How to Generate and Exchange Secrets
(extended abstract)

Andrew Chi-Chih Yao*

Department of Computer Science
Princeton University
Princeton, New Jersey 08544

Abstract

In this paper we introduce a new tool for controlling the knowledge transfer process in cryptographic protocol design. It is applied to solve a general class of problems which include most of the two-party cryptographic problems in the literature.

Specifically, we show how two parties A and B can interactively generate a random integer $N = p \cdot q$ such that its *secret*, i.e., the prime factors (p, q), is hidden from either party individually but is recoverable jointly if desired. This can be utilized to give a protocol for two parties with private values i and j to compute any polynomially computable functions $f(i,j)$ and $g(i,j)$ with minimal knowledge transfer and a strong *fairness* property. As a special case, A and B can exchange a pair of secrets s_A, s_B, e.g. the factorization of an integer and a Hamiltonian circuit in a graph, in such a way that s_A becomes computable by B when and only when s_B becomes computable by A. All these results are proved assuming only that the problem of factoring large intergers is computationally intractable.

1. Introduction

A *protocol* $M = (M_A, M_B)$ is a pair of communicating probabilistic Turing machines each with a special "send-receive"-tape. Given inputs of the form (n, i_A) and (n, i_B), the two machines will alternately send and receive message strings using the send-receive tapes; each machine will perform computation as a standard Turing machine after receiving a message string, including the computation of the next message to be sent. Eventually both machines halt within a number of steps bounded by some polynomial in n, leaving strings u_A and u_B on the output tapes. For a detailed description, see for example [GMR] [GHY]; in our case we allow each machine to have their own private input tape. For any run σ of the protocol, let $\Delta_A(\sigma)$ denote the *history* of the run from A's view, i.e. a sequence of the instataneous descriptions of M_A; similarly $\Delta_B(\sigma)$ denote the history of this run from B's view.

* This research was supported in part by National Science Foundation under grants DCR-83-08109.

The class of problems we are interested in as follows. Given inputs (n, i_A) and (n, i_B), where (i_A, i_B) is a random pair of strings distributed according to a probability distribution h_n over $\{0,1\}^* \times \{0,1\}^*$, we wish to design a protocal $M = (M_A, M_B)$ such that the outputs (u_A, u_B) is distributed according to a certain probability distribution w_{n,i_A,i_B} over $\{0,1\}^* \times \{0,1\}^*$. The sequence of distributions $(h_1, h_2, \ldots, h_n, \ldots)$ is assumed to be a *polynomial ensemble* in the sense that, there is a probabilistic Turing machine which, given input n, will generate in time polynomial in n a random string x with a distribution indistinguishable computationally from h_n; similarly we assume that w_{n,i_A,i_B} is a polynomial ensemble in that a random sample point can be generated in time polynomial in n, when the parameters n, i_A, i_B have been given. Let us call this a *(two-party) interactive computational problem* $\langle h_n, w_n \rangle$. Of special interest is the case when the probability distribution w_{n,i_A,i_B} is nonzero only at one point, call it $(f_n(i_A, i_B), g_n(i_A, i_B))$, in which case we can regard the problem as the evaluation of a pair of functions $f_n(i_A, i_B), g_n(i_A, i_B)$; write it as $\langle h_n, (f_n, g_n) \rangle$.

In addition to the *validity* requirement above, we would like the protocol to have certain *privacy* and *fairness* properties. Roughly, "privacy" means that if A behaves according to the protocol, then B will have no more information about the values of i_A, u_A than in the situation where an oracle does the computation for B and just hands B a value u_B. Classical examples where privacy is the main concern include oblivious transfer (Rabin [R], Fischer et al [FRMW]), coin tossing (Blum [B1], Cleve [C]), mental poker (Shamir, Rivest, and Adleman [SRA], Goldwasser and Micali [GM]). A general study of this problem is given in Yao [Y1,Y2]. A related problem is the *interactive proof system* (Goldwasser, Micali, and Rackoff [GMR], Galil, Haber, and Yung [GHY]) in which player A wishes to convince player B that a string τ is in a certain language L. Recently, Goldreich, Micali, and Wigderson [GMW] proved that, for players with polynomial-time computing power, any language in NP has a *minimum knowledge interactive proof*, assuming the existence of suitable one-way functions. This latter problem is closely related to, in our formulation, the case of computing a pair of functions $f_n(i_A, i_B), g_n(i_A, i_B)$, where $i_A = (s, \tau)$, $i_B = \tau$, $g_n(i_A, i_B) = 1$ if and only if s is a short *proof* for $\tau \in L$, and $f_n(i_A, i_B) \equiv 1$.

