Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 71 (2003)

Privacy in Non-Private Environments*

Markus Bläser[1]† Andreas Jakoby[2] Maciej Liśkiewicz[2]‡
Bodo Manthey[2]§

[1] Institut für Theoretische Informatik, ETH Zürich
8092 Zürich, Switzerland
mblaeser@inf.ethz.ch

[2] Institut für Theoretische Informatik, Universität zu Lübeck
Ratzeburger Allee 160, 23538 Lübeck, Germany
jakoby/liskiewi/manthey@tcs.uni-luebeck.de

Abstract

We study private computations in information-theoretical settings on networks that are not 2-connected. Non-2-connected networks are "non-private" in the sense that most functions cannot privately be computed on them. We relax the notion of privacy by introducing lossy private protocols, which generalize private protocols. We measure the information each player gains during the computation. Good protocols should minimize the amount of information they lose to the players. Throughout this work, privacy always means 1-privacy, i.e. players are not allowed to share their knowledge. Furthermore, the players are honest but curious, thus they never deviate from the given protocol.

By use of randomness by the protocol the communication strings a certain player can observe on a particular input determine a probability distribution. We define the loss of a protocol to a player as the logarithm of the number of different probability distributions the player can observe. For optimal protocols, this is justified by the following result: For a particular content of any player's random tape, the distributions the player observes have pairwise fidelity zero. Thus the player can easily distinguish the distributions.

The simplest non-2-connected networks consists of two blocks that share one bridge node. We prove that on such networks, communication complexity and the loss of a private protocol are closely related: Up to constant factors, they are the same.

*An extended abstract of this work appeared in the Proceedings of the 10th Int. Conf. on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2004) [5].

†Work done while at the Institut f ̈ur Theoretische Informatik, Universit ̈at zu L ̈ubeck, Germany

‡On leave from Instytut Informatyki, Uniwersytet Wrocławski, Poland.

§Supported by DFG research grant RE 672/3.

ISSN 1433-8092

Then we study one-phase protocols, an analogue of one-round communication protocols. In such a protocol each bridge node may communicate with each block only once. We investigate in which order a bridge node should communicate with the blocks to minimize the loss of information. In particular, for symmetric functions it is optimal to sort the components by increasing size. Then we design a one-phase protocol that for symmetric functions simultaneously minimizes the loss at all nodes where the minimum is taken over all one-phase protocols.

Finally, we prove a phase hierarchy. For any k there is a function such that every $(k-1)$-phase protocol for this function has an information loss that is exponentially greater than that of the best k-phase protocol.

1 Introduction

Consider a set of players, each knowing an individual secret. They want to compute some function depending on their secrets. But after the computation, no player should know anything about the secrets of the other players except for what he is able to deduce from his own secret and the function value. This is the aim of *private computation* (also called *secure multi-party computation*). To compute the function, the players can send messages to each other using secure links.

An example for such a computation is the "secret voting problem": The members of a committee wish to decide whether the majority votes for yes or no. But after the vote nobody should know anything about the opinions of the other members, not even about the exact number of yes and no votes, except for whether the majority voted for yes or no.

If no group of at most t players can infer anything about the input bits that cannot be inferred from the function value and their own input bits, we speak of t-privacy.

Any Boolean function can privately (in the following we identify privately with 1-privately) be computed on any 2-connected network. Unfortunately, there are many Boolean functions, even simple ones like parity, disjunction, or conjunction, that cannot privately be computed if the underlying network is not 2-connected [6].

However, many real-world networks are not 2-connected and private computation is not possible. If the players in the network have to compute something but do not trust each other, there is a natural interest of the players in privacy. What can we do? We relax the notion of privacy: One cannot require that any player learns only what he is able to deduce from his own secret and the function value. Instead we require that any player learns as little as possible about the secrets of the other players (in an information-theoretical sense) while it is still possible to compute the function.

Bridge nodes are important when considering non-2-connected networks. For all non-bridge players we can guarantee that they do not learn anything except for what they can deduce from their own bit and the function value. Thus, the bridge players are the only players that are able to learn something more. The question is, how much the bridge players need to learn such that the function can be computed. The simplest setting is a network of two blocks with one bridge node in common. (A block is a maximal 2-connected subnetwork.) This reminds one of communication complexity with a man in the middle: Alice (one block) and Bob (another block) want to compute a function depending on their input while preventing Eve (the bridge node) from learning anything about their input. Unfortunately, Eve listens to the only communication channel between Alice and Bob. In terms of communication complexity, this problem had been examined by Modiano and Ephremedis [15, 16] and Orlitsky and El Gamal [19] under cryptographic security. In contrast, we deal with information-theoretical security, i.e. the computational power of the players is unrestricted. Furthermore, we are not interested in minimizing communication but in minimizing the information learned by any player. It turns out that there is a close relation between communication and privacy, at least in this special case.

1.1 Previous Results

Private computation was introduced by Yao [22]. He considered the problem under cryptographic assumptions. Private Computation with information-theoretical security has been introduced by Ben-Or et al. [3] and Chaum et al. [8]. Kushilevitz et al. [14] proved that the set of Boolean functions that have a circuit of linear size equals the set of functions that can privately be computed using only a constant number of random bits. Some of the simulation techniques used in this paper are based on their work.

Kushilevitz [12] and Chor et al. [9] considered private computations of integer-valued functions. They examined which functions can privately be computed by two players.

Franklin and Yung [11] used directed hypergraphs for communication and described those networks on which every Boolean function can privately be computed.

