# Lower and Upper Bounds
# on the Randomness Complexity
# of Private Computations of AND

Eyal Kushilevitz[1], Rafail Ostrovsky[2], Emmanuel Prouff[3,5], Adi Rosén[4(✉)],
Adrian Thillard[5], and Damien Vergnaud[3]

[1] Department of Computer Science, Technion, Haifa, Israel
eyalk@cs.technion.ac.il
[2] Department of Computer Science and Department of Mathematics,
UCLA, Los Angeles, USA
rafail@cs.ucla.edu
[3] Laboratoire d'informatique de Paris 6, LIP6, Sorbonne Université, CNRS,
Paris, France
damien.vergnaud@lip6.fr
[4] CNRS and Université Paris Diderot, Paris, France
adiro@irif.fr
[5] ANSSI, Paris, France
{emmanuel.prouff,adrian.thillard}@ssi.gouv.fr

**Abstract.** We consider multi-party information-theoretic private protocols, and specifically their randomness complexity. The randomness complexity of private protocols is of interest both because random bits are considered a scarce resource, and because of the relation between that complexity measure and other complexity measures of boolean functions such as the circuit size or the sensitivity of the function being computed [12,17].

More concretely, we consider the randomness complexity of the basic boolean function and, that serves as a building block in the design of many private protocols. We show that and cannot be privately computed using a single random bit, thus giving the first non-trivial lower bound on the 1-private randomness complexity of an explicit boolean function, $f : \{0,1\}^n \to \{0,1\}$. We further show that the function and, on any number of inputs $n$ (one input bit per player), can be privately computed using 8 random bits (and 7 random bits in the special case of $n = 3$

players), improving the upper bound of 73 random bits implicit in [17]. Together with our lower bound, we thus approach the exact determination of the randomness complexity of and. To the best of our knowledge, the exact randomness complexity of private computation is not known for any explicit function (except for xor, which is trivially 1-random, and for several degenerate functions).

# 1   Introduction

A multi-party *private* protocol for computing a function $f$ is a distributed protocol that allows $n \geq 3$ players $P_i$, for $0 \leq i \leq n-1$, each possessing an individual secret input $x_i$, to compute the value of $f(x)$ in a way that does not reveal any "unnecessary" information to any player.[1] The protocol proceeds in rounds, where in each round each player sends a message to any other player, over a secure point-to-point channel. The privacy property of such protocol means that no player can learn "anything" (in an information-theoretic sense) from the execution of the protocol, except what is implied by the value of $f(x)$ and its own input. In particular, the players do not learn anything about the inputs of the other players. Private computation in this setting was the subject of considerable research, see e.g., [1,3,6,7,12,16,17]. In addition to its theoretical interest, this setting constitutes the foundation for many cryptographic applications (in information theoretic settings) such as electronic secret ballot.

Randomness is necessary in order to perform private computations involving more than two players (except for the computation of very degenerate functions). That is, the players must have access to (private) random sources. As randomness is regarded as a scarce resource, methods for saving random bits in various contexts have been suggested in the literature, see, e.g., [14,20] for surveys. Thus, an interesting research topic is the design of randomness-efficient private protocols, and the quantification of the amount of randomness needed to perform private computations of various functions and under various constraints. This line of research has received considerable attention, see, e.g., [4,5,11,15–17,19,21].

As in most of the work on the randomness complexity of private computations, we concentrate here on the computation of boolean functions, where the input $x_i$ of each player is a single input bit. Previous work on the randomness complexity of private computations revealed that there is a tradeoff between randomness and time (i.e., number of communication rounds) for the private computation of xor [19], or gave lower bounds on the number of rounds necessary to privately compute any function, in terms of the sensitivity of the function and the amount of randomness used [12]. However, if one is allowed an arbitrary number of rounds for the computation then, prior to the present work, there were no known lower bounds on the number of random bits necessary for private protocols computing explicit boolean functions (except that some

---

[1] The two-party case, $n = 2$, is known to be qualitatively different [7].

randomness is indeed necessary, i.e., no deterministic private protocol exists).[2]
In fact, Kushilevitz et al. [17] gave a relation between the number of random bits
necessary to privately compute a function $f : \{0,1\}^n \to \{0,1\}$, and the Boolean
circuit size necessary to compute $f$; it is proved, among other things, that the
class of boolean functions that have $O(1)$-random, 1-private, protocols is equal
to the class of boolean functions that have linear size circuits. This, perhaps
surprising, connection to circuit size explains the difficulty of proving $\omega(1)$ lower
bounds on the number of random bits necessary for the private computation of
any explicit boolean function $f$, as such a result would imply superlinear lower
bounds on the circuit size of $f$ – a notoriously difficult problem.[3] Additional
connections between the randomness complexity of the private computation of a
function to other complexity measures, such as its sensitivity, have been shown
in, e.g., [5,18,19].

This leaves the interesting, and perhaps feasible, task to determine the exact
randomness complexity of the private computation of boolean functions of lin-
ear circuit size, where each player has a single input bit. This class of functions
includes quite a few interesting functions $f$ and, in particular, the basic func-
tions xor and and. Indeed, the functions xor and and serve as basic building
blocks for private protocols that rely on the boolean-circuit representation of
$f$ (more generally, *addition* and *multiplication* are used as building blocks for
protocols that use the *arithmetic*-circuit representation of $f$). In the context of
lower bounds for *communication* complexity of private protocols, these building
blocks also serve as the center of analysis (as a first step for a more general
understanding), see, e.g., [8–10].

It is known that xor can be computed privately using a single random bit,
for any number of players, and this is optimal since no deterministic private
multiparty protocol can exist (see [19]). To the best of our knowledge, there is
no exact determination of the randomness complexity of private computation
for any other explicit function. Furthermore, prior to the present paper, there
was no lower bound showing for any explicit boolean function that it cannot

---

[2] Recently, a lower bound on the number of random bits necessary for the private
computation of the Disjointness function was obtained [21]. However, for the Dis-
jointness function each player has $m \geq 1$ bits of input. Furthermore, the obtained
lower bound is of $\Omega(m)$, and the hidden constant is less than 1. Thus, for the special
case of and (Disjointness with $m = 1$), the lower bound of [21] only implies, in our
context, the trivial claim that and cannot be privately computed by a deterministic
protocol.

[3] When the protocol has to be resilient against coalitions of $t > 1$ players (so called
$t$-private protocols) several $\omega(1)$ lower bounds on the number of random bits neces-
sary for private protocols for explicit functions have been proved. Kushilevitz and
Mansour [16] proved that any $t$-private protocol for xor requires at least $t$ random
bits. Blundo et al. [5] gave lower bounds for two special cases. Namely, they proved
that if $t = n - c$, for some constant $c$, then $\Omega(n^2)$ random bits are necessary for
the private computation of xor, and if $t \geq (2 - \sqrt{2})n$, then $\Omega(n)$ random bits are
necessary. Gal and Rosén [13] proved that $\Omega(\log n)$ random bits are necessary for
2-private computation of xor.

be privately computed (in the natural setting that we consider here, i.e., where each player has one input bit) using a single random bit.

In this paper, we give the first such lower bound, showing that the function and cannot be privately computed using a single random bit.[4] We further make the first step towards determining the exact randomness complexity of the private computation of and by showing an improved upper bound of 8 on the number of random bits necessary for such computation, and we strengthen this upper bound to 7 for the special case of 3 players.

The rest of the paper is organized as follows. In Sect. 2, we define the model and the complexity measures that we consider. In Sect. 3, we prove that the private computation of and cannot be performed with a single random bit. In Sect. 4, we give a number of upper bounds on the randomness complexity of private computation of and.

## 2   Preliminaries

Let $f : \{0,1\}^n \to \{0,1\}$ be any Boolean function. A set of $n$ players $P_i$ ($0 \leq i \leq n-1$), each possessing a single input bit $x_i$ (known *only* to $P_i$), collaborate in a protocol to compute the value of $f(x)$. The protocol operates in rounds. In each round each player may toss some (fair) random 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 see them). After sending its messages, each player receives the messages sent to it by the other players in the current round. Without loss of generality (because we are not interested in this paper in the number of rounds), we assume that the messages sent by the players consist of single bits. In addition, each player locally outputs the value of the function at a certain round. We say that the protocol computes the function $f : \{0,1\}^n \to \{0,1\}$ if for every input $x \in \{0,1\}^n$, and for any outcome of all coin tosses, the output produced by each player is always $f(x)$ (i.e., perfect correctness).

