Characterizing Linear Size Circuits in Terms of Privacy*

Eyal Kushilevitz† Rafail Ostrovsky‡ Adi Rosén§

Abstract

In this paper we prove a perhaps unexpected relationship between the complexity class of the boolean functions that have linear size circuits, and n-party private protocols. Specifically, let f be a boolean function. We show that f has a linear size circuit if and only if f has a 1-private n-party protocol in which the total number of random bits used by *all* players is constant.

From the point of view of complexity theory, our result gives a characterization of the class of linear size circuits in terms of another class of a very different nature. From the point of view of privacy, this result provides 1-private protocols that use a constant number of random bits, for many important functions for which no such protocol was previously known. On the other hand, our result suggests that proving, for any NP function, that it has no 1-private constant-random protocol, might be difficult.

1 Introduction

Proving lower bounds on circuit size is one of the central goals of complexity theory. While Shannon proved that almost all functions require exponential size circuits [19], no such lower bound for any NP function is known today. The best lower bounds on unrestricted circuits[1]

*An early version of this paper appeared in the proceedings of the *28th ACM Symp. on the Theory of Computing* (STOC), pp. 541–550, May 1996.

†Dept. of Computer Science, Technion, Haifa, Israel. e-mail: eyalk@cs.technion.ac.il; http://www.cs.technion.ac.il/~eyalk; Part of this research was done while visiting ICSI Berkeley. Supported by MANLAM Fund 120-857 and by the Fund for Promotion of Research at the Technion.

‡Bell Communications Research, MCC 1C-341R, 445 South Street, Morristown, New Jersey 07960-6438. e-mail: rafail@bellcore.com.

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

[1]If we restrict our attention to certain classes of circuits, such as constant-depth circuits or monotone circuits, much better bounds can be proved. For unrestricted circuits the best lower bound is $4n$ [21].

are linear (e.g., [3, 18, 21]). In other words, no NP function f that cannot be computed by a linear size circuit is currently known (the reader is referred to the survey of Boppana and Sipser [4] for an exposition on the current state of knowledge in circuit complexity and for an extensive list of references). A major goal is therefore to improve our understanding of the class of linear size circuits. In particular, a characterization of this class in terms of a different class of objects is desirable.

Private protocols allow n players in a distributed system, each holding a single input bit x_i, to compute a boolean function $f(x_1, \ldots, x_n)$ correctly and 1-privately; that is, in a way that no single player gets any additional information about the inputs of other players[2]. Privacy was the subject of a considerable amount of work, e.g., [1, 2, 5, 8, 9, 10, 11, 12, 13, 15, 16]. Recently, the amount of randomness required by private protocols was studied [6, 14, 17]; a protocol is said to be d-random if the maximum, over all inputs and executions, of the *total* number of random bits tossed by all players, is at most d.

In this paper we relate linear size circuits and private protocols. We prove that for every boolean function f the following holds:

$$\begin{gathered} f \text{ has a linear size circuit} \\ \text{if and only if} \\ f \text{ has 1-private } O(1)\text{-random protocol.} \end{gathered}$$

From the point of view of privacy, this result gives 1-private $O(1)$-random protocols for many important functions such as AND or MAJORITY (while such protocols were previously known only for the function XOR and degenerate functions that depend only on small number of variables). On the other hand, the result suggests that proving, for any NP function, that it has no 1-private $O(1)$-random protocol might be quite difficult. From the point of view of complexity theory, our result gives a characterization of the important class of linear size circuits in terms of another class of a very different nature. In principle, it might be the case that privacy arguments could be used for proving lower bounds on circuit sizes.

We emphasize that none of the directions of this theorem is straightforward. For one direction we show how to simulate a circuit by a private protocol. While this line of proof is common to most positive results in privacy (e.g., [5, 8, 13]), our construction is very efficient in randomness: We use only a constant number of random bits, reusing them over and over again. Previous constructions essentially use new random bits for almost each gate of the circuit[3]. The second direction is even more interesting. It shows how to transform 1-

[2]This is a special case of t-privacy discussed in the literature, where coalitions of at most t players are considered (and both the inputs and the output are not restricted to single bits).