While all Boolean functions can privately be computed on any undirected 2-connected network, Bl¨aser et al. [6] completely characterized the class of Boolean functions that can still privately be computed, if the underlying network is connected but not 2-connected. In particular, no non-degenerate function can privately be computed if the network consists of three or more blocks. On networks with two blocks, only a small class of functions can privately be computed.

Chaum et al. [8] proved that any Boolean function can privately be computed, if at most one third of the participating players are dishonest, i.e. they are cheating. We consider the setting that all players are honest, i.e. they do not cheat actively but try to acquire knowledge about the input bits of the other players only by observing their communication. For this model, Ben-Or et al. [3] proved that any n-ary Boolean function can be computed $\left\lfloor \frac{n-1}{2} \right\rfloor$-private. Chor and Kushilevitz [10] showed that if a function can be computed at least $\frac{n}{2}$-private, then it can be computed n-private as well.

The idea of relaxing the privacy constraints has been studied to some extend in a cryptographic setting. Yao [22] examined the problem where it is allowed that the probability distributions of the messages seen by the players may differ slightly for different inputs, such that in practice the player should not be able to learn anything.

Leakage of information in the information-theoretical sense has been considered only for two parties yet. Bar-Yehuda et al. [2] studied the minimum amount of information about the input that must be revealed for computing a given function in this setting.

1.2 Our Results

We study the leakage of information for *multi-party* protocols, where each player knows only a single bit of the input.

Our first contribution is the definition of *lossy private protocols*, which is a generalization of private protocols in an information-theoretical sense (Section 2.3). Here and in the following, private always means 1-private. Throughout this work, we restrict ourselves to non-2-connected (in the sense of non-2-vertex-connected) networks that are still 2-edge-connected. Every block in such a network has size at least three and private computation within such a block is possible. We measure the information any particular player gains during the execution of the protocol in an information-theoretical sense. This is the *loss* of the protocol to the player. The players are assumed to be honest but curios. This means that they always follow the protocol but try to derive as much information as possible.

We divide lossy protocols into phases. Within a phase, a bridge player may exchange messages only once with each block he belongs to. Phases correspond to rounds in communication complexity but they are locally defined for each bridge player.

In the definition of lossy protocols, the loss of a protocol to a player is merely the logarithm of the number of different probability distributions on the communication strings a player can observe. We justify this definition in Section 3.2: For a protocol with minimum loss to a player P and any particular content of P's random tape, the different distributions P observes have pairwise fidelity zero, i.e. the support of any two probability distributions is disjoint. Thus, in order to gain information, P can distinguish the distributions from the actual communication he observes and does not need to sample.

The simplest non-2-connected network consists of two blocks that share one bridge node. In Section 4 we show that the communication complexity of a function f and the loss of a private protocol for f are intimately connected: Up to constant factors, both quantities are equal.

Then we study one-phase protocols. We start with networks that consist of d blocks that all share the same bridge player P. In a one-phase protocol, P can communicate only once with each block he belongs to. However, the loss of the protocol may depend on the order in which P communicates with the blocks. In Section 5, we show that the order in which P should communicate with the blocks to minimize the loss equals the order in which d parties should be ordered on a directed line when they want to compute the function with minimum communication complexity. Particularly for symmetric functions, it is optimal to sort the components by increasing size.

Then we design a one-phase protocol (Theorem 5.9), which has the remarkable feature that it achieves minimal loss at any node for symmetric functions. Hence, it simultaneously minimizes the loss for all nodes where the minimum is taken over all one-phase protocols.

In Section 6, we prove a phase hierarchy. For any k there is a function for which every $(k-1)$-phase protocol has an exponentially greater information loss than that of the best k-phase protocol.

We conclude with two examples. The first example shows that even for symmetric functions, the order of the communication may have an exponentially large influence on the loss of the protocol. The second example is a non-symmetric function computed on a network with two bridge nodes. We show that it is impossible to minimize the information loss simultaneously by one protocol for both bridge players. This observation shows that, in contrast to symmetric functions, there are non-symmetric functions that do not have optimal one-phase protocols.

1.3 Comparison of Our Results with Previous Work

One of the important features of the two-party case is that at the beginning each party has knowledge about one half of the input. In the multi-party case each player knows only a single bit of the input.

Kushilevitz [12] examined which integer-valued functions can privately be computed by two players. He showed that requiring privacy can result in exponentially larger communication costs and that randomization does not help in this model, not even to improve on the number of rounds. Chor et al. [9] considered multi-party computations of functions over the integers. They showed that the possibility of privately computing a function is closely related to its communication complexity, and they characterized the class of privately computable Boolean functions on countable domains. Neither Kushilevitz [12] nor Chor et al. [9] examined the problem how functions that cannot privately be computed can still be computed while maintaining as much privacy as possible.

Leakage of information in the information-theoretical sense has been considered only for two parties, each holding one n-bit input of a two-variable function. Bar-Yehuda et al. [2] investigated this for functions that are not privately computable. They defined measures for the minimum amount of information about the individual inputs that must be learned during the computation and proved tight bounds on these costs for several functions. Finally, they showed that sacrificing some privacy can reduce the number of messages required during the computation and proved that at the costs of revealing k extra bits of information any function can be computed using $O(k \cdot 2^{(2n+1)/(k+1)})$ messages.

The counterpart of the two-party scenario in the distributed setting that we consider is a network that consists of two complete networks that share one node connecting them. Simulating any two-party protocol on such a network allows the common player to gain information depending on the deterministic communication complexity of the function that should be evaluated. Hence and in contrast to the two-party case, increasing the number of bits exchanged does not help to reduce the knowledge learned by the player that is part of either block. An important difference between the two-party scenario, where two parties share the complete input, and a network consisting of two 2-connected components connected via a common player (the bridge player) is that in the latter we have somewhat like a "man in the middle" (namely the bridge player) who can learn more than any other player in either component, since he can observe the whole communication.

