Amortizing Randomness in Private Multiparty Computations*

Eyal Kushilevitz† Rafail Ostrovsky‡ Adi Rosén§

Abstract

We study the relationship between the number of rounds needed to repeatedly perform a private computation (i.e., where there are many sets of inputs sequentially given to the players on which the players must compute a function privately) and the overall randomness needed for this task. For the XOR function, we show that by re-using the same ℓ random bits we can significantly speedup the round-complexity of each computation compared to what is achieved by the naive strategy of partitioning the ℓ random bits between the computations. Moreover, we prove that our protocols are optimal in the amount of randomness they require.

1 Introduction

A very basic question in the theory of computation is the *direct-sum* question: Can the complexity of solving k independent instances of a problem be smaller than the cost of independently solving the k instances? This general question was studied in various scenarios and with respect to various complexity measures, e.g., in [10, 20, 21, 23, 26, 38, 41]. To answer such a question, one typically needs to consider a problem whose complexity in the single instance case is reasonably well understood.

In this work we consider a direct sum question related to the randomness complexity of private multiparty protocols. A *1-private* (or simply, *private*) protocol $\mathcal{A}$ for computing a function f is a protocol that allows n players, P_i, $1 \leq i \leq n$, each possessing an individual secret input, x_i, to compute the value of $f(\vec{x})$ in a way that no *single* player learns about the initial inputs of other players more than what is revealed by the value of $f(\vec{x})$ and its own input[1]. Private computations in this setting were the subject of a considerable amount of work, e.g., [5, 13, 2, 3, 15, 16, 17, 18, 21, 31, 35]. In this paper, we consider this setting for the basic XOR function, and show quite unexpected results relating the rounds complexity and the randomness complexity of such computations.

*An early version of this paper appeared in the Proc. of the 17th PODC conference, 1998, pp. 81-90.

†Dept. of Computer Science, Technion, Haifa, Israel. e-mail: eyalk@cs.technion.ac.il; Part of this research was done while visiting ICSI Berkeley. Supported by MANLAM Fund. URL: http://www.cs.technion.ac.il/~eyalk

‡Bell Communications Research, MCC-1C365B, 445 South Street, Morristown, New Jersey 07960-6438; e-mail: rafail@bellcore.com

§Dept. of Computer Science, University of Toronto, Toronto, Canada. Part of this work was done while with the Dept. of Computer Science, Tel-Aviv University, Tel-Aviv, Israel, and while visiting ICSI Berkeley. e-mail: adiro@cs.toronto.edu

[1] In the literature a more general definition of t-privacy is given. The above definition is the case $t = 1$.

Randomness is an important resource in computation. As a result, various methods for saving in randomness were studied [1, 6, 14, 19, 25, 27, 28, 29, 36, 37, 40, 42, 43, 44]. In addition, there was a quantitative study of the role of randomness in specific contexts, e.g., [39, 34, 4, 11, 8, 9]. One such case is the study of randomness in private multiparty computations [7, 12, 30, 32, 33, 22]. In particular, in [7, 33, 30] the amount of randomness required for private computations of XOR (the exclusive-or function) was considered (this function was the subject of previous research in the area of privacy due to its being a basic linear operation and its relative simplicity [21, 16]).

In this paper we also concentrate on the XOR function. Previously, the following was known for privately computing XOR on a single instance (that is, where each player has a single input bit):

- there is no *deterministic* solution for the problem;
- with a single random bit the problem requires $\Theta(n)$ rounds (time) [33];
- with $\ell \geq 2$ random bits the problem requires $\Theta(\frac{\log n}{\log \ell})$ rounds [33, 22].

None of the above mentioned works addressed multiple inputs.[2] We study the round-complexity question in such multiple input settings.

Our Results: Before we make a precise statement of our general result, we start with a statement of a somewhat weaker version of our result which is simpler to state (and still is quite surprising): let us consider the case where n players are sequentially given n sets of inputs, of a single bit each, and for each such set the players wish to privately compute the XOR of these bits. In this case, by [33], if the players use only a single independent random bit for each set of inputs, they can compute all n xors, using $n/2$ rounds for each set of inputs. It is impossible to compute the XOR's privately using the single random bit (per computation) with less than $\Omega(n)$ rounds per computation. If the players wish to compute n independent XOR's with $O(1)$ rounds per computation, and using independent random bits for each computation, they will need at least $\Omega(n^{1+\varepsilon})$ random bits overall (i.e. at least $\Omega(n^{\varepsilon})$ for each input set) [22]. Surprisingly, we show how to re-use the *same* $O(n)$ random bits for all computations and achieve optimal rounds-performance each time, i.e., each computation will be performed in $O(1)$ rounds. Moreover, we accompany this result by a corresponding lower bound showing that in order to privately compute the n XOR's the players need $\Omega(n)$ random bits regardless of the number of rounds. Thus, using the minimum number of random bits possible, we achieve by a "recycling" procedure the optimal round-complexity.