To formally define a protocol we first define the notion of a *view* of a player.

**Definition 1 (View).** *The view of player $P_i$ at round $t \geq 1$, denoted $V_i^t$, consists of the input bit to player $P_i$, i.e. $x_i$, the messages received by $P_i$ in rounds 1 to $t-1$, and the results of the coin tosses performed by player $P_i$ in rounds 1 to $t$. Let $\mathcal{V}_i^t$ be the set of possible views of player $P_i$ at round $t$. Let $\hat{V}_i^t$ be the view $V_i^t$ without the coins tossed at round $t$, and let $\hat{\mathcal{V}}_i^t$ be the set of possible values of $\hat{V}_i^t$.*

**Definition 2 (Protocol).** *A protocol consists of a sequence of rounds, where each round $t \geq 1$ is formally defined by the following functions:*

---

[4] In a different setting, namely, where there are two input players, Alice and Bob, and a third output player, Charlie, Data et al. [11] also study the randomness (and communication) complexity of secure computation (in particular of the and function; see [11, Thm. 11]).

- $S_i^{t,\ell} : (\hat{\mathcal{V}}_i^t \times \{0,1\}^{\ell-1}) \to \{\texttt{stop}, \texttt{toss}\}$, *for $\ell \geq 1$, defining if another random coin is to be tossed by player $P_i$, given $\hat{\mathcal{V}}_i^t$ and the values of the $\ell - 1$ random coins tossed so far by player $P_i$ in round $t$.*
- $m_{i,j}^t : \mathcal{V}_i^t \to \{0,1\}$, *for $0 \leq i, j \leq n-1$, defining the message $P_i$ sends to $P_j$ at round $t$.*
- $O_i^t : \mathcal{V}_i^t \to \{0, 1, \perp\}$, *for $0 \leq i \leq n-1$, defining if and what value player $P_i$ outputs at round $t$. Since the views are increasing, we can require that each player outputs a non-null output only in one round.*

Sometimes it is more convenient to model the coin tossing done by each player as a set of binary random tapes $R_i$, each $R_i$ being provided to player $P_i$. The number of random coins tossed by player $P_i$ is the number of random bits it reads from its random tape.

We denote by $r_i$ a specific random tape provided to player $P_i$, by $\boldsymbol{r} = (r_1, \ldots, r_n)$ the vector of random tapes of all the players, and by $\boldsymbol{R} = (R_1, \ldots, R_n)$ the random variable for these tapes. Note that if we fix $\boldsymbol{r}$, we obtain a deterministic protocol.

**Definition 3 (Randomness Complexity).** *A d-random protocol is a protocol such that, for any input assignment x, the total number of coins tossed by* all *players in* any *execution is at most d.*

Our main question in this paper is what is the randomness complexity of the best private protocol (in terms of randomness complexity) for the boolean function `and`.

Informally, *privacy* with respect to player $P_i$ means that player $P_i$ cannot learn anything (in particular, the inputs of other players) from the messages it receives, except what is implied by its input bit, and the output value of the function $f$ being computed.[5] Formally, denote by $c_i$ a specific sequence of messages received by $P_i$, and by $C_i$ the random variable (depending also on $\boldsymbol{R}$) for the sequence of messages received by $P_i$. We define:

**Definition 4 (Privacy).** *A protocol $\mathcal{A}$ for computing a function $f$ is* private *with respect to player $P_i$ if, for any two input vectors x and y such that $f(x) = f(y)$ and $x_i = y_i$, for any sequence of messages $c_i$, and for any random tape $r_i$ provided to $P_i$,*

$$Pr[C_i = c_i | r_i, x] = Pr[C_i = c_i | r_i, y],$$

*where the probability is over the random tapes of all other players.*

*A protocol is said to be private if it is private with respect to all players.*

---

[5] In the literature, a more general notion of privacy, called $t$-privacy, is often considered, where any set of players of size at most $t$ cannot learn anything from the messages received by all of them. In this paper we consider only 1-privacy, and call it "privacy" for simplicity.

## 3    Lower Bound

In this section we prove that private computation of `and` of $n$ bits, $n \geq 3$, cannot be performed with a single random bit. We note that a lower bound on the number of random bits of private computation for $n > 3$ does not follow from a lower bound for $n = 3$, because, in general, simulating by 3 players a protocol on $n > 3$ players may violate the privacy requirement.

Our result for `and` is in contrast to the situation for the function `xor`, which can be computed privately using a single random bit. Our result constitutes, to the best of our knowledge, the first lower bound that quantifies the amount of randomness needed to privately compute an explicit boolean function (without any limitation on the protocol, such as its round complexity).

In the course of this proof, we denote by **1** the all-1 input of length $n$, and by $e_S$, for $S \subseteq \{1, \ldots, n\}$, the input assignment of all 1's except at the coordinates in $S$. Specifically, we use $e_j$ to denote the vector with 0 at position $j$ and 1's elsewhere[6] and $e_{i,j}$ to denote the vector with 0's at positions $i, j$ and 1's elsewhere.

Assume, towards a contradiction, that $\pi$ is a 1-random private protocol for `and`. We assume w.l.o.g. that $\pi$ is in a "canonical form", that we define as follows: A protocol is in canonical form if no message $m$, sent from player $P_i$ to player $P_j$ at round $t$, can be inferred from the input $x_j$, the private randomness of player $P_j$, and the messages previously received by player $P_j$ (i.e., either received in round $t' < t$, or in round $t$ from a player $P_{i'}$, for $i' < i$). In particular, no message in a protocol in canonical form is a constant message.

Obviously, for $\pi$ to compute `and` there must be at least one non-constant message defined in $\pi$. Consider any such message, $m$, sent in round $t = 1$ (since $\pi$ is in canonical form there must be at least one such message in round $t = 1$), say, from player $P_i$ to player $P_j$. Since the message $m$ is sent in round $t = 1$, it can depend only on $x_i$ and the random bits tossed by $P_i$ by round $t = 1$. To preserve privacy with respect to $P_j$, the message $m$ has to have the same distribution when $x_i = 0$ and when $x_i = 1$. Since $\pi$ is 1-random, the number of random bits tossed by any single player, in particular $P_i$, in any execution of the protocol, is at most 1. It follows that $Pr[m = 0] = 1/2$, and $Pr[m = 1] = 1/2$ regardless of the value of $x_i$, thus $P_i$ must toss a random bit by round $t = 1$ whether $x_i = 0$ or $x_i = 1$. To conclude, there is some player (the sender of $m$), w.l.o.g. denote it $P_0$, that regardless of its input, and in any execution, tosses in $\pi$ a single random bit, denote it $r$. Since $\pi$ is 1-random, no other random bit is tossed in $\pi$ in any execution (by any player). Thus, since all messages in $\pi$ can depend only on the input bits, $x_i$, $0 \leq i \leq n - 1$, and the single random bit $r$ tossed by player $P_0$, we may consider any message sent in $\pi$ as a sum of monomials in $x_i$, $0 \leq i \leq n - 1$, and $r$. We now analyze some properties of the (assumed) protocol $\pi$.

---

[6] Not to be confused with the $j$-th unit vector.

**Lemma 5.** *All messages $m$ sent in $\pi$ are of the form $m = r \oplus \sum_{i \in S \subseteq \{0,\ldots,n-1\}} x_i$ or $m = 1 \oplus r \oplus \sum_{i \in S \subseteq \{0,\ldots,n-1\}} x_i$. No player receives during the execution of $\pi$ two (distinct) messages.*

*Proof.* We prove the claim by induction on the round number $t \geq 1$, i.e., we prove that until round $t \geq 1$ all messages are of the form $m = r \oplus \sum_{i \in S \subseteq \{0,\ldots,n-1\}} x_i$ or $m = 1 \oplus r \oplus \sum_{i \in S \subseteq \{0,\ldots,n-1\}} x_i$ and that no player receives by the end of round $t$ two (distinct) messages.

