A Fair Protocol for Signing Contracts

MICHAEL BEN-OR, ODED GOLDREICH, SILVIO MICALI, AND RONALD L. RIVEST

Abstract—Two parties, A and B, want to sign a contract C over a communication network. To do so, they must "simultaneously" exchange their commitments to C. Since simultaneous exchange is usually impossible in practice, protocols are needed to approximate simultaneity by exchanging partial commitments in piece by piece manner. During such a protocol, one party or another may have a slight advantage; a "fair" protocol keeps this advantage within acceptable limits. We present a new protocol that is fair in the sense that, at any stage in its execution, the conditional probability that one party cannot commit both parties to the contract given that the other party can, is close to zero. This is true even if A and B have vastly different computing powers, and is proved under very weak cryptographic assumptions. Our protocol has the following additional properties:

- during the procedure the parties exchange probabilistic options for committing both parties to the contract;
- the protocol never terminates in an asymmetric situation where party A knows that party B is committed to the contract while he is not;
- the protocol makes use of a weak form of a third party (judge). If both A and B are honest, the judge will never be called upon. Otherwise, the judge rules by performing a simple computation. No bookkeeping is required of the judge.

I. Introduction

LET $A, B, \cdots$ be users who can exchange messages over a communication network. For example, they may be the users of the ordinary mail system, or of a telephone network, or a modern computer network.

A. Signatures

We will assume that a *signature scheme* is adopted in the network. Two key properties are required from a signature scheme. The first is *unforgeability*, which means that only user U can create U's signature on message m. The second is *universal verification*, whereby any other user should be able to verify that U's signature on message m is indeed a valid one. An instance of a signature scheme may be provided by ordinary "hand-written" signatures. Hand-written signatures are believed to be hard to forge and can be universally verified as trusted notary publics keep samples of each user's signature. Another instance of a signature scheme, more suitable for computer networks, is a "digital signature scheme." This notion was introduced by Diffie and Hellman [5] and first implemented by Rivest *et al.* [15]. The strongest notion of security for a digital signature scheme was suggested and first implemented (based on a weak and general complexity assumption) by Goldwasser *et al.* [11]. A simpler scheme that also uses a more general complexity assumption was recently proposed by [3]. A historical account of the problem of digital signature can be found in [11].

B. The Problem of Contract Signing

Two users A and B have negotiated over the network a contract C and now want to obtain each other's commitment to it. They communicate by taking turns sending messages to each other. Neither party wants to be committed to C unless the other one is. This calls for the exchange of commitments to occur simultaneously. Is this at all possible? Certainly not if we identify a *commitment* to the contract with a *signature* to the text of the contract [8]. Indeed, simultaneity cannot be met in our discrete world. Thus one should settle for protocols that are "fair" in the sense that no party may gain much advantage over the other one. Of course, besides being fair, it is *required* that if both parties follow the protocol properly. Then, at its termination, both will be committed to C.

The meaningfulness of a solution to the "contract signing problem" will depend on the acceptability of the fairness definition on which it is based. Two main approaches to fairness have been considered. The first one interprets fairness as deriving from equal computational effort. The second approach, adopted in this paper, interprets fairness as deriving from high probabilistic correlation.

C. Computational Approaches to Fairness

Even [6], Blum [1], and Even *et al.* [7], proposed a computational interpretation of fairness. This approach requires that at any stage during the execution of the protocol, the computational effort required from the parties to obtain each other's commitment to C be approximately equal. Here, the computational difficulty of a task

Manuscript received August 8, 1986; revised January 5, 1989. This work was supported in part by a Weizmann Postdoctoral Fellowship, in part by the National Science Foundation under Grants DCR-8413577 and MCS80-06938, and in part by an IBM Faculty Development Award. The material in this paper was partially presented at the 12th ICALP, Greece, 1985.

M. Ben-Or was with the Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA. He is now with the Institute of Mathematics and Computer Science, Hebrew University, Jerusalem, Israel.

O. Goldreich was with the Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA. He is now with the Computer Science Department, Technion–Israel Institute of Technology, Haifa 32000, Israel.

S. Micali and R. L. Rivest are with the Laboratory for Computer Science, Massachusetts Institute of Technology, 545 Technology Square, Cambridge, MA 02139.