More generally, we consider the setting where the players are sequentially given k input bits each, and the goal is to sequentially compute k times the XOR function immediately after the input bits for each set are provided. First, we prove that no solution exists which uses less than $(1-\frac{2}{n})k$ random bits (hence, generalizing the claim that in the single input case there is no deterministic solution for $n \geq 3$). If we have $\ell \geq k$ random bits, the naive solution would be to partition these bits into sets of $d \cong \ell/k$ random bits and to use the best single-input solution with d random bits for each sequential input. For example, if $\ell = k$ then $d = 1$ and this solution requires $\Theta(n)$ rounds, and if $\ell = \Theta(k)$ then $d = O(1)$ and this solution takes $\Theta(\log n)$ rounds [33, 22]. We present much better solutions than the above. In particular, we show that if $\ell = k = c \cdot (n-1)$ the problem can be solved in $O(1)$ rounds per computation (rather than $\Theta(n)$), and for $\ell = O(k)$ we can solve the problem in $O(\frac{\log n}{\log k})$ rounds (rather than $\Theta(\log n)$).

[2] Amortization of the *communication complexity* in private computation of XOR was shown in [21].

Organization: In Section 2 we provide the required definitions, including the model and the definition of privacy. In Section 3 we present our protocols that show our technique for recycling the random bits. Section 4 includes the lower bound which is technically more involved.

2 Preliminaries

In this section we give a description of the protocols we consider, and define the *privacy* property of protocols as well as the required complexity measures.

A set of n players P_i $(1 \leq i \leq n)$, each receiving sequentially k input bits $x_i^1, \ldots, x_i^k$ (known *only* to it), collaborate in a protocol to compute the k values

$$\begin{aligned} &\textsc{xor}(x_1^1, x_2^1, \ldots, x_n^1) \\ &\textsc{xor}(x_1^2, x_2^2, \ldots, x_n^2) \\ &\vdots \\ &\textsc{xor}(x_1^k, x_2^k, \ldots, x_n^k). \end{aligned}$$

(In general, we may be interested in computing any function f.) More specifically, a protocol works in k *phases*. In phase j each player P_i gets the input bit x_i^j. Then, the players have to compute the j'th value $\textsc{xor}(x_1^j, x_2^j, \ldots, x_n^j)$ and only after this computation is completed they get the $(j+1)$-st input bit. The computation in each phase operates in *rounds*. In each round each player P_i may toss some coins, and then sends messages to the other players (messages are sent over private channels so that other than the intended receiver no other player can listen to them). Player P_i then receives the messages sent to it by the other players. The content of these messages may depend on all the information available to P_i: its input (in the current and the previous phases), its random coins and the messages it received so far (in the current and the previous phases). Each player P_i receives during the execution of the protocol a sequence of messages C_i. We denote by C_i^j those messages sent to P_i during phase j. We also use x_i to denote the input seen by P_i during the whole protocol, i.e. $x_i = x_i^1, \ldots, x_i^k$, and $\vec{x}$ to denote the vector of inputs seen by all players, i.e. $\vec{x} = (x_1, \ldots, x_n)$. We denote by x^j the vector of inputs received by all players in the j-th phase, i.e. $x^j = (x_1^j, \ldots, x_n^j)$. Finally, we use $f^k(\vec{x})$ to denote the vector of the k outputs of the protocol, i.e. $f^k(\vec{x}) = (f(x^1), \ldots, f(x^k))$.

Informally, *privacy* with respect to player P_i means that player P_i cannot learn anything (in particular, the inputs of the other players) from C_i, except what is implied by its input bits, and the value of the function computed. Formally,

Definition 1: (Privacy) A (k phase) protocol $\mathcal{A}$ for computing a function f is *private* with respect to player P_i if for any two input vectors $\vec{x}$ and $\vec{y}$, such that $f^k(\vec{x}) = f^k(\vec{y})$ and $x_i = y_i$, for any sequence of messages C, and for any random coins, R_i, tossed by P_i,

$$\Pr[C_i = C | R_i, \vec{x}] \quad = \quad \Pr[C_i = C | R_i, \vec{y}],$$

where the probability is over the random coin tosses of *all other* players.
A protocol is called *private* if it is private with respect to every P_i.

Due to the nature of k-phase protocols, it might be convenient to consider each "piece" C_i^j of the communication separately. That is, given all the information available to P_i when phase j starts, the communication in that phase does not provide any additional information (other than what is implied by its input bits and the value of the function computed). While the two definitions are equivalent the second definition is sometimes easier to use.

Definition 2: (Privacy – 2nd variant) A (k phase) protocol $\mathcal{A}$ for computing a function f is *private* with respect to player P_i if for every phase j ($1 \leq j \leq k$) for any two input vectors $\vec{x}$ and $\vec{y}$, such that $f(x^j) = f(y^j)$ and $x_i = y_i$, for any sequence of messages C, for any history $C_i^1, \ldots, C_i^{j-1}$, and for any random coins, R_i, tossed by P_i,

$$\Pr[C_i^j = C | R_i, \vec{x}, C_i^1, \ldots, C_i^{j-1}] \;=\; \Pr[C_i^j = C | R_i, \vec{y}, C_i^1, \ldots, C_i^{j-1}],$$

where the probability is over the random coin tosses of *all other* players.
A protocol is called *private* if it is private with respect to every P_i.

