Information and Computation **158**, 71–97 (2000)
doi:10.1006/inco.2000.2865, available online at http://www.idealibrary.com on IDEAL®

Communication Protocols for Secure Distributed Computation of Binary Functions

Eytan Modiano

Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

and

Anthony Ephremides

Electrical Engineering Department, University of Maryland, College Park, Maryland 20742

A common task in parallel processing is the distributed computation of a function by a number of processors, each of which possesses partial information relevant to the value of that function. In this paper we develop communication protocols which allow for such computation to take place while maintaining the value of the function secret to an eavesdropper. Of interest is the *communication* complexity of such protocols. We begin by considering two processors and two channels, one secret and one public, and present a protocol which minimizes the number of bits exchanged over the secret channel, while maintaining ε-uncertainty about the value of the function for the eavesdropper. We show that all binary functions can be kept ε-secret using a constant number of bits independent of the size of their domain. We then generalize our results to N processors communicating over a network of arbitrary topology.

1. INTRODUCTION

This paper examines the *communication* complexity of secure distributed computation. This is a relatively new area in the field of secret communications, which studies the message exchange process that is needed to support a task such as a distributed computation or the implementation of a protocol, under secrecy constraints. Such computations may be needed in an environment where multiple computers (processors) are connected via a network. The secrecy aspect of the problem arises in many environments such as the military or the financial worlds. The problem presented in this paper is still in its infancy, but once developed it may have application ranging from defense applications such as missile guidance systems

0890-5401/00 $35.00

and distributed simulation to commercial applications such as electronic banking. In this paper we focus exclusively on the communication complexity aspect of performing this distributed computation and not on the computational aspects.

This problem in its simplest form was introduced in [Yao, 1979], and it involved two processors, P_x and P_y, which are interested in exchanging the values of two variables X and Y. Processor P_x knows the value of X and P_y knows the value of Y, and they communicate according to a predetermined protocol. In addition to exchanging the values of their variables, both processors wish to compute the value of a Boolean function $F(X, Y)$. This form of exchange may arise, for example, in cryptographic protocols where users may want to exchange information in order to compute a secret encryption key for their conversation. In such a case, the information that they exchange may include some identification code which needs to be completely exchanged, and the encryption key may be a function of that code. Yet another example may arise in a battlefield environment where users, which in this case may be aircraft or guided missiles, exchange information regarding a target, in order to determine whether the target belongs to a friendly force or an enemy force. In this case the information they exchange may include images of the target and the function they wish to compute may be the binary function identifying the target as a friend or a foe.

Nonetheless, in this paper we are not concerned with the design of a new encryption algorithm or any other issue related to computational complexity; rather, we are only concerned with the information exchange that is involved in the secret computation of the function. Originally the problem focused on the determination of the minimum number of bits that need to be exchanged to perform this computation. In [Orlitski, 1984] a secrecy aspect was introduced by considering an eavesdropper who monitors the channel in order to obtain information about the value of $F(X, Y)$ and who is aware of the protocol used by the processors and the structure of the function they wish to compute. The criterion for secrecy is that the communicating processors wish to keep the probability distribution of $F(X, Y)$, before and after the communication takes place, ε-close to each other as they appear through the channel monitored by the eavesdropper; i.e.,

$$|Pr\{F(X, Y)=1 \mid I(X, Y)\} - Pr\{F(X, Y)=1\}| \leqslant \varepsilon, \tag{1}$$

where $I(X, Y)$ denotes the observed values of the quantities communicated between P_x and P_y. To achieve this objective some of the bits are transmitted over a private secret channel that is inaccessible to the eavesdropper. The problem we consider is to find a protocol that minimizes the worst-case number of bits needed to be transmitted over the secret channel in order for inequality (1) to hold. That is, the protocol should minimize the number of bits that need to be exchanged secretly for a worst-case function. It was shown in [Orlitski, 1984] that when X and Y are *uniformly distributed* over their range, for any Boolean function F and $\varepsilon > 0$, P_x and P_y need to exchange no more than $2\log(1/\varepsilon) + 16$ bits over the secret channel. The effects of noise on the above protocol were studied in [Modiano, 1992] where it was shown that in the presence of noise in the eavesdropper's channel the transmission of fewer secret bits is required.

In this paper we generalize the problem in a number of ways. First we study the effects of the *nonuniform distribution* of X and Y. In Section 2 we provide a lower bound on the number of secret bits that any protocol would require in order to achieve the ε-secret computation of a worst-case function. In Section 3 we present an alternating bit protocol which appears (via numerical optimization techniques) to meet this bound. The use of an alternating bit protocol is of interest because it provides a natural continuation to the problem addressed in [Orlitski, 1984]. However, in Section 4, we relax the need for an alternating bit protocol and allow users to send messages of arbitrary size. In that case we are able to show that the protocol meets the lower bound of Section 2. We then generalize our protocol to N processors communicating over a broadcast channel and we discuss an extension of this protocol to a network of arbitrary topology.

1.1. Problem Definition

Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets and X and Y be random variables distributed over $\mathcal{X}$ and $\mathcal{Y}$, respectively. Let F be a function from $\mathcal{X} \times \mathcal{Y}$ to $\{0, 1\}$. Processor P_x knows the value of X and P_y knows the value of Y. The processors communicate according to a predetermined protocol in order to exchange the values of their variables for the purpose of computing F. An eavesdropper, *who knows both their protocol and the function F*, listens to their communication in order to obtain information about $F(X, Y)$. Processors P_x and P_y want to make sure that for every value (x, y) of (X, Y) the eavesdropper's probabilities of $F(X, Y) = 1$, before and after the communication takes place, are ε-close. Therefore they transmit some of the bits over a secret channel. The problem is to find a protocol that minimizes the worst-case number of bits needed to be transmitted secretly.

The two processors use a protocol P. For every $(x, y) \in \mathcal{X} \times \mathcal{Y}$, P specifies the following sequence,

$$\langle T_i, B_i, S_i \rangle, i = 1, ..., N,$$

where T_i describes the identity of the originator of the ith bit (i.e., P_x or P_y), B_i describes the ith bit itself (i.e., 0 or 1), and S_i denotes the channel used (Secret or Public); N is the total number of bits communicated. When the sequence T_i, B_i, S_i is a deterministic function of x and y, the protocol is said to be *deterministic*. In this paper we focus on deterministic protocols for the exchange of X and Y. An eavesdropper who knows the originator of each bit but can decode only the publicly communicated bits constructs the modified sequence $\langle T_i, B_i', S_i \rangle$, $i = 1, ..., N$, where

$$B_i' = \begin{cases} B_i & \text{if} \quad S_i = \text{public} \\ 0 & \text{if} \quad S_i = \text{secret.} \end{cases}$$

Note that, if $S_i = \text{Secret}$, the eavesdropper will have to guess the value of B_i.

Denote this modified sequence by $I(x, y)$.

A protocol is said to be ε-secret if for all transmission sequences $e = \langle T_i, B_i, S_i \rangle_i^N$ which have nonzero probability,

$$|Pr\{F(X, Y) = 1 \mid I(x, y)\} - Pr\{F(X, Y) = 1\}| \leqslant \varepsilon.$$

This ensures that the eavesdropper's a priori and a posteriori probability distributions of F are ε-close.

Let $C_P(x, y)$ be the number of bits communicated secretly under a protocol P when $X = x$ and $Y = y$. Then the worst–case complexity of P is defined as

$$C_P = \max C_P(x, y),$$

where the maximum is taken over all possible values x, y of X and Y. In other words, $C_P(x, y)$ is the maximum number of bits that the protocol would have to exchange secretly for any value of the variables X and Y.

The ε-complexity of F is defined as

$$C(F, \varepsilon) = \min\{C_P \mid P \text{ is an } \varepsilon\text{-secret protocol for } F\}.$$

Thus, $C(F, \varepsilon)$ is the maximum number of secret bits that the function F requires when using the best possible protocol.

Finally, our objective is to find a protocol which minimizes the worst case $C(F, \varepsilon)$ over all possible functions. In other words, we are looking for a protocol which minimizes the number of secret bits that are required for the worst possible function. Therefore this protocol may not be optimal for a particular function. However, as we will show in the remainder of this paper, it provides bounded performance for all functions. Furthermore, it will be shown that this bound is a tight bound in the sense that there is a function whose ε-complexity meets this bound. We begin, in the next section, by demonstrating a particular function whose ε-complexity is the worst possible.

2. LOWER BOUND ON COMPLEXITY

We begin by proving a lower bound on the number of secret bits that any deterministic protocol must use in order to exchange X and Y.[1] When a protocol, such as the one described in the previous section, is used to exchange the values of X and Y, the eavesdropper has access only to the bits transmitted over the public channel. Without access to the secret bit the eavesdropper is left confused with respect to the different values for X and Y for which the protocol transmits the same public bits. It is because of this confusion that such a protocol maintains the ε-secrecy of F.

[1] Note that the computation of a function $F(X, Y)$ can usually be done without the full exchange of X and Y. Here we simplify the problem slightly by assuming that this full exchange is needed even when the value of F can be obtained earlier. In other words, we decouple the problem of the optimal computation of F from the secrecy problem.