For the basis of the induction consider the first round $t = 1$ and the messages sent and received in this round. Clearly, a (non-constant) message in the first round can be sent only by player $P_0$ otherwise it consists of only the input bit of the sending player (or its negation) and the privacy property will be violated. Since no message is received before the messages of the first round are sent, the messages sent by player $P_0$ at round $t = 1$ are a function of only $r$ and $x_0$. We argue that such message has therefore to be of the form $m = r \oplus x_0$ (or $m = 1 \oplus r \oplus x_0$) or of the form $m = r$ (or $m = 1 \oplus r$): since $m$ depends only on $r$ and $x_0$, the monomials in $m$ can only be 1, $x_0$, $r$, and $rx_0$. We claim that if the monomial $rx_0$ appears in the sum representing $m$ then the privacy property is violated with respect to the player receiving the message, say player $P_j$. This is because the possible messages that include $rx_0$ are: $rx_0$, $rx_0 \oplus x_0 = (r \oplus 1)x_0$, $rx_0 \oplus x_0 \oplus r = (r \oplus 1)(x_0 \oplus 1) \oplus 1$, and $rx_0 \oplus r = r(x_0 \oplus 1)$ (and their negations). Consider the two input assignments $e_{0,j}$ and $e_j$. Observe that the distribution of each one of these messages on the inputs $e_{0,j}$ and $e_j$ is different, which violates the privacy requirement ("leaking some information" on $x_0$ to $P_j$). For example, $rx_0$ is always 0 in $e_{0,j}$ and uniformly distributed in $e_j$. The argument for the other cases is similar.

It follows that the messages sent in round $t = 1$ are of the desired form. Since only player $P_0$ can send messages in round $t = 1$, it also follows that by the end of round $t = 1$ each player receives at most a single message. Thus, the claim holds for round $t = 1$.

We now prove the claim for round $t > 1$, assuming the induction hypothesis holds for round $t - 1$. Consider player $P_i$ and the message $m_{i,j}^t$ that it sends in round $t$ to player $P_j$. Since this message is computed by player $P_i$ at round $t$, it can be expressed as a function of $x_i$, of the single message $m$ that player $P_i$ receives in some round $t' < t$ (if such a message exists) and, if $i = 0$, of the random bit $r$. We distinguish between two cases: when $i \neq 0$, and when $i = 0$.

When $i \neq 0$, the message $m_{i,j}^t$ sent by $P_i$ is the sum of a subset of the monomials 1, $x_i$, $m$, $mx_i$. If the monomial $mx_i$ does not appear in the sum, then $m_{i,j}^t$ is of the desired form (otherwise $m_{i,j}^t$ is either a messages that can be inferred by $P_j$ or a message that violates the privacy property with respect to $P_j$; in any case it cannot be part of the protocol.)[7]

---

[7] By inspection for each of the 8 subsets of 1, $x_i$, and $m$ (represented by $\{0,1\}^3$ in the natural way): (000, 100) - 0, 1: constants, not in protocol; (010, 110) - $x_i$, $1 \oplus x_i$: violates privacy; (001, 101) - $m$, $1 \oplus m$: of the desired form; (011, 111) - $x_i \oplus m$, $1 \oplus x_i \oplus m$: of the desired form.

On the other hand, we show in the following that any message defined by any of the 8 sums of monomials that include the monomial $mx_i$ violates the privacy property with respect to $P_j$, and hence such message cannot be part of the protocol. By the induction hypothesis $m = r \oplus \sum_{k \in S \subseteq \{0,\dots,n-1\}} x_k$ (or $1 \oplus r \oplus \sum_{k \in S \subseteq \{0,\dots,n-1\}} x_k$), for some $S$. Consider the former form, the latter is similar. For the message $mx_i$ (resp., $1 \oplus mx_i$) consider the inputs $e_{i,j}$ and $e_j$ and observe that on the former input the message is always 0 (resp., 1) while on the latter it is $m$ (resp., $1 \oplus m$), and hence does not exhibit the same distribution on the two inputs. Similarly, consider the messages $mx_i \oplus x_i = (m \oplus 1)x_i$, $mx_i \oplus m = m(x_i \oplus 1)$, and $mx_i \oplus x_i \oplus m = (m \oplus 1)(x_i \oplus 1) \oplus 1$ (and their negations), and observe that the message has different distributions on the inputs $e_{i,j}$ and $e_j$ (in each of the cases, for one of these two inputs the value of the message is either $m$ or its negation, i.e., the support of the distribution is of size 2, and for the other input the distribution has support of size 1).

For $i = 0$, player $P_i = P_0$ has also the random bit $r$, so the message sent by $P_0$ at round $t$ to player $P_j$ is the sum of a subset of the monomials 1, $x_0$, $m$, $mx_0$, $r$, $rx_0$, $rm$, $rmx_i$. But, no message $m$ can be received by player $P_0$ before round $t$, since any message of the form $r \oplus \sum_{k \in S \subseteq \{0,\dots,n-1\}} x_k$ (or $m = 1 \oplus r \oplus \sum_{k \in S \subseteq \{0,\dots,n-1\}} x_k$) would either violate the privacy (since $P_0$ knows $r$), or would be such that $P_0$ can compute it itself.[8] It follows that the message sent by player $P_0$ at round $t$ is the sum of a subset of the monomials 1, $x_0$, $r$, $rx_0$. But, we have proved above (when proving the base case of the induction) that in this case $m_{i,j}^t$ must be $r \oplus x_0$ (or $1 \oplus r \oplus x_0$).

We conclude that the messages sent in round $t$ are of the desired form.

Now, assume towards a contradiction that some player $P_j$ receives by the end of round $t$ two (distinct) messages which, as we proved above, must be of the desired form. Denote $q_1 = r \oplus \sum_{i \in S_1 \subseteq \{0,\dots,n-1\}} x_i$, and $q_2 = r \oplus \sum_{i \in S_2 \subseteq \{0,\dots,n-1\}} x_i$. The two messages received by player $P_j$ are therefore $m_1 = q_1$ (or $m_1 = 1 \oplus q_1$) and $m_2 = q_2$ (or $m_2 = 1 \oplus q_2$), for some sets $S_1$ and $S_2$. Consider now $Q = m_1 \oplus m_2$. Observe that $Q = \sum_{i \in S' \subseteq \{0,\dots,n-1\}} x_i$ (or $Q = 1 \oplus \sum_{i \in S' \subseteq \{0,\dots,n-1\}} x_i$) for $S' = S_1 \triangle S_2$. If $S' \subseteq \{x_j\}$ then, since $\pi$ is of canonical form, one of the two messages $m_1$ and $m_2$ (the one arriving later) cannot exist in $\pi$. It follows that $S' \not\subseteq \{x_j\}$ and the privacy property is violated with respect to player $P_j$, as $Q$ reveals information on the xor of the inputs in $S'$.[9] A contradiction to $\pi$ being private.

Therefore, the claim holds for round $t$.                                          □

**Lemma 6.** *Consider the protocol $\pi$, an arbitrary player $P_j$ and an arbitrary round $t \geq 1$. Then, player $P_j$ cannot compute the function* and *at the end of round $t$.*

---

[8] If $S = \{x_0\}$ then $P_0$ can compute the message itself. If there exists a $k \neq 0$, $k \in S$, consider the two inputs $e_0$, $e_{0,k}$ to see that the privacy property is violated.

[9] Formally, consider the two inputs $e_{j,k}$ and $e_j$, for some $k \in S'$, $k \neq j$. These two inputs agree on $x_j$ as well as on the value of the function, but the distributions of the messages that $P_j$ receives on these two inputs are not identical.

*Proof.* By Lemma 5, player $P_j$ receives by the end of round $t$ at most a single message, and this message is of the form $m = r \oplus \sum_{i \in S \subseteq \{0,\ldots,n-1\}} x_i$ or $m = 1 \oplus r \oplus \sum_{i \in S \subseteq \{0,\ldots,n-1\}} x_i$. We distinguish between two cases.

<u>Case 1:</u> For all $k \neq j$, $k \in S$. Since $n \geq 3$, there exist two distinct $k_1, k_2 \in S$, $k_1 \neq j$, $k_2 \neq j$. Consider the two inputs $\mathbf{1}$ and $e_{k_1,k_2}$. While $AND(\mathbf{1}) \neq AND(e_{k_1,k_2})$, the view of $P_j$ at the end of round $t$ is the same on $\mathbf{1}$ and $e_{k_1,k_2}$ and hence $\pi$ must err on at least one of them (note that this in particular holds when $j = 0$; for all other players, who do not know $r$, the message $m$ is uniformly distributed).