To measure the amount of randomness used by a protocol and its round complexity we give the following definitions:

Definition 3: An ℓ-random protocol is a protocol such that for every input assignment $\vec{x}$, the total number of coins tossed by all players in *every* execution (during all phases) is at most ℓ.

Definition 4: An r-round protocol is a protocol such that for every input assignment $\vec{x}$, and every sequence of coin tosses the number of rounds in each phase j is at most r.

We emphasize that the definitions allow, for example, that in different executions different players will toss the coins. This may depend both on the input of the players, and on the previous coin tosses.

3 Upper Bound

In this section we present our positive results. First, we consider the case $k = n - 1$. By the lower bound of Section 4, at least $n - 2$ random bits are needed for such a computation, regardless of the number of rounds per phase. The protocol below uses $n - 1$ random bits. Of course, there is a naive way to perform this computation using only $n - 1$ random bits, which is to use a single random bit for each of the $n - 1$ phases. However, such computation takes $\Theta(n)$ rounds per phase and by [33] this is the best one can do with a single random bit (per phase). Our protocol takes a different direction that allows it to use only $O(1)$ rounds per phase.

Lemma 1: There is a private, k-phase protocol that computes XOR on $k = n - 1$ inputs with $\ell = n - 1$ random bits and $r = O(1)$ rounds per phase.

Proof: For the proof we present an appropriate k-phase protocol.

Initialization: Player P_n chooses $n - 1$ random bits, denoted $r_1, \ldots, r_{n-1}$. It sends bit r_i to player P_i. (This step can be performed as part of phase 1.)

Phase j:

1. Each player P_i, $1 \leq i \leq n-1$, sends a bit $b_{i,j} = x_i^j + r_i$ to player P_j.
   In addition, player P_n sends to P_j the bit $b_{n,j} = x_n^j + \sum_{m=1}^{n-1} r_m$. (The summations here and elsewhere are all modulo 2.)

2. Player P_j sums the n bits $b_{i,j}$ it received in the previous step. It announces $\sum_{i=1}^{n} b_{i,j}$ as the output for the j-th phase.

For the correctness, consider the sum computed by player P_j in Step (2). This sum equals:

$$\begin{aligned} \sum_{i=1}^{n} b_{i,j} &= \sum_{i=1}^{n-1}(x_i^j + r_i) + (x_n^j + \sum_{m=1}^{n-1} r_m) \\ &= \text{XOR}(x_1^j, x_2^j, \ldots, x_n^j). \end{aligned}$$

For the privacy, note that P_n receives no message during the protocol (except the output values), hence the privacy with respect to P_n certainly holds. Also, observe that during phase j only player P_j receives any message (other than the output of the phase). Finally, C_j^j (the communication received by P_j in phase j) is a sequence of messages $b_{1,j}, \ldots, b_{n-1,j}, b_{n,j}$. Note that for every input $\vec{x}^j = (x_1^j, x_2^j, \ldots, x_n^j)$, every communication $r_j, c_1, \ldots, c_{n-1}, c_n$ which is consistent with the output has the same probability, $2^{-(n-1)}$. This is because each of $c_1, \ldots, c_{n-1}$ determines one of the $n-1$ random bits (which are all independent) and c_n (and r_j) are determined by the value of the function and the previously determined values. The privacy of the protocol follows. □

The main idea in the above protocol is that we can compute the XOR of each of the $n-1$ inputs, using $n-1$ random bits, in a way that allows to use the same $n-1$ random bits for all the $n-1$ inputs. We can use the same idea, with a bit more of precaution, to do it for other parameters. The following is a simple corollary of the previous construction.

Lemma 2: There is a private, k-phase protocol that computes XOR on $k = d(n-1)$ inputs with $\ell = d(n-1)$ random bits and $r = O(1)$ rounds.

Proof: Simply partition the $d(n-1)$ inputs into d sets of size $n-1$. For each set of $n-1$ inputs use the $(n-1)$-phase protocol of Lemma 1 that requires $n-1$ random bits and $O(1)$ rounds. If we do this each time with new and independent random bits, we get the desired result. □

Again note that, by the results of Section 4, the above lemma is (almost) optimal in terms of the number of random bits required for this computation. Moreover, if k (the number of inputs) is not divisible by $n-1$ we can always add some dummy inputs to each player so as to make the number of inputs be some k' which is divisible by $n-1$. For example, if $\frac{n-1}{2} \leq k < n-1$, then $\ell = n-1$ random bits and $r = O(1)$ rounds suffice. The only case where this is inefficient is when $k \ll n-1$; in such a case increasing the number of inputs to $k' = n-1$ would be wasteful. For such cases, we use the following construction:

Lemma 3: Let s be an integer ($1 \leq s \leq n$). There is a private, k-phase protocol that computes XOR on k inputs, $k < (n-1)/2$ with $\ell = 2k + s$ random bits and $r \leq \log n / \log s$ rounds per computation.

Proof: As in the previous protocols, P_n will be the player that makes the random choices. We partition the other $n-1$ players into g groups of size k. Assume for simplicity that $n-1$ is divisible by k. Moreover, assume that $g = (n-1)/k$ is even (later we describe the modifications required when this is not the case).