Also, it is left to the secret bits to resolve this confusion for the communicators and identify the exact values of X and Y in order to complete the exchange. Let

$$S(x, y) = \{(x', y') \in \mathcal{X} \times \mathcal{Y} \mid I(x, y) = I(x'y')\},$$

where $I(x, y)$ is the modified sequence defined earlier. That is, $S(x, y)$ is the set of pairs (x', y') for which the protocol produces the same sequence of public bits as that of a given pair (x, y). Then, by the definition of ε-secrecy we require that for every pair (x, y)

$$|P(F=1 \mid S(x, y)) - P(F=1)| \leqslant \varepsilon.$$

It is shown in [Orlitski, 1990] that for deterministic protocols every $S(x, y)$ must be a rectangular subset of $\mathcal{X} \times \mathcal{Y}$. That is, $S(x, y) = \mathcal{X}' \times \mathcal{Y}'$, where $\mathcal{X}' \subseteq \mathcal{X}$ and $\mathcal{Y}' \subseteq \mathcal{Y}$. Let $C(x, y)$ be the number of secret bits the protocol requires when $X = x$ and $Y = y$ and let $|S(x, y)|$ denote the number of distinct pairs $(x', y') \in S(x, y)$. Then

$$C(x, y) \geqslant \log(|S(x, y)|).$$

The above clearly holds because the secret bits must distinguish between $|S(x, y)|$ elements (since we are concerned with worst-case complexity, we assume that all of the sequences are of the same size). Theorem 1 follows immediately from the above observations.

THEOREM 1. *Let $S^*(x, y)$ be the smallest rectangular subset of $\mathcal{X} \times \mathcal{Y}$ containing the pair (x, y) such that*

$$|P(F=1 \mid S^*(x, y)) - P(F=1)| \leqslant \varepsilon.$$

Then, when $X = x$ and $Y = y$ any deterministic protocol must transmit

$$C(x, y) = \log((|S^*(x, y)|) \text{ secret bits},$$

and

$$C(F, \varepsilon) \geqslant \max_{x, y} \log((|S^*(x, y)|),$$

where the latter part of theorem follows directly from the definition of $C(F, \varepsilon)$.

Lower Bound. We present a lower bound on the worst-case complexity of any protocol by giving an example of a function that requires that many secret bits. Of course, there are many functions that require far fewer bits, but here we are looking for a worst-case complexity and therefore we search for the worst function that requires the most secret bits.

Consider the function

$$F_1(X, Y) = \begin{cases} 1 & \text{if} \quad X = 1 \text{ or } Y = 1 \\ 0 & \text{otherwise} \end{cases}$$

with probability mass function (PMF)

$$P(X, Y) = \begin{cases} K\alpha & \text{if} \quad X = 1 \text{ or } Y = 1 \\ \alpha & \text{otherwise,} \end{cases}$$

where $\alpha = 1/(n^2 + 2(K-1)\, n - K + 1)$, $n = |\mathscr{X}| = |\mathscr{Y}|$ and $K = \max P(x, y)/\min P(x, y))$.

The prior probability of F_1 being equal to 1 is clearly $(2n-1)\, K\alpha$. Suppose now that the pair $(1, 1)$ is contained in the rectangular subset $\mathscr{X}' \times \mathscr{Y}'$. Then it follows that

$$P(F_1 = 1 \mid (1, 1) \in \mathscr{X}' \times \mathscr{Y}') = \frac{(|\mathscr{X}'| + |\mathscr{Y}'| - 1)\, K\alpha}{(|\mathscr{X}'| + |\mathscr{Y}'| - 1)\, K\alpha + (|\mathscr{X}'| \times |\mathscr{Y}'| - |\mathscr{X}'| - |\mathscr{Y}'| + 1)\, \alpha}.$$

In order to maintain the ε-secrecy of F_1 we must have

$$\frac{(|\mathscr{X}'| + |\mathscr{Y}'| - 1)\, K\alpha}{(|\mathscr{X}'| + |\mathscr{Y}'| - 1)\, K\alpha + (|\mathscr{X}'| \times |\mathscr{Y}'| - |\mathscr{X}'| - |\mathscr{Y}'| + 1)\, \alpha} \leqslant (2n-1)\, K\alpha + \varepsilon.$$

In order to find $|S(1, 1)|$ we must determine the minimum of $|\mathscr{X}'| \times |\mathscr{Y}'|$ so that the above inequality is satisfied. By taking $|\mathscr{X}'| = |\mathscr{Y}'|$ we have

$$P(F_1 = 1 \mid (1, 1) \in \mathscr{X}' \times \mathscr{Y}') = \frac{(2\,|\mathscr{X}'| - 1)\, K}{(2\,|\mathscr{X}'| - 1)\, K + |\mathscr{X}'|^2 - 2\,|\mathscr{X}'| + 1} \leqslant (2n-1)\, K\alpha + \varepsilon,$$

from which it follows that

$$\log(|\mathscr{X}'|) \geqslant \log(K) + \log(1 - \varepsilon') - \log(\varepsilon'),$$

where $\varepsilon' = (2n-1)\, K\alpha + \varepsilon$ and $\lim_{n \to \infty} \varepsilon' = \varepsilon$. Therefore, when $\varepsilon \leqslant 1/2$ and $n \to \infty$ we have

$$\log(|\mathscr{X}'|) \geqslant \log(K) + \log\left(\frac{1}{\varepsilon}\right) - 1$$

and

$$\log(|\mathscr{X}'| \times |\mathscr{Y}'|) \geqslant 2\log(K) + 2\log\left(\frac{1}{\varepsilon}\right) - 2.$$

Therefore, $|S(1, 1)| \geqslant 2\log(K) + 2\log(\frac{1}{\varepsilon}) - 2$, and $C(F_1, \varepsilon) \geqslant 2\log(K/\varepsilon) - 2$. We have now shown a function that requires at least $2\log(K/\varepsilon) - 2$ secret bits. In the

next section we will present a protocol which can achieve the ε-secret computation of *any* binary function using no more than $2\log(K/\varepsilon)+10$ secret bits, where K denotes the ratio of the maximum to the minimum values of the PMF of (X, Y), as it did in this example. The analysis presented in the next section relies only on numerical techniques to evaluate the performance of the protocol and, therefore, we cannot conclusively claim that the protocol of the next section provides an upper bound on complexity. However, in Section 4.2 we present another protocol, for which we are able to show that the worst-case number of secret bits that need to be exchanged is $2\log(K/\varepsilon)+10$. We are therefore able to conclude that the protocol of Section 4.2 is indeed optimal (within a specified small number of bits) for worst-case functions. Nonetheless, the alternating bit protocol to be described is simple and, given the numerical evidence of its optimality, worthy of consideration.

3. THE PROTOCOL

In this section we present an alternating-bit protocol for exchanging the values of X and Y while maintaining the ε-secrecy of $F(X, Y)$. The protocol is based on a suitable partitioning of the function table for $F(X, Y)$. Partitions are formed so that all subtables in each partition have approximately the same probability of F being equal to 1. The protocol begins by P_x partitioning $\mathcal{X}$ into $\mathcal{X}'$ and $\mathcal{X}''$ and using one bit to inform P_y whse subset contains X. In the next step P_y partitions $\mathcal{Y}$ into $\mathcal{Y}'$ and $\mathcal{Y}''$ and uses one bit to inform P_x of the subset containing Y. The processors continue to partition $\mathcal{X}$ and $\mathcal{Y}$ and exchange partitioning information one bit at a time, until the exact values of X and Y are obtained by both processors (and consequently $F(X, Y)$). Figure 1 shows the partitioning of a function table for an arbitrary function $F(X, Y)$.

These exchanges take place over the public channel as long as the ε-secrecy requirement is not violated. Namely, if we let I_j denote the first j bits exchanged, then bit j is sent over the public channel as long as

$$|P(F=1 \mid I_j) - P(F=1)| \leqslant \varepsilon.$$

This, of course, ensures that the eavesdropper's probability of F being equal to 1 is within ε of the prior probability of F being equal to 1. As a consequence, the processors exchange information over the public channel until they must switch over to the secret channel, after which they send the rest of the bits over the secret channel. The key to this protocol is in the formation of the partitions of $\mathcal{X}$ and $\mathcal{Y}$. How we form these partition will determine the number of bits that must be sent over the secret channel, and is described next.

3.1. The First Partition

Let $P(x, y)$ be the PMF of X and Y which takes values in $\mathcal{X}\times\mathcal{Y}$. We assume that $\forall(x, y)\in\mathcal{X}\times\mathcal{Y}\ P(x, y)>0$ and we let

$$K=\frac{\max_{x, y} P(x, y)}{\min_{x, y} P(x, y)}.$$

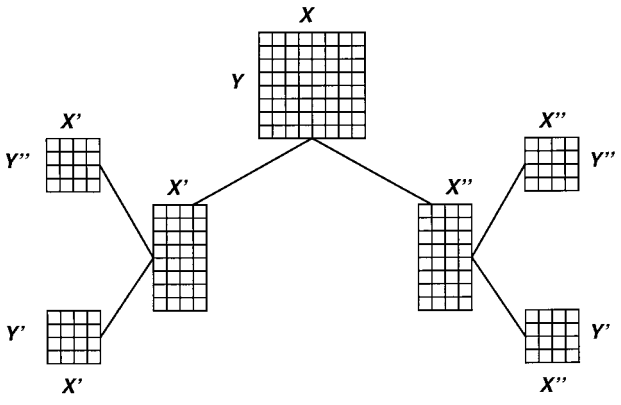

FIG. 1. The partitioning of a function table.