<u>Case 2:</u> There exists an index $k \neq j$, $k \notin S$. Consider the two inputs $\mathbf{1}$ and $e_k$. As in Case 1, $\pi$ must err on at least one of these inputs.     □

We conclude that protocol $\pi$ (a 1-random private protocol for `and`) does not exist.

**Theorem 7.** *The private computation of the function `and` cannot be performed with a single random bit.*

## 4   Upper Bounds

In this section, we provide significantly improved upper bounds on the randomness complexity of `and`. Specifically, we show that `and` on any number of players $n$ can be privately computed using 8 random bits, and can be computed with 7 random bits for the special case of $n = 3$ players.

In order to present our protocol, we first present two building blocks which are used in our constructions. They are both implementations of information-theoretic 1-out-of-2 Oblivious Transfer.

### 4.1   1-out-of-2 Oblivious Transfer

In a 1-out-of-2 Oblivious Transfer (1-2 OT) protocol two parties, Alice and Bob, engage in a protocol that allows Bob to choose which part of the information that Alice holds he wants to learn, in such a way that Alice does not learn which part of her information Bob learned.

More formally, for a 1-out-of-2 Oblivious Transfer protocol Alice has two bits $b_0$ and $b_1$, and Bob has a selection bit $s$. Alice and Bob, together with a set of helper players, $\mathcal{H}$, engage in a protocol at the end of which the following holds:

– Bob knows the value of $b_s$.
– Bob does not learn any information on $b_{1 \oplus s}$ (i.e., the transcript of Bob, given the values of $s$ and $b_s$, is identically distributed whether $b_{1 \oplus s}$ is 0 or 1).
– Alice does not learn anything (i.e., the transcript of Alice, given any choice of values for $b_0$ and $b_1$, is identically distributed; in particular, it is independent of $s$).
– The helper players do not learn anything (i.e., the transcript of each of them is distributed independently of the inputs $s, b_0, b_1$).

We now give two implementations of this building block using a different number of helper players, and a different number of random bits. Both are used in our protocols.

### 4.1.1   Implementation 1: 3 Random Bits, 1 Helper Player

This implementation is given by Beaver [2].

There is one helper player, denoted by $H$. The protocol is defined as follows.

1. The helper player $H$ tosses 2 uniformly distributed and independent random bits, $r_0$ and $r_1$ (to be used as masking bits), and one additional independent and uniformly distributed random bit $p$ (to define one of the two possible permutations of $\{0, 1\}$).
2. $H$ sends both $r_0$ and $r_1$ to Alice and it sends $p$ and $r^* = r_p$ to Bob.
3. Bob sends to Alice the message $i = s \oplus p$.
4. Alice sends to Bob the two bits $m_0 = b_0 \oplus r_i$ and $m_1 = b_1 \oplus r_{1 \oplus i}$.
5. Bob "deciphers" the value of the bit he wants to learn (i.e., $b_s$) by computing $m_s \oplus r^*$.

The fact that Bob learns the value of $b_s$ follows by simple case analysis ($p = 0$ or $p = 1$).

For completeness, we sketch a proof of the privacy of this protocol. A more detailed formal proof is incorporated within the privacy proof for our protocol that uses the $OT$ protocol as a sub-protocol. We observe the following:

- $H$ does not receive any message and hence privacy is preserved with respect to $H$.
- Alice receives from $H$ the bits $r_0$ and $r_1$ and from Bob the bit $i = s \oplus p$. All three are independent and uniformly distributed between 0 and 1 (as $r_0, r_1$, and $p$ are uniformly distributed and independent random bits).
- Bob receives from $H$ the bits $p$ and $r^* = r_p$ (but not the bit $r_{1 \oplus p}$), and from Alice the bits $m_0 = b_0 \oplus r_i$ and $m_1 = b_1 \oplus r_{1 \oplus i}$. Observe that $b_s = m_s \oplus r^*$, but $p$ and $m_{1 \oplus s}$ are both independent of $m_s$ and $r^*$, and uniformly distributed.

### 4.1.2   Implementation 2: 2 Random Bits, 2 Helper Players

Here we assume that there are two helper players, denoted $H_0$ and $H_1$. The protocol is defined as follows.

1. Alice tosses two independent random bits $p$ and $r$.
2. Alice sends the message $m_0 = b_0 \oplus r$ to player $H_p$, the message $m_1 = b_1 \oplus r$ to player $H_{1 \oplus p}$, and $p$ and $r$ to Bob.
3. Bob sends the bit 1 to player $H_{s \oplus p}$ and the bit 0 to player $H_{1 \oplus s \oplus p}$.
4. If player $H_0$ (resp., $H_1$) receives 1 from Bob, it sends to Bob the message $m$ it received from Alice (i.e, either $m_0$ or $m_1$).
   Otherwise, if player $H_0$ (resp., $H_1$) receives 0 from Bob, it sends to Bob the (constant) message 0.[10]

---

[10] Clearly in this case the relevant helper player does not need to send any message to Alice. However in order to stay coherent with the model we use, we define which bit

5. Bob "deciphers" the value of the bit he wants to learn (i.e., $b_s$) by computing $b_s = m \oplus r$, where $m$ is the message Bob got from $H_{s \oplus p}$.

The fact that Bob learns the value of $b_s$ follows from the protocol, by inspection.

For completeness, we sketch a proof of the privacy of this protocol. A more detailed formal proof is incorporated within the privacy proof for our protocol that uses the $OT$ protocol as a sub-protocol. We observe the following:

– Alice does not get any message and hence privacy is preserved with respect to Alice.
– Each of $H_0$ and $H_1$ gets a single message from Alice, which is one of her input bits xored with $r$; it also received from Bob a bit which is either 0 or 1, depending on the value $s \oplus p$; since $r$ and $p$ are uniformly random and independent, then privacy is preserved with respect to each one of the helper players $H_0$ and $H_1$.
– Bob receives $p$ and $r$ from Alice. Then, given the value of $p$ and the value of $s$, known to Bob, it receives a constant message from one of $H_0$ or $H_1$, and the message $b_s \oplus r$ from the other (if $s \oplus p = 1$ Bob receives the constant message from $H_0$, and if $s \oplus p = 0$ Bob receives the constant message from $H_1$.) Hence, given the value of $b_s$, the transcript of Bob is distributed uniformly.

## 4.2   The AND Protocol

We first present a protocol, $\Pi_{\text{odd}}$, applicable to an odd number of players; this protocol uses 8 random bits. This protocol serves to introduce the main ideas of our protocols, and is also the basis for a somewhat improved protocol for $n = 3$, that uses 7 random bits. Extending $\Pi_{\text{odd}}$ to work also for even number of players and keeping the 8 random bits bound requires some more effort, and we give such a protocol in Subsect. 4.2.2, applicable to any $n \geq 4$.

### 4.2.1   Odd Number of Players

We describe our protocol $\Pi_{\text{odd}}$ for odd number of players, $n$, denoted $P_0, P_1, \ldots, P_{n-1}$.

**Initialization Phase.** In the initialization phase player $P_0$ tosses 3 random bits and plays the role of the helper player of implementation 1 of $OT$ for all pairs of players where Alice is an odd player and Bob is the successive even player. Player $P_{n-1}$ does the same for all pairs of players where Alice is an even player and Bob is the successive odd player. Specifically:

Player $P_0$ tosses 3 random bits $r_0^1$, $r_1^1$, and $p^1$. It sends the bits $r_0^1$, $r_1^1$ to all odd players, and sends the bits $p^1$ and $r_{p^1}^1$ to all even players.

---

is sent in each round between any two players, unless for all inputs and all random coins values, no message is sent between the two.

Player $P_{n-1}$ tosses 3 random bits $r_0^0$, $r_1^0$, and $p^0$. It sends the bits $r_0^0$, $r_1^0$ to all even players, and sends the bits $p^0$ and $r_{p^0}^0$ to all odd players.