Initialization: Player P_n chooses k random bits, denoted $r_1, \ldots, r_k$. It sends the bit r_i to the i-th player of each of the g groups. P_n chooses additional s bits $\alpha_1, \ldots, \alpha_s$ to be used later.

Phase j:

1. The i-th player in each group t ($1 \leq i \leq k$, $1 \leq t \leq g$), denoted $P_{i,t}$, sends a bit $b^j_{i,t} = x^j_{i,t} + r_i$ to player $P_{j,t}$ (where $x^j_{i,t}$ denotes the input of $P_{i,t}$ in the current phase, j).

2. Each player $P_{j,t}$ computes $Y^j_t = \sum_{i=1}^{k} b^j_{i,t}$.

3. The j-th players of all groups together with P_n involve in a private protocol to compute the sum of $g+1$ bits: $Y^1_j, \ldots, Y^g_j$ and x^j_n. They announce the output as the XOR of the j-th input. The players do this computation using the protocol of [33]. This protocol, when using s random bits, terminates within $\log(g+1)/\log s$ rounds. In addition, all the random bits in this protocol are chosen by one player which other than that receives no message during the protocol; we choose this player to be P_n and $\alpha_1, \ldots, \alpha_s$ to be these random bits (note that P_n uses the *same* s random bits in all k phases).

For the correctness note that

$$\begin{aligned}
\sum_{t=1}^{g} Y^t_j \;+\; x^j_n \;&=\; \sum_{t=1}^{g}\sum_{i=1}^{k} b^j_{i,t} + \; x^j_n \\
&=\; \sum_{t=1}^{g}\sum_{i=1}^{k} (x^j_{i,t} + r_i) + \; x^j_n \\
&=\; \sum_{i=1}^{n} x^j_i + g \cdot (\sum_{i=1}^{k} r_i).
\end{aligned}$$

Since we assumed that g is even, the last term contributes 0 to the sum (modulo 2) and so the j-th output is $\sum_{i=1}^{n} x^j_i$, as needed. The number of random bits used is $k+s$.

For the privacy, note again that P_n only sends messages during the whole protocol and that in phase j only the j-th player of each group receives messages. Each of the players $P_{j,t}$ receives in Step 1 from the members of its group, k bits $b^j_{1,t}, \ldots, b^j_{k,t}$ which are distributed uniformly and independently. Then, it involves in a private protocol which guarantees that no matter what is the input to the protocol each player sees the same distribution of communications, for all the possible inputs that $P_{j,t}$ may have, and all possible outputs. (Note that the s random bits used for this sub-protocol are independent of the random bits used in Step 1.) Altogether, the privacy follows.

Now, there are some technical issues that we still need to take care of. First, if the number of groups g is odd then P_n can always make sure that there will be no contribution of random bits to the result by using as its input in Step 3 of phase j the bit x^j_n xored with these random bits. Another technical issue that has to be dealt with is the case that $n-1$ is not divisible by k. In this

case the g-th group is of size $k' < k$ and hence cannot use the above protocol. We solve this by letting P_n choose for them k' special random bits. The messages $b^j_{i,g}$ will be sent to the j-th player of group 1 instead of the j-th player of group g (who may not exist). Since this is done with new random bits the privacy still holds, and the total number of random bits is still at most $2k + s$. □

Combining Lemma 2 and Lemma 3 we get the following theorem:

Theorem 4: Let s be an integer ($1 \leq s \leq n$). There is a private k-phase protocol that computes XOR on $k = d(n-1)+q$ inputs, with $\ell = d(n-1)+2q+s$ random bits and $r \leq \log n / \log s$ rounds.

An interesting case is the following:

Corollary 5: There is a private, k-phase protocol that computes XOR on k inputs, with $\ell = O(k)$ random bits and $r \leq \log n / \log k$ rounds per phase.

This means that we can use $O(k)$ random bits overall, and compute the function is each phase in the optimal time for computing XOR on a single input using k bits (i.e., $O(\log n / \log k)$ rounds [33, 22]).

4 Lower Bound

In this section we prove a lower bound on the number of random bits required for a k-phase, private computation of XOR (on k instances). This lower bound holds for any number of rounds used by the protocol. We prove the following theorem.

Theorem 6: Let A be a d-random, k-phase, n-player, private protocol to compute XOR. Then $d \geq (1 - \frac{2}{n}) \cdot k$.

Before we proceed, we give a technical proposition and a technical definition which will be used in the proof.