IEEE Log Number 8933051.

should be thought as the number of machine operations needed to complete the task. Typically, during the execution of a protocol (constructed according to this approach), the computational difficulty of reconstructing the counterpart's commitment to C should decrease (for both parties) by approximately equal amounts at each step, until it vanishes. An advantage of this approach is that it does not require the intervention of a third party (in any form).[1] Unfortunately, this approach suffers from three major weaknesses.

First Weakness — Equal Computing Power Assumption: This fairness condition only relates the computational efforts of the two parties in terms of the number of operations. However, if party A can execute 100000 operations per second while B can execute 1000000000 operations per second, then this notion of "fairness" is hardly acceptable. (Suppose that at some step of the protocol, each party needs 10^{12} operations to obtain his counterpart's commitment to the contract. Then A needs four months to obtain the commitment, while B can obtain it in less than 17 min.) Thus the computational approach to fairness is meaningful only if both parties are assumed to have equal computing power as was indeed assumed in [1], [6], [7].

We feel that to assume equal computing power is both unrealistic in practice and undesirable from a theoretical point of view. In real life parties often may have different computing power (e.g., consider a large commercial firm and an individual). Also, it is difficult for a party to estimate the other's computing ability, as one can always pretend to be less powerful. Jumping ahead, we remark that the probabilistic approach, as well as our protocol, is valid even if the parties have different—but still feasible —computing powers.

Second Weakness — The Disrupting Effect of Early Stopping: In case one party stops prematurely, this approach guarantees that both parties face the same computational effort. However, the approach does not specify instructions for the honest party in this case. We believe that this deficiency is not a coincidence, as there are no reasonable alternatives. Note that party A is in a difficult situation if the contract specifies that he must take some actions (such as buying stocks) within one week, yet party B has terminated the contract signing procedure at the point where A has one year of computing to do to find out if B is really committed. A major difficulty here is that there is no near-term deadline by which the signing process cleanly terminates with either both parties bound to the contract or neither party bound.

Third Weakness — The Difficulty of Proving Correctness: This difficulty may not be inherent but is certainly a formidable one. Proving that certain problems cannot be efficiently solvable seems to be hard. Proving that the computational difficulty of certain problems decreases by a fixed amount, when releasing some "partial information," seems even harder. Typically, the correctness of Even *et al.* protocol [7] follows from the assumption that "ideal" trapdoor one-way permutations exist and the assumption that "uniformly hard" one-way functions f exist. By this they mean that given $f(x)$ and k bits of (information about) x, where x is n-bit long, the best algorithm for computing x essentially consists of trying all the 2^{n-k} possibilities for the remaining bits, a strong assumption indeed.[2]

By contrast, the correctness of our protocol follows from a much weaker complexity assumption: the existence of one-way functions f. Such f's need not be "ideal" or uniformly hard. In particular, it may be that half of the bits of x are easy to compute on input $f(x)$. We only require that (all of) x is not efficiently computable on input $f(x)$. If the communication network is a computer network or a telephone network, we also need particularly strong digital signature schemes. Such schemes have been proven to exist under simple complexity assumption [11].

D. A Probabilistic Approach to Fairness

Rabin [14] proposed to exchange commitments to a contract with the help of a trusted third party. The third party proposed by Rabin publicly broadcasts, at regular intervals (say each day), a randomly chosen integer between 1 and 100. Parties a and B can then "sign contracts" by agreeing on a future date D and exchanging signed messages of the form: "I am committed to C if integer i is chosen on date D." A goes first. Party B will send his ith message *only* if he gets A's ith message. Similarly A sends her $i+1$st message *only* if she receives B's ith message. All messages should be exchanged prior to date D. Party B may try to cheat by not sending his ith message after receiving A's ith message. The advantage he gains by doing so is not too large: the probability that he will have a "signed contract," on date D, but A will not, equals $1/100$. In this sense, the protocol is "fair."