In addition, player $P_0$ tosses 2 additional random bits $q_0$ and $q_1$. It sends $q_0$ to all odd players, and $q_1$ to all even players. $P_0$ also locally computes $y_0 = q_0 \oplus q_1 \oplus x_0$.

**Computation Phase.** This phase runs in $n-1$ rounds. The inductive invariant that we maintain is that at the end of round $i \geq 1$ player $P_i$ has the value $y_i = q_0 \oplus q_1 \oplus \Pi_{j=0}^{i} x_j$. In each round, the protocol will run an $OT$ protocol. We give a detailed description of the rounds below.

**Final Phase.** At the end of the computation phase, player $P_{n-1}$ has (by the inductive invariant) the value $y_{n-1} = q_0 \oplus q_1 \oplus \Pi_{j=0}^{n-1} x_j$. It sends this value to $P_0$ who xors it with $q_0 \oplus q_1$ to obtain $\Pi_{j=0}^{n-1} x_j = AND(x_0, x_1, \ldots, x_{n-1})$. Player $P_0$ then sends this value to all other players.

We now define how the computation phase is implemented so as to maintain the inductive invariant (and privacy, as we prove later). The invariant clearly holds at the end of round 0 (i.e., before the computation phase starts) since player $P_0$ has $x_0$, $q_0$, and $q_1$ and hence can compute $y_0$.

Now, in round $i \geq 1$ players $P_{i-1}$ and $P_i$ engage in a 1-2 $OT$ protocol, using Implementation 1 described above. The values that Alice (i.e., player $P_{i-1}$) holds for the $OT$ protocol are $b_0 = q_{i \bmod 2}$ and $b_1 = y_{i-1}$. Observe that Alice has $b_0$ from the initialization phase, and $b_1$ by the inductive invariant. The selection bit of Bob (i.e., player $P_i$) is $s = x_i$. The random bits used for the $OT$ protocol are $r_0^k$, $r_1^k$, and $p^k$, where $k = (i+1) \bmod 2$. Observe that $P_i$ and $P_{i-1}$ receive in the initialization phase the random bits needed in order to simulate Alice and Bob of the $OT$ protocol (i.e., $r_0^k$, $r_1^k$ for Alice and $p^k$, $r_{p^k}^k$ for Bob). Let $v_i$ denote the output (i.e., the bit learned by Bob) of the $OT$ protocol run in the $i$-th round.

It follows that at the end of the $OT$ protocol, if the value that player $P_i$ holds is $x_i = 0$, then it gets from player $P_{i-1}$ the value $v_i = q_{i \bmod 2}$, and if $x_i = 1$ then it gets the value $v_i = y_{i-1} = q_0 \oplus q_1 \oplus \Pi_{j=0}^{i-1} x_j$.

Now, if $x_i = 0$ then $\Pi_{j=0}^{i} x_j = 0$, and player $P_i$ has $y_i$ by calculating $q_{i \bmod 2} \oplus q_{(i+1) \bmod 2}$, where the former is just $v_i$ and the latter is received from $P_0$ in the initialization phase.

If $x_i = 1$ then $\Pi_{j=0}^{i} x_j = \Pi_{j=0}^{i-1} x_j$ and player $P_i$ just sets $y_i = v_i$.

The total number of random bits used in this protocol is 8: the protocol uses 3 bits for each of the two sets of $OT$ protocols, and 2 additional masking bits, $q_0$ and $q_1$.

It remains to prove that privacy is preserved with respect to all players. Intuitively, there are $n-1$ invocations of the $OT$ protocol. Each internal player (i.e., all players except $P_0$ and $P_{n-1}$) participates in two $OT$ invocations, once as Alice (with the following player) and once as Bob (with the preceding player), each of these two invocations using different sets of random bits, one set from $P_0$ and one set from $P_{n-1}$. Players $P_0$ and $P_{n-1}$ participate each in a single invocation of the $OT$ protocol, $P_0$ as Alice with $P_1$ and $P_{n-1}$ as Bob with $P_{n-2}$. Hence the number of players must be odd (to guarantee that the random bits

used by the $OT$ protocol of $P_{n-1}$ and $P_{n-2}$ come from $P_0$ and not from $P_{n-1}$). Formally,

**Theorem 8.** *The AND protocol $\Pi_{odd}$ is private for $n$ odd, $n \geq 3$.*

*Proof.* We first prove the claim for players $P_i$, $0 < i < n - 1$, and then for $P_0$ and for $P_{n-1}$.

For $0 < i < n - 1$, observe that player $P_i$ receives messages pertaining to exactly two $OT$ invocations, one in which it plays the role of Alice, and one where it plays the role of Bob. In addition, $P_i$ receives from player $P_0$ either the bit $q_0$ or the bit $q_1$ and, at the end of the protocol, the computed value of the function.

We prove the claim for $i$ even (the case of $i$ odd is analogous, switching the roles of the random bits, i.e., flipping their superscripts, and switching $q_0$ and $q_1$). The messages that such player $P_i$ receives are:

1. During the initialization phase: bits $r_0^0, r_1^0, p^1, r_{p^1}^1, q_0$.
2. During the $OT$ protocol with player $P_{i-1}$ (i.e., when playing the role of Bob):
   - $q_1 \oplus r_j^1$ and $y_{i-1} \oplus r_{1\oplus j}^1$, where $j = x_i \oplus p^1$.
3. During the $OT$ protocol with player $P_{i+1}$ (i.e., when playing the role of Alice):
   - $x_{i+1} \oplus p^0$.
4. In the final phase, from player $P_0$, $AND(x_0, x_1, \ldots, x_{n-1})$.

Observing the nine messages received by $P_i$, one can verify that:

1. The messages received in Stage 1 are just the random bits $r_0^0, r_1^0, p^1, r_{p^1}^1, q_0$.
2. For the messages of Stage 2 we distinguish between two cases depending on the value of $x_i$.
   If $x_i = 0$ then the first message is $q_1 \oplus r_{p^1}^1$ and the second one is $y_{i-1} \oplus r_{1\oplus p^1}^1$. We have that the first message includes a xor operation with $q_1$, and the second one a xor with $r_{1\oplus p^1}^1$.
   If $x_i = 1$ then the first message is $q_1 \oplus r_{1\oplus p^1}^1$ and the second one is $y_{i-1} \oplus r_{p^1}^1$. In that case, the first message includes a xor operation with $r_{1\oplus p^1}^1$ and the second one a xor with $q_1$ (since by the inductive invariant, $y_{i-1} = q_0 \oplus q_1 \oplus \Pi_{j=0}^{i-1} x_j$.)
3. The message received in Stage 3 includes a xor operation with $p^0$.
4. The message received in Stage 4 is the value of the function.

Inspecting the distribution of the above messages, the last message (Stage 4) is, by definition, the value of the function; all other 8 messages are independent and uniformly distributed (in correspondence with the 8 random bits that are used): the bits $r_0^0, r_1^0, p^1, r_{p^1}^1, q_0$ in Stage 1, the two messages of Stage 2 one includes a xor operation with $r_{1\oplus p^1}^1$ and the other with $q_1$, and the message received in Stage 3 which includes a xor operation with $p^0$. Hence, the privacy with respect to $P_i$ follows.

Almost the same argument applies to player $P_{n-1}$ as well. It receives a subset of the messages received by players $P_i$, $0 < i < n - 1$, namely, those of Stages 1, 2, 4 above. In addition it knows the value of the random bit $p^0$. But, since the

message of Stage 3 is not received by $P_{n-1}$, the privacy with respect to $P_{n-1}$ holds.

As to player $P_0$, it receives the messages listed under Stages 1 and 3 above, and (from player $P_{n-1}$ at the final phase) the message $y_{n-1} = q_0 \oplus q_1 \oplus \Pi_{j=0}^{n-1} x_j$. In addition, player $P_0$ knows the values of the random bits $r_0^1$, $r_1^1$, $q_0$ and $q_1$. We have that the messages received in Stages 1 and 3 each includes a xor operation with an independent (uniformly distributed) random bit not known to $P_0$. The message received in the final phase is determined by the value of the function, $q_0$ and $q_1$. Hence, privacy with respect to player $P_0$ holds as well.     □

### 4.2.2   At Least 4 Players

If one attempts to apply the above protocol $\Pi_{\text{odd}}$ to an even number of players then privacy will not be preserved. This is because when players $P_{n-2}$ and $P_{n-1}$ engage in their $OT$ protocol, they will do that with the random bits tossed by player $P_{n-1}$ (while in the case of odd $n$ these bits are tossed by the "helper" $P_0$).