The "fairness" requirement means that a cheater should not be able to obtain the desired output while denying the other party to find the proper output. The problem of exchanging secrets (Blum[B2], Luby, Micali, and Rackoff [LMR],Vazirani and Vazirani [VV]) has this requirement as its main concern.

In this paper, we will give a protocol for performing the computation $\langle h_n, w_n \rangle$ which achieves validity, privacy and fairness as stated in Theorem 3, under the assumption that factoring large integers is computationally intractable. Many of the existing results mentioned are special cases of this theorem. A particular interesting special case (stated as Theorem 2) is that it allows two parties to exchange *secrets*, such as the factorization of a publicized integer or the Hamiltonian circuit of a publicized large graph, in a way that the probability of successful cheating can be made arbitrarily small. It is somewhat surprising that the two secrets being exchanged could be of very different apparent complexity, and one would have thought that it is difficult to find an equitable ratio of bits to be maintained during the process of swapping the secrets. Previously Blum [B2] gave a protocol for exchanging prime factors based on several assumptions (see Hastad and Shamir [HS] for discussions); Luby, Micali, and Rackoff [LMR], Vazirani and Vazirani [VV] gave protocols for exchanging secrets which are one bit long.

The proof of Theorem 3 depends on the ability of two parties to generate a random integer $N = p \cdot q$, with its *secret* (p, q) hidden from each party, but recoverable at a later time by a joint effort. This result is of independent interest, and should serve as another useful tool for cryptographic protocol design. A general form of this result is given as Theorem 1.

Compared with [Y1,Y2], which deals with the same general problem, the addition of the fairness property is the new motive which leads to the present work. We should mention that all the protocols considered in this paper for solving the interactive copmputational problems $\langle h_n, w_n \rangle$ will be independent of h_n, as is traditional in considering cryptographic protocols. The main results (Theorems 1,2, and 3) are all proved under the following assumption. Let W_n be the set of all integers N of the form $p \cdot q$ where $p \equiv q \equiv 3 \bmod 4$ are prime numbers.

The Intractability Assumption of Factoring(IAF): Let $k > 0$ be any fixed number. For any polynomial time probabilistic algorithm for factoring integers, the probability of success for a random input integers from W_n is less than $1/n^k$ for all large n.

2. Terminology

Let $\mathcal{S} = (S_1, S_2, \ldots)$ and $\mathcal{S}' = (S_1', S_2', \ldots)$ be polynomial ensembles, where $S_n = (X_n, Y_n)$ and $S_n' = (X_n', Y_n')$ each is a probability distribution over $\{0,1\}^* \times \{0,1\}^*$. Furthermore, assume that the two ensembles $(X_1, X_2, \ldots)$ and $(X_1', X_2', \ldots)$ are indistinguishable for polynomial-time computations.

Let $\mathcal{D} = (d_1, d_2, \ldots)$ be a sequence of predicates d_n:$\{0,1\}^* \to \{0,1\}$, where given d and n, $d_n(x)$ can be computed probabilistically in time polynomial in n. Define a *guessing algorithm* Q_B to be a probabilistic algorithm which takes (n, y) as input, where $y \in \{0,1\}^*$, and outputs a 0 or 1 in time polynomial in n.

Consider the experiment of taking a random (x_n, y_n) distributed according to (X_n, Y_n). Suppose that one observes the value y_n and tries to guess what the value of $d_n(x_n)$ is by using a guessing algorithm Q; let us denote by $r(d_n, X_n, Y_n, Q)$ the probability that a correct guess will be made. We use the notation o(poly-small) to denote any sequence $(b_1, b_2, \ldots)$ that has the property $b_n = o(1/n^k)$ for all fixed k.

Definition We write $I_n(X_n \mid Y_n) \preceq I_n(X_n' \mid Y_n')$ if $\forall\, \mathcal{D}$ and $Q\ \exists$ Q' such that $r(d_n, X_n', Y_n', Q') - r(d_n, X_n, Y_n, Q) \geq$o(poly-small).

