Protocols for Secure Computations
(extended abstract)

Andrew C. Yao
University of California Berkeley, California 94720

1 Introduction

Two millionaires wish to know who is richer; however, they do not want to find out inadvertently any additional information about each other's wealth. How can they carry out such a conversation?

This is a special case of the following general problem. Suppose m people wish to compute the value of a function $f(x_1, x_2, x_3, \ldots, x_m)$, which is an integer-valued function of m integer variables x_i of bounded range. Assume initially person P_i knows the value of x_i and no other x's. Is it possible for them to compute the value of f, by communicating among themselves, without unduly giving away any information about the values of their own variables? The millionaires' problem corresponds to the case when $m = 2$ and $f(x_1, x_2) = 1$ if $x_1 < x_2$, and 0 otherwise. In this paper, we will give precise formulation of this general problem and describe three ways of solving it by use of one-way functions (i.e., functions which are easy to evaluate but hard to invert). These results have applications to secret voting, private querying of database, oblivious negotiation, playing mental poker, etc. We will also discuss the complexity question "How many bits need to be exchanged for the computation", and describe methods to prevent participants from cheating. Finally, we study the question "What cannot be accomplished with one-way functions".

Before describing these results, we would like to put this work in perspective by first considering a unified view of secure computation in the next section.

2 A Unified View of Secure Computation

Since one-way functions were first proposed in 1976 (Diffie and Hellman [1]), they have been used in two kinds of applications. The first kind is concerned with the encryption and transmission of messages to make them unreadable and unalterable for eavesdroppers and saboteurs [1, 2, 3, 4]. The second kind of applications includes "mental poker" (Shamir, et.al. [5]), in which two players deal cards by communicating over a telephone line, and "coin flipping" (Blum [6]), in which two mutually suspecting parties are to generate an unbiased bit. It would be desirable to have a unified framework where all these applications can be related, and where common proof techniques can be developed for proving the security of protocols. More fundamentally, such a framework is essential if we are ever to understand the intrinsic power and limitation of one-way functions. For example, without a precise model it would be hard to answer a question such as "Is it possible for three mutually suspecting parties to interactively generate a bit with bias $1/e$?"

In response to this need, we propose to adopt the following view. Two parties Alice and Bob, in possession of private variables i and j respectively, wish to communicate so that Alice can evaluate a function $f(i, j)$, and Bob a function $g(i, j)$. There may be some eavesdroppers or saboteurs on the communication line. The purpose of a protocol would be to design an algorithm for Alice and Bob to follow, such that certain *security constraints* (against saboteur) and *privacy constraints* (Alice may not wish to reveal the exact value of i) can be satisfied.

In one extreme, when the computation component is trivial, e.g. if $f =$ constant and $g(i, j) = i$, then we get the first kind of applications mentioned before, in which the basic concern is eavesdropping and sabotage. In the other extreme, when such external threats can be ignored, but the computation of f and g is nontrivial, then we get the problem which is to be studied in this paper. (Mental poker and coin flipping represent a stochastic version of this problem which will also be discussed.) Note that, although we have used Alice and Bob in the above description, all discussions can be extended to the case of m parties communicating.

It would be natural to discuss these two special cases together. However, due to length considertion, we will report here only the results corresponding to the computation-intense case with no outside saboteurs. Results on the other case will be reported elsewhere.

3 Deterministic Computations

3.1 Solutions to the Millionaires' Problem

In this abstract, we will describe in detail only one of the three solutions we have.

For definiteness, suppose Alice has i millions and Bob has j millions, where $1 < i, j < 10$. We need a protocol for them to decide whether $i < j$, such that this is also the only thing they know in the end (aside from their own values). Let M be the set of all N-bit nonnegative integers, and Q_N be the set of all 1-1 onto functions from M to M. Let E_a be the public key of Alice, generated by choosing a random element from Q_N.