To remedy this problem, we stop the computation phase one round earlier, that is, we run it for $n-2$ rounds only, at the end of which player $P_{n-2}$ has, as in $\Pi_{\text{odd}}$, the value $y_{n-2} = q_0 \oplus q_1 \oplus \Pi_{j=0}^{n-2} x_j$. We then perform the last $OT$ protocol of the computation phase using Implementation 2 defined above and fresh random bits. This, however, increases the total number of random bits used, and further requires that the total number of players is at least 4 (as required by Implementation 2 of $OT$). While requiring at least 4 players is not an issue since we have another variant of the protocol for odd number of players, in order not to increase the total number of random bits used, we generate and distribute the random bits needed for the various $OT$ invocations in a more efficient way. That is, while each internal player still participates in 2 $OT$ invocations, we do not need totally separate 2 sets of random bits. Rather, it is sufficient to ensure that no player will receive two messages (of two different $OT$ invocations) that "use" the same random bit. The resulting protocol uses a total of 8 random bits and is applicable to any $n \geq 4$.

We now formally describe our protocol for $n \geq 4$ players, denoted $P_0, P_1, \ldots, P_{n-1}$. As indicated above, the high level structure of the protocol is the same as that of $\Pi_{\text{odd}}$, with some modifications, most notably a different way to produce and distribute the random bits.

**Initialization Phase.** In the initialization phase player $P_{n-1}$ tosses 4 random bits $u_0, u_1, u_2, u_4$ and defines a sequence of bits $r_0, r_1, \ldots, r_\ell$, for $\ell = 2(n-2)$, recursively as follows:[11]

---

[11] Here and in the following we sometimes abuse notation and consider indices that involve summations over both $\mathcal{N}$ and $\mathcal{F}_2$ (denoted with the operands $+$ and $\oplus$, respectively).

$r_0 = u_0, r_1 = u_1, r_2 = u_2, r_4 = u_4$, and

$$r_j = \begin{cases} r_{j-3+r_{j-1}} & j > 1, j \text{ odd} \\ r_{j-6+(1\oplus r_{j-4})} & j > 4, j \text{ even} \end{cases} . \tag{1}$$

Player $P_{n-1}$ then sends to each player $P_i$, $0 \le i \le n-2$ the two bits $r_{2i}, r_{2i+1}$.

In addition, player $P_0$ tosses 2 additional random bits $q_0$ and $q_1$. It sends $q_0$ to all odd players, and $q_1$ to all even players. $P_0$ also locally computes $y_0 = q_0 \oplus q_1 \oplus x_0$.

**Computation Phase.** This phase runs in $n-1$ rounds. The inductive invariant that we maintain is that at the end of round $i \ge 1$ player $P_i$ has the value $y_i = q_0 \oplus q_1 \oplus \Pi_{j=0}^{i} x_j$. In each round, the protocol will run an $OT$ protocol. We give a detailed description of the rounds below.

**Final Phase.** At the end of the computation phase, player $P_{n-1}$ has (by the inductive invariant) the value $y_{n-1} = q_0 \oplus q_1 \oplus \Pi_{j=0}^{n-1} x_j$. It sends this value to $P_0$ who xors it with $q_0 \oplus q_1$ to obtain $\Pi_{j=0}^{n-1} x_j = AND(x_0, x_1, \ldots, x_{n-1})$. Player $P_0$ then sends this value to all other players.

The following lemma gives the properties of the sequence of bits $r_j$, necessary both for the correctness of the protocol and for its privacy. The proof of this lemma is quite technical and the reader may wish to skip this proof.

**Lemma 9.** *For any $1 \le i \le n-2$, the five bits $r_j$, $2(i-1) \le j \le 2(i+1)$, are such that*

1. *$r_{2i+1} = r_{2(i-1)+r_{2i}}$.*
2. *The four bits $r_{2(i-1)}, r_{2(i-1)+1}, r_{2i}, r_{2(i+1)}$ are independent and uniformly distributed.*

*Proof.* We prove the lemma by induction on $i$. For the base of the induction ($i = 1$), observe that Point (2) is satisfied since $r_0, r_1, r_2, r_4$ are set to be $u_0, u_1, u_2, u_4$, respectively, and these are independent and uniformly distributed random bits tossed by Player $P_{n-1}$. As to Point (1), $r_3$ is set to be equal to $r_{r_2}$ according to Eq. (1) (first part, with $j = 3$).

We now prove the lemma for $i+1$ assuming it is correct for $i$. Note that the 5-tuple that corresponds to $i+1$ partially overlaps the 5-tuple that corresponds to $i$.

Point (1) holds for $i+1$ because by the first part of Eq. (1) (taking $j$ to be $2(i+1)+1$) $r_{2(i+1)+1} = r_{2i+1+r_{2(i+1)}}$.

As to Point (2), we consider both parts of Eq. (1) and the value of $r_{2i}$. There are two cases:

1. If $r_{2i} = 0$:
   - $r_{2i+1} = r_{2(i-1)}$           (taking $j = 2i + 1$ for the first part of Eq. (1)).
   - $r_{2(i+2)} = r_{2(i-1)+1}$  (taking $j = 2(i+2)$ for the second part of Eq. (1)).
2. If $r_{2i} = 1$:
   - $r_{2i+1} = r_{2(i-1)+1}$    (taking $j = 2i + 1$ for the first part of Eq. (1)).
   - $r_{2(i+2)} = r_{2(i-1)}$        (taking $j = 2(i+2)$ for the second part of Eq. (1)).

It follows that the 4-tuple of bits with indices $2i, 2i + 1, 2(i + 1)$ and $2(i + 2)$, i.e., the 4-tuple $(r_{2i}, r_{2i+1}, r_{2(i+1)}, r_{2(i+2)})$ is equal to either the 4-tuple $(r_{2i}, r_{2(i-1)}, r_{2(i+1)}, r_{2(i-1)+1})$ (if $r_{2i} = 0$) or to $(r_{2i}, r_{2(i-1)+1}, r_{2(i+1)}, r_{2(i-1)})$ (if $r_{2i} = 1$), which are two permutations of the same 4 bits. By the induction hypothesis, these 4 bits are independent and uniformly distributed. Hence also Point (2) holds for $i + 1$. $\qquad\square$

We now define how the computation phase is implemented so as to maintain the inductive invariant (and privacy, as we prove later). The invariant clearly holds at the end of round 0 (i.e., before the computation phase starts) since player $P_0$ has $x_0, q_0$, and $q_1$ and hence can compute $y_0$.

Similarly to protocol $\Pi_{\text{odd}}$, in round $1 \le i \le n - 2$ (but not in round $n - 1$, as is the case for $\Pi_{\text{odd}}$) players $P_{i-1}$ and $P_i$ engage in a 1-2 $OT$ protocol, using Implementation 1 described above. The values that Alice (i.e., player $P_{i-1}$) holds for the $OT$ protocol are $b_0 = q_{i \bmod 2}$ and $b_1 = y_{i-1}$. Observe that Alice has $b_0$ from the initialization phase, and $b_1$ by the inductive invariant. The selection bit of Bob (i.e., player $P_i$) is $s = x_i$. The random bits used for the $OT$ protocol are $r_{2(i-1)}$ and $r_{2(i-1)+1}$ held by Alice (player $P_{i-1}$) and $r_{2i}$ held by Bob (player $P_i$). Observe that $P_i$ and $P_{i-1}$ receive these bits from $P_{n-1}$ during the initialization phase. Further, by Lemma 9 the bits $r_{2(i-1)}$ and $r_{2(i-1)+1}$, held by player $P_{i-1}$, and the bit $r_{2i}$, held by player $P_i$, all satisfy the properties required for the $OT$ protocol to be correct (and private).