[3]All these constructions were described in the more general t-private setting. However, even simplifying these constructions to the 1-privacy case requires new random bits for almost every gate. In fact, the secret sharing stage, with which all these constructions start, already uses a total of $n \log n$ random bits, even in its 1-private version.

private $O(1)$-random protocols into linear size circuits. We remark that this transformation is independent of the computational complexity or the communication complexity of the protocol.

2 Preliminaries

2.1 Circuits

We consider circuits having as input n bits $x_1, \ldots, x_n$, and their negations $\bar{x}_1, \ldots, \bar{x}_n$. A circuit consists of AND and OR gates of fan-in 2, and arbitrary fan-out. The size of a circuit, denoted m_g, is the number of gates in the circuit. (Alternatively, one can consider the measure m_e, the size of the circuit in edges. Note that these two measures are essentially the same: $m_g \leq m_e \leq 2m_g$.)

For a function $f : \{0,1\}^* \to \{0,1\}$ denote by f_n the restriction of f to the domain $\{0,1\}^n$. A function $f : \{0,1\}^* \to \{0,1\}$ is in the class of linear size circuits if there is a constant k such that for all n there exists a circuit of size at most $k \cdot n$ that computes the restricted function f_n. If f is in this class we say that f has a linear size circuit.

Remark: Many variants of this definition are equivalent. For example, we could allow different types of gates with constant fan-in c. We could allow negations to appear in any place in the circuit, etc. The circuit size under all these variants is the same if we ignore constant factors.

2.2 Private Protocols

We give a description of the protocols we consider, and define the *privacy* property of protocols. We use here the private channels model (as in [5, 8]) in which information theoretic privacy is considered, as opposed to computational privacy (as in [20, 13]).

Let $f_n : \{0,1\}^n \to \{0,1\}$ be a boolean function. A set of n players P_i $(1 \leq i \leq n)$, each player P_i possessing a single input bit x_i (known *only* to P_i), collaborate in a protocol to compute the value of $f_n(\vec{x})$. Each player P_i is provided with a (read only) random tape R_i, each of its bits is uniformly distributed and the bits are all independent. The protocol operates in rounds. In each round each player may toss some coins (that is, it reads some random bits from its random tape), 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). The player 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.

Each player P_i receives during the execution of the protocol a sequence of messages. Let c_i be a random variable of the communication string received by player P_i, and let C_i be a particular communication string received by P_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) An n-party protocol $\mathcal{A}$ for computing a function f_n is *private* with respect to player P_i if for any two input vectors $\vec{x}$ and $\vec{y}$, such that $f_n(\vec{x}) = f_n(\vec{y})$ and $x_i = y_i$, for any sequence of messages C_i, and for any random tape R_i provided to P_i,

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

where the probability is over the random tapes of *all other* players. A protocol $\mathcal{A}$ is 1-private (or private, for short) if it is private with respect to every player P_i. A function f_n is 1-private (or private, for short) if there exists a 1-private protocol $\mathcal{A}$ that computes the function f_n.

The number of coin tosses done (or the amount of randomness used) by player P_i is defined as the position of the rightmost bit read from its random tape R_i. To measure the amount of randomness used by a protocol we give the following definition:

Definition 2: (Randomness) 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 say that a function $f : \{0,1\}^* \rightarrow \{0,1\}$ has a 1-private $O(1)$-random protocol if there exists a constant d such that for all n there exists a d-random 1-private protocol that computes $f_n : \{0,1\}^n \rightarrow \{0,1\}$.

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 the outcome of previous coin tosses.

3 Main Theorem

Our main result in this paper is the following theorem:

Theorem 1: Let f be a boolean function. The function f has a linear size circuit if and only if f has a 1-private $O(1)$-random protocol.

The theorem follows immediately from the following two, more general, lemmas that we prove. The first lemma shows how to construct, from a circuit of size $\beta \cdot n$, a 1-private $O(\beta)$-random protocol. (The proof of this lemma appears in Section 4.)

Lemma 2: If f_n can be computed using a circuit of size m_g, then f_n can be computed using a 1-private $O(\frac{m_g}{n})$-random protocol.

For the second direction, from a 1-private d-random protocol, we construct a circuit of size $\beta \cdot n$ where β is exponential in d. (The proof of this lemma appears in Section 5.)

Lemma 3: If f_n can be computed using a 1-private d-random protocol, then f_n can be computed using a circuit of size $2^{O(d)} \cdot n$.

