A Randomness-Rounds Tradeoff in Private Computation

Eyal Kushilevitz[1] and Adi Rosén[2]

[1] Dept. of Computer Science, Technion, Haifa, Israel*
[2] Dept. of Computer Science, Tel-Aviv University, Tel-Aviv, Israel

Abstract. We study the role of randomness in multi-party private computations. In particular, we give several results that prove the existence of a randomness-rounds tradeoff in multi-party private computation of **xor**. We show that with a single random bit, $\Theta(n)$ rounds are necessary and sufficient to privately compute **xor** of n input bits. With $d \geq 2$ random bits, $\Omega(\log n/d)$ rounds are necessary, and $O(\log n/\log d)$ are sufficient.
More generally, we show that the private computation of a boolean function f, using $d \geq 2$ random bits, requires $\Omega(\log S(f)/d)$ rounds, where $S(f)$ is the sensitivity of f. Using a single random bit, $\Omega(S(f))$ rounds are necessary.

1 Introduction

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 more about the initial inputs of other players than what is revealed by the value of $f(\vec{x})$ and its own input[1]. The players are assumed to be honest but curious. Namely, they all follow the prescribed protocol $\mathcal{A}$ but they could try to get additional information by considering the messages they receive during the execution of the protocol. Private computations in this setting were the subject of a considerable amount of work, e.g., [BGW88, CCD88, BB89, CK89, K89, B89, FY92, CK92, CGK90, CGK92]. One crucial ingredient in private protocols is the use of *randomness*. Quantifying the amount of randomness needed for computing functions privately is the subject of the present work.

Randomness as a resource was extensively studied in the last decade. Methods for saving random bits range over pseudo-random generators [BM84, Y82, N90],

* Work on this paper by the first author was supported by the E. and J. Bishop Research Fund, and by the Fund for the Promotion of Research at the Technion. Part of his research was performed while he was at Aiken Computation Laboratory, Harvard University, supported by research contracts ONR-N0001491-J-1981 and NSF-CCR-90-07677.

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

techniques for re-cycling random bits [IZ89, CW89], sources of weak randomness [CG88, VV85, Z91], and construction of different kinds of small probability spaces [NN90, AGHP90, S92, KM93] (which sometimes even allow to eliminate the use of randomness). A different direction of research is a quantitative study of the role of randomness in specific contexts, e.g., [RS89, KPU88, BGG90, CG90, BGS94, BSV94]. In this work, we initiate a quantitative study of randomness in private computations. We will mainly concentrate on the specific task of computing the **xor** of n input bits. However, some of our results extend to any boolean function. The task of computing **xor** was the subject of previous research due to its being a basic linear operation and its relative simplicity [FY92, CK92].

It is not difficult to show that private computation of **xor** cannot be carried out deterministically (for $n \geq 3$). On the other hand, with a single random bit this becomes possible: At the first round player P_n chooses a random bit r and sends to P_1 the bit $x_n \oplus r$. Then, in round i $(2 \leq i \leq n)$ player P_{i-1} xors its bit x_{i-1} with the message it received in the previous round, and sends the result to P_i. Finally, P_n xors the message it received with the random bit r. Both the correctness and privacy of this protocol are easy to verify. The main drawback of this protocol is that it takes n rounds. Another protocol for this task computes **xor** in 2 rounds but requires a linear number of random bits: In the first round each player chooses a random bit r_i. It sends $x_i \oplus r_i$ to P_1 and r_i to P_2. In the second round P_2 xors all the (random) bits it received in the first round and sends the result to P_1 which xors all the messages it received during the protocol to get the value of the function. Again, both the correctness and privacy are not hard to verify.

In this work we prove that there is a tradeoff between the amount of randomness and the number of rounds in private computations of **xor**. For example, we show that while with a single random bit $\Theta(n)$ rounds are necessary and sufficient[2], with two random bits $O(\log n)$ rounds suffice. Namely, with a second additional random bit, the number of rounds is significantly reduced. Additional bits give a much more "modest" saving. More precisely, we prove that with $d \geq 2$ random bits $O(\log n/\log d)$ rounds suffice and $\Omega(\log n/d)$ rounds are required. Our upper bound is achieved using a new method that enables us to use linear combinations of random bits again and again (while preserving the privacy). The lower bounds are proved using combinatorial arguments, and they are strong in the sense that they also apply to protocols that are allowed to make errors, and that they actually show a lower bound on the *expected* number of rounds. In the final version of this paper, we will show that if protocols are restricted to a certain natural type (that includes, in particular, the protocol that achieves the upper bound) we can even improve the lower bound and show that $\Theta(\log n/\log d)$ rounds are necessary and sufficient.

Our lower bound techniques apply not only to the **xor** function, but in fact give lower bounds on the number of rounds for any boolean function in terms of the *sensitivity* of the function. Namely, we prove that with $d \geq 2$ random bits

[2] More precisely, $n/2$ rounds; and this can be achieved by a slight modification of the first protocol above.

$\Omega(\log S(f)/d)$ rounds are necessary to privately compute a boolean function f, whose sensitivity is $S(f)$. With a single random bit ($d = 1$) $\Omega(S(f))$ rounds are necessary.