It should be noted that the above probabilistic statements were made with respect to future actions of a third party. We claim that reference to a third party, in such statements, is inherent. Suppose that A and B are alone in the world (i.e., no third parties exist). Thus, when party A cuts the communication party B is left by himself. Whatever B is able to compute in feasible time at the moment communication is terminated equals whatever B may be able to compute in the near future, since no additional information will be received. Thus either B has A's com-

[1] When we say that a contract signing scheme does not require the intervention of a third party in any form, we mean intervention for the mere technical purpose of determining whether or not the contact is properly signed. This technical question (which nowadays is taken for granted in disputes) should not be confused with the essential role of the trusted third party (i.e., the judge as representing the legal system) in determining the obligations and compensations arising from the contract (which may be subject to legal interpretation) and offering means for forcing the parties to fulfill his judgement.

[2] Even's protocol [*E*] (as well as its simplified version [*G*]) requires an additional nonmathematical assumption: that the contract's content has a fixed and known value. The correctness of Blum's protocol [*B*1] was based on the assumption that a particular computational problem, related (but not known to be computationally equivalent) to integer factorization, was infeasible. Unfortunately, this problem has been shown to be efficiently solvable by Hastad and Shamir [HS].

mitment to C (and can check that this is the case) or he does not. Only inserting a third party into the picture changes the situation. The third party and "unpredictable" actions he may conduct in the future provide the basis for probabilistic statements (which can be interpreted as speculations on the "verdict" of the third party).

Thus the third party's actions will determine whether a party is committed to C. Therefore, it is natural to think of the third party as being a judge and call his actions the verdict. In general, we will consider a probability space generated by the parties and possibly the third party (the judge). This will typically define two random variables, one associated with each party, which represent whether that party has some option or capability associated with the contract (such as having the commitment of the other party, or the option to cause the judge to commit both parties to the contract). These random variables should be closely related throughout the execution of the protocol. A precise definition of the relation between these variables is the essence of the probabilistic interpretation of fairness (see Section II). Typically, the probability that the judge will rule that A has this option or capability increases throughout the execution of a protocol from zero to one. This increase is gradual so that the probability that A has such an option is at all times nearly equal to the probability that B has such an option.

E. Intervention by a Third Party

In this paper we adopt a probabilistic approach to fairness. As argued before, such an approach requires the existence and possible intervention of a third party. Relying on the existence of a trusted third party is indeed a drawback, but we believe it to be preferable to the major disadvantages of the computational approach to fairness discussed in Section I-C.

Because we rely on a trusted third party in solutions to the contract signing problem, the quality of the solution we obtain will depend on the role such a party plays in it: the more "inconspicuous" and efficient the third party is, the better the solution is. The third party we use in our solution is a simple probabilistic algorithm, called the *judge*. The judge is inactive until he is invoked. The judge can rule on whether a contract is binding on both parties, in the presence of only one party. Furthermore, the judge is invoked only in case of dispute and does not need to store past verdicts. The mildness of the (third party's) intervention in our solution can be demonstrated by comparing it with other forms of trusted third parties proposed in previous solutions to the "contract signing problem."

A simple folklore solution to signing contracts is a cancellation center which stores all invalidated contracts (see [7]). Parties commit themselves to a contract C by exchanging messages of the form: "I'm committed to C unless I've deposited a cancellation notice in the center by date D," where D is some future date. This naive solution suffers from serious practical drawbacks. The paperwork involved in maintaining the cancellation center is tremendous. Even more disturbing is the fact that A must contact the cancellation center if he wants to verify whether C is binding (as B could have cancelled C without telling A). This is the case even if both A and B are honest and wish to execute the protocol properly.

As discussed in Section I-D, Rabin [14] proposed the use of a trusted third party who broadcasts randomly chosen integers at fixed times. We point out that this third party must always be active, regardless of the honesty of all parties and of whether his output is being referred to.

II. The Definition of Fairness

The protocol Rabin proposed for contract signing involved the exchange of (partial) *commitments*. In his protocol an interval of time always arises when one party is committed to the contract, but the other is not; however, the protocol is designed so that the parties are not able to recognize this interval. Nonetheless, this represents a weakness in Rabin's protocol because, if the protocol were stopped during this interval, an asymmetrical situation would arise. In the protocol proposed here, this intrinsic weakness appears in a weaker form. This is accomplished by having the parties exchange "privileges" rather than commitments, where privilege is the power to cause the judge to rule that the contract is binding on both parties. (We say that a party is privileged when s/he is capable of causing the judge to so rule.) It is true that in our protocol one party may be privileged while the other is not; however, the privileged party does not *know* he is so privileged, and furthermore, this privilege is merely the power to force a symmetric situation. In Rabin's protocol, one party may know that the other party is committed while he is not, whereas in our protocol any attempt to resolve the uncertainty by appealing to the judge forces the situation to become symmetric.