2 Preliminaries

2.1 Notations

For $i \in \mathbb{N}$, let $[i] := \{1, \ldots, i\}$. We define $\mathbb{B} = \{0, 1\}$.

Let $x = x_1 x_2 \ldots x_n \in \{0,1\}^n$ be a string of length n. Throughout the paper, we often use the string operation $x_{I \leftarrow \alpha}$ defined as follows: For $x \in \{0,1\}^n$, $I \subseteq [n]$, and $\alpha \in \{0,1\}^{|I|}$, $x_{I\leftarrow\alpha}$ is defined by

$$(x_{I\leftarrow\alpha})_i = \begin{cases} x_i & \text{if } i \notin I, \\ \alpha_j & \text{if } i \in I \text{ and } i \text{ is the } j\text{th smallest element in } I, \end{cases}$$

for all $i \in [n]$. In case $I = \{q\}$ is a singleton, we write $x_{q\leftarrow\alpha}$. For a function $f : \{0,1\}^n \rightarrow \{0,1\}$, a set of indices $I \subseteq [n]$, and a string $\alpha \in \{0,1\}^{|I|}$, $f_{I\leftarrow\alpha} : \{0,1\}^{n-|I|} \rightarrow \{0,1\}$ denotes the function obtained from f by specializing the positions in I to the values given by α, i.e. for all $x \in \{0,1\}^{n-|I|}$,

$$f_{I\leftarrow\alpha}(x) = f(({0^n}_{I\leftarrow\alpha})_{\overline{I}\leftarrow x}) ,$$

where $\overline{I} = [n] \setminus I$. For a string $x \in \{0,1\}^n$ and a set $I \subseteq [n]$, we define $x_I \in \{0,1\}^{|I|}$ as follows: For all $j \leq |I|$, $(x_I)_j = x_i$ if i is the jth smallest element in I.

An undirected graph $G = (V, E)$ is called *2-connected*, if the graph obtained from G by deleting an arbitrary node is still connected. For a set $U \subseteq V$, let $G|_U := (U, E|_U)$ denote the graph induced by U, where $E|_U = \{\{x,y\} \in E \mid x, y \in U\}$. A subgraph $G|_U$ is called a *block* of G, if $G|_U$ is 2-connected and there is no proper superset U' of U such that $G|_{U'}$ is 2-connected. We here consider a graph with two nodes and one edge connecting these two nodes as 2-connected. A graph is called *2-edge-connected* if after removal of one edge, the graph is still connected. Note that a graph is 2-edge-connected if it is connected and has no block of size 2. A node belonging to more than one block is called a *bridge node*. The other nodes are called *internal nodes*. The blocks of a graph are arranged in a tree structure. For more details on graphs, see e.g. Berge [4].

A Boolean function is called *symmetric*, if the function value depends only on the number of 1s in the input. See for instance Wegener [21] for a survey on Boolean functions.

2.2 Private Computations

We consider the computation of Boolean functions $f : \{0,1\}^n \to \{0,1\}$ on a network of n players. In the beginning, each player knows a single bit of the input x. Each player P_i is equipped with a random tape R_i. The players can send messages to other players via point-to-point communication using secure links where the link topology is given by an undirected graph $G = (V, E)$. When the computation stops, all players should know the value $f(x)$. The goal is to compute $f(x)$ such that no player learns anything about the other input bits in an information-theoretical sense except for the information he can deduce from his own bit and the result. Such a protocol is called private.

Definition 2.1 *Let C_i be a random variable of the communication string seen by player P_i, and let c be a particular string seen by P_i. A protocol $\mathcal{A}$ for computing a function f is* private with respect to player P_i *if for any pair of input vectors x and y with $f(x) = f(y)$ and $x_i = y_i$, for every c, and for every random string r provided to P_i,*

$$\Pr[C_i = c \mid R_i = r, x] \;=\; \Pr[C_i = c \mid R_i = r, y]\;,$$

where the probability is taken over the random strings of all other players. A protocol $\mathcal{A}$ is private *if it is private with respect to every player P_i.*

In the following, we use a strengthened definition of privacy of protocols: We allow only one player, say P_i, to know the result. The protocol has to be private with respect to P_i according to Definition 2.1. Furthermore, for all players $P_j \neq P_i$, for all inputs x, y with $x_j = y_j$, and for all random strings r we require $\Pr[C_j = c \mid R_j = r, x] = \Pr[C_j = c \mid R_j = r, y]$, independently of $f(x)$ and $f(y)$. In such a protocol, P_i is the only player that learns the function value. The other players do not learn anything.

This definition does not restrict the class of functions computable by private protocols according to Definition 2.1. Every function f in this class can be computed by a protocol $\mathcal{A}$ fulfilling the conditions above. To achieve this additional restriction, P_i generates a random bit r. Then we use a private protocol for computing $r \oplus f(x)$. Since the protocol used is private, no player except for P_i learns anything about the function value that cannot be derived from its own input bit.

2.3 Information Source

The definition of privacy basically states the following: The probability that a player P_i sees a specific communication string during the computation does not depend on the input of the other players. Thus, P_i cannot infer anything about the other inputs from the communication he observes.

If private computation is not possible since the graph is not 2-connected, it is natural to weaken the concept of privacy in the following way: We measure the information player P_i can infer from seeing a particular communication string. This leads to the concept of *lossy private protocols*. The less information any player can infer, the better the protocol is.