Claim 1: For any non-negative values $a_{i,j}$ ($1 \leq j \leq q$, $1 \leq i \leq p$),

$$\prod_{j=1}^{q} (\sum_{i=1}^{p} a_{i,j}) \geq p^q \min_{1 \leq i \leq p} \{\prod_{i=1}^{q} a_{i,j}\}$$

Proof:

$$\begin{aligned}
\prod_{j=1}^{q} \sum_{i=1}^{p} a_{i,j} &= p^q \prod_{j=1}^{q} \left(\frac{\sum_{i=1}^{p} a_{i,j}}{p} \right) \\
&\geq p^q \prod_{j=1}^{q} \left(\prod_{i=1}^{p} a_{i,j} \right)^{\frac{1}{p}} \\
&= p^q \left(\prod_{i=1}^{p} \prod_{j=1}^{q} a_{i,j} \right)^{\frac{1}{p}} \\
&\geq p^q \min_{1 \leq i \leq p} \left(\prod_{j=1}^{q} a_{i,j} \right) ,
\end{aligned}$$

where the first inequality uses the theorem of the arithmetic and geometric means (cf. [24] p. 17). □

Definition 5: Denote by $View_i^t(\vec{x}, \vec{R})$ the view of player P_i at time t on input $\vec{x}$ and vector of random tapes $\vec{R}$. This view consists of the inputs to player P_i received so far, the random tape of player P_i, i.e. R_i, and the communication received by player P_i until, and including, time $t-1$.

In our proof we argue about the number of different views that players can see in various executions of the protocol. The following lemmas are useful for this argument.The proof of this theorem is based on the following two lemmas. The first lemma (Lemma 7) is very similar to a lemma in [33] and its proof appears here mainly for self-containment.

Lemma 7: [33] Consider a private d-random, k-phase, protocol to compute a boolean function f. Fix the random tapes of the players to be $\vec{R}$. Then, for any P_i, the view $View_i^t(\vec{y}, \vec{R})$ can assume at most 2^{2k+d} different values (over the 2^{kn} values of $\vec{y}$).

Proof: In the first step of the proof, we fix an arbitrary input $\vec{x}$ and consider the possible values of $View_i^t(\vec{x}, \vec{R})$ over all different choices of random tapes $\vec{R} = (R_1, \ldots, R_n)$. The d-randomness of the protocol implies that the total number of coins tossed is at most d; however, in different executions these coins can be tossed by different players. Nevertheless, we claim that the number of different values that $View_i^t(\vec{x}, \vec{R})$ can assume is at most 2^d. For each execution we can order the coin tosses of all players (i.e., the readings from the local random tapes) according to the phases of the protocol, within each phase according to the rounds, and within each round according to the index of the players that toss them. The identity of the player to toss the first coin is fixed by $\vec{x}$. The identity of the player to toss any next coin is determined by $\vec{x}$, and the outcome of the previous coins. Therefore, the different executions on input $\vec{x}$ can be described using the following binary tree: In each node of the tree we have a name of a player P_j that tosses a coin. The two outgoing edges from this node, labeled 0 and 1 according to the outcome of the coin, lead to two nodes labeled P_k and P_ℓ respectively (j, k and ℓ need not be distinct) which are the identities of the players to toss the next coin depending on the outcome of the random choice made by P_j. If no additional coin toss occurs, the node is labeled "nil"; there are no outgoing edges from a nil node. By the d-randomness property of the protocol, the depth of the above tree is at most d, hence it has at most 2^d root-to-leaf paths. Every possible run of the protocol is described by one root-to-leaf path. Such a path determines all the messages sent in the protocol, which player tosses coins and when, and the outcome of these coins. In particular each such path determines for any P_i the communication $View_i^t(\vec{x}, \vec{R})$. Hence, $View_i^t(\vec{x}, \vec{R})$ can assume at most 2^d different values.

In the second step of the proof, we first fix a vector of random tapes for the players $\vec{\rho} = (\rho_1, \ldots, \rho_n)$. We now consider the deterministic protocol $\mathcal{A}'$ derived from the private protocol $\mathcal{A}$ by fixing these random tapes. We partition the input assignments $\vec{x}$ into 2^{2k} groups according to the input value of x_i (0 or 1) in each of the k phases, and to the output value (0 or 1) in each of the k phases. We argue that the number of different values that the communication string $View_i^t(\vec{x}, \vec{\rho})$ can assume in $\mathcal{A}'$, on the different input assignments within each such group, is at most 2^d. For this, fix $\vec{x}$ in one of these 2^{2k} groups and consider any other $\vec{y}$ pertaining to the same group. If the value of $View_i^t(\vec{y}, \vec{\rho})$ is some communication C_i, then by the privacy requirement (with respect to player P_i), communication C_i must also occur (in $\mathcal{A}$) when the input is $\vec{x}$, and the random tapes are some $R'_1, \ldots, R'_n$, where $R'_i = \rho_i$. Thus, the value of $View_i^t(\vec{y}, \vec{\rho})$ must also appear as $View_i^t(\vec{x}, \vec{R})$

for some random tapes $\vec{R}$. However, by the first step of the proof, for a fixed $\vec{x}$, the communication string $View_i^t(\vec{x}, \vec{R})$ can assume at most 2^d values (over the random tapes $\vec{R}$). Since this is true for each group, the lemma follows. □

Lemma 8: Let A be a deterministic (non-private), k-phase n-player protocol to compute XOR. Then, there is at least one player that has at least $2^{(3-\frac{2}{n})k}$ views over the 2^{kn} input assignments.

First, we show that the above lemmas imply the theorem.