In this section we present a probabilistic definition of fairness. We assume that we will be dealing with viable protocols, in the following sense.

Definition: We say that a protocol is *viable* if, when both parties follow the protocol properly, the protocol terminates with both parties being committed to the contract.

The probability space over in which events will be defined must be specified. In general, we will consider a probability space generated by the parties and possibly the third party (the judge). We define two random variables associated with the state of being privileged of each party. These random variables will be closely related throughout the execution of the protocol. We first quantify the statement that two $\{0,1\}$-random variables are "closely related."

A. Closely Related Events: A Priori Versus Conditional Probability

Again, the essence of our probabilistic approach resides in requiring that the event "A is privileged" and the event "B" is privileged" are closely related. How can this be

made formal? We consider two reasonable alternatives:

1) requiring that the probability that "A is privileged but B is not" is always small (similarly for B privileged and A not);
2) requiring that the conditional probability that "B is not privileged" given that "A is privileged" is always small (unless the event "A is privileged" is very unlikely). (Similarly for B privileged and A not.) (Without the relaxation no viable protocol can satisfy the condition, since at some step one of the parties becomes privileged with nonzero probability while his counterpart is privileged with zero probability.)

Notice that the second requirement implies the first. We believe that the second requirement is a better measure for fairness, since it better models the intuitive notion of fairness as simultaneity ("if A is privileged then B is privileged too" and vice versa). To illustrate the difference between the alternatives, we consider Rabin's protocol (see Section I-D). At every stage in that protocol, the probability that A is committed but B is not committed is $1/100$. However, after A has sent i messages, but before B has replied, the conditional probability that "B is not committed to the contract" given that "A is committed to the contract" equals $1/i$ (not $1/100$).

B. Formal Setting

We let $v > 0$ determine our measure of "negligible" probability; namely, we will not care above events occurring with probability less than v. For example, one may choose $v = 2^{-100}$. We will let ϵ denote an upper bound on the probability that one party is not privileged given that the other is privileged (provided that this event does not have negligible probability).

Definition: A contract signing protocol is (v, ϵ)-*fair for* A if the following holds, for any contract C, when A follows the protocol properly. At any step of the protocol in which the probability that B is privileged is greater than v, the conditional probability that "A is not privileged," given that "B is privileged," is at most ϵ.

A protocol is (v, ϵ)-fair if it is (v, ϵ)-fair for both A and B.

III. On the Number of Messages Exchanged in a (v, ϵ)-Fair Protocol

In this section we derive a lower bound on the length of the shortest possible (v, ϵ)-fair protocol. The proof also yields the optimal sequence of probabilities $p_1, p_2, \cdots$, such that after the ith transmission the recipient is privileged with probability p_i.

A protocol for exchanging privileges is a sequence of message exchanges hereafter called *steps*. At each step, one of the parties sends a message to the other. Without loss of generality, no party sends messages in two consecutive steps; i.e., each party alternately sends and receives messages. Let us denote the party which sends (respectively, receives) a message in step i by S_i (respectively, R_i). Let E_i denote the event "after step i, party R_i is privileged" (i.e., the judge when invoked by R_i will rule that the contract is binding on both parties).