The first partition of $\mathcal{X}$ into $\mathcal{X}'$ and $\mathcal{X}''$ is formed by first splitting $\mathcal{X}$ into m_0 subsets, $\mathcal{X}^1, \mathcal{X}^2, ..., \mathcal{X}^{m_0}$, where

$$\mathcal{X}^i = \left\{ x : \frac{m_0 - i}{m_0} \leqslant P(F=1 \mid X=x) < \frac{m_0 - i + 1}{m_0} \right\}.$$

These $\mathcal{X}^i$'s are subsets of $\mathcal{X}$ for which the value of $P(F=1)$ lies within a narrow range; clearly the value of m_0 denotes the *step-size* or *granularity* of the splitting. We next proceed to divide the elements of each of the $\mathcal{X}^i$'s between $\mathcal{X}'$ and $\mathcal{X}''$ so that $P(F=1 \mid \mathcal{X}') \approx P(F=1 \mid \mathcal{X}'')$. We do so by using the following procedure:

For all i, let us arrange the members of $\mathcal{X}^i$, $x_1^i, x_2^i, ..., x_{k_i}^i$, in decreasing order of probability value; that is

$$P(x_1^i) \geqslant P(x_2^i) \geqslant \cdots \geqslant P(x_{k_i}^i),$$

where k_i is the size of the set $\mathcal{X}^i$. Partition each of the sets $\mathcal{X}^i$ into $\mathcal{X}^{i\prime}$ and $\mathcal{X}^{i\prime\prime}$ using the following inductive set of steps:

Step 1. Let $\mathcal{X}^{1\prime}$ contain odd-indexed members of $\mathcal{X}^1$ and let $\mathcal{X}^{1\prime\prime}$ contain even-indexed members of $\mathcal{X}^1$.

Step i. If $\sum_{l=1}^{i-1} P(\mathcal{X}^{l\prime}) \leqslant \sum_{l=1}^{i-1} P(\mathcal{X}^{l\prime\prime})$ then let $\mathcal{X}^{i\prime}$ contain odd members of $\mathcal{X}^i$ and let $\mathcal{X}^{1\prime\prime}$ contain even members of $\mathcal{X}^i$; otherwise let $\mathcal{X}^{i\prime\prime}$ contain odd members of $\mathcal{X}^i$ and let $\mathcal{X}^{i\prime}$ contain even members of $\mathcal{X}^i$

Next, let $\mathcal{X}' = \bigcup_{i=1}^{i=m_0} \mathcal{X}^{i\prime}$ and $\mathcal{X}'' = \bigcup_{i=1}^{i=m_0} \mathcal{X}^{i\prime\prime}$. Figure 2 shows the splitting of $\mathcal{X}$ into m_0 subsets and forming $\mathcal{X}'$ and $\mathcal{X}''$.

Finally, P_x notifies P_y if X is in $\mathcal{X}'$ or $\mathcal{X}''$ by sending one bit x_1, where

$$x_1 = \begin{cases} 0 & \text{if} \quad X \in \mathcal{X}' \\ 1 & \text{if} \quad X \in \mathcal{X}''. \end{cases}$$

We call $\mathcal{X}'$ and $\mathcal{X}''$ first-level subsets of $\mathcal{X}$. We denote the first-level subset containing X by $\mathcal{X}_1$.

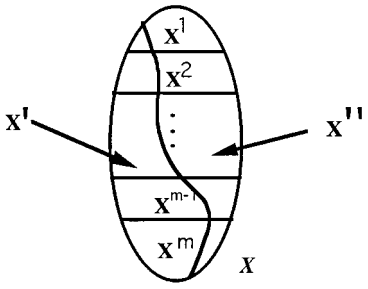

FIG. 2. Partitioning $\mathscr{X}$ into $\mathscr{X}'$ and $\mathscr{X}''$.

In the Appendix we show that (for $K \leq n/3$),

$$|P(F=1 \mid \mathscr{X}_1) - P(F=1)| \leqslant \frac{3K}{n} + \frac{1}{m_0},$$

where $n = |\mathscr{X}|$. The above represents a measure of the information disclosed by revealing $\mathscr{X}_1$.

Upon receiving x_1, P_y forms the first partition of $\mathscr{Y}$ into $\mathscr{Y}'$ and $\mathscr{Y}''$ using the distribution $P(Y \mid X \in \mathscr{X}_1)$ and the same algorithm used by P_x. First, P_y partitions $\mathscr{Y}$ into m_0 subsets $\mathscr{Y}^1, \mathscr{Y}^2, \ldots, \mathscr{Y}^{m_0}$, where

$$\mathscr{Y}^i = \left\{ y : \frac{m_0 - i}{m_0} \leqslant P(F=1 \mid Y=y, X \in \mathscr{X}_1) < \frac{m_0 - i + 1}{m_0} \right\}.$$

Then $\mathscr{Y}'$ and $\mathscr{Y}''$ are formed to approximate the condition $P(F=1 \mid \mathscr{X}_1, \mathscr{Y}') = P(F=1 \mid \mathscr{X}_1, \mathscr{Y}'')$. We proceed as follows: For all i, label the members of $\mathscr{Y}^i$, $y_1^i, y_2^i, \ldots, y_{k_i}^i$, such that

$$P(y_1^i \mid \mathscr{X}_1) \geqslant P(y_2^i \mid \mathscr{X}_1) \geqslant \cdots \geqslant P(y_{k_i}^i \mid \mathscr{X}_1),$$

where k_i now denotes the size of the set $\mathscr{Y}^i$. As before, partition each of the sets $\mathscr{Y}^i$ into $\mathscr{Y}^{i\prime}$ and $\mathscr{Y}^{i\prime\prime}$ using the following inductive set of steps:

Step 1. Let the set $\mathscr{Y}^{1\prime}$ contain odd members of $\mathscr{Y}^1$ and the set $\mathscr{Y}^{1\prime\prime}$ contain even members of $\mathscr{Y}^1$.

Step i. If $\sum_{l=1}^{i-1} P(\mathscr{Y}^{l\prime} \mid \mathscr{X}_1) \leqslant \sum_{l=1}^{i-1} P(\mathscr{Y}^{l\prime\prime} \mid \mathscr{X}_1)$ then let the set $\mathscr{Y}^{i\prime}$ contain odd members of $\mathscr{Y}^i$ and the set $\mathscr{Y}^{i\prime\prime}$ contain even members of $\mathscr{Y}^i$; otherwise, let $\mathscr{Y}^{i\prime\prime}$ contain odd members of $\mathscr{Y}^i$ and let $\mathscr{Y}^{i\prime}$ contain even members of $\mathscr{Y}^i$.

Next, we let $\mathscr{Y}' = \bigcup_{i=1}^{i=m_0} \mathscr{Y}^{i\prime}$ and $\mathscr{Y}'' = \bigcup_{i=1}^{i=m_0} \mathscr{Y}^{i\prime\prime}$.

Finally, P_y notifies P_x if Y is in $\mathscr{Y}'$ or $\mathscr{Y}''$ by sending one bit y_1, where,

$$y_1 = \begin{cases} 0 & \text{if} \quad Y \in \mathscr{Y}' \\ 1 & \text{if} \quad Y \in \mathscr{Y}''. \end{cases}$$

We call $\mathscr{Y}'$ and $\mathscr{Y}''$ first-level subsets of $\mathscr{Y}$. We denote the first-level subset containing Y by $\mathscr{Y}_1$.

Again, it is shown in the Appendix that,

$$|P(F=1 \mid \mathscr{X}_1, \mathscr{Y}_1) - P(F=1 \mid \mathscr{X}_1)| \leqslant \frac{3K}{n} + \frac{1}{m_0}.$$

This quantity is a measure of the information disclosed by revealing $\mathscr{Y}_1$. Here, and for the remainder of this paper, we assume that $|\mathscr{X}| = |\mathscr{Y}| = n$. This assumption is made for simplicity of presentation. However, this restriction can be easily removed without significantly altering the results.

3.2. The jth Partition

The information exchange continues by forming successive partitions and revealing the partition to which the variables belong. Having exchanged the $2(j-1)$ bit-sequence $x_1, y_1, ..., x_{j-1}, y_{j-1}$ both P_x and P_y know that X and Y are contained in $\mathscr{X}_{j-1} \times \mathscr{Y}_{j-1}$, where $\mathscr{X}_{j-1}$ and $\mathscr{Y}_{j-1}$ are the $(j-1)$st-level subsets of X and Y respectively. The jth-level subsets are then formed according to the same rules used in forming the first subsets. However, these subsets are formed using the distribution of X given that X and Y are contained in $\mathscr{X}_{j-1} \times \mathscr{Y}_{j-1}$.

The set $\mathscr{X}_{j-1}$ is partitioned into $\mathscr{X}'_{j-1}$ and $\mathscr{X}''_{j-1}$ by first splitting $\mathscr{X}_{j-1}$ into $m_{(j-1)}$ subsets and then dividing the elements of each of those subsets between $\mathscr{X}'_{j-1}$ and $\mathscr{X}''_{j-1}$ according to the same rules used in forming the first partition of $\mathscr{X}$.

Again, P_x notifies P_y if X is in $\mathscr{X}'_{j-1}$ or $\mathscr{X}''_{j-1}$ by sending one bit x_j. We call $\mathscr{X}'_{j-1}$ and $\mathscr{X}''_{j-1}$, the jth-level subsets of $\mathscr{X}$. We denote the jth-level subset containing X by $\mathscr{X}_j$.