In the following, $c_1, c_2, c_3, \ldots$ denotes a fixed enumeration of all communication strings seen by any player during the execution of $\mathcal{A}$. (We could also use a fixed standard enumeration of all strings. In the latter case, we would get probability distributions with a finite support on countable probability spaces instead of probability distributions on finite spaces. The concepts arising would be completely the same.)

Definition 2.2 *Let C_i be a random variable of the communication string seen by player P_i while executing $\mathcal{A}$. Then for $a, b \in \{0,1\}$ and for every random string r provided to P_i, define the* information source *of P_i on a, b, and r as*

$$\mathcal{S}_{\mathcal{A}}(i,a,b,r) := \{(\mu_x(c_1), \mu_x(c_2), \ldots) \mid x \in \{0,1\}^n \wedge x_i = a \wedge f(x) = b\}$$

where $\mu_x(c_k) := \Pr[C_i = c_k | R_i = r, x]$ and the probability is taken over the random strings of all other players.

Basically $\mathcal{S}_{\mathcal{A}}(i,a,b,r)$ is the set of all different probability distributions on the communication strings observed by P_i when the input x of the players varies over all possible bit strings with $x_i = a$ and $f(x) = b$ and P_i's random tape is fixed to r.

The *loss* of a protocol $\mathcal{A}$ on a, b with respect to player P_i is

$$\ell = \max_r \log |\mathcal{S}_{\mathcal{A}}(i,a,b,r)| \,.$$

Thus the protocol looses ℓ bits of information to P_i. We call such a protocol ℓ-*lossy* on a, b with respect to P_i.

If a uniform distribution of the input bits is assumed, then the self-information of an assignment to the players $P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n$ is equal to $n-1$ [20]. In this case the maximum number of bits of information that can be extracted by P_i is $n-1$. If $\mathcal{A}$ is 0-lossy for all $a, b \in \{0,1\}$ with respect to P_i, then we say that $\mathcal{A}$ is *lossless* with respect to P_i. $\mathcal{A}$ is lossless to P_i if and only if $\mathcal{A}$ is private to P_i. Thus the notion of lossy private protocols generalizes the notion of private protocols.

Next we treat the loss to each player.

Definition 2.3 *A protocol $\mathcal{A}$ computing a function f in a network G is $\ell_{\mathcal{A}}$-lossy, with $\ell_{\mathcal{A}} : [n] \times \{0,1\}^2 \to \mathbb{R}_0^+$, if for all $a, b \in \{0,1\}$,*

$$\ell_{\mathcal{A}}(i,a,b) = \max_r \log |\mathcal{S}_{\mathcal{A}}(i,a,b,r)| \,.$$

Let f be an n-ary Boolean function and let $G = (V,E)$ be a network with $|V| = n$. We define $\ell_G : [n] \times \{0,1\}^2 \to \mathbb{R}_0^+$ by

$$\ell_G(i,a,b) := \min_{\mathcal{A}} \{\ell_{\mathcal{A}}(i,a,b) \mid \mathcal{A} \textit{ is an } \ell_{\mathcal{A}}\textit{-lossy protocol for } f \textit{ in } G\} \,.$$

The loss of a protocol $\mathcal{A}$ is *bounded by* $\lambda \in \mathbb{N}$, if $\ell_{\mathcal{A}}(i,a,b) \leq \lambda$ for all i, a, and b. $\ell_G(i,a,b)$ is obtained by locally minimizing the loss to each player P_i over all protocols. It is a priori not clear whether there is one protocol with $\ell_G(i,a,b) = \ell_{\mathcal{A}}(i,a,b)$ for all i,a,b. We show that this is the case for symmetric functions and one-phase protocols (as defined in Section 2.4).

Sometimes we will use the size of the information source instead of $\ell_{\mathcal{A}}$. Therefore, for a protocol $\mathcal{A}$, we define

$$\begin{array}{lcl} s_{\mathcal{A}}(i,a,b,r) & = & |\mathcal{S}_{\mathcal{A}}(i,a,b,r)| \;, \\ s_{\mathcal{A}}(i,a,b) & = & \max_r s_{\mathcal{A}}(i,a,b,r) \text{ , and} \\ s_{\mathcal{A}}(i,a) & = & s_{\mathcal{A}}(i,a,0) + s_{\mathcal{A}}(i,a,1) \;. \end{array}$$

By definition, $\ell_{\mathcal{A}}(i,a,b) = \log s_{\mathcal{A}}(i,a,b)$. If the underlying protocol is clear from the context, we omit the subscript $\mathcal{A}$. Let f be an n-ary Boolean function. For a network $G = (V,E)$ with $|V| = n$, we define $s_G(i,a,b) := \min_{\mathcal{A}} s_{\mathcal{A}}(i,a,b)$ and $s_G(i,a) := \min_{\mathcal{A}} s_{\mathcal{A}}(i,a)$ If a player P_i is an internal node of the network, then it is possible to design protocols that are lossless with respect to P_i (see Section 3.1). Players that are bridge nodes are in general able to infer some information about the input.

2.4 Phases in a Protocol

Without loss of generality we will assume that for any protocol the players communicate with each other in rounds such that in every round each player P_q may send one bit to one of its neighbors in the underlying communication network, or may receive one bit from one of its neighbors, or P_q may be idle. We call such protocols *synchronous*. The fact that each player receives or sends at most one bit per round is only made to simplify some of the following definitions. (If a player sends or receives more than one bit in a single round in a given protocol, then we can design a new protocol that fulfills this restriction by simulating this one round by several rounds and sending bits consecutively.) For a player P_q we encode a complete communication string as a sequence $c = c[1], c[2], \ldots, c[t]$ such that every item $c[i]$ completely describes the i-th communication round performed by P_q (in particular $c[i]$ encodes the name of the player P_q has communicated with in the i-th round and the content of the message).