An analysis follows of the implication of the (v, ϵ)-fairness requirement on the number of steps in a viable protocol. For simplicity, we assume here that the number of steps is independent of the contract C, and denote this number by $\#(v, \epsilon)$. By the viability requirement, upon termination of the protocol each of the two parties has to be privileged with probability 1. Thus

$$\Pr(E_{\#(v,\epsilon)-1}) = 1.$$

By the (v, ϵ)-fairness, for every i, $1 \le i \le \#(v, \epsilon)$, if $\Pr(E_i) \ge v$ then

$$\Pr(E_{i-1} | E_i) \ge 1 - \epsilon.$$

This implies

$$\Pr(E_{i-1}) \ge (1 - \epsilon) \cdot \Pr(E_i)$$

(since $\Pr(A)/\Pr(B) \ge \Pr(A|B)$ for any two events A and B). We get

$$\Pr(E_{\#(v,\epsilon)-1+i}) \ge (1 - \epsilon)^i.$$

Finally, $\Pr(E_1) \le v$, so that the fairness requirement is not violated by the first step. Thus, $\#(v, \epsilon)$ and the $\Pr(E_i)$ are easily bounded by expressions depending on v and ϵ, as in the following.

Theorem 1: Every viable (v,ϵ)-fair protocol for exchanging privileges has length

$$\#(v, \epsilon) \ge \left\lceil \frac{\log(v)}{\log(1 - \epsilon)} \right\rceil + 2.$$

This is approximately equal to $\epsilon^{-1} \cdot \log(v^{-1})$. Furthermore, the probability that after step i a party is privileged does not exceed $v/(1 - \epsilon)^i$.

IV. The Proposed Scheme

Our protocol makes use of the signature scheme of the network. For our purpose, it will be convenient to decouple the analysis of a protocol for exchanging privileges from the security of the underlying signature scheme used in its implementation. For example, ordinary hand-written signatures are impossible to analyze mathematically but could be used in our protocol with the ordinary postal mail system. This decoupling can be done by assuming that the signatures used in the implementation are totally unforgeable, namely, for each message m and user A, the probability that another user (B) can produce A's signature on m is zero (independent of B's computing power). It should be stressed that *digital* signature schemes cannot possibly be unforgeable in this sense, so problems might arise when implementing our protocol with a "concrete" digital signature scheme. In fact, the "security" of a protocol cannot exceed the "security" of the underlying signature scheme, but it may be *much* less. This is so because the protocol, in

its concrete implementation, may interact badly (in an unpredictable manner) with the signature scheme. However, this is not the case with respect to the protocol presented in this paper. *Our protocol remains fair when implemented with a signature schemes that is (existentially) unforgeable (even under an adaptive chosen message attack)* as the ones in [11], [3]. A formal version of this statement and its proof are omitted.

A. The Protocol for A and B

We now present our protocol for A and B to exchange their commitments to a contract C. In this protocol A and B will exchange signed messages of the form, "With probability p, the contract C shall be valid. (Signed)." The recipient of such a message is privileged with probability p; that is, if s/he goes to the judge with this message, with probability p the judge will find that the contract is binding on both parties. Party A(respectively, B) uses a local variable λ_A (resp. λ_B) denoting the probability mentioned in the most recent such message signed by A (resp. B).

(*Timeout Procedure:* If this protocol does not terminate normally by date D, then either party may invoke the "early stopping procedure" described later.)

- Parties A and B agree who will "go first"; here we assume that A was chosen to go first. We assume that the contract defines a date D by which the signing process is to be completed.
- Party A chooses a probability v which is "negligible" in the sense that A is willing to accept a chance of v that B is privileged while A is not.
- Party A also chooses a value $\alpha > 1$ for which it must be guaranteed that at each step the conditional probability that A is privileged, given that B is privileged, should be at least α^{-1} (unless, of course, the probability that B is privileged is $< v$).
- Similarly, B chooses a value $\beta > 1$ corresponding to the symmetrical condition.
- The protocol is initialized with both λ_A and λ_B set to zero.

Iterations: Parties A and B alternate steps until both have received and sent signed messages of the form "With probability 1, the contract C shall be valid."

A-step: Let p be the probability mentioned in the last signed message received by A ($p = 0$ if no such message was received). Party A checks whether $p \geq \lambda_A$; if not, A terminates the protocol, and considers B to have "stopped the protocol early."[3] Party A sets $\lambda_A \leftarrow \max(v, \min(1, p \cdot \alpha))$, and sends B a signed message saying, "With probability λ_A, the contract C shall be valid. (Signed) A."