As was done for the first partition of $\mathscr{Y}$ we compute the probability that the function equals 1 given the jth partition as a function of that probability given the previous partitions. We begin by noting that

$$P(F=1 \mid \mathscr{X}_j, \mathscr{Y}_{j-1}) = \frac{P(F=1, \mathscr{X}_j \mid \mathscr{X}_{j-1}, \mathscr{Y}_{j-1})}{P(\mathscr{X}_j \mid \mathscr{X}_{j-1}, \mathscr{Y}_{j-1})}.$$

Again, it can be shown (using the same reasoning as was used for the first partitions in the Appendix) that

$$P(x \mid \mathscr{X}_{j-1}, \mathscr{Y}_{j-1}) \leqslant \frac{K}{|\mathscr{X}_{j-1}|}$$

and therefore it follows that

$$|P(F=1 \mid \mathscr{X}_j, \mathscr{Y}_{j-1}) - P(F=1 \mid \mathscr{X}_{j-1}, \mathscr{Y}_{j-1})| \leqslant \frac{3K}{|\mathscr{X}_{j-1}|} + \frac{1}{m_{(j-1)}}. \tag{2}$$

The proof of inequality (2) is identical to the proof presented for the first partition and is therefore omitted.

Upon receiving x_j, P_y forms the jth partition of $\mathcal{Y}$ using the distribution $P(Y \mid X \in \mathcal{X}_j, Y \in \mathcal{Y}_{j-1})$. The set $\mathcal{Y}_{j-1}$ is partitioned into $\mathcal{Y}'_{j-1}$ and $\mathcal{Y}''_{j-1}$ by first splitting $\mathcal{Y}_{j-1}$ into $m_{(j-1)}$ subsets and then dividing the elements of each of these subsets between $\mathcal{Y}'_{j-1}$ and $\mathcal{Y}''_{j-1}$ in the same way as was done for the first partition.

Finally, P_y notifies P_x if Y is in $\mathcal{Y}'_{j-1}$ or $\mathcal{Y}''_{j-1}$ by sending one bit y_j. We call $\mathcal{Y}'_{j-1}$ and $\mathcal{Y}''_{j-1}$, the jth-level subsets of $\mathcal{Y}$. We denote the jth-level subset containing Y by $\mathcal{Y}_j$.

Here, again, it can be shown that

$$P(x \mid \mathcal{X}_j, \mathcal{Y}_{j-1}) \leqslant \frac{K}{|\mathcal{Y}_{j-1}|}$$

and therefore it follows that

$$|P(F=1 \mid \mathcal{X}_j, \mathcal{Y}_j) - P(F=1 \mid \mathcal{X}_j, \mathcal{Y}_{j-1})| \leqslant \frac{3K}{|\mathcal{Y}_{j-1}|} + \frac{1}{m_{(j-1)}}. \tag{3}$$

3.3. Complexity Analysis

The objective of our protocol is to minimize the number of secret bits that must be exchanged, while maintaining the ε-security of the function. The protocol continues to partition $\mathcal{X}$ and $\mathcal{Y}$ and communicate the partitioning information over a public channel until the posterior probability of the function being equal to 1 exceeds the prior probability of F being equal to 1 by more than ε. At that point P_x and P_y exchange the rest of the information over the secure channel. We analyze our protocol by first computing the probability of the function being equal to 1, given an ith-level partition of $\mathcal{X}$ and $\mathcal{Y}$.

Now, by repeatedly applying Eqs. (2) and (3) and accounting for the sizes of the partitions$(\mathcal{X}_i, \mathcal{Y}_i)$ we have that the probability, P_i, of the function F being equal to 1 after i partitions is bounded by

$$P_i \leqslant P_0 + 3K \sum_{j=0}^{i-1} \left[\frac{2^{j+1}}{n - \sum_{l=0}^{j-1} m_l 2^l} + \sum_{j=0}^{i-1} \frac{2}{m_j} \right], \tag{4}$$

where P_0 is the prior probability of the function F being equal to 1, m_{i-1} is the number of subsets used in forming the ith partition, n is the cardinality of the ranges of X and Y (assumed to be the same), and K is defined by:

$$K = \frac{\max_{x,\,y} P(x,\,y)}{\min_{x,\,y} P(x,\,y)}.$$

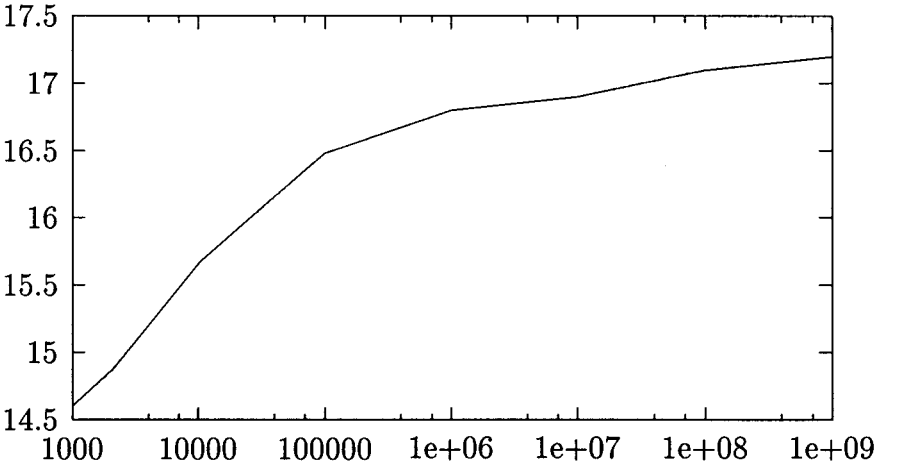

FIG. 3. $C(\varepsilon)$ vs. n with $K=1$ and $\varepsilon=0.1$.

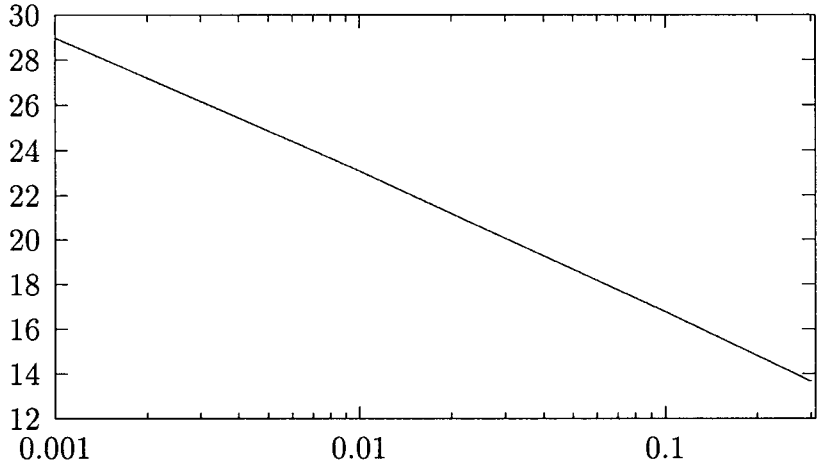

FIG. 4. $C(\varepsilon)$ vs. ε with $K=1$ and $n=10^6$.

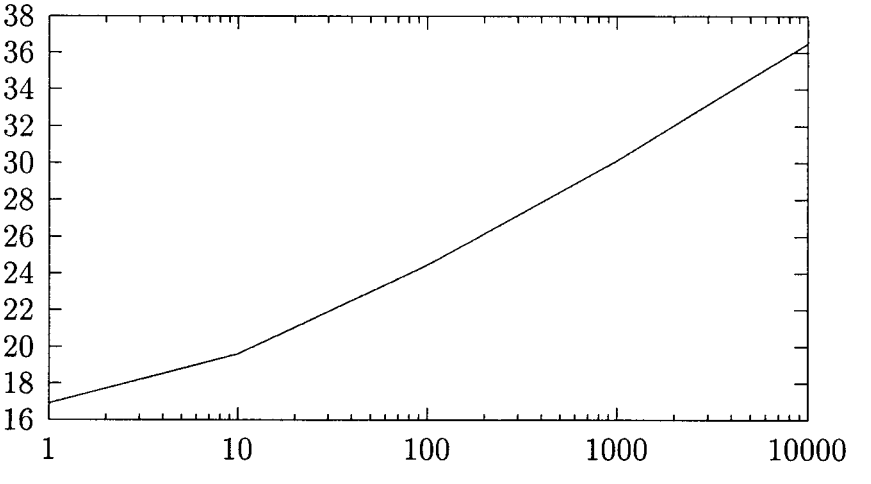

FIG. 5. $C(\varepsilon)$ vs. K with $n=10^7$ and $\varepsilon=0.1$.

We still need to compute the number of bits that are left to transmit over the secret channel. This, of course, is the logarithm of the size of the ith partition and can be computed as follows:

$$|\mathscr{X}_0| = n$$

$$|\mathscr{X}_1| \leqslant \frac{|\mathscr{X}_0|}{2} + m_0$$

$$|\mathscr{X}_i| \leqslant \frac{n}{2^i} + \sum_{j=0}^{i-1} \frac{m_j}{2^{i-j}}.$$

Now, since the same holds for $|\mathscr{Y}_i|$, the number of secret bits, a_i, that remain to be transmitted is twice the log of the above quantity. Hence

$$a_i \leqslant 2 \log \left[\frac{n}{2^i} + \sum_{j=0}^{i-1} \frac{m_j}{2^{i-j}} \right]. \tag{5}$$

In order to determine the worst-case number of secret bits that our protocol requires, we must minimize the number of secret bits, a_i, subject to the constraint that $P_i \leqslant P_0 + \varepsilon$. The resulting minimum, a_{min}, is the smallest table-size possible that does not violate inequality (1). The actual number of secret bits transmitted in the worst case is given by $\lceil \log(a_{\min}) \rceil$. This minimization should be done over all possible values of i and all values of $m_0, m_1, ..., m_i$. Since analytical minimization does not seem feasible we resort to numerical computation of the minimum for various values of n, K, and ε by using techniques from constrained optimization [Bertsekas, 1982]. Figure 3 plots the number of secret bits, $C(\varepsilon)$, versus n, when $\varepsilon = 0.1$ and $K = 1$. Similarly, Fig. 4 plots $C(\varepsilon)$ vs ε, when $K = 1$ and $n = 10^6$, and Fig. 5 plots $C(\varepsilon)$ versus K, when $n = 10^7$ and $\varepsilon = 0.1$. We performed this minimization for various values of n, K, and ε and our results indicate that $C(\varepsilon)$ is upper-bounded by $2 \log(K/\varepsilon) + 10$. This implies that our protocol can achieve ε-secrecy *for any binary function F* and any distribution of its arguments X and Y by using a constant number of bits that is independent of n.