Finally, in round $i = n - 1$, players $P_{n-2}$ and $P_{n-1}$ engage in an $OT$ protocol as in previous rounds, but using Implementation 2 and using additional new random bits.[12] Specifically, $P_{n-1}$ is Bob of the OT protocol and $P_{n-2}$ is Alice; the helper players are $P_0$ ($H_0$) and $P_1$ ($H_1$), and we denote the two fresh random bits tossed by $P_{n-2}$ by $u_5$ and $u_6$ (see Sect. 4.1.2). The use of Implementation 2 of the $OT$ protocol in this round is the reason that the protocol described here works only for $n \ge 4$.

As in protocol $\Pi_{\text{odd}}$, let $v_i$ denote the output (i.e., the bit learned by Bob) of the $OT$ protocol run in the $i$-th round, $1 \le i \le n - 1$. It follows that at the end of the $OT$ protocol, if the value that player $P_i$ holds as input is $x_i = 0$, then it gets from player $P_{i-1}$ the value $v_i = q_{i \bmod 2}$, and if $x_i = 1$ it gets the value $v_i = y_{i-1} = q_0 \oplus q_1 \oplus \Pi_{j=0}^{i-1} x_j$.

Now, if $x_i = 0$ then $\Pi_{j=0}^i x_j = 0$, and player $P_i$ has $y_i$ by calculating $q_{i \bmod 2} \oplus q_{(i+1) \bmod 2}$, where the former is just $v_i$ and the latter is received from $P_0$ in the initialization phase. If $x_i = 1$ then $\Pi_{j=0}^i x_j = \Pi_{j=0}^{i-1} x_j$ and player $P_i$ just sets $y_i = v_i$.

The total number of random bits used in this protocol is 8: $u_0, u_1, u_2, u_4$ and $q_0, q_1$ are tossed by player $P_0$, and $u_5, u_6$ are tossed by player $P_{n-2}$.

It remains to prove that privacy is preserved with respect to all players. Intuitively, there are $n - 1$ invocations of the $OT$ protocol. Each internal player

---

[12] It is also possible to perform the $OT$ protocol of this round using Implementation 1 with a separate set of 3 random bits, tossed by another player, say player $P_0$, but this results in a larger total number of random bits for the protocol.

(i.e., all players except $P_0$ and $P_{n-1}$) participates in two $OT$ invocations, once as Alice and once as Bob, and the privacy property with respect to these players will follow from the properties of the sequence of bits $r_j$ (Lemma 9). We now prove that the protocol is private.

**Theorem 10.** *The AND protocol for $n \geq 4$ is private.*

*Proof.* We first prove the claim for players $P_i$, $1 < i < n - 2$, and then for the players having special roles, $P_0$, $P_1$, $P_{n-2}$, $P_{n-1}$.

For $1 < i < n - 2$, observe that player $P_i$ receives messages pertaining to exactly two $OT$ invocations, one in which it plays the role of Alice, and one where it plays the role of Bob (as implemented in Sect. 4.1.1). In addition, $P_i$ receives from player $P_0$ either the bit $q_0$ or the bit $q_1$ and, at the end of the protocol, the computed value of the function.

We prove the claim for $i$ even (the case of $i$ odd is analogous, switching the roles of $q_0$ and $q_1$). The messages player $P_i$ receives are:

1. During the initialization phase: bits $r_{2i}, r_{2i+1}, q_0$.
2. During the $OT$ protocol with player $P_{i-1}$ (i.e., when playing the role of Bob):
   - $q_1 \oplus r_{2(i-1)+j}$ and $y_{i-1} \oplus r_{2(i-1)+(1\oplus j)}$, where $j = x_i \oplus r_{2i}$.
3. During the $OT$ protocol with player $P_{i+1}$ (i.e., when playing the role of Alice):
   - $x_{i+1} \oplus r_{2(i+1)}$.
4. In the final phase, from player $P_0$, $AND(x_0, x_1, \ldots, x_{n-1})$.

Observing the seven messages received by $P_i$, one can verify that:

1. The messages received in Stage 1 are the bits $r_{2i}, r_{2i+1}, q_0$.
2. For the messages of Stage 2, we distinguish between two cases depending on the value of $x_i$.
   If $x_i = 0$ then the first message is $q_1 \oplus r_{2(i-1)+r_{2i}}$ and the second one is $y_{i-1} \oplus r_{2(i-1)+(1\oplus r_{2i})}$. In this case, the first message includes a xor operation with $q_1$, and the second one a xor operation with $r_{2(i-1)+(1\oplus r_{2i})}$.
   If $x_i = 1$ then the first message is $q_1 \oplus r_{2(i-1)+(1\oplus r_{2i})}$ and the second one is $y_{i-1} \oplus r_{2(i-1)+r_{2i}}$. In this case, the first message includes a xor operation with $r_{2(i-1)+(1\oplus r_{2i})}$ and the second one a xor operation with $q_1$ (since by the inductive invariant, $y_{i-1} = q_0 \oplus q_1 \oplus \Pi_{j=0}^{i-1} x_j$).
3. The message received in Stage 3 includes a xor operation with $r_{2(i+1)}$.
4. The message received in Stage 4 is the value of the function.

The last message (Stage 4) is, by definition, the value of the function. From the observations above, it follows that the distribution of the other 6 messages is the same as the distribution of the tuple $r_{2i}, r_{2i+1}, q_0, q_1, r_{2(i-1)+(1\oplus r_{2i})}, r_{2(i+1)}$ (if $x_i = 0$) or the tuple $r_{2i}, r_{2i+1}, q_0, r_{2(i-1)+(1\oplus r_{2i})}, q_1, r_{2(i+1)}$ (if $x_i = 1$). But, using Lemma 9, we can conclude that both of these 6-tuples are uniformly distributed over the $2^6$ possible binary vectors. Thus, privacy is preserved for all players $P_i$, $1 < i < n - 2$.

Similar arguments apply to the remaining four players. Let $\hat{b}_0 = q_{(n-1) \bmod 2} \oplus u_6$ and let $\hat{b}_1 = y_{n-2} \oplus u_6$ (recall that player $P_{n-2}$ tosses two

random bits $u_5, u_6$, to be used by the $OT$ protocol, Implementation 2, in round $n-1$).

Player $P_0$: Player $P_0$ receives the following messages: those listed under Stages 1 and 3 above; in round $n-1$ of the computation phase, when Implementation 2 of $OT$ is invoked, $P_0$ receives from $P_{n-2}$ either the message $\hat{b}_0 = q_{n-1 \bmod 2} \oplus u_6$ or the message $\hat{b}_1 = y_{n-2} \oplus u_6$ and from $P_{n-1}$ the message $x_{n-1} \oplus u_5 \oplus 1$; and from $P_{n-1}$, at the final phase, the message $y_{n-1} = q_0 \oplus q_1 \oplus \Pi_{j=0}^{n-1} x_j$. In addition, player $P_0$ has the values of the random bits $q_0$ and $q_1$ tossed by itself. Therefore, the messages received in Stages 1 and 3, as well as the messages received from $P_{n-2}$ and $P_{n-1}$, each includes a xor operation with an independent (uniformly distributed) random bit not known to $P_0$. The message received in the final phase is (together with $q_0$ and $q_1$) the value of the function. Hence, privacy with respect to $P_0$ holds.

Player $P_1$: Player $P_1$ receives the following messages: the messages listed under Stages 1, 3 and 4 above; in round $n-1$ of the computation phase, when Implementation 2 of $OT$ is invoked, $P_1$ receives from $P_{n-2}$ either the message $\hat{b}_0 = q_{n-1 \bmod 2} \oplus u_6$ or the message $\hat{b}_1 = y_{n-2} \oplus u_6$ and from $P_{n-1}$ the message $x_{n-1} \oplus u_5$. Therefore, the messages received in Stages 1 and 3, as well as the messages received from $P_{n-2}$ and $P_{n-1}$, each includes a xor operation with an independent (uniformly distributed) random bit not known to $P_1$. The message received in Stage 4 is the value of the function. Hence privacy with respect to $P_1$ holds.

Player $P_{n-2}$: The set of messages that player $P_{n-2}$ receives is a subset of the messages received by players $P_i$, $1 < i < n-2$. None of these messages depend on $u_5$ or $u_6$ tossed by $P_{n-2}$. The privacy with respect to player $P_{n-2}$ thus follows from the proof for $P_i$, $1 < i < n-2$.