4 From Circuits to Protocols (Proof of Lemma 2)

As is done in (almost) all known private protocols, given a circuit (with m_g gates) that computes a function f_n, the players simulate the circuit in a gate-by-gate manner (from bottom to top). As opposed to most other constructions of private protocols, in our simulation different players have different roles[4]. First, we partition the players into two sets:

$$\Delta_1 = \{P_1, \ldots, P_{\lfloor \frac{n}{2} \rfloor}\}$$

and

$$\Delta_2 = \{P_{\lfloor \frac{n}{2} \rfloor + 1}, \ldots, P_n\}.$$

We call the players in Δ_2 the *input players*. The players in Δ_1 are further partitioned as follows:

- The player P_1, which is called the *randomness player*.
- $k = \lfloor \frac{|\Delta_1|-1}{3} \rfloor \approx \frac{n}{6}$ *teams*, each consists of three players. Every i'th team, T_i, is further partitioned into a set of two *in-players*, denoted A_i and B_i, and the third player, the *out-player*, denoted C_i.

We first give a high-level description of the simulation: Each input player $P_i \in \Delta_2$, will be responsible during the simulation for its own input bit x_i (and its negation) and also for the bit $x_{i-\lfloor \frac{n}{2} \rfloor}$ (and its negation) which is an input of a player in Δ_1 (if n is odd then player P_n is responsible only for its own input). To do that, player P_i will receive a message from $P_{i-\lfloor \frac{n}{2} \rfloor}$ which will consist of the value $x_{i-\lfloor \frac{n}{2} \rfloor}$ xored with a random bit. Thus the privacy requirement with respect to P_i will still hold. Each of the k teams will be responsible for the simulation of (at most) $\ell = \lceil \frac{m_g}{k} \rceil$ gates. When a team T_i simulates a gate g, the two in-players A_i and B_i will be responsible for the (two) inputs of the gate while the out-player C_i will be responsible for the output of the gate. Again, to guarantee the privacy requirement

[4]One exception for this is in [20] where there are two players who play non-symmetric roles.

with respect to these players, both the inputs and the output of the gate will be xored with random bits in a way that will be described below. Note that each of the inputs for the gate g is either an output of a gate g' which was already simulated by some team T_j, or one of the $2n$ inputs of the circuit. In the former case, the corresponding in-player of T_i (i.e., A_i or B_i) will get a message from the out-player C_j of the team T_j that simulated g'. In the latter case, the corresponding in-player of T_i will receive a message from the relevant input player (in Δ_2). The simulation itself consists of the exchange of a number of messages between the players of the team T_i (and the randomness player P_1), which results in the out-player C_i having a bit which is the value of the gate g xored with a random bit.

All the random bits required in the protocol are chosen by the randomness-player, P_1. These bits are logically partitioned as follows:

- r_0 – used to mask the inputs of players in Δ_1 when sent to the corresponding players in Δ_2.
- $r_1, \ldots, r_\ell$ – used to mask the inputs of the gates before sending them to the in-players (guarantees the privacy with respect to the in-players).
- $s_1, \ldots, s_\ell$ – used to mask the outputs of the gates at the out-players (guarantees the privacy with respect to the out-players).
- $\alpha \cdot \ell$ random bits used for the simulation of the gates ($\alpha = 10$). Every team uses α independent random bits for each of the ℓ simulations that it performs. However, different teams use the *same* random bits.

Altogether (as $\alpha = O(1)$) we get that the number of random bits used is $O(\ell) = O(\frac{m_g}{n})$. The protocol works as follows:

PROTOCOL:

1. (Coin Tossing)
   P_1 tosses all the coins described above. It sends r_0 to all players in Δ_1. It sends $r_1, \ldots, r_\ell$ to all input players (i.e., Δ_2) and to all out-players (i.e., the C_i's). It sends the $\alpha\ell$ random bits needed for the simulation to the in-players (i.e., the A_i's and B_i's).
2. (Inputs Transfer)
   Each player $P_i \in \Delta_1$ sends a bit $y_i = x_i \oplus r_0$ to $P_{i+\lfloor \frac{n}{2} \rfloor}$.
3. (Simulation)
   The gates of the circuits are partitioned into ℓ sets each of size at most k. Each of the k teams of players is responsible for (at most) one gate in each of the ℓ sets. The

assignment of teams to gates, and the role of each player in each team is fixed and known to all players. The simulation of the circuit goes in a bottom-up manner, each gate g being simulated by a team T_i only after all gates below it (in the circuit) have been simulated. The result of gate g being simulated is that the out-player of team T_i holds a value which is the value of the gate xored with one of the random bits $\{s_j\}$. In what follows we describe how team T_i simulates a gate g that falls in the d'th set of gates. For a given input $\vec{x}$, denote by a and b the inputs of gate g in the original circuit, and by c its output. The players do the following:

