosschovakis, **Descriptive set theory**, North Holland, Amsterdam, 1980.

[Na] M. Naor, Bit commitment using pseudorandomness, *Journal of Cryptology* 4(2), 1991, 151-158.

[Ow] G. Owen, **Game Theory** 2nd ed. Academic Press, New York 1982.

[Pa] C. Papadimitriou, Games against nature, *Jour. Comp. System Sci.* 31(1985) 288 - 301.

[Pe] J. Pearl, **Heuristics: intelligent search strategies for computer solving problems**, Addison-Wesley, 1984.

[PSL] M. Pease, R. Shostak, and L. Lamport, Reaching agreement in the presence of faults, *J. ACM* 27(1980) 228 - 234.

[Ra1] M. O. Rabin, Decidability of second-order theories and automata on infinite trees, *Trans. AMS* 141(1969) 1 - 35.

[Ra2] M. O. Rabin, Probabilistic algorithms, in **Algorithms and Complexity** (J. Traub ed.) Academic Press, 1976, 21 - 39.

[Ra3] M. O. Rabin, Randomized Byzantine Generals, *Proc. 24th Annual IEEE Symposium on Foundations of Computer Science* (1983) 403 - 409.

[RB] T. Rabin and M. Ben-Or, Verifiable secret sharing and mutiparty protocols with honest majority, *Proc. 21st Annual ACM Symp. on Theory of Computing* (1989) 73 - 85.

[Ri] R. Rivest, Cryptography, in **Handbook of Theoretical Computer Science, vol. A**, (J. van Leeuwen ed.), The MIT Press/Elsevier, (1990) 719 - 755.

[RG] J. S. Rosenschein and M. R. Genesereth, Deals Among Rational Agents, in **The Ecology of Computation**, (B. A. Huberman ed.), North-Holland, 1988, 117 - 132.

[RW] S. Russell and E. Wefald, **Do the Right Thing** MIT Press, 1991.

[S] M. Saks, A robust noncryptographic protocol for collective coin flipping, *SIAM J. Discrete Math.* 2(1989) 240 - 244.

[SW] M. Saks and A. Wigderson, Probabilistic decision trees and the complexity of evaluating game trees, *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science* (1985) 29-38.

[SZ] M. Saks and F. Zaharoglou, Wait-free k-set agreement is impossible: The topology of public knowledge, *Proc. 25th ACM Symp. on Theory of Computing* (1993) 101 - 110.

[SV] M. Santha and U. V. Vazirani, Generating quasi-random sequences from semi-random sources, *Jour. Computer and System Sci.*, 33(1986) 75 - 87.

[Sh1] A. Shamir, How to share a secret, *Comm. ACM* 22(1979) 612 - 613.

[Sh2] A. Shamir, **IP = PSPACE** *Proc. 31st Annual IEEE Symposium on Foundations of Computer Science* (1990) 11 - 15.

[SS] H. A. Simon, and J. Schaeffer, The game of chess, This handbook.

[ST] D. Sleator and R. E. Tarjan, Amortized efficiency of list update and paging rules, *Comm. ACM* 28(1985) 202 - 208.

[So] S. Sorin, Repeted games with complete information, This handbook.

[Ta] M. Tarsi, Optimal Search on Some Game Trees, *J. ACM* 30(1983) 389-396.

[V] U. V. Vazirani, Randomness, adversaries and computation, PhD thesis, U. C. Berkeley, 1986.

[VV] U. V. Vazirani and V. V. Vazirani, Random polynomial time is equal to semi-random polynomial time, *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science* (1985) 417 - 428.

[Yao] A. C. Yao, Probabilistic computation: Towards a unified measure of complexity, *Proc. 18th Annual IEEE Symposium on Foundations of Computer Science* (1977) 222 - 227.

[Y1] A. C. Yao, Some compleixty questions related to distributed computing, *Proc. 11th Annual ACM Symp. on Theory of Computing* (1979) 209 - 213.

[Y2] A. C. Yao, Prototcols for secure communication, *Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science* (1982) 160 - 164.

[Y3] A. C. Yao, How to generate and exchange secrets, *Proc. 27th IEEE Symp. on the Foundations of Computer Science* (1986) 162 - 167.

[ZR] G. Zlotkin and J. S. Rosenschein, Incomplete Information and Deception in Multi-Agent Negotiation, Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, 1991, Sydney, Australia, pages 225–231.

[Zu] D. Zuckerman, Simulating BPP using general weak random source, *Proc. 32nd IEEE Symp. on the Foundations of Computer Science* (1991) 79 - 89.