We say that P_q who corresponds to a bridge node makes an *alternation* if he finishes the communication with one block and starts to communicate with another block. (This is well-defined, since each player receives or sends at most one bit per round.) Formally, we say that P_q makes an alternation between the round i and $i+1$, if there exists a subsequence $c[i-h], \ldots, c[i], c[i+1]$, with $h \geq 0$, such that $c[i-h]$ and $c[i+1]$ encode P_q's communication with players belonging to different blocks and, in case $h > 0$, the internal items $c[i-h+1], \ldots, c[i]$ encode P_q's idle communication period. During such an alternation, information can flow from one block to another.

The alternations of a communication string $c = c[1], c[2], \ldots, c[t]$ partition in a natural way the sequence into blocks: $c[1..i_1], c[(i_1+1)..i_2], \ldots, c[(i_{t-1}+1)..t]$. In this paper $c[i..j]$ denotes subsequence $c[i], c[i+1], \ldots, c[j]$. Formally we define by

$$\mathrm{block}_j(c) := c[(i_{j-1}+1)..i_j]$$

a subsequence of c such that

- $i_{j-1} = 0$ or P_q makes an alternation between the rounds i_{j-1} and $i_{j-1}+1$,
- $i_j = t$ or P_q makes an alternation between the rounds i_j and i_j+1, and
- P_q makes no alternation inside $\mathrm{block}_j(c)$.

Next we partition the work of P_q into phases as follows. P_q starts at the beginning of the first phase and it initiates a new phase when, after an alternation, it starts to communicate again with a block it already has communicated with previously in the phase.

Definition 2.4 *A protocol $\mathcal{A}$ is a k-*phase protocol *for a bridge node P_q if for every input string and contents of the random tapes of all players, P_q works in at most k phases. $\mathcal{A}$ is called a k-*phase protocol *if it is a k-phase protocol for every bridge node.*

The start and end round of each phase does not need to be the same for each player. Of particular interest are one-phase protocols. In such a protocol, each bridge player may only communicate once with each block he belongs to. Such protocols seem to be natural, since they have a local structure. Once the computation is finished in one block, the protocol will never communicate with this block again.

For k-phase protocols we define $\ell^k_G(i, a, b)$ and $s^k_G(i, a, b)$ in a similar way as $\ell_{\mathcal{A}}$ and s_G in the general case, but we minimize over all k-phase protocols.

During each phase a player communicates with at least two blocks. The order in which the player communicates within a phase can matter. The *communication order* σ_q of a bridge node P_q specifies the order in which P_q communicates with the blocks during the whole computation. Formally, σ_q is a finite sequence of (the indices of) blocks P_q belongs to and the length of σ_q is the total number of alternations made by P_q plus one. We say that a protocol is σ_q-ordered for P_q if for all inputs and all contents of the random tapes, the communication order of P_q is consistent with σ_q. Let $P_{q_1}, \ldots, P_{q_k}$ with $q_1 < q_2 < \ldots < q_k$ be an enumeration of all bridge players of a network G and $\sigma = (\sigma_{q_1}, \ldots, \sigma_{q_k})$ be a sequence of communication orders. We call a protocol *σ-ordered* if it is σ_{q_j}-ordered for every P_{q_j}. Finally, define

$$s_G(i, a, b, \sigma) := \min\{s_{\mathcal{A}}(i, a, b) \mid \mathcal{A} \text{ is a } \sigma\text{-ordered protocol for } f \text{ on } G\} .$$

2.5 Communication Protocols

For comparing the communication complexity of a certain function with the loss of private protocols while computing this function, we need the following definitions. For more about communication complexity, see e.g. Kushilevitz and Nisan [13].

Definition 2.5 *Let $f : \mathbb{B}^{m_1} \times \mathbb{B}^{m_2} \to \mathbb{B}$ be a Boolean function. Let $x \in \mathbb{B}^{m_1}$, $y \in \mathbb{B}^{m_2}$. Then x is the input for the first party, Alice, and y is the input for the second party, Bob.*

Let $\mathcal{B}$ be a deterministic two-party communication protocol for computing f according to the distribution described above. Then $\mathrm{DC}(\mathcal{B}, x, y)$ is the total number of bits exchanged by Alice and Bob when executing $\mathcal{B}$ on x and y. The deterministic communication complexity *of $\mathcal{B}$ is*

$$\mathrm{DC}(\mathcal{B}) = \max_{(x,y) \in \mathbb{B}^{m_1+m_2}} \mathrm{DC}(\mathcal{B}, x, y) .$$

$\mathrm{CP}(\mathcal{B})$ *denotes the number of different communication strings that occur, i.e. the* protocol partition number. *Finally,*

$$\begin{array}{rcl} \mathrm{DC}(f) & = & \min_{\mathcal{B} \textit{ for } f} \mathrm{DC}(\mathcal{B}) \textit{ and} \\ \mathrm{CP}(f) & = & \min_{\mathcal{B} \textit{ for } f} \mathrm{CP}(\mathcal{B}) . \end{array}$$

The protocol partition number is the number of leaves in the protocol tree. The messages sent so far determine whether Alice or Bob sends the next message, i.e. we require that in every round of communication the set of all possible messages is prefix-free.