Definition We write $I_n(X_n \mid Y_n) \approx I_n(X_n' \mid Y_n')$ if $I_n(X_n \mid Y_n) \preceq I_n(X_n' \mid Y_n')$ and $I_n(X_n' \mid Y_n') \preceq I_n(X_n \mid Y_n)$.

A *puzzle ensemble* $\mathcal{P} = (L, \mathcal{F})$ consists of a language $L \in$BPP and a polynomial-time ensemble $\mathcal{F} = (F_1, F_2, \ldots)$ where each F_n is a distribution over $\{0,1\}^* \times \{0,1\}^*$; we require further that a random (s, τ) distributed according to F_n will satisfy $(n, s, \tau) \in L$ with probability 1-o(poly-small); we will call s a *secret* of the *text* τ. Let $T_{\mathcal{P},n}$ denote the distribution of a random τ taken from the second component of a random (s, τ) distributed as F_n. We will call $\mathcal{P}$ *intractable* if, for every probabilistic polynomial-time algorithm S, when given as input a pair (s, τ) where τ is distributed according to $T_{\mathcal{P},n}$, will fail with probabilty 1-o(poly-small) to produce an s satisfying $(n, s, \tau) \in L$.

For example, let $L = \{(n; p, q, N) \mid N = p \cdot q,\ p, q$ are n-bit primes $\}$, and F_n be the uniform distribution over the set W_n. Under the Intractability of Factorization Assumption, the *factorization puzzle ensemble* $\mathcal{P} = (L, \mathcal{F})$ is an intractable puzzle ensemble.

3. Generating a Secret.

Let $\mathcal{P} = (L, \mathcal{F})$, where $\mathcal{F} = (F_1, F_2, \ldots, F_n, \ldots)$, be an intractable puzzle ensemble. We wish to design a protocol $\mathcal{M} = (M_A, M_B)$ which has the following properties for any given n: 1) $\mathcal{M}$ generates implicitly a pair (s, τ) distributed according to F_n, 2) the text τ will be found out by M_A and M_B as their outputs, and 3) the secret s, while computable by A and B jointly based on the information they have at the end of the execution of the protocol, is completely hidden from each party by itself, even if one of them cheats during the execution of the protocol.

To simplify the presentation of the result, we restrict ourselves to puzzles that have unique secrets. Let us call a puzzle ensemble $\mathcal{P} = (L, \mathcal{F})$ *uniquely decipherable* if, for every n, τ, there is at most one s satisfying $(n, s, \tau) \in L$. For example, the factorization puzzle ensemble is uniquely decipherable. For a uniquely decipherable puzzle ensemble, we can write it as $\mathcal{P} = (\alpha, \mathcal{D})$, where $\alpha = (a_1, a_2, \ldots)$ and $\mathcal{D} = (D_1, D_2, \ldots)$ are given by $a_n(\tau) = s$ and $D_n = T_{\mathcal{P},n}$.

Let $\mathcal{P} = (\alpha, \mathcal{D})$ be a uniquely decipherable intractable puzzle ensemble, where $\alpha = (a_1, a_2, \ldots)$ and $\mathcal{D} = (D_1, D_2, \ldots)$. Formally, we define our requirements on a protocol $\mathcal{M} = (M_A, M_B)$ for generating a secret for $\mathcal{P}$ as follows. (A or B will sometimes output u_A or u_B =CHEATING; informally we say that A or B has detected that the other party is cheating.)

Validity.

If both A and B follow the protocol, then
(i) With probability $1 - o(\text{poly-small})$, $u_A = u_B$, and their common value τ is distributed according to a distribution indistinguishable (by polynomial-time computations) from D_n;

(ii) $I_n^{(J)}(a_n(\tau) \mid \tau, \Delta_j) \approx I_n^{(\mathcal{L})}(a_n(\tau) \mid \tau)$ for $j \in \{A, B\}$, where J is the stochastic process induced by the execution of the protocol $\mathcal{M} = (M_A, M_B)$, and $\mathcal{L}$ is the stochastic process of fetching τ according to D_n.

(iii) $\exists$ a protocol $\mathcal{N} = (N_A, N_B)$ which, given input (Δ_A, Δ_B), computes outputs v_A, v_B, with the property that $v_A = v_B = a_n(\tau)$ with probability 1-o(poly-small).