The protocol proceeds as follows:

0272-5428/82/0000/0160$00.75

1. Bob picks a random N-bit integer, and computes privately the value of $E_a(x)$; call the result k.
2. Bob sends Alice the number $k - j + 1$;
3. Alice computes privately the values of $y_u = D_a(k - j + u)$ for $u = 1, 2, \ldots, 10$.
4. Alice generates a random prime p of $N/2$ bits, and computes the values $z_u = y_u \pmod p$ for all u; if all z_u differ by at least 2 in the $\bmod p$ sense, stop; otherwise generates another random prime and repeat the process until all z_u differ by at least 2; let p, z_u denote this final set of numbers;
5. Alice sends the prime p and the following 10 numbers to B: $z_1, z_2, \ldots, z_i$ followed by $z_i + 1, z_{i+1} + 1, \ldots, z_{10} + 1$; the above numbers should be interpreted in the $\bmod p$ sense.
6. Bob looks at the j-th number (not counting p) sent from Alice, and decides that $i \geq j$ if it is equal to $x \bmod p$, and $i < j$ otherwise.
7. Bob tells Alice what the conclusion is.

This protocol clearly enables Alice and Bob to decide correctly who is the richer person. To show that it meets the requirement that they cannot get any more information about the wealth of the other party, we need to define a precise model which will be done in Section 3.2. Here we will informally argue why the requirement is met.

Firstly Alice will not know anything about Bob's wealth j, except for the constraint on j implied by the final result that Bob told her, because the only other information coming from Bob is that Bob knows the vaue of $D_a(s)$ for some s between $k - j + 1$ to $k - j + 10$. As the function E_a is random all the 10 possibilities are equally likely.

What does Bob know? He knows y_j (which is x) and hence z_j. However, he has no information about the values of other z_u, and by looking at the numbers Alice sent him, he cannot tell if they are z_u or $z_u + 1$.

This has not finished the argument yet, as Alice or Bob might try to figure out the other person's value by making more calculations. For example, Bob might try to randomly choose a number t and check if $E_a(t) = k - j + 9$; if he succeeds, he then knows the value y_9 to be t, and knows the value of z_9, which enables him to find out whether $i \geq 9$. That would be an extra piece of information that Bob is not supposed to find out, if $i \geq j$ has been the outcome of the previous conclusion. Thus, one also has to include in the formal definition that not only the participants do not gain information as a result of the exchange specified by the protocol, but also they cannot perform calculation within a reasonable amount of time to gain this information. In the formal definition to be given in Section 3.2, we will define this precisely.

One may have noticed the possibility that some party may cheat in the process, by deviating from the agreed protocol. For example, Bob may lie to Alice in the final step and tell Alice the wrong conclusion. Is there a way of designing a protocol such that the chance of a successful cheating becomes vanishingly small, without revealing the values of i and j? We will show that this is possible in Section 3.3. (Note that this is a stronger requirement than the verifiability requirement used in the mental poker protocol in Shamir et. al. [5].)

We have two other solutions to the millionaires' problem based on different principles. The first of them assumes that Alice and Bob each owns a private one-way function, where these functions satisfy the commutativity property, i.e. $E_a E_b(x) = E_b E_a(x)$. The other solution makes use of a probabilistic encryption method invented by Goldwasser and Micali [2].

3.2 Model for the General Problem

As these three solutions base their security on different assumptions, a precise model has to be specified in detail for each solution. In this abstract, we will only give the model that corresponds to the first solution.

For simplicity, we will only give the definitions and results for the case when f is $0-1$ valued and $m = 2$ (Alice and Bob). Generalization of the results to general m will be briefly discussed in Section 5. The proofs for the general case involve additional technical complications, and there are extra security considerations such as possible "collusions" that are absent in the 2-person case.