The question whether private computations can be carried out in constant number of rounds was discussed in [BB89, BFKR91]. In light of our results, a promising approach to investigate this question may be by proving that if a constant number of rounds is sufficient then a large number of random bits is required.

The rest of the paper is organized as follows: In Section 2 we give some definitions. In Section 3 we give an upper bound on the number of rounds required to privately compute **xor**. In section 4 we give lower bounds on the number of rounds to privately compute a boolean function, in terms of its sensitivity. We conclude in Section 5 with lower bounds on the *expected* number of rounds in terms of the *average* sensitivity of the function being computed.

2 Preliminaries

We give here a description of the protocols we consider, and define the *privacy* property of protocols. More rigorous definitions of the protocols are given in Section 4.1.

A set of n players P_i ($1 \leq i \leq n$), each possessing a secret input bit x_i, collaborate in a protocol to compute a function $f(\vec{x})$. The protocol operates in rounds. In each round each player 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). It then receives the messages sent to it by the other players. In addition, each player at a certain round chooses to output the value of the function. We assume that each player knows its serial number and the total number of players n.

Each player P_i receives during the execution of the protocol a sequence of messages C_i. 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 bit, and the value of the function computed. Formally,

Definition 1. (Privacy) A 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 $f(\vec{x}) = f(\vec{y})$ and $x_i = y_i$, for any sequence of messages C, and for any random coins, R_i, tossed by P_i, $Pr[C_i = C|R_i, \vec{x}] = Pr[C_i = C|R_i, \vec{y}]$, where the probability is over the random coin tosses of *all other* players.

3 Upper Bound

This section presents a protocol which allows n players to use $d \geq 2$ random bits for computing **xor** privately. This protocol takes $O(\log_d n)$ rounds. (For the case $d = 1$ a similar protocol can be presented, that uses $n/2$ rounds.)

Consider the following protocol (which we call the *basic protocol*): First organize the n players in a tree. The degree of the root of the tree is $d+1$, and that of any other internal node is d (assume for simplicity that n is such that this forms a complete tree). The computation starts from the leaves and goes towards the root by sending messages (each of them of a single bit) as follows: Each leaf player P_i sends its input bit x_i to its parent in the tree. Each internal node, after receiving messages from its d children, sums them up (modulo 2) together with its input bit x_i and sends the result to its parent. Finally, the root player sums up the $d+1$ messages it receives together with its input bit and the result is the output of the protocol.

While a simple induction shows the correctness of this protocol, and it clearly runs in $O(\log_d n)$ rounds, it is obvious that it does not maintain the required privacy. The second idea will be to "mask" each of the messages sent in the basic protocol by an appropriate random bit (constructed using the d random bits available), in a way that these masks will disappear at the end, and we will be left with the output. To do so we assign the nodes of the above tree vectors in $GF[2^d]$ as follows (the meaning of those vectors will become clear soon): Assign to the root the vector $(0,\ldots,0)$. The children of the root will be assigned $d+1$ (non-zero) vectors such that the vectors in any d-size subset of them are linearly independent and the sum of all the $d+1$ vectors is $(0,\ldots,0)$ (for example, the d unit vectors together with the $(1,\ldots,1)$ vector satisfy these requirements). Finally, in a recursive way, given an internal node which is assigned a vector v, we assign to its d children d linearly independent vectors whose sum is v (note that in particular none of these vectors is the $\vec{0}$ vector)[3].

We now show how to use the vectors we assigned to the nodes, so as to get a private protocol. We will assume that the random bits $b_1,\ldots,b_d$ are chosen by some external processor. We will later see that this assumption can be eliminated easily. Let v be the vector assigned to some player which is a *leaf* in the tree. We will give this player a single bit $r_v = v \cdot b$, where $b=(b_1,\ldots,b_d)$ is the vector consisting of the d random bits, and the product is an inner product. The players will use the basic protocol, described above, with the modification that a player in a leaf also adds to its message the bit r_v it received (the other players behave exactly as before). We claim that for every player P_i, if in the basic protocol it sends the message m when the input vector is $\vec{x}$, then in the modified protocol it sends the message $m+(v_i \cdot b)$, where v_i is the vector assigned to this player. The proof goes by induction: It is trivially true for the leaf players. For internal nodes the message is calculated by summing the incoming messages, thus, using the induction hypothesis it is $\sum_{k=1}^{d}[m^k+(v^k \cdot b)]$, where m^k is the message received from the k'th child in the basic protocol, and v^k is the vector assigned to the k'th child. Since the construction is such that v_i, the vector assigned to P_i, satisfies $v_i=\sum_{k=1}^{d} v^k$, then a simple algebraic manipulation proves the induction step.

[3] For example, such a collection of d vectors can be constructed as follows: Since $v \neq \vec{0}$ there exists an index i such that $v_i=1$. The first $d-1$ vectors will be the $d-1$ unit vectors $e_1,\ldots,e_{i-1},e_{i+1},\ldots,e_d$. The last vector will be $v-\sum_{j\neq i} e_j$. Obviously the sum of these d vectors is v and they are linearly independent.