B-step: Let p be the probability mentioned in the last signed message received by B. Party B checks whether $p \geq \lambda_B$; if not, B terminates the protocol and considers A to have "stopped the protocol early." Party B sets $\lambda_B \leftarrow \min(1, p \cdot \beta)$, and sends A a signed message saying, "With probability λ_B, the contract C shall be valid. (Signed) B."

Early Stopping Procedure: If the entire protocol is not completed by date D, party X invokes the judge, providing him with the contract C and the most recent message received from the other party Y.

B. The Judge's Procedure

When the judge is invoked regarding a contract C specifying a date D, the judge does nothing until date D has passed. Then, he examines the supplied message of the form, "With probability p the contract C shall be valid. (Signed) Y" and checks the validity of Y's signature. If the signature is invalid the judge does nothing; otherwise, he proceeds. If the judge has never before given a verdict concerning contract C, he chooses a random value ρ_C according to a uniform distribution over the interval $[0,1]$. Otherwise, s/he retrieves the previously computed value ρ_C. Then, if $p \geq \rho_C$, the judge rules that C is binding by sending a signed verdict to both parties. If $p < \rho_C$ the judge notifies both parties that a (signed) message regarding contract C was examined, but that the value of p it contained was too small to find the contract valid.

C. Discussion

1) If both parties are honest and complete the protocol by date D, the judge will never be invoked. In fact, each will have a message bearing probability 1 from the other party, so that this message is itself clear evidence that both parties are committed to this contract.

2) The protocol terminate normally even if one party, say A, never increases the probability λ_A that his counterpart is privileged above his current probability p of being privileged. In fact, increasing λ_A above p is A's prerogative. (Note that α is known only to A.) Progress is guaranteed as long as one party keeps increasing its λ by some constant factor.

3) Problems may arise only if one of the parties prematurely terminates the protocol. In such a case the judge may be invoked by either party and may rule even in the absence of the other party.

4) Since the judge sends his verdict to both parties, whenever the judge rules that the contract is committing, both parties will be equally committed to it (i.e., either both are committed or neither is committed). Thus the advantage one party may have on his counterparts is merely in the ability to initiate such a verdict.

5) The parties do not know the value of ρ_C during the execution of the protocol, since the judge will not provide a judgment until after date D has passed.

6) Let $\epsilon = \max(1 - \alpha^{-1}, 1 - \beta^{-1})$. The manner in which the probabilities in the messages are chosen makes the protocol (v, ϵ)-fair. Furthermore, if $\alpha = \beta$, the sequence of probabilities is identical to the sequence of probabilities

[3] Party A does check whether B has increased the probability above the value λ_A, and in particular that $p = \beta \cdot \lambda_A$. ($A$ does not even know β.) See point 2 in Section IV-C-2.

given in Theorem 1. In this case our protocol is of shortest length among all viable (v,ϵ)-fair protocols. A similar statement could be proved for arbitrary (nonequal) α and β.

7) The fact that the judge, upon receiving some (possibly partial) evidence from one party rules that the contract is not committing does not mean that he will rule so also when supplied with more evidence. However, both parties have almost the same amount of evidence.

From the previous discussion it is not difficult to prove the following.

Theorem 2: For every (v,ϵ), there exists a viable, (v,ϵ)-fair protocol, for contract signing of length

$$\#(v,\epsilon)=\left\lceil \frac{\log(v)}{\log(1-\epsilon)} \right\rceil +2.$$

Furthermore, these protocols can be efficiently constructed.

V. Improving the Efficiency of the Judging Procedure

Observe that the judge only needs to specify ρ_C to enough accuracy to determine whether $p<\rho_C$ or $p\geq\rho_C$. Recall that the judge is required always to use the same value ρ_C for the same contract C. Failing to do so may create conflicting verdicts and could give advantage to a party who keeps appealing to court many times with the same contract. In Section IV implementing the protocol by randomly choosing ρ_C once and for all, storing the pair (C,ρ_C), and using ρ_C in all future verdicts regarding C was suggested. This solution can hardly be considered efficient, as the judge must keep a record of all previous ρ_C. It is our goal to free the judge from any bookkeeping. We can do this by using the "pseudorandom functions" of Goldreich *et al.* [10].