Protocol. Assume Alice has a public one-way function E_a, whose inverse function D_a is known only to Alice; similarly Bob has a public E_b, and a private inverse D_b. Assume that E_a and E_b are independently and randomly drawn from Q_N, the set of all the possible 1-1 onto functions on N-bit integers. A protocol A for computing a function $f(i, j)$ specifies exactly how Alice and Bob should communicate as follows. Alice and Bob send strings to each other alternately. Each time after Bob has finished transmission, Alice examines the information so far in her possession, which consists of some sequence of strings $\alpha_1, \alpha_2, \ldots, \alpha_t$, and some relations among the strings (e.g. $E_b(\alpha_3) = \alpha_9$, α_8 has an odd number of 1's); based on the bits that have so far been transmitted between her and Bob, the protocol specifies how she should compute in private strings $\alpha_{t+1}, \alpha_{t+2}, \ldots, \alpha_s$ where each new string α_u is a function of the earlier strings, or of the form $E_a(y)$, $E_b(y)$ or $D_a(y)$ where y is a string already obtained. The choice of which function to apply or whether to evaluate E_b or D_a is in general probabilistic, i.e. she will decide to evaluate $E(4)$, or to compute $\alpha_2 + 3\alpha_8$ based on the outcomes of some coin tosses. After she has finished this computation, she will send a string to Bob, again the string is chosen probabilistically. Now it is Bob's turn to compute strings and send a string according to the protocol. We agree that there is a special symbol whose appearance means the end of the execution of the protocol. By that time, the

protocol has an instruction for each participant to compute the function value f in private. Finally, we require that, in a protocol, the total number of evaluations of E's and D's by Bob and Alice be bounded by $O(N^k)$, where k is an integer chosen in advance.

Privacy Constraint. Let $\epsilon, \delta > 0$, and $f(i,j)$ be a 0-1 valued function. Assume that initially all pairs of (i,j) values are equally likely. Suppose Bob and Alice carry out the computation faithfully according to the protocol. At the end, Alice can in principle, from her computed value v of the function and the strings in her possesion, compute a probability distribution of the values of j; call this $p_i(j)$. A protocol is said to *satisfy the* (ϵ, δ)*-privacy constraint* if the following conditions are satisfied:

1. $p_i(j) = \frac{1}{|G_i|}\big(1 + O(\epsilon)\big)$ for $j \in G_i$, and 0 otherwise, where G_i is the set of j for which $f(i,j) = v$,
2. if Alice tries afterwards to perform more calculations with no more than $O(N^k)$ evaluations of E's and D's, then with probability at least $1 - \delta$ she will still get the above distribution on j, and
3. the above requirement is also true for Bob.

Theorem 1 *For any $\epsilon, \delta > 0$ and any function f, there exists a protocol for computing f that satisfies the (ϵ, δ)-privacy constraint.*

It is possible to consider the more general case when the initial distribution of (i,j) is nonuniform. We will not go into that here. In Section 4, that becomes a special case of probabilistic computations.

3.3 Additional Requirements

(A) Complexity. The solution given earlier for the millionaires' problem will become impractical if the range n of i, j become large, since the number of bits transmitted is proportional to n. An interesting question is then, to determine the minimum number of bits needed by any protocol to compute f that satisfies the (ϵ, δ)-privacy constraint. Conceivably, there are functions that are easily computable without the privacy requirement, but become infeasible with the extra privacy constraint. Fortunately, we can prove that this is not the case. Let A be a protocol, let $T(A)$ denote the maximum number of bits exchanged between Alice and Bob when A is used.

Theorem 2 *Let $1 > \epsilon, \delta > 0$ and $f(i,j)$ be a 0-1 function. If f can be computed by a boolean circuit of size $C(f)$, then there is a protocol A computing f satisfying the (ϵ, δ)-privacy constraint such that: $T(A) = O\left(C(f) \log \frac{1}{\epsilon\delta}\right)$.*

In fact, if f can be computed by a Turing machine in time S, then the protocol can be implemented such that both Alice and Bob have Turing machine algorithms to execute the protocol with a time bound $O\big(S \log(1/\epsilon\delta)\big)$.

However, there exist functions that need exponentially many bits transmitted between Bob and Alice with the privacy constraint. Let F_n be the family of 0-1 valued function $f(i,j)$ with i and j being n-bit integers. Clearly, at most n bits of transmitted information can compute f, in the absence of the privacy constraint (See Yao [7] for further discussions).