We also consider multi-party communication with a referee, which is a generalization of two-party communication. Therefore, we consider Boolean functions $f : \mathbb{B}^{m_1} \times \mathbb{B}^{m_2} \times \ldots \times \mathbb{B}^{m_k} \to \mathbb{B}$. Let $A_1, \ldots, A_k$ be k parties and R be a referee, all with unlimited computational power. For computing $f(x_1, \ldots, x_k)$ for and $x_i \in \mathbb{B}^{m_i}$, the parties and the referee proceed as follows:

- Initially, A_i only knows x_i ($i \in [k]$). The referee R does not know anything about any x_i.
- The protocol proceeds in rounds. In a single round, R can communicate (i.e. receive or send a message) only with a single player.
- After finishing the communications, R computes the result of $f(x_1, \ldots, x_k)$.

Definition 2.6 *Let $\mathcal{B}$ be a deterministic communication protocol for computing f with k parties and a referee as described above. Then $\mathrm{DC}(\mathcal{B}, x_1, \ldots, x_k)$ is the total number of bits exchanged by the parties and the referee when executing $\mathcal{B}$ on $x_1, ldots, x_k$. The* deterministic communication complexity of $\mathcal{B}$

$$\mathrm{DC}^{\mathrm{R}}(\mathcal{B}) = \max_{(x_1,\ldots,x_k) \in \mathbb{B}^{m_1+\ldots+m_k}} \mathrm{DC}(\mathcal{B}, x_1, \ldots, x_k) .$$

$\mathrm{CP}^{\mathrm{R}}(\mathcal{B})$ *denotes the number of different communication strings that occur, i.e. the* protocol partition number. $\mathrm{DC}^{\mathrm{R}}(f)$ *and* $\mathrm{CP}^{\mathrm{R}}(f)$ *are defined analogously to Definition 2.5 by minimizing over all protocols for computing* f.

3 The Suitability of the Model

The aim of this section is to justify the definitions given in Section 2. We have restricted ourselves to considering networks that are 2-edge-connected. Thus, any block has size at least three. Every Boolean function can be computed with three or more players [3]. Hence, it is possible to compute functions privately within any block.

In the next subsection, we argue that it is sufficient to consider bridge players when talking about the loss of a protocol. In Subsection 3.2, we prove that in optimal protocols, the probability distributions observed by any player have pairwise fidelity zero. Thus, any player can easily distinguish the different probability distributions he observes.

3.1 Internal Players do not Learn Anything

Throughout this paper, we restrict ourselves to considering the loss of protocols to bridge players. The aim of this section is to justify this restriction. We prove that any protocol can be modified without increasing the loss to each bridge player such that no internal player (i.e. player who is not a bridge player) learns anything.

Theorem 3.1 *For any protocol $\mathcal{A}$ on an 2-edge-connected G there exists a protocol $\mathcal{A}'$ on G computing the same function as $\mathcal{A}$ such that*

1. *the loss of $\mathcal{A}'$ to each internal player is zero and*

2. *the loss of $\mathcal{A}'$ to each bridge player is at most the loss of $\mathcal{A}$ to this bridge player.*

Proof: We assume that $\mathcal{A}$ is synchronous. Thus, the communication string a player receives in any round depends on the input bits, the random tapes, and the communication prior to this round of all players. Let $C_{i,t}$ denote the communication received by P_i up to round t. We have

$$C_{i,t+1} = f_{i,t}(C_{1,t}, \ldots, C_{n,t}, x_1, \ldots, x_n, r_1, \ldots, r_n)$$

for some suitable function $f_{i,t}$. Since we only consider graphs where each block has size at least three, we can compute $f_{i,t}$ privately according to the protocol of Kushilevitz et al. [14] such that for any $i \in [n]$ and any round t we have the following properties:

- If P_i is an internal player, then he knows $C_{i,t}$ masked by sufficiently many random bits while some other player knows these random bits.

- If P_i is a bridge player, he knows $C_{i,t}$.

The protocol $\mathcal{A}'$ presented is clearly lossless with respect to any internal player. Furthermore, the loss to any bridge player is the same as in the protocol $\mathcal{A}$. □

3.2 Extracting Information from Probability Distributions

We consider arbitrary 1-connected networks. Let f be a Boolean function and $\mathcal{A}$ be a protocol for computing f on a 1-connected network G. Let P_q be a bridge player of G, $a, b \in \mathbb{B}$, and r_q be the random string provided to P_q. We define

$$X = \{x \in \mathbb{B}^n \mid x_q = a \wedge f(x) = b\}$$

and for any communication string c

$$\psi(c) = \{x \in X \mid \mu_x(c) > 0\} ,$$

where $\mu_x(c) = \Pr[C_q = c \mid R_q = r_q, x]$. For every communication string c that can be observed by P_q on some input $x \in X$, P_q can deduce that $x \in \psi(c)$. If $s_{\mathcal{A}}(q, a, b) = s_G(q, a, b) = 1$, then we have either $\psi(c) = X$ or $\psi(c) = \emptyset$. Thus P_q does not learn anything in this case.

Theorem 3.2 *If $s_G(q, a, b) > 1$, then for any protocol $\mathcal{A}$ and every communication string c that can be observed by P_q on input $x \in X$, $\psi(c)$ is a non-trivial subset of X, i.e. $\emptyset \neq \psi(c) \subsetneq X$, and there exist at least $s_G(q, a, b)$ different such sets. Hence, from seeing c on $x \in X$, P_q always gains some information and there are at least $s_G(q, a, b)$ different pieces of information that can be extracted by P_q on inputs from X.*

The next result says that $s_G(q, a, b)$ is a tight lower bound on the number of pieces of information: the lower bound is achieved when performing an optimal protocol on G. Let μ and μ' be two probability distributions over the same set of elementary events. The *fidelity* is a measure for the similarity of μ and μ' (see e.g. Nielsen and Chuang [17]) and is defined by

$$F(\mu, \mu') = \sum_c \sqrt{\mu(c) \cdot \mu'(c)} .$$

Theorem 3.3 *If $\mathcal{A}$ is an optimal protocol for player P_q on a and b, i.e. $s_{\mathcal{A}}(q, a, b) = s_G(q, a, b)$, then for every random string r_q and all probability distributions $\mu \neq \mu'$ in $\mathcal{S}_{\mathcal{A}}(q, a, b, r_q)$ we have $F(\mu, \mu') = 0$.*

Theorem 3.2 follows directly from the Lemmas 3.4 and 3.5 below. Theorem 3.3 follows from Lemma 3.6.