To discuss the validity concept when one party, say B, may misbehave. Let d_n be the probability for runs in which $v_A \neq$CHEATING, and let D'_n be the probability distribution for τ when restricted to such runs. If d_n is negligible, then A will almost always catch B cheating, and no further requirement is needed. On the other hand, if $d_n = \Omega(1/n^t)$ for some fixed $t > 0$, then we require the following two constraints to be true:

(iv) $\mathcal{P}' = (\alpha, \mathcal{D}')$ is a uniquely decipherable intractable puzzle ensemble, where $\mathcal{D}' = (D'_1, D'_2, \ldots)$.

(v) $I_n^{(J)}(a_n(\tau) \mid \tau, \Delta_B) \approx I_n^{(\mathcal{L})}(a_n(\tau) \mid \tau)$, where J is the stochastic process induced by the execution of the protocol $\mathcal{M} = (M_A, M'_B)$, and $\mathcal{L}$ is the stochastic process of fetching τ according to D'_n.

To develop the concept of fairness, consider the execution of protocol $\mathcal{M}$ and then $\mathcal{N}$. If A follows the protocols, we require that the probability for B to obtain s while A cannot recover s to be small.

Fairness.

(i) [B may cheat.] Suppose protocols $\mathcal{M} = (M_A, M_B)$ and $\mathcal{N} = (N_A, N_B)$ are run with machine pairs $(M_A, M'_B), (N_A, N'_B)$. There exists a polynomial-time probabilistic algorithm Y (dependent on M'_B and N'_B), which takes a history pair for $\mathcal{M}$ and $\mathcal{N}$ as input, and outputs a string w. We require that if A follows the protocols $\mathcal{M}$ and $\mathcal{N}$ and then runs Y, the probability that $u_A = \tau$, $v_B = a_n(\tau)$ while $w \neq a_n(\tau)$ is o(poly-small).

(ii) [A may cheat.] (Interchange the roles of A and B in (i).)

Theorem 1. Let $\mathcal{P}$ be an intractable puzzle ensemble that is uniquely decipherable. There exists a protocol $\mathcal{M} = (M_A, M_B)$ for generating a secret from $\mathcal{P}$ that achieves validity and fairness.

4. Exchange of Secrets.

Let $\mathcal{P}_A = (L_A, \mathcal{F}_A)$ and $\mathcal{P}_B \doteq (L_B, \mathcal{F}_B)$ be two intractable puzzle ensembles. Let (s_A, τ_A), (s_B, τ_B) be random puzzles distributed according to $F_{A,n}$ and $F_{B,n}$; give (n, s_A, τ_A, τ_B) as input to A and (n, s_B, τ_A, τ_B) to B. We wish to design a protocol $\mathcal{M} = (M_A, M_B)$ that will enable A and B to exchange their secrets s_A and s_B so that neither will be swindled. We state the criteria for the protocol below.

Validity.

If both parties follow the protocol, then with probability $1 - o(\text{poly-small})$, $u_B = s_A$ and $u_A = s_B$.

Fairness.

(i) [If B gets s_A, then A can compute s_B.]
If A follows the protocol and B does not (i.e., $M'_B \neq M_B$), then for any fixed k, there exists a probabilistic polynomial-time algorithm S which takes (n, Δ_A) as input and outputs a number v such that the following is true:

$$\Pr\{(u_B = s_A) \wedge (v \neq s_B)\} = O(\frac{1}{n^k}).$$

(ii) [If A gets s_B, then B can compute s_A.]

(Interchange A and B in (i).)

Theorem 2. Let $\mathcal{P}_A$ and $\mathcal{P}_B$ be any two intractable ensembles. There exists a protocol $\mathcal{M} = (M_A, M_B)$ for exchanging secrets between $\mathcal{P}_A$ and $\mathcal{P}_B$ that achieves validity and fairness.

5. General Computation.

Consider an interactive computational problem $\langle h_n, w_n \rangle$ as defined in Section 1. We will show that there exists a protocol for it that satisfies some strong privacy and fairness constraints. In this extended abstract, we restrict ourselves to the case when w_n represents a pair of functions (f_n, g_n) to be evaluated, i.e., given inputs i_A, i_B, A and B wish to compute $f_n(i_A, i_B)$ and $g_n(i_A, i_B)$. The results can be extended to the general case.