Theorem 3 *Let $\frac{1}{5} > \epsilon, \delta > 0$ be fixed. Let f be a random element of F_n, then any protocol A that computes f with (ϵ, δ)-privacy constraint must have $T(A) > 2^{n/2}$ for all large n.*

(B) Mutually-Suspecting Participants. So far the discussions have assumed that Bob and Alice observe the rules specified by an agreed protocol. What if either of them might cheat in order to gain additional information or to mislead the other party to receive a wrong answer? It is true that with our protocol, any cheating will be discovered if there is a verification stage afterwards where both parties are required to reveal all their private computation. However, that will force both parties to reveal their variables. As will become clear in the applications to be given later, this sometimes can be a serious drawback. The following results will show that one can thwart cheating, without asking either to reveal the variable.

Since a protocol can never prohibit Alice (or Bob) from behaving as if she had a different variable value i', the most that a protocol can achieve is to make sure that this is the only cheating that Alice (or Bob) can do.

Definition 1 *Consider an instance in the execution of a protocol. We will consider it to be a* ***successful cheating*** *by Alice, if Alice does not behave consistently with any value of i and yet Bob does not detect it. A successful cheating by Bob is defined similarly.*

Theorem 4 *Let $1 > \gamma > 0$. Under the same assumption of Theorem 2, there exists a protocol A for computing f such that*

1. *$T(A) = O\left(C(f) \log \frac{1}{\epsilon\delta\gamma} \log \frac{1}{\gamma}\right)$, and*
2. *if one participant behaves according to A, the probability of a successful cheating by the other participant is at most γ.*

3.4 Applications

Secret Voting. Suppose a committee of m members wish to decide on a yes-no action. Each member is to write an opinion x_i, and the final action can be regarded as a function $f(x_1, x_2, x_3, \ldots, x_m)$. The results obtained in this paper means that it is possible to agree on the final action f, without anyone knowing the opinion of any other member's. Furthermore, the protocol makes the

probability of anyone having a successful cheating very remote.

Oblivious Negotiation. Suppose that Alice is trying to sell Bob a house. In principle, each one has a strategy of negotiation in mind. If we number all the possible strategies of Alice as $A_1, A_2, \ldots, A_t$, and those of Bob' as $B_1, B_2, \ldots, B_u$, then the outcome (no deal, or sell at x dollars, ...) will be decided once the actual strategies A_i, B_j used have been determined. Write the outcome as $f(i,j)$, then it is possible to carry out the negotiation obliviously, in the sense that Alice will not gain any information on Bob's negotiation tactics except that it is consistent with the outcome, and vice versa.

Private Querying of Database. The theorems we have proved can be extended to the case when each person P_i is computing a different function f_i. In particular, Alice may wish to compute a function $f(i,j)$ and Bob wishes to compute a trivial function $g(i,j) =$ constant, meaning that Bob will know nothing about i in the end. If we regard Bob as a database query system with j the state of the database, and Alice is asking query number i, then Alice can get answer to the query without knowing anything else about the data in it, while the database system does not know what Alice has queried.

4 Probabilistic Computations

Let us consider the case with two parties Bob and Alice ($m = 2$). Let V and W be finite sets. A function p from $V \times W$ to the interval $[0,1]$ is called a *probability density* if the sum of $p(v,w)$ over v and w is equal to 1. Let $P(V,W)$ be the set of all such probability densities.

Let I, J be finite sets of integers. Let $F = \{f_{ij} | i \in I, j \in J\} \subseteq P(V,W)$ be a family of probability densities. Initially, Alice knows the value of $i \in I$, and Bob knows $j \in J$; the values of (i,j) obey a certain initial probability desity $q \in P(I,J)$. They wish to send messages between them, so that at the end Alice will obtain a value $v \in V$ and Bob a value $w \in W$ with probability $f_{ij}(v,w)$. The privacy constraint is that the information Alice can obtain about j and w is no more than what can be inferred from her values of i and v (plus a corresponding constraint on Bob). This statement can be made precise in terms of q and F; we omit its full generality here but simply give an illustration for the special case $q =$ constant. In this special case, the distribution $h(w)$ that Alice can infer from the computation she has done should, according to the privacy constraint, is equal to

$$\frac{1}{|J|} \sum_{j \in J} \frac{f_{ij}(v,w)}{\sum_{x \in W} f_{ij}(v,x)}$$

For example, mental poker would correspond to the following situation: $I = J = \{0\}$, q is a constant, $V = W$ is the set of all 5-element subsets of $\{1, 2, \ldots, 52\}$, $f_{00}(v,w)$ is 0 if v and w are not disjoint and equal to a constant otherwise.