Lemma 3.4 *Assume $\mathcal{A}$ is a protocol for computing f. Then we have the following implications for every communication string $\hat{c}$:*

(i) if $\psi(\hat{c}) = \emptyset$, then for all $x \in X$ we have $\mu_x(\hat{c}) = 0$ and

(ii) if $\psi(\hat{c}) = X$, then $s_G(q, a, b) = 1$.

Proof: Item (i) follows from the definition of ψ in a straight-forward way. To prove Item (ii) assume that there exists a communication string $\hat{c}$ with $\psi(\hat{c}) = X$. From Lemma 4.7, for $w = \hat{c}$ we can construct a communication protocol $\mathcal{B}$ for computing $f_{q \leftarrow a}$ such that $\mathcal{B}$ on every $x \in X$ generates the same communication string. Using the simulation presented in the proof of Lemma 4.6 we get $s_G(q, a, b) \leq 1$. □

Lemma 3.5 *Let $\mathcal{S}_{\mathcal{A}}(q, a, b, r_q) = \{\mu_1, \mu_2, \ldots, \mu_m\}$. Let $\mathcal{M} = \{\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_m\}$ be a set of communication strings such that for every $i \in [m]$, $\hat{c}_i$ is the lexicographically first string in $\{c \mid \mu_i(c) > 0\}$. Then $|\mathcal{M}| \geq s_G(q, a, b)$ and for every pair of different $\hat{c}_i, \hat{c}_j$ we have $\psi(\hat{c}_i) \neq \psi(\hat{c}_j)$.*

Proof: Let Γ denote the alphabet for the communication strings and let $\gamma \in \Gamma$ be the lexicographically first symbol in Γ. Denote by τ the maximum length of communication strings $c_1, c_2, c_3, \ldots$. From Lemma 4.7, we get that for $w = \gamma^\tau$, we obtain the communication protocol $\mathcal{B}$ for computing $f_{q \leftarrow a}$ such that the number of different communication strings of $\mathcal{B}$ on all $x \in X$ is equal to $|\mathcal{M}|$. Using the simulation presented in the proof of Lemma 4.6 we get $s_G(q, a, b) \leq |\mathcal{M}|$.

To prove the second part of the lemma, note that for any $\hat{c}_i$ we have $\psi(\hat{c}) \neq \emptyset$. Now assume that $\hat{c}_i, \hat{c}_j$ are two different communication strings with $\psi(\hat{c}_i) = \psi(\hat{c}_j)$. It follows that for all $x, x' \in X$ we have $\mu_x(\hat{c}_i) > 0$ if and only if $\mu_{x'}(\hat{c}_i) > 0$. Hence

$$\hat{c}_i, \hat{c}_j \in \{c \mid \mu_i(c) > 0\} \cap \{c \mid \mu_j(c) > 0\}\,.$$

Because we have chosen $\hat{c}_i$ as the lexicographically first element in $\{c \mid \mu_i(c) > 0\}$ and $\hat{c}_j$ as the lexicographically first element in $\{c \mid \mu_j(c) > 0\}$, by the property above we have both $\hat{c}_i \leq_{\text{lex}} \hat{c}_j$ and $\hat{c}_j \leq_{\text{lex}} \hat{c}_i$, where $\leq_{\text{lex}}$ means lexicographically smaller. Hence $\hat{c}_i = \hat{c}_j$, a contradiction. □

Lemma 3.6 *Let $\mathcal{S}_{\mathcal{A}}(q, a, b, r_q) = \{\mu_1, \mu_2, \ldots, \mu_m\}$ and assume that for some $i \neq j \in [m]$ we have $F(\mu_i, \mu_j) > 0$. Then $s_G(q, a, b) < m$.*

Proof: Assume that for some $i \neq j \in [m]$ we have $F(\mu_i, \mu_j) > 0$ and let $\hat{c}$ be such a communication string with $\mu_i(\hat{c}), \mu_j(\hat{c}) > 0$. From Lemma 4.7 we get that for $w = \hat{c}$ we can construct a communication protocol $\mathcal{B}$ for $f_{q \leftarrow a}$ such that the number of different communication strings of $\mathcal{B}$ on all $x \in X$ is smaller or equal to $m - 1$. Using the simulation presented in the proof of Lemma 4.6 we get $s_G(q, a, b) \leq m - 1$. □

4 Communication Complexity and Private Computation

In this section we investigate the relations between deterministic communication complexity and the minimum size of an information source in a connected network with one bridge node. To distinguish between protocols in terms of communication complexity and protocols in terms of private computation, we will call the former communication protocols.

4.1 Two-Party Model

The communication complexity of two-party protocol and the protocol partition number are closely related.

Lemma 4.1 (Kushilevitz and Nisan [13]) $\log(\mathrm{CP}(f)) \leq \mathrm{DC}(f) \leq 3 \cdot \log(\mathrm{CP}(f))$.

In this subsection we investigate the relation between the protocol partition number and the size of an information source on graphs G of n nodes that consist of two blocks sharing one bridge node P_q. Let $m_1 + 1$ be the size of the first block and $m_2 + 1$ be size of the second one.