In particular, as the root is assigned the vector $(0, \ldots, 0)$, its output equals the output of the basic protocol. Hence, the correctness follows.

We now prove the privacy property of the protocol. The leaf players do not receive any message, hence there is nothing to prove. Let P_j be an internal node in the tree. Denote by $s^1, \ldots, s^d$ the messages it receives. We claim that for every vector $w = (w_1, \ldots, w_d) \in GF[2^d]$, for any input vector, we have

$$Pr[(s^1 = w_1) \wedge \ldots \wedge (s^d = w_d)] = \frac{1}{2^d},$$

where the probability is over the random choice of $b_1, \ldots, b_d$ (note that in this protocol the players do not make internal random choices). In other words, fix any specific input vector $\vec{x}$, then for every vector w, there exists exactly one choice of values for $b_1, \ldots, b_d$, such that the messages that P_j receives, when the protocol is executed with input $\vec{x}$, are the vector w. Denote by $\vec{v}^1, \ldots, \vec{v}^d$ the vectors corresponding to the d children of P_j in the tree, and let $m^1, \ldots, m^d$ be the messages they have to send in the basic protocol given a specific input vector $\vec{x}$. As claimed, for every $1 \leq k \leq d$, the message that the k-th child sends in the modified protocol can be expressed as $s^k = m^k + (\vec{v}^k \cdot \vec{b})$. With this notation, for having $s^1 = w_1, \ldots, s^d = w_d$ the following linear system has to be satisfied:

$$\begin{bmatrix} \vec{v}^1 \\ \vdots \\ \vec{v}^d \end{bmatrix} \cdot \vec{b} = \begin{bmatrix} w_1 - m^1 \\ \vdots \\ w_d - m^d \end{bmatrix}$$

Since $\vec{v}^1, \ldots, \vec{v}^d$ are linearly independent, this system has exactly one solution, as needed.

As for the root player the same argument can be applied to any fixed d-size subset of the $d+1$ messages it receives. This gives us that given any input vector $\vec{x}$, for all d-size messages vectors $\vec{w}$,

$$Pr[(s^1 = w_1) \wedge \ldots \wedge (s^d = w_d)] = \frac{1}{2^d} .$$

Now take two input vectors $\vec{x}$ and $\vec{y}$ such that $x_{\mathrm{root}} = y_{\mathrm{root}}$ and such that $f(\vec{x}) = f(\vec{y})$. Then by the correctness of the algorithm, given a specific d-size messages-vector, the $d+1$'st message is the same for $\vec{x}$ and $\vec{y}$. Thus the privacy property holds with respect to the root too.

Finally, note that we assumed that the random choices were made by some external processor. However, we can let one of the leaf players randomly choose the bits $b_1, \ldots, b_d$ and to supply each of the leaf players with the appropriate bit r_v. As the leaf players only send messages in the protocol, the special processor that selects the random bits gets no advantage.

Note that if a player is non-honest it can easily prevent the other players from computing the correct output. However, it cannot get any additional information in the above protocol, since the only message each player gets after sending its own message is the value of the function. We have thus proved the following theorem:

Theorem 2. *The function* **xor** *can be computed privately using $d \geq 2$ random bits in $O(\log n / \log d)$ rounds.*

4 Lower Bounds

In this section we prove several lower bounds on the number of rounds required to privately compute a boolean function, given that the total number of random bits the players can toss is d. The lower bound is given in terms of the sensitivity of the function. In Section 4.1 we give some formal definitions. In Section 4.2 we present a lemma, central to our proofs, about sensitivity of functions. The proof of the lower bound appears in Section 4.3.

4.1 Preliminaries

We first give a formal definition for the protocols. A protocol operates in rounds. In each round each player P_i, based on the value of its input bit x_i, the values of the messages received in previous rounds, and the values of the coins tossed in previous rounds, tosses a certain number of additional coins, and sends messages to the other players. The values of these messages may depend on all of the above, including the coins just tossed. Then the player receives the messages sent to it by the other players. In addition, each player, at a certain round, chooses to output the value of the function as calculated by it. To define the protocol more formally we give the following definition:

Definition 3. (View)

- A time-t *partial* view of player P_i consists of its input bit x_i, the messages it has received in the first $t-1$ rounds, and the coins it tossed in the first $t-1$ rounds. We denote it by $PView_i^t$.
- A *time-t* view of player P_i consists of its input bit x_i, the messages it has received in the first $t-1$ rounds, and the coins it tossed in the first t rounds. We denote it by $View_i^t$.

Intuitively, the *partial view* of a player in round t determines how many coins (if at all) it will toss in round t. Then, its *view* (which includes those newly tossed coins) determines the messages it will send in round t. Formally,

Definition 4. A protocol consists of a set of functions $R_i^k(PView_i^k)$ which determine how many coins are tossed by P_i in round k, and a set of functions $M_{i \to j}^k : View_i^k \to M$, $1 \leq i, j \leq n$ (where M is a finite domain of possible message values), which determine the message sent by P_i to P_j at round k.

Definition 5. A d-random protocol is a protocol such that for any input assignment, the total number of coins tossed by all players in *any* execution is at most d.