4. PROTOCOL EXTENSIONS

In the previous section we considered two processors communicating over a broadcast channel and using an alternating bit protocol to exchange the values of their variables. Here we consider some alterations to the model. We begin by extending our protocol of the previous section to N processors over a broadcast channel. We then propose an alteration to the alternating-bit protocol so that each processor can transmit messages of arbitrary size. Finally, we propose a scheme that would allow this exchange to take place over a network of arbitrary topology.

4.1. N Processors over a Broadcast Channel

The contents of this section form a simple extension of the alternating bit protocol of the previous section to N processors communicating over a broadcast

channel. The broadcast channel model requires that all of the nodes in the network are connected to each other and, thus, every transmission by a node is received directly by all other nodes. We assume that both the public and the secure channels are broadcast channels.

We consider N processors, $P_1, ..., P_N$, each with the value of a random variable V_i which takes values in the finite set $\mathcal{V}_i$. The processors, again, wish to exchange the values of their variables, using a deterministic protocol, while keeping the value of a binary function ε-secret. The definitions of a deterministic protocol and of ε-secrecy are simple extensions of the ones from the previous section and for the sake of brevity will not be repeated here. The objective, again, is to perform this ε-secret exchange with the minimum number of secret bits. We extend our protocol from the previous section in the following way: Processor P_1 forms its first partition of $\mathcal{V}_1$ into $\mathcal{V}'_1$ and $\mathcal{V}''_1$ using the same algorithm as the one from the previous section with m_0 subsets. It then uses one bit to inform the rest of the processors where V_1 lies. The processors proceed in order to form their first-level partitions based on the conditional distribution of their variables given the partitioning information of the other variables and using m_0 preliminary subsets. As in the previous protocol, second-level partitions are formed using m_1 subsets and ith-level partitions are formed using m_{i-1} subsets. Applying Propositions 2 and 3 from the Appendix, together with the repeated application of Eqs. (2)–(3) we can now express the amount of information disclosed about the value of F after i partitions as follows,

$$P_i \leqslant P_0 + 3KN \sum_{j=0}^{i-1} \left[\frac{2^j}{n - \sum_{l=0}^{j-1} m_l 2^l} + \sum_{j=0}^{i-1} \frac{N}{m_j} \right], \tag{6}$$

where P_0 is the prior probability of the function F being equal to 1, m_{i-1} is the number of subsets used in forming the ith partition, n is the cardinality of the ranges of the V_i's (assumed to be the same), and K is defined by:

$$K = \frac{\max_{v_1, ..., v_N} P(v_1, ..., v_N)}{\min_{v_1, ..., v_N} P(v_1, ..., v_N)}.$$

As before, the size of the ith subset can now be bounded by:

$$|\mathcal{V}_i| \leqslant \frac{n}{2^i} + \sum_{j=0}^{i-1} \frac{m_j}{2^{i-j}}.$$

Since the same holds for all the variables, the number of secret bits, a_i, that remain to be transmitted is N times the log of the above quantity; hence

$$a_i \leqslant N \log \left[\frac{n}{2^i} + \sum_{j=0}^{i-1} \frac{m_j}{2^{i-j}} \right]. \tag{7}$$

As before, to optimize the performance of the protocol we must choose the number of bits, i, and the m_i's, so that to minimize the number of secret bits, a_i, while maintaining F, ε-secret. This minimization is similar to that of the previous section. We note here that a_i is exactly $N/2$ times what it was in the previous section and the secrecy constraint is also $N/2$ times that of the previous section. We are therefore confronted with the same minimization problem as before, with a secrecy level $\varepsilon' = 2\varepsilon/N$. Since we concluded before that $2\log(K/\varepsilon) + 10$ secret bits were required, it immediately follows that in this case the requirement is for $N\log(K/\varepsilon') + 10 = N\log(KN/(2\varepsilon)) + 10$ bits.

4.2. Arbitrary Message Size Protocol

So far we have considered an alternating bit protocol for exchanging the values of X and Y. The desired secrecy level was maintained by each processor partitioning the range of its variables into two parts at each step and using one bit to reveal the subset containing the value of the variable. We wish to generalize our protocol to allow for exchanges of arbitrary size between the processors. To do so, we let each processor partition its range into a variable number of subsets at each step. Our model is the same as the two-processors model from the previous section. We begin by describing the protocol and proceed to show how such partitions are formed. Processor P_x starts by partitioning $\mathcal{X}$ into $2^{n_{x0}}$ subsets and then uses n_{x0} bits to disclose to P_y, whose subset contains XW. Upon receiving that information, processor P_y partitions $\mathcal{Y}$ into $2^{n_{y0}}$ subsets and uses n_{y0} bits to disclose which subset contains Y. In the next step, each processor partitions the range of its variale into $2^{n_{x1}}$ and $2^{n_{y1}}$ subsets, respectively. In the ith step they partition the range of their variables into $2^{n_{x(i-1)}}$ and $2^{n_{y(i-1)}}$ subsets. This process continues until the values of X and Y are disclosed. As before, the processors communicate over the public channel as long as they can maintain the ε-secrecy of F. When they can no longer exchange partitioning information over the public channel without violating the security requirement they switch over to the secret channel and complete their exchange. The partitions are formed according to a procedure described by the following proposition.

Let X be a random variable distributed over the range $\{1 \cdots n\}$ with PMF $P(X)$ and let $F(X)$ be a random binary function. That is, for all x, $F(x)$ is equal to either 0 or 1 with some probability. Also, let $P(F=1)$ be the prior probability of F being equal to one. That is

$$P(F=1) = \sum_{x \in \mathcal{X}} P(F=1 \mid x)\, P(x).$$

Then the following holds,

PROPOSITION 1. *$\mathcal{X}$ can be partitioned into J subsets $\{\mathcal{X}_0, \mathcal{X}_1, ..., \mathcal{X}_{J-1}\}$, such that*

$$\forall i \qquad |P(F=1 \mid \mathcal{X}_i) - P(F=1)| \leqslant \frac{3JK}{n} + \frac{2}{m},$$

where $K = (\max_x P(x))/(\min_x P(x))$ *and* m *is an arbitrary positive integer. Also*

$$\forall i \qquad \left| |\mathcal{X}_i| - \frac{n}{J} \right| \leqslant m.$$

Proof. We prove this proposition by construction of the subsets. As in the previous section we begin by partitioning $\mathcal{X}$ into m parts $\mathcal{X}^1, ..., \mathcal{X}^m$, where

$$\mathcal{X}^i = \left\{ x : \frac{m-i}{m} \leqslant P(F=1 \mid X=x) < \frac{m-i+1}{m} \right\}.$$

We then proceed to divide the elements of these subsets between the $\mathcal{X}_i$'s so that the probability of F being equal to one is approximately the same for all of these subsets. This is done as follows:

For all i, label the members of $\mathcal{X}^i$, $x_0^i, x_1^i, ..., x_{k_i-1}^i$, so that

$$P(x_0^i) \geqslant P(x_1^i), ..., \geqslant P(x_{k_i-1}^i),$$

where $K_i = |\mathcal{X}^i|$. Now form the partitions according to the following inductive set of rules:

Step 1. Let $\mathcal{X}_j$ contain members of $\mathcal{X}^1$ whose index mod(J) is equal to j.

Step i. Define $P_i(j) = P(\mathcal{X}_j \cap (\mathcal{X}^1 \cup \cdots \cup \mathcal{X}^i))$. Rank the elements of the collection $\{\mathcal{X}_0, \mathcal{X}_1, ..., \mathcal{X}_{J-1}\}$ in increasing order of $P_i(j)$ and let $R_i(j)$ be the rank of $\mathcal{X}_i$ in that list. That is,

$$R_i(k) > R_i(l) \leftrightarrow P_i(k) > P_i(l).$$

Now for all $i > 1$ let $\mathcal{X}_j$ contain members of $\mathcal{X}^i$ whose index mod(J) is equal to $R_i(j)$.

Continue until $i = m$.

The above algorithm attempts to divide the elements of each of the $\mathcal{X}^i$'s between the $\mathcal{X}_j$'s so that they all have approximately the same probability of F being equal to one. The proof of Proposition 1 follows directly from the above algorithm constructing the partition. The details of the proof are similar to those of Propositions 2 and 3 in the Appendix and are omitted here for brevity.