We will consider two variants, which differ in their input-output format. Model I is the natural one, in which the inputs and outputs are as specified in the previous paragraph. However, since in general A does not have any control over the value of i_B, a dishonest B can pretend that i_B is an arbitrary value y. As a result, the fairness constraint can at most force B to compute the value of $f_n(i_A, y)$ for *some* y. In Model II, A will be given as input $(i_A, p_A, q_A, N_A, N_B, E_{N_B}(i_B))$ where $N_A = p_A \cdot q_A$ is the product of two large primes, and N_B is another integer of two large prime factors used to encode B's parameter i_B as $E_{N_B}(i_B)$; B has as input $(i_B, p_B, q_B, N_B, N_A, E_{N_A}(i_A))$. The integers N_A and N_B are generated from some distributions such that factoring these numbers are computationally intractable; E_N can be any probabilistic encryption scheme that is provably secure under the Intractability Assumption of Factoring (e.g. the ones in Alexi, Chor, Goldreich and Schnorr [ACGS] or Blum and Goldwasser [BS]). In the output, A, B obtain (u_A, v_A, w_A) and (u_A, v_A, w_A). When both parties behave honestly, $u_A = f_n(i_A, i_B)$, $v_A = (p'_A, q'_A, N'_A)$, and $w_A = (N'_B, E_{N'_B}(g_n(i_A, i_B)))$, where N'_A and N'_B are integers which are the products of two large prime factors; similarly for the output of B. Such formats arise quite naturally in concatenating protocols. The privacy and fairness constraints in this model are stronger than the first model.

Model I.

We first define the constraints when both parties follow the protocol.

Validity.
With probabilty 1-o(poly-small), $u_A = f_n(i_A, i_B)$ and $u_B = g_n(i_A, i_B)$.

Privacy.

(i) $I_n^{(J)}(i_A, i_B, u_A \mid \Delta_B, i_B, u_B) \approx I_n^{(\mathcal{L})}(i_A, i_B, u_A \mid i_B, u_B)$where J is the stochastic process of running M with input from h, and $\mathcal{L}$ is the stochastic process of having (i_A, i_B) distributed according to h and $u_A = f_n(i_A, i_B)$, $u_B = g_n(i_A, i_B)$.
(ii) (Interchange A and B in (i).)

We now define the constraints when A follows the protocol while B may cheat, that is, B executes the protocol with any M'_B. Since M'_B is arbitrary, B can always generates an i'_B and acts as if this is the input value i_B. As A can never detect this method of cheating, A has to cooperate and let B know the value of $g_n(i_A, i'_B)$. This enables B to control A's output u_A, and to probe somewhat into the value of i_A; this cannot be prevented if we want to preserve the validity condition for honest parties described above. We shall require that B cannot do more than that. Let us denote by Z the set $\{0,1\}^* \cup \{$ CHEATING $\}$, and extend the function f_n by $f_n(i_A, y)$ =CHEATING if $y =$ CHEATING.

Let M'_B be any communicating Turing machine, and let U_n be the distribution corresponding to u_A when (M_A, M'_B) are used to carry out the protocol.

Validity.

There exists a probabilistic polynomial-time algorithm S that takes input (n, i_B) and produces a random $y \in Z$ such that U_n is indistinguishable from the distribution corresponding to $f_n(i_A, y)$.

Let J be the stochastic process of running (M_A, M'_B) with input (i_A, i_B) distributed according to h_n. Let $\mathcal{S}$ be the set of all probabilistic polynomial-time algorithms S that take input (n, i_B) and produce random $y \in Z$. For any $S \in \mathcal{S}$, let $\mathcal{L}(S)$ be the stochastic process of generating (i_A, i_B) according to h_n, run S on input (n, i_B) to produce a random y, and define $u_A = f_n(i_A, y)$, $u_B = g_n(i_A, y)$. Intuitively, B may use S to generate a random y, and after that, acts as if y were the value of i_B, and otherwise follows the protocol; $\mathcal{L}(S)$ is clearly a less informative process for B in which A simply tells B the value of $u_B = g_n(i_A, y)$ without other communications taking place.

Privacy.
For any M'_B, there exists an $S \in \mathcal{S}$ such that $I_n^{(J)}(i_A, i_B, u_A \mid \Delta_B, i_B, u_B) \preceq I_n^{(\mathcal{L}(S))}(i_A, i_B, u_A \mid i_B, u_B)$.