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

Definition 6. A protocol to compute a function f is a protocol such that for any input vector $\vec{x}$ and every player P_i, P_i correctly outputs the value of $f(\vec{x})$ with probability 1.

Protocols that are allowed to err will be considered in Section 5.1.

We will slightly modify our view of the protocol in the following way. Fix an arbitrary binary encoding for the messages in M. We will consider a protocol where each player sends instead of a single message from M, a set of boolean messages that represent the binary encoding of the message to be sent in the original protocol. These messages are sent "in parallel" in the same round. Henceforth when we refer to messages we refer to these binary messages. Clearly, the number of rounds remains the same.

We further modify the model, with respect to its randomness. By the above definitions in a d-random protocol each player can locally toss coins, and we are assured that no more than d coin tosses occur. Assume we have an additional external agent that has a tape with d random bits. Then, we can replace the local coins tossing by a primitive with which each player communicates with this external agent, asking it for a random bit. In response, the external agent (immediately) provides the next random bit on its tape.

Given a d-random private protocol in which the players locally toss coins, there is a protocol that uses the external agent, sends the same messages, and runs in the same number of rounds, while the privacy requirements are not violated in any of the players. The external agent will never be asked for more than the d random bits it has on its tape. Note that the external agent may be able to learn things, but in our lower bound proof we will use the privacy requirement only with respect to the original players. Thus, without loss of generality, we prove our lower bounds on the number of rounds required by a d-random private protocol that uses an external agent as its source of randomness.

With this model in mind we can regard a d-random protocol as a distribution of 2^d *deterministic* protocols, each derived from the randomized protocol by a specific random tape of length d. Furthermore, $View_i^t$, for any i and t, is a function of the random tape $\vec{b}$ and the input assignment $\vec{x}$. We can thus write it as $View_i^t(\vec{x}, \vec{b})$.

Denote by $T_i(\vec{x}, \vec{b})$ the round number in which P_i outputs its result, given input assignment $\vec{x}$ and random tape $\vec{b}$ (Note that since $\vec{b}$ is an argument for T, then T is a deterministic function). The following definition defines the number of rounds of a protocol.

Definition 7. (Rounds Complexity)

- An r-round protocol to compute a function f is a protocol to compute f such that for all i, $\vec{x}$, $\vec{b}$, $T_i(\vec{x}, \vec{b}) \leq r$.

- An *expected* r-round protocol to compute a function f is a protocol to compute f such that for all i and $\vec{x}$, $E_{\vec{b}}[T_i(\vec{x}, \vec{b})] \leq r$.

We include here some definitions related to functions $f : \{0,1\}^n \rightarrow D$, where D is some finite domain.

Definition 8. (Sensitivity)

- A function f is *sensitive* to its i-th variable on assignment Y, if $f(Y) \neq f(Y^{(i)})$, where $Y^{(i)}$ is the same as Y with the i-th variable flipped.
- $\mathcal{S}_f(Y)$ is the set of variables to which the function f is sensitive on assignment Y.
- The *sensitivity* of a function f, denoted $S(f)$, is $S(f) \triangleq \max_Y\{|\mathcal{S}_f(Y)|\}$.
- The *average sensitivity* of a function f, denoted $AS(f)$, is the average of $|\mathcal{S}_f(Y)|$. That is, $AS(f) \triangleq \frac{1}{2^n}\sum_{Y\in\{0,1\}^n}|\mathcal{S}_f(Y)|$.
- The set of variables on which f *depends*, denoted $\mathcal{D}(f)$, is $\mathcal{D}(f) \triangleq \{i : \exists Y \text{s.t.} i \in \mathcal{S}_f(Y)\}$. If $i \in \mathcal{D}(f)$ we say that f depends on its i-th variable.

The following claim gives a lower bound on the degree of error if we evaluate a function f by means of another function g, in terms of their average sensitivities.

Claim 9. *Consider two functions $f, g : \{0,1\}^n \rightarrow D$. Then $f(\vec{x}) = g(\vec{x})$ for at most $2^n \cdot (1 - \frac{AS(f)-AS(g)}{2n})$ input assignments $\vec{x}$.*

Proof. Consider the n-dimensional hypercube. An *f-good edge* is an edge $e = (\vec{x}, \vec{y})$ such that $f(\vec{x}) \neq f(\vec{y})$. By the definitions, the number of f-good edges is exactly $\frac{2^n AS(f)}{2}$. Therefore, there are at least $2^n\frac{AS(f)-AS(g)}{2}$ edges which are f-good but not g-good. For each such edge $e = (\vec{x}, \vec{y})$ either $f(\vec{x}) \neq g(\vec{x})$ or $f(\vec{y}) \neq g(\vec{y})$. Since the degree of each vertex in the hypercube is n there must be at least $2^n \cdot \frac{AS(f)-AS(g)}{2n}$ inputs on which f and g do not agree. □

4.2 A Lemma on Sensitivity

In this section we prove a lemma that bounds the growth of the sensitivity of a combination of functions. This lemma plays a central role in the proofs of our lower bounds, and any improvement on it will immediately improve our lower bounds.