(a) A_i gets a value which corresponds to the input a:
If a is an output of some previous gate, then there is an out-player C_j that holds this value a xored with some bit $s_{d'}$. Player C_j sends to A_i the value $(a \oplus s_{d'}) \oplus r_d$.
If a is some variable x_j, for $j \geq \lfloor \frac{n}{2} \rfloor + 1$, then P_j sends to A_i the value $x_j \oplus r_d$.
If a is $\bar{x}_j$ for $j \geq \lfloor \frac{n}{2} \rfloor + 1$ then P_j sends to A_i the value $\bar{x}_j \oplus r_d$.
If a is x_j for $j \leq \lfloor \frac{n}{2} \rfloor$ then $P_{j+\lfloor \frac{n}{2} \rfloor}$ sends to A_i the value $y_j \oplus r_d = (x_j \oplus r_0) \oplus r_d$.
If a is $\bar{x}_j$ for $j \leq \lfloor \frac{n}{2} \rfloor$ then $P_{j+\lfloor \frac{n}{2} \rfloor}$ sends to A_i the value $y_j \oplus 1 \oplus r_d = (\bar{x}_j \oplus r_0) \oplus r_d$.
Note that in all cases the message that A_i gets can be represented as $a \oplus j_a \oplus r_d$, where a is the input value to the gate in the original circuit, and the bit j_a is either the constant 0 or some random value independent of $r_1, \ldots, r_\ell$.

(b) In the same way B_i gets a value that corresponds to b (represented as $b \oplus j_b \oplus r_d$).

(c) Players A_i, B_i, C_i and P_1 evaluate the gate g. This evaluation consists of each of A_i, B_i and P_1 sending a single message to C_i who will be able to compute the value $c \oplus s_d$. A detailed description of this process is given later in the sequel.

4. (OUTPUT)
The out-player C_j of the team that computes the final output gate of the circuit, sends the value it computes, which is $f_n(\vec{x}) \oplus s_{d'}$, to P_1. Player P_1 then xors this message with the random bit $s_{d'}$ (which is known to it) to get the desired output $f_n(\vec{x})$, and broadcasts the output to all other players.

Although we have postponed the description of Stage (3c) of the simulation, we have described already, for all but the out-players, *all* the messages that they receive. Therefore, we can already show the following properties of the protocol:

1. Player P_1, although taking a very active role in the protocol, gets only a *single* message which is $f_n(\vec{x}) \oplus s_{d'}$ (in the OUTPUT stage). Thus, given the value of the function $f_n(\vec{x})$ and the random bits (chosen by P_1) this message is fixed. The privacy requirement with respect to P_1 thus holds.

2. The communication sent to each player P_j in Δ_2 consists of a sequence of random bits $r_1, \ldots, r_\ell$ (sent to P_j in the COIN TOSSING stage), the bit $y_{j-\lfloor \frac{n}{2} \rfloor} = x_{j-\lfloor \frac{n}{2} \rfloor} \oplus r_0$ (sent to P_j in the INPUTS TRANSFER stage), and the value of the function (as broadcast in the OUTPUT stage).[5] That is, given any input assignment $\vec{x}$, the last message is fixed by $f_n(\vec{x})$, and for the previous $\ell + 1$ messages, each of the possible sequences of messages $w_1, \ldots, w_{\ell+1}$ (for $w_i \in \{0,1\}$) has probability $2^{-(\ell+1)}$ (since $r_0, r_1, \ldots, r_\ell$ are all independent and uniformly distributed). Thus the privacy requirement with respect to the players in Δ_2 holds.