The fairness concept refers to obtaining the information that one is entitled to. Suppose B wishes to know the value of $g_n(i_A, y)$ for some y other than i_B. As we mentioned earlier, B can succeed by pretending $i_B = y$ and otherwise follow the protocol; in the process A will obtain also the value of $f_n(i_A, y)$. The privacy constraint stipulates that no information other than the value of $g_n(i_A, y)$ is conveyed from A to B. The fairness constraint is concerned with whether it is possible for B to stop at some point, once B has the information about $g_n(i_A, y)$, and deny the knowledge of $f_n(i_A, y)$ to A. The following formulation is not the strongest possible version of this constraint, but it is sufficient for some applications such as exchanging secrets of sufficient length. Let L_n denote the set of all possible values of $f_n(i_A, i_B)$ when (i_A, i_B) is distributed according to h_n.

Definition. A *recovery algorithm* R for A is a probabilistic polynomial-time algorithm which takes (n, Δ_A) as input, and outputs a string v.

Consider a pair (i_A, i_B) distributed according to h_n. Let G denote a probabilistic polynomial-time Turing machine that takes i_B as input and outputs z; let $\beta_n(G)$ denote the probability that $z = g_n(i_A, i_B)$.

Fairness.
For any fixed k, there exists a recovery algorithm R (dependent on M'_B) whose output v satisfies the following condition:

$$\Pr\{(u_B = g_n(i_A, i_B)) \wedge (v \notin L_n)\} \leq \beta_n(G) + O(\frac{1}{n^k})$$

for some G.

To complete the definitions, we interchange the roles of A and B in the above discussions, which gives the constraints of validity, privacy, and fairness when B follows the protocol and A is cheating.

Definition. A protocol is said to achieve validity, privacy and fairness, if all the above constraints are satisfied.

Theorem 3. For any interactive computational problem $\langle h_n, (f_n, g_n)\rangle$, there exists a protocol M that achieves validity, privacy and fairness.

In Model II, the constraints are simpler. The input-output formats are as explained at the beginning of this section.

Model II.

When both A and B follow the protocol, the following is required.

Validity.
With probabilty 1-o(poly-small), $u_A = f_n(i_A, i_B)$, $v_A = (p'_A, q'_A, N'_A)$ and $w_A = E_{N'_B}(g_n(i_A, i_B))$, where N'_A is the product of n-bit primes p'_A and q'_A; the distribution of N'_A is such that it is intractable to factor. A dual constraint on B is also required.

Privacy.

(i) $I_n^{(J)}(i_A, i_B, u_A \mid \Delta_B, i_B, u_B, v_B, w_B) \approx I_n^{(\mathcal{L})}(i_A, i_B, u_A \mid i_B, u_B)$where J is the stochastic process of running M with input from h, and $\mathcal{L}$ is the stochastic process of having (i_A, i_B) distributed according to h and $u_A = f_n(i_A, i_B)$, $u_B = g_n(i_A, i_B)$.
(ii) (Interchange A and B in (i).)

If A follows the protocol, but B may cheat and executes some M'_B. Let d_n be the probability for runs in which $v_A \neq$CHEATING, and let U_{n,i_B} be the probability distribution for u_B when restricted to such runs and with i_B being an input for B. If d_n is negligible, then A will almost always catch B cheating, and no further requirement is needed. On the other hand, if $d_n = \Omega(1/n^t)$ for some fixed $t > 0$, then we require the following constraints of Validity and Privacy.

Validity.

With probability d_n−o(poly-small), $u_A = f_n(i_A, i_B)$, $v_A = (p'_A, q'_A, N'_A)$ and $w'_A = E_{N'_B}(g_n(i_A, i_B))$, where N'_A is the product of n-bit primes p'_A and q'_A; the distribution of N'_A is such that it is intractable to factor.

Privacy.

$I_n^{(J)}(i_A, i_B, u_A \mid \Delta_B, i_B, u_B, v_B, w_B) \approx I_n^{(\mathcal{L})}(i_A, i_B, u_A \mid i_B, u_B)$, where J is the stochastic process induced by the execution of the protocol $M = (M_A, M'_B)$, and $\mathcal{L}$ is the stochastic process of being given the value of u_B distributed according to U_{n,i_B}.