In [10] it is shown how to transform any one-way (in a weak sense [13]) function into a set of efficiently computable functions that are "indistinguishable from random functions." In particular, the transformation defines a mapping from the set of n-bit strings into a set of functions (from n-bit strings to n-bit strings) hereafter called the set of polyrandom functions. The n-bit string r is mapped into the function f_r: $\{0,1\}^n\rightarrow\{0,1\}^n$. The transformation is efficient in the sense that a polynomial-time algorithm exists that, on input of index r and an argument x, outputs $f_r(x)$ (i.e., the value of f_r on the argument x). These functions are indistinguishable from random functions in the following sense: no polynomial-time algorithm exists that, by querying an oracle on inputs of its choice, can distinguish the case where the oracle implements a truly random function from the case where the oracle implements a randomly chosen polyrandom function. (By a random function, we mean a function randomly selected with uniform probability from the set of all functions from n-bit strings to n-bit strings.) Thus a polyrandom function possesses all the statistical properties of truly random functions with respect to an observer with polynomially bounded resources.

Let us now show how to use this result to free the judge from bookkeeping. The judge randomly selects an n-bit string r once and for all and keeps it in secret. No other random choices are ever made by the judge, and he does not need to keep any records other than the secret value of r. When invoked with C and all other inputs, the judge computes $f_r(C)$ and uses it (instead of flipping coins) to determine the value of ρ_C.[4] This way of operating essentially maintains (v,ϵ)-fairness. It can be shown that the conditional probability that "B is privileged" given that "A is privileged" is greater than $1-\epsilon-1/n^c$, for all constants $c>0$ and sufficiently large n. (This follows from the properties of the polyrandom functions, unless no one-way functions exist.) We stress that the (v,ϵ)-fairness of the protocol holds even if the parties are given the ρ's associated with other contracts of their choice.

A. Many Judges

Our solution easily extends to allow many judges to rule consistently without requiring them to coordinate. To this end, the judges only need to share a randomly selected n-bit string r specifying a function f_r of a polyrandom collection. The function is used by each judge in the same manner described earlier.

B. The Contract Need not have Length n

In the foregoing we assumed that all contracts have length n. This assumption is unnecessary: it is sufficient that each contract starts with an n-bit string S uniquely associated with it (and containing also the timeout date D). To ensure uniqueness of S, it should contain (in addition to the timeout D) the names of A and B, a unique substring chosen by A, and a unique substring chosen by B. Instead of applying f_r to the entire text of C, the judge computes $\rho_C=f_r(S)$. As long as one party never uses the same substring for two different contracts, no matter how "tricky" the second party will choose his part of S, the same security is maintained. The messages exchanged can now be made shorter, by replacing "the contract C" with "the contract with identification number S" in them. Before the previously given protocol begins, the parties need to exchange signed messages asserting that the string S is the agreed-upon identification number for the contract C.

C. Why Polyrandom Functions

Polyrandom functions are based on and improve previous results of Blum and Micali [4] and Yao [16] on pseudorandom number generators. Pseudorandom number generators are deterministic algorithms that transform a truly random (but short) secret seed to a long pseudoran-

[4] We assume here that the length of C equals n. This assumption will be relaxed later.

dom bit-sequence passing polynomial time statistical tests. Such pseudorandom number generators may not be straightforwardly applied in our context. For example,

1) using C or part of it as input to the generator does not work (the parties may also do so and determine ρ_C as well);
2) using the bit-by-bit EXCLUSIVE OR of C and some fixed key r (randomly and secretly selected by the judge) as input to the generator may not work. In fact, these generators are proved to produce "random" outputs only on randomly and independently selected inputs; they do not guarantee that the outputs produced by closely related inputs are not related. In our setting, knowledge of the ρ's associated with previous contracts, may help in predicting the ρ_C associated with a new contract C.

VI. Summary and Further Discussion

Let us first sum up the properties of our solution (which consists of a two-party protocol and a judge-procedure)

1) It satisfies both the viability and the fairness requirements, without using the "equal computing power" assumption. It is not prone to "early stopping." Furthermore, unlike other solutions, our solution requires the intervention of a trusted third party and a "time out" mechanism in a very mild sense.