Lemma 10. *Let $\mathcal{F} = \{f_j\}, 1 \leq j \leq m$ be a set of m functions $f_j : \{0,1\}^n \rightarrow \{0,1\}$, for some n. Assume $S(f_j) \leq C$ for all j. Define the function $F(Y) \triangleq (f_1(Y), \ldots, f_m(Y))$. If F assumes at most 2^d different values (different vectors), then the sensitivity of F is at most $C \cdot 2^d - 1$.*[4]

[4] An obvious bound is $S(F) \leq C \cdot m$. However, for reasons that will become clear soon we are interested in bounds which are independent of m.

Proof. Assume towards a contradiction that F has a larger sensitivity. Then, there is an assignment Y such that $|\mathcal{S}_F(Y)| \geq C \cdot 2^d$. Consider Y and assume without loss of generality that $F(Y) = (0, \ldots, 0)$. We will show the existence of $2^d + 1$ different values of F, contradicting the conditions of the lemma. Pick the smallest value j_1 such that $\mathcal{S}_{f_{j_1}}(Y) \neq \emptyset$. Thus, there exists an $i \in \mathcal{S}_{f_{j_1}}(Y)$ which implies that $F(Y^{(i)}) = (0, \ldots, 0, f_{j_1}(Y^{(i)}) \neq 0, *, \ldots, *)$, where each $*$ denotes an arbitrary value. From now on we disregard all values k such that $k \in \mathcal{S}_{f_{j_1}}(Y)$ (those variables to which f_{j_1} is sensitive on Y). According to the conditions of the lemma there are at most C such variables. We now pick a new value j_2, the smallest $j_2 > j_1$ such that there exists a new variable numbered ℓ, still under consideration, and $\ell \in \mathcal{S}_{f_{j_2}}(Y)$. Since f_{j_1} is not sensitive to the ℓ'th variable on Y, we have that $F(Y^{(\ell)}) = (0, \ldots, 0, f_{j_2}(Y^{(l)}) \neq 0, *, \ldots, *)$ which is different from the previous value for F that we have built. We continue this process and at each such step we eliminate at most C variable to which F is sensitive on Y. Since we assumed that $|\mathcal{S}_F(Y)| \geq C \cdot 2^d$, we can continue this process for at least 2^d steps, creating 2^d different values for F. Together with $F(Y) = (0, \ldots, 0)$ this makes more that 2^d different values. □

4.3 Lower Bound on the Number of Rounds

In this subsection we prove the following theorem.

Theorem 11. *Let $\mathcal{A}$ be an r-round d-random ($d \geq 2$) private protocol to compute a boolean function f. Then, $r = \Omega(\log S(f)/d)$.*

Using similar arguments we can show that with a single random bit ($d = 1$) $\Omega(S(f))$ rounds are required. The proof of this case is omitted.

In the following proof we restrict our attention to a specific deterministic algorithm derived from the original protocol by a specific random tape $\vec{B}$. In such a deterministic protocol the views of the players are functions of only the input assignment $\vec{x}$.

Lemma 12. *Consider a private d-random protocol to compute a boolean function f. For a given random tape $\vec{B}$, recall that $View_i^k(\vec{y}, \vec{B})$ is the view of player P_i at round k. Then, for any P_i, $View_i^k(\vec{y}, \vec{B})$ can assume at most 2^{d+2} different values.*

Proof. Partition the input assignments $\vec{x}$ into 4 groups according to the values of x_i (0 or 1), and the value of $f(\vec{x})$ (0 or 1). We argue that the number of different values the view can assume within each such group is at most 2^d.

Fix $\vec{x}$ in one of these 4 groups and consider any other $\vec{y}$ pertaining to the same group. Since the two input assignments are in the same group, by the privacy requirement the value of $View_i^k(\vec{y}, \vec{B})$ must appear also as $View_i^k(\vec{x}, \vec{b})$ for some random tape $\vec{b}$. However, there are only 2^d different random tapes. □

Lemma 13. *Consider a private d-random protocol to compute a boolean function f, and consider a specific random tape $\vec{B}$. Then for any player P_i, the function $View_i^k(\vec{x}, \vec{B})$ (as a function of $\vec{x}$) has sensitivity of at most $T(k) \triangleq (2^{d+2})^{k-1}$.*

Proof. First note that as we consider a fixed random tape the views of the players are functions of the input assignment $\vec{x}$ only.

We prove the lemma by induction. For $k = 1$ the view of any player depends only on its single input bit. Thus, the claim is obvious. For $k > 1$ assume the claim holds for any $\ell < k$. This implies in particular that all messages received by player i and included in the view under consideration have sensitivity of at most $T(k-1)$. Denote by F the view of the player without its input bit. Assume towards a contradiction that $View_i^k$ has sensitivity strictly greater than $T(k)$. Since the view consists of the input bit of the player and the messages received, then F has sensitivity of at least $T(k)$. Using Lemma 10 with $C = T(k-1)$ we conclude that F assumes more than 2^{d+2} different values, contradicting Lemma 12. (Note that Lemma 10 allows us to give a bound which does not depend on the *number* of messages received by P_i.) □