The results in Section 3 have generalizations to the probabilistic case. Basically, a reasonable probabilistic computation remains feasible when the privacy constraints are imposed. We will not give the details here.

One interesting corollary of our results is that mental poker can be played with any general public-key system. It differs from Shamir et. al's solution [5] in that we do not require the one-way functions used to be commutative, and that we can play it with a public-key system (instead of using private keys). (A solution with a special one-way function with publicized keys for playing mental poker was known in [2], but that solution depends on the special properties of the one-way function involved.) Moreover, the present solution uses much fewer bits as the number of cards becomes greater. Suppose we have a deck of n cards, and Alice and Bob each want to draw a random card from the deck in turn. All the previously known solutions transmit cn bits of information between Bob and Alice, while our scheme only needs about $c(\log n)^2$ bits.

5 Generalization to m-Party Case

When m parties $A_1, A_2, \ldots, A_m$ collaborate to compute a function $f(x_1, x_2, \ldots, x_m)$, more than one parties may collude to cheat. We will show that even the most severe constraint can be met in the following sense: No matter how many participants may collude, any cheating act will be detected and identified by all the honest parties (even if as many as $m-1$ dishonest guys try to help cover up). We now make it precise.

Let V be the range of the function $f(x_1, x_2, \ldots, x_m)$, where $x_i \in X_i$. For any nonempty $K \subseteq \{1, 2, \ldots, m\}$, define $H_K = X_{t_1} \times X_{t_2} \times \cdots \times X_{t_{|K|}}$, where $\{t_1, t_2, \ldots, t_{|K|}\} = K$. Let $K' = \{1, 2, \ldots, m\} - K$, and define $H_{K'}$ similarly. For any $i \in H'_K$ and $v \in V$, let $G_i(v) \subseteq H_K$ be the set of all $j \in H_K$ such that the (unique vector) $x = (x_1, x_2, \ldots, x_m)$, whose projection on $H_{K'}$ and H_K equals i and j respectively, satisfies $f(x) = v$. Let $q_{i,v}(j) = 1/|G_i(v)|$ for $j \in G_i(v)$ and 0 otherwise. (If all the participants A_r with $r \in K'$ collude to infer the probabilty distribution of the variable values of other participants, and if the only information available, in addition to their only variable values i, is that the function f has value v, then $q_{i,v}(j)$ is the distribution they can infer.) Let $\epsilon, \delta > 0$. A protocol A is said to *satisfy the* (ϵ, δ)*-private constraint* if for every nonempty K, even if the participants in K are allowed to perform in private an amount of calculation polynomial in $T(A)$, they will still infer, with probability at least $1 - \delta$, that the distribution on j is equal to $q_{i,v}(j)(1 + O(\epsilon))$. A *successful cheating* by K' (with respect to a protocol A) is an instance of the execution of A, in which at least one participant A_r with $r \in K'$ behaves inconsistently with any $x_r \in X_r$ without being detected by all the participants of K.

Theorem 5 *For any $\epsilon, \delta, \gamma > 0$, there exists a protocol A for computing f which satisfies the (ϵ, δ)-private constraint and which has the property that, for any $K' \neq \{1, \ldots, m\}$, the probability for K' to have a successful cheating can not be more than γ.*

The value of $T(A)$ in the above theorem is $O(|X_1| \cdot |X_2| \cdot \ldots \cdot |X_m| \cdot |V|)$, which is almost optimal in general as the next theorem shows.