As stated earlier the processors partition their range into variable size partitions. These partitions are formed according to the rules of the above proposition. In the ith step, they partition their range into $2^{n_{x(i-1)}}$ and $2^{n_{y(i-1)}}$ subsets, respectively, using $m_{x(i)}$ and $m_{y(i)}$ preliminary subsets to form these partitions. It follows immediately from the proposition that if we let $\hat{\mathcal{X}}_i$ and $\hat{\mathcal{Y}}_i$ be the ith level subsets containing X and Y, then

$$|P(F=1 \mid \hat{\mathcal{X}}_i, \hat{\mathcal{Y}}_i) - P(F=1)| \leqslant \sum_{j=0}^{i-1} \left[\frac{3(2^{n_{xj}})\, K}{|\mathcal{X}_j|} + \frac{2}{m_{xj}} + \frac{3(2^{n_{yj}})\, K}{|\mathcal{Y}_j|} + \frac{2}{m_{yj}} \right],$$

where $|\mathscr{X}_j|$ and $|\mathscr{Y}_j|$ represent the size of the ith subsets containing X and Y and are expressed by

$$|\mathscr{X}_0| = |\mathscr{Y}_0| = n$$

and

$$\left| |X_i| - \frac{n}{2^{\sum_{j=0}^{i-1} n_{xj}}} \right| \leqslant \sum_{j=0}^{i-1} \frac{m_{xj}}{2^{\sum_{l=j+1}^{i-1} n_{xl}}}$$

and

$$\left| |Y_i| - \frac{n}{2^{\sum_{j=0}^{i-1} n_{yj}}} \right| \leqslant \sum_{j=0}^{i-1} \frac{m_{yj}}{2^{\sum_{l=j+1}^{i-1} n_{yl}}}.$$

We now consider two special cases of this protocol. The first corresponds to letting $n_{xj} = n_{yj} = 1$, for all j, in which case we obtain the alternating bit protocol of Section 3. Alternatively, we can let $i = 1$ and allow n_x and n_y to vary. This is the case where we limit the number of transmissions to one message by each processor, but allow the message to be arbitrarily long. Using the above results we obtain

$$|P(F=1 \mid \hat{\mathscr{X}}_1, \hat{\mathscr{Y}}_1) - P(F=1)| \leqslant \frac{3(2^{n_x})\,K}{n} + \frac{2}{m_x} + \frac{3(2^{n_y})\,K}{n} + \frac{2}{m_y},$$

and

$$\left| |\hat{\mathscr{X}}_1| - \frac{n}{2^{n_x}} \right| \leqslant m_x;$$

similarly

$$\left| |\hat{\mathscr{Y}}_1| - \frac{n}{2^{n_y}} \right| \leqslant m_y.$$

To optimize the performance of this protocol we want to choose n_x, n_y, m_x, and m_y to minimize the worst-case size of the subsets containing X and Y while maintaining the function ε-secret. That is, we need that we find

$$\min_{n_x, n_y, m_x, m_y} \{\log(|\mathscr{X}_1|) + \log(|\mathscr{Y}_1|)\}$$

such that

$$\frac{3(2^{n_x})\,K}{n} + \frac{2}{m_x} + \frac{3(2^{n_y})\,K}{n} + \frac{2}{m_y} \leqslant \varepsilon.$$

By symmetry we let $n_x = n_y$ and $m_x = m_y$ and require to find

$$\min_{n_x, m_x} \log\left(\frac{n}{2^{n_x}} + m_x\right) \tag{8}$$

such that

$$\frac{6(2^{n_x})\,K}{n}+\frac{4}{m_x}\leqslant\varepsilon. \tag{9}$$

The above minimization can be easily solved to obtain

$$m_x=\left\lceil \frac{4}{\varepsilon-\dfrac{6(2^{n_x})K}{n}} \right\rceil$$

and

$$n_x=\left\lfloor \log\left((0.1)\,\frac{\varepsilon n}{K}\right)\right\rfloor .$$

These results lead us to an upper bound on the number of secret bits that our protocol requires.

THEOREM 2. *Given any $\varepsilon>0$ and any binary function $F(X, Y)$, it is possible to keep the function ε-secret using no more than*

$$2\log\left(\frac{K}{\varepsilon}\right)+10$$

secret bits.

Proof. The proof follows immediately from Eqs. (8) and (9) when the optimal values of m_x and n_x are used.

Before we go on, we wish to consider a subcase of the above problem which will prove useful in the next section. Suppose that only P_x wants to send the value of X to P_y, without needing to know the value of Y. Clearly, P_x can use the above one message protocol, to send X, while keeping F ε-secret. If P_x used n_x public bits and m_x preliminary subsets, the difference between the eavesdropper's prior and posterior probabilities of F being equal to one is upper-bounded by $(3(2^{n_x})\,K)/n+(2/m_x)$ and the number of bits which remain to be transmitted over the secret channel is upperbounded by $\log((n/2^{n_x})+m_x)$. Using the optimal values of n_x and m_x obtained for the two-way protocol we arrive at the following corollary.

COROLLARY 1. *Given any $\varepsilon>0$ and any binary function $F(X,Y)$, P_x can transmit the value of X to P_y while keeping the function ε-secret using no more than*

$$\log\left(\frac{K}{\varepsilon}\right)+5$$

secret bits.

4.3. *Arbitrary Network Topology*

So far we have considered processors communicating over a broadcast channel. In many cases a broadcast channel is not available. We would like to be able to perform these secure exchanges over a network of arbitrary topology. In this section we consider one approach toward achieving this goal. We assume, as we must, that our network is connected. That is, every node in the network has a path to every other node. Also, since we are only concerned with minimizing the number of secret bits exchanged, we assume that every node can communicate with every other node over the public channel at no cost. This is equivalent to assuming that our public channel behaves as if it were a broadcast channel. Therefore, our restricted network topology applies only to the secret channel. In order to model a network of arbitrary topology we consider a fully connected network with various costs on the links. By varying the costs on the different links, we are able to create any desired network topology. Figure 6 shows a fully connected network with costs on the links. Note that setting the cost on a given link to infinity amounts to eliminating that link.

Clearly, if we can arrive at an optimal solution for the ε-secret exchange over this fully connected network, we will have obtained the optimal solution for a network of any topology by simply varying the costs on the links to reflect the desired topology. Since we no longer use a broadcast channel, we make the following alteration to the problem. Instead of requiring that all processors obtain the values of the variables of all other processors, we simply require that one processor obtains the values of all of the variables and computes F. This modification makes sense in view

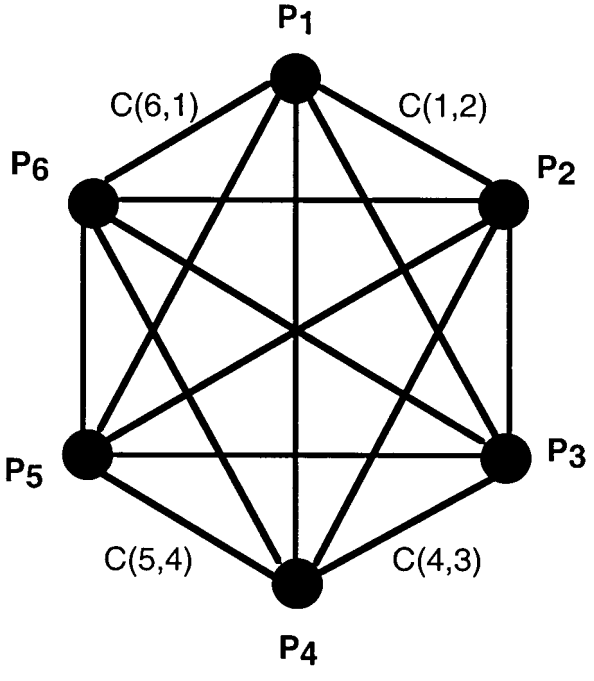

FIG. 6. Fully connected network with costs $C(i, j)$ on the link between nodes i and j.

of the fact that, in the absence of a broadcast channel, the requirement of a complete exchange will result in a significant additional cost, while not affecting the objective of computing F. We are therefore led to considering the following modified model.

We have N processors, $P_1, ..., P_N$, each with the value of a variable $V_i, ..., V_N$. Each processor has a secret channel to all the other processors. That is, processor P_i has a secret channel to P_j and the cost of transmitting a bit over that channel is $C_{(i,j)}$. The objective is that one processor, for simplicity say P_N, obtains the values of all the variables (and thereby the value of F) while keeping F ε-secret. If we were only interested in an exchange, each node would send its value to P_N independently. However, since we want to maintain the ε-secrecy of F, the nodes must coordinate their transmissions so that the combined effect of the information revealed by each node individually does not violate the secrecy requirement.

There may be many ways to achieve this objective. Here we describe one which uses the techniques developed in the previous section and which we believe to be efficient. Consider a path through the network which visits every node at least once. We call such a path a *spanning path* for the network. If every processor were to transmit the value of its variable along such a path terminating in P_N, P_N will obtain the values of all the variables. Consider such a spanning path and without loss of generality, assume that the newly visited nodes along this path are named $P_1, ..., P_N$ (e.g., the nodes are visited in order) and that the cost of transmitting from node i to node $i+1$, $C_{(i,i+1)}$, is simply denoted by C_i. This cost may include the cost of traversing multiple links as the path may visit nodes more than once, and therefore the path from node P_i to P_{i+1} may go through previously visited nodes and include multiple links. Now, clearly, the cost of sending a one-bit message from node one to node N along this path is equal to $\sum_{i=1}^{N-1} C_i$. Our objective is to have every node send the value of its variable to P_N along this path without violating the ε-secrecy requirement. The motivation is as follows: We showed in the previous section that one processor can transmit the value of its variable to another using no more than $\log(K/\varepsilon)+5$ secret bits, a constant number, which is independent of the size of the range of the variable. Once processor P_i obtains the values of $V_1, ..., V_{i-1}$ it can form a new variable combining the values of $V_1, ..., V_{i-1}$ with its own and disclose the value of this new variable to P_{i+1} using no more than $\log(K/\varepsilon)+5$ secret bits. At that point, P_{i+1} will have the values of $V_1, ..., V_i$ and proceed in a similar manner to inform P_{i+2}. Our protocol proceeds as follows: Processor P_1 uses a one-way protocol, as described in the previous section, to disclose the value of V_1 to P_2, while keeping the value of F, ε_1-secure with $\varepsilon_1 < \varepsilon$. To do this P_1 sends $\log(K/\varepsilon_1)+5$ secret bits to P_2, at a cost of $C_1(\log(K/\varepsilon_1)+5)$. Also, in doing so P_1 publicly discloses to everyone (including the intruder) that V_1 lies in a range $\mathscr{V}_1'$. Only P_2, which receives the secret bits, actually knows the exact value of V_1. At this point, P_2 can form a new variable, V_2', which lies in $\mathscr{V}_1' \times \mathscr{V}_2$. P_2 can now communicate the value of this new variable to P_3 while revealing less than ε_2 additional information about F, using no more than $\log(K/\varepsilon_2)+5$ secret bits, at a cost of $C_2(\log(K/\varepsilon_2)+5)$. The processors continue in this way until they disclose their values to P_N. The ε_i's are the amount of additional information that each processor is allowed to reveal about the value of