Theorem 14. *Given a private d-random protocol ($d \geq 2$) to compute a boolean function f, consider the deterministic protocol derived from it by any given random tape $\vec{B}$. For any player P_i, there is at least one input assignment $\vec{x}$ such that $T_i(\vec{x}, \vec{B}) = \Omega(\log S(f)/d)$.*

Proof. Consider a fixed but arbitrary player P_i. Denote by t the largest round number in which P_i outputs a value, i.e. $t = \max_{\vec{x}}\{T_i(\vec{x}, \vec{B})\}$. We claim that as long as the sensitivity of the view of P_i does not reach $S(f)$, there is at least one input assignment $\vec{x}$ for which P_i cannot output the correct value $f(\vec{x})$. The reason is that the sensitivity of the view of P_i, while deciding on its output, is a bound on the sensitivity of the output of P_i. On the other hand the value $S(f)$ is obtained by some assignment Y such that the value of $f(Y)$ is different from the value of f on $S(f)$ of Y's "neighbors". Hence, the output must be wrong on either Y or on at least one of these "neighbors". Thus, t is such that $S(View_i^t(\vec{x}, \vec{B})) \geq S(f)$. By Lemma 13, $2^{(d+2)(t-1)} \geq S(f)$, i.e., $t \geq \frac{\log S(f)}{(d+2)} + 1$. □

This proves Theorem 11. Note that this proves not only that there is an input assignment $\vec{x}$ and a random tape $\vec{b}$ for which the protocol runs "for a long time", but also that for each random tape $\vec{b}$ there is such input assignment.

Corollary 15. *Let $\mathcal{A}$ be an r-round d-random private protocol ($d \geq 2$) to compute* **xor** *of n bits. Then $r = \Omega(\log n/d)$.*

As stated at the top of the section, using similar arguments we can show that with a single random bit ($d = 1$) $\Omega(n)$ rounds are required to privately compute **xor**, which is tight.

5 Lower Bounds on the Expected Number of Rounds

In this section we prove lower bounds on the *expected* number of rounds, in terms of the average sensitivity of the computed function. In particular, we prove an $\Omega(\log n/d)$ lower bound on the *expected* number of rounds required by protocols that privately compute **xor** of n bits. We further strengthen the result in the next subsection where we allow the protocol to make errors (we will formally define this notion shortly). In particular, we prove that the expected number of rounds of such protocols that compute **xor** remains $\Omega(\log n/d)$. We start with the following theorem:

Theorem 16. *Let f be a boolean function and let $\mathcal{A}$ be an* expected *r-round d-random private protocol ($d \geq 2$) to compute the function f.*
Then, $r = \Omega(AS(f)\log AS(f)/nd)$.

Consider a protocol $\mathcal{A}$ and a player P_i. We say that the protocol is *late* on $\vec{x}$ and $\vec{b}$ if $T_i(\vec{x},\vec{b}) \geq \frac{\log AS(f)}{2(d+2)}+1$. We first show that for any deterministic protocol derived from a private protocol to compute f, not only there is at least one input on which it is late, but that this happens for a large fraction of the inputs.

Lemma 17. *Consider player P_i. For any random tape $\vec{b}$, there are at least $2^n(\frac{AS(f)-\sqrt{AS(f)}}{2n})$ input assignments $\vec{x}$ such that the protocol is late on $\vec{x}$, $\vec{b}$.*

Proof. Consider the views of P_i, $View_i^t$, given random tape $\vec{b}$. For any round t such that $t < \frac{\log AS(f)}{2(d+2)}+1$, by Lemma 13, $S(View_i^t) < 2^{(d+2)\frac{\log AS(f)}{2(d+2)}} = \sqrt{AS(f)}$. Any function g computed from such a view can have at most the same sensitivity, and thus clearly an average sensitivity of at most $\sqrt{AS(f)}$. By Claim 9, g can have the correct value for the function f for at most $2^n(1-\frac{AS(f)-\sqrt{AS(f)}}{2n})$ input assignments. Since we assume that $\mathcal{A}$ is correct for all input assignments, it follows that at least $2^n\frac{AS(f)-\sqrt{AS(f)}}{2n}$ input assignments are late. □

Lemma 18. *Consider player P_i. There is at least one input assignment $\vec{x}$ for which $\frac{1}{2^d}\sum_{\vec{b}\in\{0,1\}^d} T_i(\vec{x},\vec{b}) \geq (\frac{AS(f)-\sqrt{AS(f)}}{2n})(\frac{\log AS(f)}{2(d+2)}+1)$.*

Proof. By the previous lemma the total number of pairs $\vec{x}$, $\vec{b}$ on which the protocol is late is at least $2^d 2^n(\frac{AS(f)-\sqrt{AS(f)}}{2n})$. It follows that there is at least one input assignment $\vec{x}$ for which there are at least $2^d(\frac{AS(f)-\sqrt{AS(f)}}{2n})$ random tapes $\vec{b}$ such that the protocol is late on $\vec{x}$, $\vec{b}$. For each such tape $T_i(\vec{x},\vec{b}) \geq \frac{\log AS(f)}{2(d+2)}+1$. □

Theorem 16 follows from the above lemma.