3. To claim the privacy with respect to the in-players, note that the sequence of messages received by each in-player P_j consists of $\alpha\ell$ random bits, followed by a sequence of $\ell' \leq \ell$ single bit messages (one bit for each of the gates that the player simulates), followed by a message which is the value of the function. The sequence of ℓ' messages that correspond to the simulated gates is of the form $a'_1 \oplus r_1, \ldots, a'_{\ell'} \oplus r_{\ell'}$, where a'_i is an input value to some gate (in the original circuit), xored with a bit j_i which is independent of the bits $r_1, \ldots, r_\ell$, and of the $\alpha\ell$ random bits communicated to the in-player earlier. Therefore, given any input assignment $\vec{x}$, the output message is fixed by $f_n(\vec{x})$, and the probability of each possible communication sequence for the previous $\alpha\ell + \ell'$ messages is $2^{-(\alpha\ell+\ell')}$. This implies the privacy requirement with respect to each of the in-players.

We can conclude the above with the following claim.

Claim 1: The protocol described above computes the function f_n correctly, is $O(\frac{m_g}{n})$-random, and is 1-private with respect to the random player P_1, the players in Δ_2, and the in-players.

The only players for which the privacy is not claimed yet are the out-players. For this, we first need to describe how the actual computation is done, and which messages the out-players receive (in Stage (3c) of the SIMULATION stage). We remark that there are several ways of doing this computation, based on various private protocols in the literature. The one we use here is based on ideas from [11].

- STAGE (3C)
  Recall that player A_i has some value $a' = a \oplus j_a \oplus r_d$, player B_i has some value $b' = b \oplus j_b \oplus r_d$ and the goal is for out-player C_i to have the value $c' = c \oplus s_d$, where c is the output of the gate on the inputs a and b. Obviously, P_1 does not know a and b (this may violate the privacy), however it knows all the random bits. Thus, using the actual

[5] If n is odd then player P_n does not get a y message in the INPUTS TRANSFER stage.

values of the random bits, P_1 can prepare a 2×2 matrix Z such that each of its rows corresponds to a possible value of a', each column corresponds to a possible value of b', and each (a', b') entry of the matrix contains the corresponding value c'. Now A_i, B_i and P_1 use $\alpha = 10$ random bits for their messages (these α bits are already known to all three of them). These α bits are viewed as two 2×2 boolean matrices Z^A and Z^B and two additional bits k_A and k_B. Player P_1 uses the random bit k_A to permute the rows of Z (i.e., if $k_A = 0$ it does nothing and if $k_A = 1$ it switches the rows) and uses k_B to permute the columns of Z. Denote by Z' the resulted matrix. P_1 now computes a 2×2 matrix $Z^* = Z' \oplus Z^A \oplus Z^B$. Player P_1 then sends Z^* to C_i. Player A_i knows a' and it also knows k_A (received from P_1 in the COIN TOSSING stage) so it knows the row $a^* = a' \oplus k_A$ of Z^* which contains the value c'. It sends this index to C_i (since k_A is random this index gives no information on a') and also $Z^A_{a^*}$, the corresponding row of Z^A (these are just two random bits). Similarly, B_i knows b' and it also knows k_B so it knows the column $b^* = b' \oplus k_B$ of Z^* which contains the value c'. It sends this index to C_i and also $Z^B_{b^*}$, the corresponding column of Z^B. Player C_i uses the row $Z^A_{a^*}$ it got from A_i and the column $Z^B_{b^*}$ it got from B_i to "reveal" the (a^*, b^*) entry of Z^* and to get c' as needed (note that for all other entries of Z player C_i misses some of the random bits either from Z^A or Z^B and hence has no information on these entries; it follows that C_i gets no information other than $c' = c \oplus s_d$).

To formally prove the privacy property with respect to the out-player C_i, we proceed as follows. The output of each such simulation is $c' = c \oplus s_d$. Since s_d is uniformly distributed then so is c' and hence the output gives no information on the input $\vec{x}$. Now, fix the value c' and fix in addition values for a', b' and Z'. For each such choice we claim that every communication seen by C_i, which is consistent with the value of c', has the same probability. More precisely, note that the communication seen by C_i consists of 10 bits $Z^*, a^*, b^*, Z^A_{a^*}$ and $Z^B_{b^*}$. Out of the 2^{10} possible values 2^9 are consistent with c'. We claim that each of them appears with probability exactly 2^{-9}. To see this, note that each of $Z^A_{a^*}$ and $Z^B_{b^*}$ is just a pair of random bits. The value a^* is the xor of a' with a random bit, k_A, hence it is uniformly distributed and so is b^*. Finally note that in Z^* one bit is determined by the other values and the value of c' but since Z^A and Z^B are random matrices then each of the other 3 entries of Z^* is uniformly distributed. This implies the privacy with respect to a single simulation. Finally, note that in all ℓ simulations in which C_i participates α independent random bits are used, and also that $s_1, \ldots, s_\ell$ are all independent. This implies the privacy with respect to the out-players and concludes our construction.