2) A third party (a judge) intervenes only in case of dispute. In this case the judge can rule even if only one party appears in court. Furthermore, no bookkeeping is required from the judge (other than remembering his own secret, of course).

3) Our solution allows many judges to be used without the need to coordinate their actions. Still, all judges will give the same verdict once presented the same case.

4) The protocol can be implemented and proven fair under what seems to be the minimum possible intractability assumption: the existence of secure signature schemes (see [11]). The judge-procedure, which involves the use of random functions, requires a slightly different assumption: the existence of one-way functions (see [10] and [13]).

5) The protocol is easy to execute. Also, the judging-procedure is conceptually simple and only requires the examination of *one* message (the last received secret). (If the messages exchanged only refer to a contract C by its identification number then the judge does not need to see the actual contract C.)

6) The protocol is optimal in the sense that it uses the minimum number of message exchanges needed to satisfy the (v, ϵ)-fairness requirement.

A. Historical Remark

> "The idea [presented in this paper] is not really novel at all: it is but a sophistication of an idea proposed by one of the authors (Rivest) five years ago at a Crypto (81 or 82, I can't recall exactly) rump session. I never saw Rivest's idea in print afterwards and I felt sorry about it because I thought it was rather clever despite Rivest's attitude (then) that it was not very serious (this happens often in Crypto rump talks)."
>
> from a referee report

Indeed the basic idea was suggested by Rivest in Crypto81. However, at the time it seemed to have only disadvantages with respect to the protocols using the computational approach to fairness, since the weaknesses of that approach were not yet understood. We believe that pointing out these weaknesses is one of the contributions of this paper.

Acknowledgment

We wish to thank Shimon Even, Charlie Rackoff, and Adi Shamir for illuminating discussions concerning the issues of this paper. We are also grateful to the referees for useful comments.

References

[1] M. Blum, "How to exchange (secret) keys," *ACM Trans. Comp. Syst.*, vol. 1, no. 2, pp. 175–193, 1983.
[2] ____, "Mail certification by randomization," unpublished manuscript.
[3] M. Bellare and S. Micali, "How to sign given any trapdoor function," in *Proc. 20th ACM Symp. Theory of Computing*, 1988, pp. 32–42.
[4] M. Blum and S. Micali "How to generate cryptographically strong sequences of pseudo-random bits," *SIAM J. Computing*, vol. 13, pp. 850–864, Nov. 1984.
[5] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, no. 6, pp. 644–654, Nov. 1976.
[6] S. Even, "A protocol for signing contracts," Comput. Sci. Dept., Technion, Haifa, Israel, Tech. Rep. 231, 1982 (presented at Crypto81).
[7] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *Commun. ACM*, vol. 28, no. 6, pp. 637–647, 1985.
[8] S. Even and Y. Yacobi, "Relations among public key signature systems," Comput. Sci. Dept., Technion, Haifa, Israel, Tech. Rep. 175, 1980.
[9] O. Goldreich, "A simple protocol for signing contracts," in *Advances in Cryptology: Proceedings of Crypto83*, D. Chaum, Ed., New York: Plenum Press, 1984, pp. 133–136.
[10] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *J. ACM*, vol. 33, no. 4, pp. 792–807, 1986.
[11] S. Goldwasser, S. Micali, and R. L. Rivest, "A digital signature scheme secure against adaptive chosen-message attack," *SIAM J. Computing*, vol. 17, no. 2, pp. 281–308, 1988.
[12] J. Hastad and A. Shamir, "The cryptographic security of truncated linearly related variables," in *Proc. 17th ACM Symp. Theory of Computing*, 1985, pp. 356–362.
[13] L. A. Levin, "One-way functions and pseudorandom generators," in *Proc. 17th ACM Symp Theory of Computing*, 1985, pp. 363–365.
[14] M. O. Rabin, "Transaction protection by beacons," Aiken Computation Lab., Harvard Univ., Cambridge, MA, Tech. Rep. 29-81, 1981.
[15] R. L. Rivest, A. Shamir, and L. Adelman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, pp. 120–126, Feb. 1978.
[16] A. C. Yao, "Theory and application of trapdoor functions," in *Proc. 23rd IEEE Symp. Foundation of Computer Science*, 1982, pp. 80–91.