Player $P_{n-1}$: Player $P_{n-1}$ receives a subset of the messages received by the players $P_i$, $1 < i < n-2$, namely those of Stage 1 and of Stage 4. In addition, it receives, while engaging in Implementation 2 of the $OT$ protocol with player $P_{n-2}$ and helpers $P_0, P_1$, the following messages:

(1) the messages $u_5$ and $u_6$ from player $P_{n-2}$,
(2) the message $M_0 \cdot (x_{n-1} \oplus u_5 \oplus 1)$ from player $P_0$, where $M_0$ is the message $P_0$ receives from $P_{n-2}$ in the $OT$ protocol, Implementation 2,
(3) the message $M_1 \cdot (x_{n-1} \oplus u_5)$ from player $P_1$, where $M_1$ is the message $P_1$ receives from $P_{n-2}$ in the $OT$ protocol, Implementation 2.

We now have four cases depending on the values of $x_{n-1}$ and $u_5$. In each of the four cases, the two messages received from $P_0$ and $P_1$ can be written as follows:

- $x_{n-1} = 0, u_5 = 0$: $P_{n-1}$ receives from $P_1$ the message 0, and from $P_0$ the message $M_0 = \hat{b}_0 = q_{(n-1) \bmod 2} \oplus u_6$.
- $x_{n-1} = 0, u_5 = 1$: $P_{n-1}$ receives from $P_0$ the message 0, and from $P_1$ the message $M_1 = \hat{b}_0 = q_{(n-1) \bmod 2} \oplus u_6$.
- $x_{n-1} = 1, u_5 = 0$: $P_{n-1}$ receives from $P_0$ the message 0, and from $P_1$ the message $M_1 = \hat{b}_1 = y_{n-2} \oplus u_6$.

  &minus; $x_{n-1} = 1, u_5 = 1$: $P_{n-1}$ receives from $P_1$ the message 0, and from $P_0$ the message $M_0 = \hat{b}_1 = y_{n-2} \oplus u_6$.

    It can be verified that, given the values of $x_{n-1}$ and of $AND(x_0, x_1, \ldots, x_{n-1})$, the distribution of the messages received by $P_{n-1}$ is identical in all four cases. Indeed, given the value $x_{n-1}$, the value of $u_5$ determines which of the two messages above is a constant, and which includes a xor operation with $q_{(n-1) \bmod 2}$. Now recall that $y_{n-2} = q_0 \oplus q_1 \oplus \Pi_{j=0}^{n-2} x_j$, thus the rest of the messages (except $AND(x_0, x_1, \ldots, x_{n-1})$) include a xor operation with a distinct random bit, other than $q_{(n-1) \bmod 2}$, all are uniformly distributed and independent. Hence, privacy is preserved with respect to $P_{n-1}$.    □

### 4.2.3  The Case of $n = 3$

This case can be slightly improved compared to the general case. We can privately compute the **and** of 3 players using 7 random bits instead of 8.

    The protocol is simple to define: run the protocol $\Pi_{\text{odd}}$, but fix the bit $q_1$ to be 0 (rather than it being a random bit).

    The correctness of the protocol clearly holds since it holds for $\Pi_{\text{odd}}$ with any choice of random bits. To see that privacy is still preserved with respect to all three players, observe that both player $P_0$ and player $P_1$ get $q_1$ in the original protocol ($P_0$ tosses it, and $P_1$ gets it in the initialization phase). Therefore, fixing it to 0 leaves the privacy with respect to these two players intact. As to player $P_2$, note that the $OT$ protocol performed between $P_1$ and $P_2$ does not change in the modified protocol. Therefore, if $x_2 = 0$ then $P_2$ gets $q_1$ (which is fixed to 0), and no other information. If $x_2 = 1$ then the only information $P_2$ gets is $q_0 \oplus q_1 \oplus \Pi_{j=0}^1 x_j = q_0 \oplus \Pi_{j=0}^1 x_j$, from which it can compute, using the bit $q_0$ that it got in the initialization phase, the value of $\Pi_{j=0}^1 x_j$. But this value can be inferred, in the case of $n = 3$ and $x_2 = 1$, from the value of the function and $x_2$, so privacy is preserved with respect to $P_2$ too.

## 5   Conclusions

We consider the randomness complexity of the information-theoretic multi-party private computation of the function **and**. We show that this computation cannot be done using a single random bit, thus giving the first non-trivial lower bound on the randomness complexity of the private computation of an explicit boolean function. We further give an improved upper bound on the randomness complexity of the private computation of **and**, thus approaching the exact determination of that measure for **and**. To the best of our knowledge, for no explicit function $f$ is the exact randomness complexity of the private computation of $f$ known (except for **xor**, which is trivially 1-random, and degenerate functions). We leave the exact determination of the randomness complexity of private computations of **and** for further research.

# References

1. Asharov, G., Lindell, Y.: A full proof of the BGW protocol for perfectly secure multiparty computation. J. Cryptology **30**(1), 58–151 (2017)
2. Beaver, D.: Precomputing oblivious transfer. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 97–109. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-44750-4_8
3. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2–4, 1988, Chicago, Illinois, USA, pp. 1–10 (1988)
4. Blundo, C., Galdi, C., Persiano, P.: Randomness recycling in constant-round private computations. In: Jayanti, P. (ed.) DISC 1999. LNCS, vol. 1693, pp. 140–149. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48169-9_10
5. Blundo, C., De Santis, A., Persiano, G., Vaccaro, U.: Randomness complexity of private computation. Comput. Complex. **8**(2), 145–168 (1999)
6. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2–4, 1988, Chicago, Illinois, USA, pp. 11–19 (1988)
7. Chor, B., Kushilevitz, E.: A zero-one law for Boolean privacy. SIAM J. Discrete Math. **4**(1), 36–47 (1991)
8. Chor, B., Kushilevitz, E.: A communication-privacy tradeoff for modular addition. Inf. Process. Lett. **45**(4), 205–210 (1993)
9. Damgård, I., Nielsen, J.B., Ostrovsky, R., Rosén, A.: Unconditionally secure computation with reduced interaction. In: Fischlin, M., Coron, J.-S. (eds.) EURO-CRYPT 2016. LNCS, vol. 9666, pp. 420–447. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_15
10. Damgård, I., Nielsen, J.B., Polychroniadou, A., Raskin, M.: On the communication required for unconditionally secure multiplication. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9815, pp. 459–488. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53008-5_16
11. Data, D., Prabhakaran, V.M., Prabhakaran, M.M.: Communication and randomness lower bounds for secure computation. IEEE Trans. Inf. Theory **62**(7), 3901–3929 (2016)
12. Gál, A., Rosén, A.: A theorem on sensitivity and applications in private computation. SIAM J. Comput. **31**(5), 1424–1437 (2002)
13. Gál, A., Rosén, A.: Omega(log n) lower bounds on the amount of randomness in 2-private computation. SIAM J. Comput. **34**(4), 946–959 (2005)
14. Goldreich, O.: Modern Cryptography, Probabilistic Proofs and Pseudorandomness, vol. 17 of Algorithms and Combinatorics. Springer, Berlin (1998)
15. Jakoby, A., Liśkiewicz, M., Reischuk, R.: Private computations in networks: topology versus randomness. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 121–132. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36494-3_12

16. Kushilevitz, E., Mansour, Y.: Randomness in private computations. SIAM J. Discrete Math. **10**(4), 647–661 (1997)
17. Kushilevitz, E., Ostrovsky, R., Rosén, A.: Characterizing linear size circuits in terms of privacy. J. Comput. Syst. Sci. **58**(1), 129–136 (1999)
18. Kushilevitz, E., Ostrovsky, R., Rosén, A.: Amortizing randomness in private multiparty computations. SIAM J. Discrete Math. **16**(4), 533–544 (2003)
19. Kushilevitz, E., Rosén, A.: A randomness-rounds tradeoff in private computation. SIAM J. Discrete Math. **11**(1), 61–80 (1998)
20. Nisan, N., Ta-Shma, A.: Extracting randomness: a survey and new constructions. J. Comput. Syst. Sci. **58**(1), 148–173 (1999)
21. Rosén, A., Urrutia, F.: A new approach to multi-party peer-to-peer communication complexity. In: Blum, A. (ed.) 10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10–12, 2019, San Diego, California, USA, vol. 124 of LIPIcs, pp. 64:1–64:19. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2019)