We thus showed the following claim, which together with Claim 1 implies Lemma 2:

Claim 2: The protocol described is 1-private with respect to the out-players.

5 From Protocols to Circuits (Proof of Lemma 3)

In what follows we make a series of transformations from a 1-private protocol which computes a function f_n using d random bits until we get a circuit that computes the function f_n. First, we want all messages in the protocol to be single bits. Any protocol can be made to satisfy this condition by "breaking" each of the original messages into its binary representation[6]. Henceforth, when we refer to messages, we refer to these binary messages.

The next step relies on the following lemma from [17]. This lemma says that the number of different communication strings that a player may see in a protocol which is 1-private and uses a "small" amount of randomness is "small". This will be a key fact for our transformation. More precisely, we restrict our attention to a specific deterministic protocol derived from the original protocol by fixing specific random tapes $\hat{R}_1, \ldots, \hat{R}_n$ for the n players (in such a deterministic protocol the communication is a function of the input assignment $\vec{x}$ only) and we give an upper bound on the number of communication strings that can be seen by every player P_i in such a deterministic protocol.

Lemma 4: Consider a private d-random protocol $\mathcal{A}$ to compute a boolean function f_n. Let $C_i(\vec{x}, R_1, \ldots, R_n)$ denote the communication seen by player P_i on input $\vec{x}$, when the n random tapes for the players are $R_1, \ldots, R_n$. Fix the random tapes of the n players to some $\hat{R}_1, \ldots, \hat{R}_n$. Then, for any P_i, the communication $C_i(\vec{y}, \hat{R}_1, \ldots, \hat{R}_n)$ can assume at most 2^{d+2} different values (over all choices of input assignments $\vec{y} \in \{0,1\}^n$).

Proof: In the first step of the proof, we fix an arbitrary input $\vec{x}$ and consider the possible values of $C_i(\vec{x}, R_1, \ldots, R_n)$ over all different choices of random tapes $R_1, \ldots, R_n$. 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 $C_i(\vec{x}, R_1, \ldots, R_n)$ 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 rounds of the protocol 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

[6] Formally, let M be the set of all different messages that can be sent in the protocol in all different runs. Fix an arbitrary fixed-length binary encoding for the messages in M (note that M is finite). We consider a protocol where each player sends instead of a single message $m \in M$, a sequence of boolean messages that represent the binary encoding of m.

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 $C_i(\vec{x}, R_1, \ldots, R_n)$. Hence, $C_i(\vec{x}, R_1, \ldots, R_n)$ 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 $\hat{R}_1, \ldots, \hat{R}_n$. 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 4 groups according to the value of x_i (0 or 1), and the value of $f_n(\vec{x})$ (0 or 1). We argue that the number of different values that the communication string $C_i(\vec{x}, \hat{R}_1, \ldots, \hat{R}_n)$ 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 4 groups and consider any other $\vec{y}$ pertaining to the same group. If the value of $C_i(\vec{y}, \hat{R}_1, \ldots, \hat{R}_n)$ 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 $R'_i = \hat{R}_i$. Thus, the value of $C_i(\vec{y}, \hat{R}_1, \ldots, \hat{R}_n)$ must also appear as $C_i(\vec{x}, R_1, \ldots, R_n)$ for some random tapes $R_1, \ldots, R_n$. However, by the first step of the proof, for a fixed $\vec{x}$, the communication string $C_i(\vec{x}, R_1, \ldots, R_n)$ can assume at most 2^d values (over the random tapes $R_1, \ldots, R_n$). □