Theorem 6 *There exist functions f for which any protocol A satisfying the conditions in Theorem 5 must have $T(A) = \Omega\left(\left(|X_1| \cdot |X_2| \cdot \ldots \cdot |X_m|\right)^{1/4}\right)$.*

In special cases protocols can be designed with better running time than the bound given in Theorem 5. For example, the *parity function* $f(x_1, x_2, \ldots, x_m) = x_1 \oplus x_2 \oplus \cdots \oplus x_m$ and the *tally function* $f(x_1, x_2, \ldots, x_m) =$ # of 1's in the x's (the x's are boolean variables) both have protocols satisfying Theorem 5 with running time polynomial in m.

The security measure as we considered above is a strong one. For some purposes, less stringent measures would be adequate. (For example, one may only require that no subset K' be able to force the outcome of the computation to be a certain value.) Sometimes protocols with better running time can be designed under such less stringent requirements. For example, there is a protocol with running time $O(p(m) \log q)$, where $p(m)$ is a polynomial, for m parties to compute the function $f(x_1, x_2, \ldots, x_m) = x_1 + x_2 + \cdots + x_m (\mathrm{mod}\ q)$, under a security criterion only slightly relaxed from that given in Theorem 5.

6 What Cannot be Done?

There are security constraints that can not be achieved by any protocols. We will only mention two results here.

The first impossibility result is valid for all three models given in this paper. Suppose m people try to generate a bit with bias α. It is easy to see how it can be done for $m > 2$. For example A generates a random unbiased bit α_1 and sends it to B, B generates a random α_2 and sends it to C, C generates a random α_3 and sends it to A. Now let $\alpha = \alpha_1 + \alpha_2 + \alpha_3$, and we get an unbiased α with the property that it remains unbiased even if one of the persons cheats by generating a biased bit. Let us call a protocol for generating a bit with bias α *robust*, if the bias remains correct if somebody has cheated.

Theorem 7 *No protocol A with finite $T(A)$ which generates a bit with a transcendental bias α can be robust.*

The second result is valid for the model defined in Section 3.2. Suppose Alice and Bob wish to exchange a pair of solutions x, y with $E_a(x) = 1$ and $E_b(y) = 1$. Is there a protocol such that an honest party will not be double-crossed, i.e. swindled out of its secret without getting the secret from the other party.

Theorem 8 *Let A be any protocol for exchanging secrets. Then either Alice or Bob will be able to double-cross successfully with probability at least $1/2$.*

It is of interest to mention that a different type of exchanging secrets is possible in the same model. Suppose Alice wants to know the solution y to $E_b(y) = w$ and Bob wants to know the solution x to $E_a(x) = u$, but Bob does not know the value of w and Alice does not know u. Let N be the number of bits that the encryption functions E_a and E_b operate on.

Theorem 9 *Let $\epsilon > 0$ be fixed. There is a protocol A with polynomial (in N) running time which exchanges secrets $D_b(w)$ and $D_a(u)$, and under which the probability of anyone double-crossing successfully is bounded by ϵ.*

Different kinds of exchanging secrets have been considered previously. Blum [6] showed that it is possible to exchange factors of a large composite numbers (a special type of secrets) with vanishing chance for cheating. Even (private communication, 1981) also devised some protocols for exchanging secrets.

References

[1] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[2] S. Goldwasser and S. Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proceedings of the 14th ACM Symposium on Theory of Computing (STOC'82)*, pages 365–377, San Francisco, CA, USA, May 1982.

[3] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report LCS/TR-212, Massachusetts Institute of Technology, 1979.

[4] R. L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[5] Adi Shamir, R. L. Rivest, and Leonard M. Adleman. Mental poker. Technical Report LCS/TR-125, Massachusetts Institute of Technology, April 1979.

[6] Manuel Blum. Three applications of the oblivious transfer: Part I: Coin flipping by telephone; part II: How to exchange secrets; part III: How to send certified electronic mail. Technical report, University of California, Berkeley, CA, USA, 1981.

[7] Andrew C. Yao. Some complexity questions related to distributive computing. In *Conference Record of the 11th ACM Symposium on Theory of Computing (STOC'79)*, pages 209–213, Atlanta, GA, USA, April 1979.