the function. The sum of these values must, of course, be less than ε in order to comply with the secrecy requirement. Therefore we wish to choose the ε_i's so that the number of secret bits is minimized. Specifically we want to find

$$\min_{\varepsilon_i' s} \sum_{i=1}^{N-1} C_i \left(\log \left(\frac{K}{\varepsilon_i} \right) + 5 \right),$$

so that

$$\sum_{i=1}^{N-1} \varepsilon_i \leqslant \varepsilon.$$

This minimization is trivial to perform using Lagrange multipliers. It can be shown that

$$\frac{C_i}{\varepsilon_i} = \frac{C_j}{\varepsilon_j} \; \forall i, j.$$

Therefore

$$\varepsilon_i = \left(\frac{C_i}{C_j} \right) \varepsilon_j$$

and consequently

$$\varepsilon_i = \frac{\varepsilon}{\sum_{j=1}^{N-1} \left(\frac{C_j}{C_i} \right)} = \frac{\varepsilon C_i}{\sum_{j=1}^{N-1} C_j}.$$

The cost of the secret bits, $C(\varepsilon)$, can be expressed in terms of C, the cost of sending one bit along the spanning path ($C = \sum_{j=1}^{N-1} C_j$), as,

$$C \log \left(\frac{KC}{\varepsilon} \right) + 5C - \sum_{i=1}^{N-1} C_i \log(C_i).$$

Clearly, the choice of a spanning path has a critical effect on the complexity of this exchange. Since in the above expression $C \log(KC/\varepsilon)$ is the dominant term, we wish to choose a spanning path which minimizes that term. Of course, that path would be the one which minimizes C or, in other words, the minimum weight spanning path. Of course, in this paper we are concerned only with the communication complexity and not with the complexity of the task of finding such a path.

The results of this section provide a first approach to solving the problem for a network of arbitrary topology. This approach yields an upper-bound on the number of secret bits required for any network. Although we attempt to minimize the communication complexity of this algorithm, we do not know if the algorithm itself is efficient and leave this question as an open problem for future work.

5. CONCLUSION

This paper expands upon a new problem in the area of communication complexity of distributed compution. This problem still has many unexplored, interesting aspects. The most obvious is an extension to nonbinary functions. In [Orlitski, 1984] it is shown that such an extension is possible. The results of this paper, as well as other works referenced in this paper, are worst-case results; that is, the protocols are designed to be optimal for a worst-case function. In reality, an actual function of interest could require far fewer bits than a worst-case function. It would therefore be helpful to identify possible functions of interest and design protocols which are optimal for those specific functions, or, alternatively, to consider average rather than worst-case functions. Finally, in this paper, we just touched upon the problem of performing these exchanges over a network of arbitrary topology. This problem needs to be studied in greater depth. To start, it would be interesting to identify a lower bound on the complexity of the exchange and, as was done in the two-processor case, devise a protocol which meets that bound. Since such an approach may prove to be overly ambitious, it may be interesting to consider some specific networks of interest which are commonly used for parallel computation, such as a hypercube or a mesh network.

APPENDIX

A.1. The Probability of F Being Equal to 1 Given $\mathscr{X}_1$

We compute the probability of F being equal to 1 given $\mathscr{X}_1$ as follows:

$$P(F=1 \mid \mathscr{X}_1)=\frac{P(F=1, \mathscr{X}_1)}{P(\mathscr{X}_1)}.$$

PROPOSITION 2. *The following inequality holds*

$$\frac{1}{2}-\frac{K}{2n} \leqslant P(\mathscr{X}_1) \leqslant \frac{1}{2}+\frac{K}{2n}.$$

Proof. Let $\mathscr{X}^i{}_{odd}$ denote the odd members of $\mathscr{X}^i$ and let $\mathscr{X}^i{}_{even}$ denote the even members of $\mathscr{X}^i$; then, for all i we have

$$P(\mathscr{X}^i{}_{odd})-P(\mathscr{X}^i{}_{even})=P(x_1^i)-(P(x_2^i)-P(x_3^i))-\cdots-(P(x_{k_i-1}^i)-P(x_{k_i}^i)) \leqslant P(x_1^i).$$

Note that only the first term in the long expression of differences above is positive. Also, in the above expression we implicitly assume that k_i is odd; however, the same inequality can be shown for k_i even. Next we proceed to upper-bound the probability of any single element x, thereby producing an upper bound on $P(x_1^i)$.

$$\begin{aligned} P(x) &= \sum_{y \in \mathscr{Y}} P(x, y) \leqslant |\mathscr{Y}| \max_{x, y} P(x, y) \\ &\geqslant |\mathscr{Y}| \min_{x, y} P(x, y) \end{aligned}$$

Therefore

$$\frac{\max_x P(x)}{\min_x P(x)} \leqslant K.$$

Now, since $\min_x P(x) \leqslant \frac{1}{n}$, we have that $\max_x P(x) \leqslant \frac{K}{n}$.

It should be also noted that the probability of the even members of $\mathscr{X}^i$ is never greater than that of the odd members; that is,

$$P(\mathscr{X}^i{}_{odd}) - P(\mathscr{X}^i{}_{even}) = (P(x_1^i) - P(x_2^i)) + (P(x_3^i) - P(x_4^i)) + \cdots + P(x_{k_i}^i) \geqslant 0.$$

Clearly, the above is true because all of the summands in the above expression are nonnegative. Therefore, for all i, we have

$$0 \leqslant P(\mathscr{X}^i{}_{odd}) - P(\mathscr{X}^i{}_{even}) \leqslant \frac{K}{n}.$$

We show next that for all i we have

$$\left| \sum_{l=1}^{i} P(\mathscr{X}^{i\prime}) - \sum_{l=1}^{i} P(\mathscr{X}^{i\prime\prime}) \right| \leqslant \frac{K}{n}. \tag{10}$$

This is done by induction on i. Clearly, it is true for $i = 1$. Assume now that it is also true for $i-1$. Then we consider two cases:

1. If $\sum_{l=1}^{i-1} P(\mathscr{X}^{l\prime}) \leqslant \sum_{l=1}^{i-1} P(\mathscr{X}^{l\prime\prime})$, the set $\mathscr{X}^{i\prime}$ contains odd members of $\mathscr{X}^i$ and the set $\mathscr{X}^{i\prime\prime}$ contains even members of $\mathscr{X}^i$; furthermore, since $0 \leqslant P(\mathscr{X}^i{}_{odd}) - P(\mathscr{X}^i{}_{even}) \leqslant \frac{K}{n}$, the desired inequality holds for i as well.

2. If $\sum_{l=1}^{i-1} P(\mathscr{X}^{l\prime}) > \sum_{l=1}^{i-1} P(\mathscr{X}^{l\prime\prime})$ the set $\mathscr{X}^{i\prime\prime}$ contains odd members of $\mathscr{X}^i$ and the set $\mathscr{X}^{i\prime}$ contains even members of $\mathscr{X}^i$ and by a similar argument it is clear that the inequality is true for i as well. Thus the induction step is valid.

Since inequality (10) holds for all values of i, we obtain

$$|P(\mathscr{X}') - P(\mathscr{X}'')| = \left| \sum_{l=1}^{m_0} P(\mathscr{X}^{i\prime}) - \sum_{l=1}^{m_0} P(\mathscr{X}^{i\prime\prime}) \right| \leqslant \frac{K}{n}.$$

In addition, we have $P(\mathscr{X}') + P(\mathscr{X}'') = 1$.

Adding and subtracting the above two equations yields

$$\frac{1}{2} - \frac{K}{2n} \leqslant P(\mathscr{X}') \leqslant \frac{1}{2} + \frac{K}{2n}$$

and

$$\frac{1}{2} - \frac{K}{2n} \leqslant P(\mathscr{X}'') \leqslant \frac{1}{2} + \frac{K}{2n}.$$

Now, since $\mathscr{X}_1$ is equal to either $\mathscr{X}'$ or $\mathscr{X}''$ we have proved our proposition.