We now use the above lemma. Given a d-random private protocol $\mathcal{A}$, we can transform it into a deterministic protocol $\mathcal{A}'$ in which each player can see (over all 2^n inputs) at most $t = 2^{d+2}$ different communication strings. This is done by fixing the random tapes of all players (e.g., to the strings of 0's) and letting the players simulate the protocol $\mathcal{A}$ using these random tapes (i.e., whenever in $\mathcal{A}$ a player tosses a random coin, in $\mathcal{A}'$ the player behaves as if this bit is 0). By Lemma 4, protocol $\mathcal{A}'$ is a deterministic protocol in which each player can see at most t different communication strings. In addition, $\mathcal{A}'$ is a correct protocol to compute f_n since $\mathcal{A}'$ generates a possible run of the (correct) protocol $\mathcal{A}$. Note that $\mathcal{A}'$ is *not* a private protocol, and that all the other transformations presented below do not depend on privacy.

The next transformation converts $\mathcal{A}'$ into another (deterministic) protocol $\mathcal{A}''$ in which we limit the total number of messages transmitted. Although in $\mathcal{A}'$ the number of different possible communication strings of each player is restricted to t, it might be that the number of bits that the player receives is much larger. In $\mathcal{A}''$ the number of bits received by each player will be bounded by $t - 1$ bits.[7] The intuition is that if there are more than $t - 1$

[7] Ideally, one could hope for each player P_i to receive only $\log_2 t$ bits; this however may not be possible since

bits that a player receives, then many of these bits are induced by other bits and hence can be omitted (we have to make sure, however, that this omission still enables each player to compute in time the messages it has to send). We associate a tree with the bits received by a player P_i. Level (k,j) of this tree corresponds to the k'th bit that player P_j sends to P_i (and the levels appear from top to bottom in a lexicographic order). We look at each node in level (k,j) and the path leading to it and we put an outgoing edge with value 0 (respectively 1) if there is an input assignment which will cause player P_i to see the previous bits with values as indicated by the path leading to this node, and for which the k'th bit that player P_j sends to P_i is 0 (respectively 1). Observe that the root-to-leaves paths of this tree correspond to the (at most) t communication strings that P_i may see. Thus, there are at most t leaves in this tree, and hence the tree contains at most $t-1$ nodes with out degree 2. If there is such a node in level (k,j), then in $\mathcal{A}''$ player P_j will keep sending to P_i the k'th bit as in $\mathcal{A}'$; otherwise we omit this bit from $\mathcal{A}''$. Since the total number of such nodes is at most $t-1$, then in $\mathcal{A}''$ player P_i will receive at most $t-1$ bits. All the bits that P_i receives in nodes with out-degree 1 are fixed by previous messages and hence P_i can deduce their value without actually receiving the bits. Player P_i can therefore compute any message it has to send by simulating $\mathcal{A}'$. Thus, for all P_i's, the protocol $\mathcal{A}''$ is well defined. Note that the protocol $\mathcal{A}''$ is *oblivious*; that is, the senders, the receivers, and the rounds in which messages are sent are independent of the input.

Our last stage is to transform $\mathcal{A}''$ into a circuit. We will consider the messages sent in protocol $\mathcal{A}''$, and add to those a single "virtual output message" which will be the value of the function computed. To do that we pick an arbitrary player P_i and define the additional message as the value of the function as computed by this player locally. As each player receives at most $t-1$ bits, altogether (including the output message) at most $n \cdot (t-1)+1$ bits are exchanged (this does *not* mean that each player sends at most $t-1$ bits). To build the circuit we employ a simple idea: The circuit is built out of sub-circuits, each computing one of the messages. Each message sent by a player P_i in $\mathcal{A}''$ will be simulated by a sub-circuit that gets as inputs x_i (the input of P_i), and all the (at most $t-1$) bits that P_i received so far from other players. The bit to be sent by P_i is some function g of these (at most t) bits. Clearly, using a simple construction we can implement a circuit that computes g with size $O(t \cdot 2^t)$, e.g., using the DNF form of g. However, this can be improved using the following observation. As explained above, each player can see at most t different views. Therefore, it suffices that each sub-circuit will compute meaningful results only for these t possible views. Such a circuit can be easily constructed to have size of $O(t^2)$ (e.g., by using a DNF representation with a single term for each of the t possible views on which 1 is the output). Altogether, there are $O(t \cdot n)$ sub-circuits of size $O(t^2)$ each, and the total size of the circuit

messages are sent to P_i by various senders which may have only partial information about the communication exchanged by other players.

is $O(t^3 n) = 2^{O(d)} \cdot n$. The output of the circuit is the output of the sub-circuit that computes the additional "virtual message". Finally, note that the implementation of the sub-circuits as described above may use negation gates (in addition to OR and AND gates). To get a circuit that conforms to the definition we gave, we apply a standard transformation to the whole circuit, so that only input variables are negated; this transformation may result in a penalty of a factor of 2 in the size of the circuit. This concludes our construction for the proof of Lemma 3.