As in Model I, let G denote a probabilistic polynomial-time Turing machine that takes i_B as input and outputs z; let $\beta_n(G)$ denote the probability that $z = g_n(i_A, i_B)$.

Fairness.

For any fixed k, there exists a recovery algorithm R (dependent on M'_B) whose output s satisfies the following condition:

$$\Pr\{(u_B = g_n(i_A, i_B)) \wedge (s \neq f_n(i_A, i_B))\} \leq \beta_n(G) + O(\frac{1}{n^k})$$

for some G.

When B follows the protocol and A may cheat, we have a set of requirements obtained from the above ones by switching the roles of A and B.

Theorem 3 is also true for Model II.

6. Connections.

We have chosen a semantic definition of privacy, namely, the communications will not enable one to compute more accurately any polynomial-time predicates. In recent literature, an elegant concept of *minimum-knowledge transfer* protocol was introduced in [GMR] (and generalized in [GHV]) to capture the notion of no unintended information disclosure. It is easy to define privacy constraints for our problems in terms of this concept. We can show that the protocols used to prove the theorems also satisfy the minimum knowledge transfer requirements.

Theorem 4. For any interactive computational problem $\langle h_n, (f_n, g_n)\rangle$, there exists a minimum knowledge transfer protocol M that achieves validity and fairness.

The proofs of the theorems are lengthy, and will be given in the complete paper.

References.

[ACGS] W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr, "RSA/Rabin bits are 1/2+1/2poly(log n) secure," *Proceedings of 25th Annual IEEE Symposium on Foundations of Computer Science*, 1984, 449-457.

[B1] M. Blum, "Coin flipping by phone," *COMPCON* (1982), 133-137.

[B2] M. Blum, "How to exchange (secret) keys," *ACM Transactions on Computer Systems* 1(1983), 175-193.

[BS] M. Blum and S. Goldwasser, "An efficient probabilistic PKCS as secure as factoring," *Proceedings of Crypto 84*, 1984.

[C] R. Cleve, "Limits on the security of coin flips when half of the processors are faulty," *Proceedings of 18th Annual ACM Symposium on Theory of Computing*, 1986, 364-369.

[FMRW] M. Fischer, S. Micali, C. Rackoff, and D. Wittenberg, "An oblivious transfer protocol," 1985, to appear.

[GHY] Z. Galil, S. Haber, and M. Yung, "A private interactive test of a Boolean predicate and minimum-knowledge public-key cryptosystems," *Proceedings of 26th Annual IEEE Symposium on Foundations of Computer Science*, 1985, 360-371.

[GM] S. Goldwasser and S. Micali, "Probabilistic encryption and how to play mental plker keeping secret all partial information," *Proceedings of 14th Annual ACM Symposium on Theory of Computing*, 1982, 365-377.

[GMR] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *Proceedings of 17th Annual ACM Symposium on Theory of Computing*, 1985, 291-304.

[GMW] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity and a methodology of cryptographic protocol design," *Proceedings of 27th Annual IEEE Symposium on Foundations of Computer Science*, 1986.

[HS] J. Hastad and A. Shamir, "The cryptographic security of truncated linearly related variables," *Proceedings of 17th Annual ACM Symposium on Theory of Computing*, 1985, 356-362.

[LMR] M. Luby, S. Micali, and C. Rackoff, "How to simultaneously exchange a secret bit by flipping a symmetrically-based coin," *Proceedings of 24th Annual IEEE Symposium on Foundations of Computer Science*, 1985, 11-22.

[R] M. Rabin, "How to exchange secrets," 1981, unpublished manuscript.

[SRA] A. Shamir, R. Rivest, and L. Adleman, "Mental Poker," MIT Technical Report, 1978.

[T] T. Tedrick, "How to exchange half a bit," *Crypto '83*.

[VV] U. Vazirani and V. Vazirani, "Trapdoor pseudo-random number generators, with applications to protocol design," *Proceedings of 24th Annual IEEE Symposium on Foundations of Computer Science*, 1985, 23-30.

[Y1] A. Yao, "Protocols for secure computations," (extended abstract) *Proceedings of 21st Annual IEEE Symposium on Foundations of Computer Science*, 1982.

[Y2] A. Yao, "Protocols for secure computations," in preparation.

[Y3] A. Yao, "Theory and applications of trapdoor functions," *Proceedings of 21st Annual IEEE Symposium on Foundations of Computer Science*, 1982.