Corollary 19. *Let $\mathcal{A}$ be an* expected *r-round d-random private protocol ($d \geq 2$) to compute **xor** of n bits. Then, $r = \Omega(\log n/d)$.*

Proof. Follows from Theorem 16 and the fact that $AS(\mathbf{xor}) = n$. □

5.1 Weakly Correct Protocols

We first formally define a protocol that is allowed to make a certain amount of errors. Given a protocol $\mathcal{A}$, denote by $\mathcal{A}_i(\vec{x}, \vec{b})$ the output of the protocol in player P_i, given input assignment $\vec{x}$ and random tape $\vec{b}$.

Definition 20. For $\delta < 1/2$, a $(1 - \delta)$-correct protocol to compute a function f is a protocol such that for any player P_i and any input vector $\vec{x}$, $Pr_{\vec{b}}[\mathcal{A}_i(\vec{x}, \vec{b}) = f(\vec{x})] \geq (1 - \delta)$.

We prove lower bounds on the number (and the expected number) of rounds of such d-random protocols.

Theorem 21. *Let f be a boolean function.*

- *Let $\mathcal{A}$ be an $(1 - \delta)$-correct r-round d-random private protocol ($d \geq 2$) to compute f. If $\delta < \frac{AS(f)-\sqrt{AS(f)}}{2n}$ then $r = \Omega(\log AS(f)/d)$.*
- *Let $\mathcal{A}$ be an $(1 - \delta)$-correct d-random private protocol ($d \geq 2$) to compute f. Then the* expected *number of rounds is $\Omega((1 - \sqrt{2\delta}) \cdot (\frac{AS(f)-\sqrt{AS(f)}}{2n} - \sqrt{\frac{\delta}{2}}) \cdot \log AS(f)/d)$.*

Proof. We first prove a lower bound on the number of rounds, and then turn our attention to the expected number of rounds. The correctness requirement implies that for any player P_i, $Pr_{\vec{b}}[\mathcal{A}_i(\vec{x}, \vec{b}) = f(\vec{x})] \geq 1-\delta$, for all $\vec{x}$. This implies that there is at least one random tape $\vec{b}$ such that for at least $2^n(1 - \delta)$ input assignments $\vec{x}$, $\mathcal{A}_i(\vec{x}, \vec{b}) = f(\vec{x})$. By the same arguments as those in the proof of Lemma 17, it follows that before round number $\frac{\log AS(f)}{2(d+2)} + 1$, the protocol can be correct on at most $2^n(1 - \frac{AS(f)-\sqrt{AS(f)}}{2n})$ inputs (with random tape $\vec{b}$). Since we require that at least $2^n(1 - \delta)$ are correct, we have that at least

$$2^n(1 - \delta) - 2^n(1 - \frac{AS(f) - \sqrt{AS(f)}}{2n}) = 2^n(\frac{AS(f) - \sqrt{AS(f)}}{2n} - \delta)$$

inputs are late. For a lower bound on r for an r-round protocol it is sufficient to have a single input vector $\vec{x}$ such that the execution on $(\vec{x}, \vec{b})$ is "long". For this, note that if $\delta < \frac{AS(f)-\sqrt{AS(f)}}{2n}$ then (for random tape $\vec{b}$) the number of late inputs is greater than 0. This gives us a lower bound of $r = \Omega(\log AS(f)/d)$ for any $(1 - \delta)$-correct r-round d-random protocol, with δ as above.

We now prove a lower bound on the *expected* number of rounds of $(1 - \delta)$-correct protocols. Again the correctness requirement implies that for any player P_i $Pr_{\vec{b}}[\mathcal{A}_i(\vec{x}, \vec{b}) = f(\vec{x})] \geq 1 - \delta$, for all $\vec{x}$. By a counting argument, it follows that for at least $2^d(1 - \sqrt{2\delta})$ random tapes, $\mathcal{A}_i(\vec{x}, \vec{b}) = f(\vec{x})$ for at least $2^n(1 -$

$\sqrt{\frac{\delta}{2}})$ input assignments $\vec{x}$. Consider those random tapes, and the deterministic protocols derived by them. For each such protocol there are at least

$$2^n(1-\sqrt{\frac{\delta}{2}}) - 2^n(1-\frac{AS(f)-\sqrt{AS(f)}}{2n}) = 2^n(\frac{AS(f)-\sqrt{AS(f)}}{2n} - \sqrt{\frac{\delta}{2}})$$

late input assignments. Thus the total number of late pairs $\vec{x}$, $\vec{b}$ is at least

$$2^d(1-\sqrt{2\delta})\cdot 2^n(\frac{AS(f)-\sqrt{AS(f)}}{2n} - \sqrt{\frac{\delta}{2}})\ .$$

It follows that there is at least one input assignment $\vec{x}$ for which the number of random tapes $\vec{b}$ such that $\vec{x},\vec{b}$ is late is at least $2^d(1-\sqrt{2\delta})\cdot(\frac{AS(f)-\sqrt{AS(f)}}{2n}-\sqrt{\frac{\delta}{2}})$. □

Corollary 22. *For fixed $\delta < 1/2$ let $\mathcal{A}$ be a $(1-\delta)$-correct d-random* expected *r-round private protocol to compute* **xor** *of n bits. Then $r = \Omega(\log n/d)$. (Obviously the same lower bound holds for r-round protocols.)*

Proof. Follows from Theorem 21 and the fact that $AS(\mathbf{xor}) = n$. Note that the term $(1-\sqrt{2\delta})(\frac{1}{2}-\sqrt{\frac{\delta}{2}}-\frac{1}{2\sqrt{n}})$ is greater than 0 for any $\delta < 1/2$ (and sufficiently large n). □

Acknowledgments We thank Benny Chor for useful comments.