6 Extensions and Open Problems

Looking at our constructions, one can see that they not only give a relationship between the size of circuits and the amount of randomness used by the protocol but in addition, they maintain the relationship between the *depth* of the circuit and the number of *rounds* of the private protocol. Namely, given a circuit of depth ℓ our protocol can operate in $O(\ell)$ rounds[8]. For the other direction, given a d-random protocol that operates in r rounds, the circuit that we obtain is of depth $O(d) \cdot r$.

The most obvious open problem raised by this work is to prove, for some NP function f, that it has no 1-private protocol that uses $d = O(1)$ random bits. This would imply that this function has no linear size circuit. A more modest goal would be to prove such a lower bound for a specific constant d. Although we know that any boolean function can be computed privately using a randomized protocol [5, 8], as of today the only known lower bound on the amount of randomness necessary to 1-privately compute a function f_n, is that deterministically almost no function can be computed 1-privately (i.e., at least 1 random bit is necessary).

Finally, we remark that the results of the present work (in particular, the randomness-efficient protocols) were subsequently extended to deal with the more general t-privacy case [7].

Acknowledgments

We wish to thank Oded Goldreich and Yuval Ishai for useful comments.

[8]The protocol, as defined in Section 4, operates in more rounds, but can be easily converted to run in $O(\ell)$ rounds by simulating concurrently all the gates which are in the same level of the circuit.

References

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

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

[3] N. Blum, "A Boolean Function Requiring $3n$ Network Size", *Theoretical Computer Science*, Vol. 28, pp. 337-345, 1984.

[4] R. B. Boppana, and M. Sipser, "The Complexity of Boolean Functions", in *Handbook of Theoretical Computer Science*, Vol. A, Chapter 14, pp. 757-804, 1990.

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

[6] C. Blundo, A. De-Santis, G. Persiano, and U. Vaccaro, "On the Number of Random Bits in Totally Private Computations", Proc. of 22nd ICALP, Springer-Verlag, Lecture Notes in Computer Science, Vol. 944, pp. 171–182, 1995.

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

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

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

[10] B. Chor, M. Geréb-Graus, and E. Kushilevitz, "Private Computations Over the Integers", FOCS 90 and *SIAM Jour. on Computing*, Vol. 24, No. 2, pp. 376-386, 1995.

[11] U. Feige, J. Kilian, and M. Naor, "A Minimal Model for Secure Computation", Proc. of 26th STOC, pp. 554-563, 1994.

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

[13] O. Goldreich, S. Micali, and A. Wigderson, "How to Play Any Mental Game", Proc. of 19th STOC, pp. 218-229, 1987.

[14] E. Kushilevitz, and Y. Mansour, "Randomness in Private Computations", Proc. of 15th PODC, pp. 181–190, 1996.

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

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

[17] E. Kushilevitz, and A. Rosén, "A Randomness-Rounds Tradeoff in Private Computation", Crypto94 and *SIAM J. Disc. Math.* (to appear).

[18] W.J. Paul, "A $2.5n$ Lower Bound on the Combinatorial Complexity of Boolean Functions", *SIAM Jour. on Computing*, Vol. 6, No. 3, pp. 427-443, 1977.

[19] C. E. Shannon, "A Mathematical Theory of Communication", *Bell System Tech. Jour.*, Vol. 27, pp. 379-423, 623-656, 1948.

[20] A. C. Yao, "How to Generate and Exchange Secrets", Proc. of 27th FOCS, pp. 162-167, 1986.

[21] U. Zwick, "A 4n Lower Bound on the Combinational Complexity of Certain Symmetric Boolean Functions over the Basis of Unate Dyadic Boolean Functions", *SIAM Jour. on Computing*, Vol. 20, 1991.