Proof of Theorem 6. By Lemma 7, if we fix the random tapes of the players then each player can see (over the different inputs) at most 2^{2k+d} different views. But, by Lemma 8, for the protocol to be correct there must be at least one player that sees at least $2^{(3-\frac{2}{n})k}$ views. Thus $2^{(3-\frac{2}{n})k} \leq 2^{2k+d}$, and the theorem follows. □

It remains to prove Lemma 8. We now turn to the main technical claim of this section. For the purpose of the proof, we consider k-phase (deterministic, non-private) protocols that compute XOR, but such that for the first instance only m of the n players get inputs (alternatively, we can assume that the input of $n-m$ of the players for the first instance is 0). For $k \geq 1$ and $1 \leq m \leq n$ let $\mathcal{A}(k,m)$ be the set of k-phase protocols that correctly compute XOR with the above restriction. We prove the following lemma.

Lemma 9: Let $A \in \mathcal{A}(k,m)$. Let V_i^A be the number of different views player P_i can see over the $2^{(k-1)n+m}$ inputs. Then $\Pi_{s=1}^n V_i^A \geq 2^{(3n-2)(k-1)+n+2(m-1)}$.

Proof: We prove the claim by induction on both k and m, where the base case is $k=1, m=1$. Let $A \in \mathcal{A}(1,1)$. That is, one player has an input bit and A has to ensure that all players "compute" the value of this bit. Obviously for all P_i we have $V_i^A \geq 2$ (as there are two output values), which gives $\Pi_{s=1}^n V_i^A \geq 2^n$, as required. For the induction step, let $A \in \mathcal{A}(k,m)$, for $k > 1$ *or* $m > 1$. We consider two cases, $m > 1$ and $m = 1$.

$m > 1$ **(and $k \geq 1$):** Before the first XOR value is computed by any player there must be one non-constant message sent in the protocol. That is, there must be some player P_i that sends a message to player P_j, and this message is not constant over all input assignments. Consider the first such non-constant message and, without loss of generality, let it be sent from player P_i to player P_j. Denote this message by M. Since M is the first non-constant message it depends on the first input of P_i only. Without loss of generality, assume that P_i sends the value of its input bit. Let ℓ_s^0 (resp. ℓ_s^1) be the number of possible views of player P_s given that the value of M is 0 (resp. 1). We get that

- $V_i^A = \ell_i^0 + \ell_i^1$.
- $V_j^A = \ell_j^0 + \ell_j^1$.
- $\forall k \neq i,j, V_k^A \geq \max(\ell_k^0, \ell_k^1)$.

Therefore,

$$\begin{aligned}\Pi_{s=1}^{n} V_s^A &\geq (\ell_i^0+\ell_i^1)(\ell_j^0+\ell_j^1)\Pi_{s\neq i,j}\max(\ell_s^0,\ell_s^1)\\ &\geq 4\min(\ell_i^0\ell_j^0,\ell_i^1\ell_j^1)\Pi_{s\neq i,j}\max(\ell_s^0,\ell_s^1)\\ &= 4\ell_i^0\ell_j^0\Pi_{s\neq i,j}\max(\ell_s^0,\ell_s^1)\\ &\geq 4\Pi_{s=1}^{n}\ell_s^0\ ,\end{aligned}$$

where the second inequality is by Claim 1 and the equality is by assuming, without loss of generality, that $\ell_i^0\ell_j^0 \leq \ell_i^1\ell_j^1$.

Now, consider a protocol A_0 defined as follows. It is the protocol A with the modification that P_i has no input, and behaves as if its input is 0. Since we assume that A is a correct protocol, then A_0 is a correct protocol as well in the class $\mathcal{A}(k, m-1)$. [3] Also, we know that A_0 sends 0 as the value of M. Therefore for any s, $1 \leq s \leq n$, we have $V_s^{A_0} = \ell_s^0$. We get

$$\begin{aligned}\Pi_{s=1}^{n} V_s^A &\geq 4\Pi_{s=1}^{n}\ell_s^0\\ &= 4\Pi_{s=1}^{n} V_s^{A_0}\\ &\geq 4\cdot 2^{(3n-2)(k-1)+n+2(m-2)}\ ,\end{aligned}$$

where the last inequality follows from the induction hypothesis. We get that

$$\Pi_{s=1}^{n} V_s^A \geq 2^{(3n-2)(k-1)+n+2(m-1)}\ ,$$

which concludes the proof of the first case.

$m = 1$ **(and $k > 1$):** This is the case where in the first iteration there is a single player who has an input bit. The value of the function on this input has to be computed by all players before they get the next input to be computed. Therefore, the first step of the protocol must be that all players receive messages from which each player can conclude if this first input is 0 or 1. It follows for each P_i, that $V_i^A = \ell_i^0 + \ell_i^1$, where ℓ_i^0 (resp. ℓ_i^1) is the number of different views of player P_i given that the first input bit is 0 (resp. 1). Also note that all players agree on the output. We get,

$$\Pi_{s=1}^{n} V_i^A = \Pi_{s=1}^{n}(\ell_i^0+\ell_i^1)\ .$$

Using Claim 1, we have

$$\Pi_{s=1}^{n}(\ell_i^0+\ell_i^1) \geq 2^n \min(\Pi_{s=1}^{n}\ell_i^0, \Pi_{s=1}^{n}\ell_i^1)\ ,$$