References

[AGHP90] N. Alon, O. Goldreich, J. Hastad, and R. Peralta, "Simple constructions of almost k-wise independent random variables", Proc. of 31st FOCS, 1990, pp. 544–553.

[BB89] 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.

[B89] D. Beaver, "Perfect Privacy for Two-Party Protocols", TR-11-89, Harvard University, 1989.

[BFKR91] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway, "Security with Low Communication Overhead", 1991.

[BGG90] M. Bellare, O. Goldreich, and S. Goldwasser, "Randomness in Interactive Proofs", Proc. of 31st FOCS, 1990, pp. 563–571.

[BGW88] 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.

[BM84] 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.

[BGS94] C. Blundo, A. Giorgio Gaggia, and D. R. Stinson, "On the Dealer's Randomness Required in Secret Sharing Schemes", *Proc. of EuroCrypt94*.

[BSV94] C. Blundo, A. De-Santis, and U. Vaccaro, "Randomness in Distribution Protocols", To appear in *Proc. of 21st ICALP*, 1994.

[CG90] R. Canetti, and O. Goldreich, "Bounds on Tradeoffs between Randomness and Communication Complexity", FOCS 90 and *Computational Complexity* Vol. 3, (1993), 141-167.

[CCD88] D. Chaum, C. Crepeau, and I. Damgard, "Multiparty Unconditionally Secure Protocols", Proc. of 20th STOC, 1988, pp. 11–19.

[CK89] B. Chor, and E. Kushilevitz, "A Zero-One Law for Boolean Privacy", STOC 89 and *SIAM J. Disc. Math.* Vol. 4, (1991), 36–47.

[CK92] B. Chor, and E. Kushilevitz, "A Communication-Privacy Tradeoff for Modular Addition", *Information Processing Letters*, Vol. 45, 1993, pp. 205–210.

[CGK90] B. Chor, M. Geréb-Graus, and E. Kushilevitz, "Private Computations Over the Integers", Proc. of 31st FOCS, 1990, pp. 335–344.

[CGK92] B. Chor, M. Geréb-Graus, and E. Kushilevitz, "On the Structure of the Privacy Hierarchy", *Journal of Cryptology*, Vol. 7, No. 1, 1994, pp. 53-60.

[CG88] 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.

[CW89] A. Cohen, and A. Wigderson, "Dispersers, Deterministic Amplification, and Weak Random Sources", Proc. of 30th FOCS, 1989, pp. 14–19.

[FY92] M. Franklin, and M. Yung, "Communication complexity of secure computation", Proc. of 24th STOC, 1992, pp. 699–710.

[IZ89] R. Impagliazzo, and D. Zuckerman, "How to Recycle Random Bits", Proc. of 30th FOCS, 1989, pp. 248–253.

[KM93] D. Koller, and N. Megiddo, "Constructing Small Sample Spaces Satisfying Given Constraints", Proc. of 25th STOC, 1993, pp. 268–277.

[KPU88] D. Krizanc, D. Peleg, and E. Upfal, "A Time-Randomness Tradeoff for Oblivious Routing", Proc. of 20th STOC, 1988, pp. 93–102.

[K89] E. Kushilevitz, "Privacy and Communication Complexity", FOCS 89, and SIAM Jour. on Disc. Math., Vol. 5, No. 2, May 1992, pp. 273–284.

[NN90] J. Naor, and M. Naor, "Small-Bias Probability Spaces: Efficient Constructions and Applications", STOC 90, and *SIAM J. on Computing,* Vol 22, No. 4, 1993, pp. 838–856.

[N90] N. Nisan, "Pseudorandom Generator for Space Bounded Computation", Proc. of 22nd STOC, 1990, pp. 204–212.

[RS89] P. Raghavan, and M. Snir, "Memory vs. Randomization in On-Line Algorithms", Proc. of 16th ICALP, 1989, pp. 687–703.

[S92] L. J. Schulman, "Sample Spaces Uniform on Neighborhoods", Proc. of 24th STOC, 1992, pp. 17–25.

[VV85] U. Vazirani, and V. Vazirani, "Random Polynomial Time is Equal to Slightly-Random Polynomial Time", Proc. of 26th FOCS, 1985, pp. 417–428.

[Y82] A. C. Yao, "Theory and Applications of Trapdoor Functions" Proc. of 23rd FOCS, 1982, pp. 80–91.

[Z91] D. Zuckerman, "Simulating BPP Using a General Weak Random Source" Proc. of 32nd FOCS, 1991, pp. 79–89.