PROPOSITION 3. *The following inequality holds*:

$$\frac{P(F=1)}{2}-\frac{K}{2n}-\frac{1}{4m_0}\leqslant P(F=1, \mathscr{X}_1)\leqslant\frac{P(F=1)}{2}+\frac{K}{2n}+\frac{1}{4m_0}.$$

Proof. We have

$$P(F=1, \mathscr{X}')=\sum_{i=1}^{i=m_0} P(F=1, \mathscr{X}', \mathscr{X}^i)=\sum_{i=1}^{i=m_0} P(F=1 \mid \mathscr{X}', \mathscr{X}^i)\times P(\mathscr{X}', \mathscr{X}^i).$$

By the definition of $\mathscr{X}^i$, we know that

$$\forall x\in\mathscr{X}^i \qquad \frac{m_0-i}{m_0}\leqslant P(F=1 \mid x)<\frac{m_0-i+1}{m_0}.$$

Therefore

$$\sum_{i=1}^{i=m_0}\left(\frac{m_0-i}{m_0}\right)\times P(\mathscr{X}', \mathscr{X}^i)\leqslant P(F=1, \mathscr{X}')<\sum_{i=1}^{i=m_0}\left(\frac{m_0-i+1}{m_0}\right)\times P(\mathscr{X}', \mathscr{X}^i).$$

Similarly, we can obtain an inequality involving $\mathscr{X}''$; i.e.,

$$\sum_{i=1}^{i=m_0}\left(\frac{m_0-i}{m_0}\right)\times P(\mathscr{X}'', \mathscr{X}^i)\leqslant P(F=1, \mathscr{X}'')<\sum_{i=1}^{i=m_0}\left(\frac{m_0-i+1}{m_0}\right)\times P(\mathscr{X}'', \mathscr{X}^i)$$

which yields

$$P(F=1, \mathscr{X}')-P(F=1, \mathscr{X}'')\leqslant\sum_{i=1}^{i=m_0}[P(\mathscr{X}', \mathscr{X}^i)-P(\mathscr{X}'', \mathscr{X}^i)]\times\left(\frac{m_0-i}{m_0}\right)+\left(\frac{P(\mathscr{X}')}{m_0}\right).$$

Let $\Delta_i=P(\mathscr{X}', \mathscr{X}^i)-P(\mathscr{X}'', \mathscr{X}^i)$. We show first that

$$\sum_{i=1}^{i=m_0}\Delta_i\times\left(\frac{m_0-i}{m_0}\right)\leqslant\left(\frac{m_0-1}{m_0}\right)\times\frac{K}{n}. \tag{11}$$

Proof is by induction on m_0. Inequality (11) clearly holds for $m_0=1$; assume that it is true for $m_0=j-1$ and show that it must then hold for $m_0=j$. By the induction hypothesis we have

$$\sum_{i=1}^{i=j-1}\Delta_i\times\left(\frac{j-i-1}{j-1}\right)\leqslant\left(\frac{j-2}{j-1}\right)\frac{K}{n}.$$

Noting that the $(j-1)$st term of this sum is equal to 0 we have

$$\sum_{i=1}^{i=j-2}\Delta_i\times\left(\frac{j-i-1}{j-1}\right)\leqslant\left(\frac{j-2}{j-1}\right)\frac{K}{n}.$$

Multiplying both sides of the inequality by $(j-1/j)$ we obtain

$$\sum_{i=1}^{i=j-2} \Delta_i \times \left(\frac{j-i-1}{j}\right) \leqslant \left(\frac{j-2}{j}\right)\frac{K}{n}$$

which in turn yields

$$\sum_{i=1}^{i=j-2} \Delta_i \times \left(\frac{j-i}{j}\right) \leqslant \left(\frac{j-2}{j}\right)\frac{K}{n} + \sum_{i=1}^{i=j-2} \Delta_i \times \left(\frac{1}{j}\right).$$

Now, when $m_0 = j$ we have

$$\sum_{i=1}^{i=j} \Delta_i \times \left(\frac{j-i}{j}\right) = \sum_{i=1}^{i=j-2} \Delta_i \times \left(\frac{j-i}{j}\right) + \left(\frac{\Delta_{j-1}}{j}\right).$$

Using our induction hypothesis yields

$$\sum_{i=1}^{i=j} \Delta_i \times \left(\frac{j-i}{j}\right) \leqslant \left(\frac{j-2}{j}\right)\frac{K}{n} + \sum_{i=1}^{i=j-2} \Delta_i \times \left(\frac{1}{j}\right) + \frac{\Delta_{j-1}}{j}$$

$$= \left(\frac{j-2}{j}\right)\frac{K}{n} + \sum_{i=1}^{i=j-1} \Delta_i \times \left(\frac{1}{j}\right) \leqslant \left(\frac{j-1}{j}\right) \times \frac{K}{n}.$$

This is because, as was shown in the proof of Proposition 2, for all j, we have that $\sum_{i=1}^{i=j} \Delta_i \leqslant \frac{K}{n}$. So, we have shown that

$$P(F=1, \mathscr{X}') - P(F=1, \mathscr{X}'') \leqslant \left(\frac{m_0-1}{m_0}\right) \times \frac{K}{n} + \left(\frac{1}{m_0}\right) P(\mathscr{X}').$$

Finally, we upper-bound $P(\mathscr{X}')$ according to Proposition 2 and obtain.

$$P(F=1, \mathscr{X}') - P(F=1, \mathscr{X}'') \leqslant \frac{K}{n} + \left(\frac{1}{2m_0}\right).$$

Combining the above inequality with the fact that $P(F=1, \mathscr{X}') + P(F=1, \mathscr{X}'') = P(F=1)$ we obtain Proposition 3.

Now we are ready to upper-bound the probability of F being equal to 1 given $\mathscr{X}_1$. We do this by using the upper bound from Proposition 3 for the numerator and the lower bound from Proposition 2 for the denominator; that is,

$$P(F=1 \mid \mathscr{X}_1) = \frac{P(F=1, \mathscr{X}_1)}{P(\mathscr{X}_1)} \leqslant \frac{\dfrac{P(F=1)}{2} + \dfrac{K}{2n} + \dfrac{1}{4m_0}}{\dfrac{1}{2} - \dfrac{K}{2n}}.$$

Similarly we obtain a lower bound on the above probability and by simple algebraic manipulation we obtain

$$|P(F=1 \mid \mathscr{X}_1) - P(F=1)| \leqslant \frac{3K}{n-K} + \frac{n}{2m_0(n-K)}.$$

If we restrict our values of K to be less than $(\frac{n}{3})$,[2] we obtain the following simplified expression,

$$|P(F=1 \mid \mathscr{X}_1) - P(F=1)| \leqslant \frac{3K}{n} + \frac{1}{m_0}.$$

A.2. The Probability of F Being Equal to 1 Given $\mathscr{X}_1$ and $\mathscr{Y}_1$

The probability of F being equal to 1 given $\mathscr{X}_1$ and $\mathscr{Y}_1$ is computed as follows,

$$P(F=1 \mid \mathscr{X}_1, \mathscr{Y}_1) = \frac{P(F=1, \mathscr{Y}_1 \mid \mathscr{X}_1)}{P(\mathscr{Y}_1 \mid \mathscr{X}_1)}.$$

As before, we bound this probability by upper- and lower-bounding the numerator and denominator, respectively. First we prove a basic property of the probability density function of Y given $\mathscr{X}_1$.

PROPOSITION 4. *The following inequality holds*

$$\max_y P(y \mid \mathscr{X}_1) \leqslant \frac{K}{n}.$$

Proof. We have

$$P(y \mid \mathscr{X}_1) = \frac{\sum_{x \in \mathscr{X}_1} P(y, x)}{P(\mathscr{X}_1)} \leqslant \frac{|\mathscr{X}_1| \max_{x,y} P(x, y)}{P(\mathscr{X}_1)}$$

$$\geqslant \frac{|\mathscr{X}_1| \min_{x,y} P(x, y)}{P(\mathscr{X}_1)}.$$

Therefore,

$$\frac{\max_y P(y \mid \mathscr{X}_1)}{\min_y P(y \mid \mathscr{X}_1)} \leqslant \frac{\max_{x,y} P(x, y)}{\min_{x,y} P(x, y)} \leqslant K.$$

Now, since $\min_y P(y \mid \mathscr{X}_1) \leqslant \frac{1}{n}$, the proposition has been proved.

[2] We know from the lower-bound of the previous section that when $K = \Omega(n)$ essentially all of the bits have to be sent over the secret channel; therefore, this restriction does not limit the generality of the protocol.

Using the above result, and following the proofs of Propositions 2 and 3, we obtain the following equations:

$$\frac{1}{2}-\frac{K}{2n}\leqslant P(\mathscr{Y}_1 \mid \mathscr{X}_1)\leqslant\frac{1}{2}+\frac{K}{2n}$$

and

$$\left|P(F=1, \mathscr{Y}_1 \mid \mathscr{X}_1)-\frac{P(F=1 \mid \mathscr{X}_1)}{2}\right|\leqslant\frac{K}{2n}+\frac{1}{4m_0}.$$

Hence

$$|P(F=1 \mid \mathscr{X}_1, \mathscr{Y}_1)-P(F=1 \mid \mathscr{X}_1)|\leqslant\frac{3K}{n}+\frac{1}{m_0}.$$

Received December 29, 1996

REFERENCES

Bertsekas, D. P. (1982), "Constrained Optimization and Lagrange Multiplier Methods," Academic Press, New York.

Modiano, E., and Ephremides, A. (July 1992), Communication complexity of secure distributed computation in the presence of noise, *IEEE Trans. Inform. Theory*.

Orlitsky, A., and El Gamal, A. (April 1984), Communication with secrecy constraints, *in* "Proc. of the 16th Annual ACM Symposium on the Theory of Computing, Atlanta, GA," pp. 217–224.

Orlitsky, A., and El Gamal, A. (January 1990), Average and randomized communication complexity, *IEEE Trans. Inform. Theory*.

Yao, A. C. (May 1979), Some complexity questions related to distributed computing, *in* "Proc. of the 11th Annual Symposium on the Theory of computing, Washington D.C.," pp. 209–213.