and assuming, without loss of generality, that $\Pi_{s=1}^{n}\ell_i^0 \leq \Pi_{s=1}^{n}\ell_i^1$ we get

$$\Pi_{s=1}^{n} V_i^A \geq 2^n \Pi_{s=1}^{n}\ell_i^0\ .$$

By the same arguments as those for the first case, we now consider a protocol $A_0 \in \mathcal{A}(k-1, n)$ defined using protocol A, and have that $V_i^{A_0} = \ell_i^0$, for any P_i. Using the induction hypothesis we have

$$\begin{aligned}\Pi_{s=1}^{n} V_i^A &\geq 2^n \Pi_{s=1}^{n}\ell_i^0\\ &= 2^n \Pi_{s=1}^{n} V_i^{A_0}\\ &\geq 2^n s^{(3n-1)(k-2)+n+2(n-1)}\\ &= 2^{(3n-2)(k-1)+n}\ ,\end{aligned}$$

[3] In case $\ell_i^0\ell_j^0 > \ell_i^1\ell_j^1$ we consider a protocol A_1, that behaves as if the input to P_i is 1, but also negates the outputs of the first set of inputs.

which concludes the proof for the second case. □

We can now complete the proof of Lemma 8:

Proof of Lemma 8. Let $A \in \mathcal{A}(k,n)$. Then, by Lemma 9,

$$\Pi_{s=1}^{n} V_i^A \geq 2^{(3n-2)(k-1)+n+2(n-1)} = 2^{(3n-2)k}.$$

Therefore there is at least one player P_i such that $V_i^A \geq 2^{(3-\frac{2}{n})k}$. □

Acknowledgments

We wish to thank Pino Persiano for several interactions regarding this paper and Yuval Ishai for discussions on the definitions. We also wish to thank the anonymous PODC referees for useful comments.

References

[1] N. Alon, O. Goldreich, J. Hastad, and R. Peralta, "Simple constructions of almost k-wise independent random variables", FOCS 90 and *Random Structures & Algorithms*, Vol 3, pp. 289-304, 1992. (Addendum: Vol 4. pp. 119-120, 1993).

[2] J. Bar-Ilan, and D. Beaver, "Non-Cryptographic Fault-Tolerant Computing in a Constant Number of Rounds", Proc. of 8th PODC, 1989, pp. 201–209.

[3] D. Beaver, "Perfect Privacy for Two-Party Protocols", TR-11-89, Harvard University, 1989.

[4] M. Bellare, O. Goldreich, and S. Goldwasser, "Randomness in Interactive Proofs", Proc. of 31st FOCS, 1990, pp. 563–571.

[5] M. Ben-or, S. Goldwasser, and A. Wigderson, "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation", Proc. of 20th STOC, 1988, pp. 1–10.

[6] M. Blum, and S. Micali "How to Generate Cryptographically Strong Sequences Of Pseudo- Random Bits", FOCS 82 and *SIAM J. on Computing*, Vol 13, 1984, pp. 850–864.

[7] C. Blundo, A. De-Santis, G. Persiano, and U. Vaccaro, "On the Number of Random Bits in Totally Private Computations", *Proc. of 22nd ICALP*, 1995.

[8] C. Blundo, A. Giorgio Gaggia, and D. R. Stinson, "On the Dealer's Randomness Required in Secret Sharing Schemes", EuroCrypt94 and *Designs, Codes and Cryptography*, Vol. 11(2), pp. 235–259, 1997.

[9] C. Blundo, A. De-Santis, and U. Vaccaro, "Randomness in Distribution Protocols", ICALP 94 and *Information and Computation*, Vol. 131(2), pp. 111-139, 1996.

[10] Bshouty, N. H., "On The Extended Direct Sum Conjecture", *Proc. of 21th ACM Symposium on Theory of Computing*, 1989, pp. 177-185.

[11] R. Canetti, and O. Goldreich, "Bounds on Tradeoffs between Randomness and Communication Complexity", FOCS 90 and *Computational Complexity* Vol. 3, (1993), 141-167.

[12] R. Canetti, E. Kushilevitz, R. Ostrovsky, and A. Rosén, "Randomness vs. Fault-Tolerance", *Proc. of 16th PODC*, pp. 35-44, 1997.

[13] D. Chaum, C. Crepeau, and I. Damgard, "Multiparty Unconditionally Secure Protocols", Proc. of 20th STOC, 1988, pp. 11–19.

[14] B. Chor, and O. Goldreich, "Unbiased Bits from Sources of Weak Randomness and Probabilistic Communication Complexity", FOCS 85 and *SIAM J. Computing* Vol. 17, (1988), 230–261.

[15] B. Chor, and E. Kushilevitz, "A Zero-One Law for Boolean Privacy", STOC 89 and *SIAM J. Disc. Math.* Vol. 4, (1991), 36–47.

[16] B. Chor, and E. Kushilevitz, "A Communication-Privacy Tradeoff for Modular Addition", *Information Processing Letters*, Vol. 45, 1993, pp. 205–210.

[17] B. Chor, M. Geréb-Graus, and E. Kushilevitz, "Private Computations Over the Integers", FOCS 90 and *SIAM Jour. on Computing*, Vol. 24(2), pp. 376-386, 1995.

[18] B. Chor, M. Geréb-Graus, and E. Kushilevitz, "On the Structure of the Privacy Hierarchy", *Journal of Cryptology*, Vol. 7(1), pp. 53-60, 1994.

[19] A. Cohen, and A. Wigderson, "Dispersers, Deterministic Amplification, and Weak Random Sources", Proc. of 30th FOCS, 1989, pp. 14–19.

[20] T. Feder, E. Kushilevitz, M. Naor, and N. Nisan, "Amortized Communication Complexity", FOCS 91 and *SIAM J. Computing*, Vol. 24(4), pp. 736-750, 1995.

[21] M. Franklin, and M. Yung, "Communication Complexity of Secure Computation", Proc. of 24th STOC, 1992, pp. 699–710.

[22] A. Gál, and A. Rosén, "A Theorem on Sensitivity and Applications in Private Computation". In *Proc. of the 31st Ann. ACM Symposium on the Theory of Computing*, May 1999. To Appear.

[23] G. Galibati, and M. J. Fischer, "On The Complexity of 2-Output Boolean Networks", *Theoretical Computer Science*, Vol 16, 1981, pp. 177–185.

[24] Hardy, Littlewood, Polya, **Inequalities**, Cambridge University Press.

[25] R. Impagliazzo, and D. Zuckerman, "How to Recycle Random Bits", Proc. of 30th FOCS, 1989, pp. 248–253.

[26] M. Karchmer, E. Kushilevitz, and N. Nisan, "Fractional Covers and Communication Complexity", *SIAM J. Discrete Math.*, Vol. 8, No. 1, pp. 76-92, 1995. (Early version in *7th IEEE Structure in Complexity Theory*, pp. 262-274, 1992.)

[27] D. Karger, and D. Koller, "(De)randomized Construction of Small Sample Spaces in *NC*", FOCS 94 and *JCSS*, Vol. 55(3), pp. 402-413, 1997.

[28] D. Koller, and N. Megiddo, "Constructing Small Sample Spaces Satisfying Given Constraints", STOC 93 and *SIAM Jour. on Discrete Math.*, Vol. 7(2), pp. 260-274, 1994.

[29] H. Karloff, and Y. Mansour, "On Construction of k-wise Independent Random Variables", STOC 94 and *Combinatorica*, Vol. 17(1), pp. 91-107, 1997.

[30] E. Kushilevitz, and Y. Mansour, "Randomness in Private Computations", PODC 96 and *SIAM Jour. on Discrete Math.*, Vol. 10(4), pp. 647–661, 1997.

[31] E. Kushilevitz, S. Micali, and R. Ostrovsky, "Reducibility and Completeness in Multi-Party Private Computations", Proc. of 35th FOCS, 1994, pp. 478-489.

[32] E. Kushilevitz, R. Ostrovsky, and A. Rosén, "Characterizing Linear Size Circuits in Terms of Privacy", Proc. of 28th STOC, pp. 541–550, 1996.

[33] E. Kushilevitz, and A. Rosén, "A Randomness-Rounds Tradeoff in Private Computation", Crypto 94 and *SIAM Jour. on Discrete Math.*, Vol. 11(1), pp. 61–80, 1998.

[34] D. Krizanc, D. Peleg, and E. Upfal, "A Time-Randomness Tradeoff for Oblivious Routing", Proc. of 20th STOC, 1988, pp. 93–102.

[35] E. Kushilevitz, "Privacy and Communication Complexity", FOCS 89, and SIAM Jour. on Disc. Math., Vol. 5(2), pp. 273–284, 1992.

[36] J. Naor, and M. Naor, "Small-Bias Probability Spaces: Efficient Constructions and Applications", STOC 90, and *SIAM J. on Computing,* Vol 22(4), pp. 838–856, 1993.

[37] N. Nisan, "Pseudorandom Generator for Space Bounded Computation", Proc. of 22nd STOC, 1990, pp. 204–212.

[38] W. Paul, "Realizing Boolean Function on Disjoint Sets of Variables", *Theoretical Computer Science* 2, 1976, pp. 383-396.

[39] P. Raghavan, and M. Snir, "Memory vs. Randomization in On-Line Algorithms", Proc. of 16th ICALP, 1989, pp. 687–703.

[40] L. J. Schulman, "Sample Spaces Uniform on Neighborhoods", Proc. of 24th STOC, 1992, pp. 17–25.

[41] Q. F. Stout, "Meshes with multiple buses", *Proc. of 27th IEEE Symposium on Foundations of Computer Science,* 1986, pp. 264-273.

[42] U. Vazirani, and V. Vazirani, "Random Polynomial Time is Equal to Slightly-Random Polynomial Time", Proc. of 26th FOCS, 1985, pp. 417–428.

[43] A. C. Yao, "Theory and Applications of Trapdoor Functions", Proc. of 23rd FOCS, 1982, pp. 80–91.

[44] D. Zuckerman, "Simulating BPP Using a General Weak Random Source", FOCS 91 and *Algorithmica*, Vol. 16, pp. 367-391, 1996.