Let $f : \mathbb{B}^n \to \mathbb{B}$ be an arbitrary function. We will relate the minimum size of an information source $s_G(q, a)$ for f and the optimum partition number for function $f_{q\leftarrow a}$, for any $a \in \mathbb{B}$. In the model of private computation the input bits are distributed among n players whereas the input bits in a communication protocol are distributed among the two parties. We identify the input $\vec{a}$ of Alice with the m_1 input bits known by players of the first block of the network and the input $\vec{b}$ of Bob with the m_2 input bits known by the players of the second block. Additionally, we assume that the value of P_q's input bit x_q is known by both Alice and Bob, so they know which function to compute: $f_{q\leftarrow 0}$ or $f_{q\leftarrow 1}$.

Lemma 4.2 *For $a \in \mathbb{B}$, we have $s_G(q, a) \leq \mathrm{CP}(f_{q\leftarrow a})$.*

Proof: Consider an optimal deterministic protocol for computing $f_{q\leftarrow a}$. Then there are two functions $A : \mathbb{B}^{m_1} \times \mathbb{B}^\star \to \mathbb{B}^\star$ and $B : \mathbb{B}^{m_2} \times \mathbb{B}^\star \to \mathbb{B}^\star$ that describe the messages sent by Alice and Bob, respectively:

$$w_i = \begin{cases} \lambda & \text{if } i = 0, \\ w_{i-1}A(y_1, w_{i-1}) & \text{if } i > 0 \text{ is odd, and} \\ w_{i-1}B(y_2, w_{i-1}) & \text{if } i > 0 \text{ is even.} \end{cases}$$

By an appropriate encoding, we can assert that all messages sent are of equal length and that the number of rounds is the same for all inputs. This does not increases the communication size.

To compute f player P_q first broadcasts the value of x_q to the remaining players to inform them which functions should be computed next. Note that both A and B may depend on a. Then P_q computes the values w_i iteratively using A and B. The both functions can privately be computed in the first and second block, respectively, while P_q is the only player who knows the w_is. P_q computes finally the result. The number of different probability distributions observable by P_q on the different inputs equals $\mathrm{CP}(f_{q\leftarrow a})$ by construction.

Broadcasting the value of the input bit seems to be very unusual in the context of private computations. In fact, this implies that the above protocol is not private with respect to the internal players. However, using Theorem 3.1, one can easily modify our protocol in such a way that it becomes private with respect to all internal players. □

Lemma 4.3 *For $a \in \mathbb{B}$, we have $\mathrm{CP}(f_{q\leftarrow a}) \leq s_G(q, a)$.*

Proof: Let $P_1, \ldots, P_q$ and $P_q, \ldots, P_n$ be the players of the first and second block, respectively. Let $\mathcal{A}$ be a protocol for computing f that is private with respect to all players except for P_q and such that the size $s_{\mathcal{A}}(q, a)$ is minimal.

We construct a communication protocol by simulating $\mathcal{A}$ and searching the lexicographical minimal communication sequence for P_q that has positive probability.

Let $c = c[1], c[2], \ldots, c[k]$ be the communication P_q has sent so far, where $c[i]$ for odd i is received by the first block and $c[i]$ for even i is received by the second block. Let $r_1, \ldots, r_n$ be the content of the random tape of player $P_1, \ldots, P_n$, respectively. Without loss of generality, we fix P_q's random tape to some fixed value. Thus,

$$c[i] = \begin{cases} B(c[1], \ldots, c[i-1], x_q, \ldots, x_n, r_{q-1}, \ldots, r_n) & \text{if } i \text{ is even and} \\ A(c[1], \ldots, c[i-1], x_1, \ldots, x_q, r_1, \ldots, r_{q-1}) & \text{if } i \text{ is odd,} \end{cases}$$

where A and B can be evaluated by Alice and Bob, respectively.

Let $\mathcal{R}_A^0$ and $\mathcal{R}_B^0$ be the sets of possible contents of $R_1, \ldots, R_{q-1}$ and $R_{q+1}, \ldots, R_n$, respectively. We iteratively restrict these sets. Therefore, let $\mathcal{R}_A^i$ and $\mathcal{R}_B^i$ be the possible contents after receiving or sending $c[i]$.

We simulate the private protocol as follows: For odd i, Alice computes $c[i]$ as

$$c[i] = \min\{A(c[1], \ldots, c[i-1], x_1, \ldots, x_q, r_1, \ldots, r_{q-1}) \mid (r_1, \ldots, r_{q-1}) \in \mathcal{R}_A^{i-1}\}\ .$$

Thus, $c[i]$ is the lexicographical minimal communication string that can occur on the given input while the previous communication has been observed. Then Alice computes $\mathcal{R}_A^i \subseteq \mathcal{R}_A^{i-1}$ as the set of possible contents of the random tapes that result in $c[i]$. Furthermore, we have $\mathcal{R}_B^i = \mathcal{R}_B^{i-1}$.

For even i, Bob computes $c[i]$ as

$$c[i] = \min\{B(c[1], \ldots, c[i-1], x_q, \ldots, x_n, r_{q+1}, \ldots, r_n) \mid (r_{q+1}, \ldots, r_n) \in \mathcal{R}_B^{i-1}\}\ .$$

and $\mathcal{R}_B^i \subseteq \mathcal{R}_B^{i-1}$ as the set of possible contents of the random tapes that result in $c[i]$. Furthermore, we have $\mathcal{R}_A^i = \mathcal{R}_A^{i-1}$.

After computation, the message $c[i]$ is sent to the other party.

If a party eventually knows the function value, the party sends it to the other party. Then the communication stops. This last message will be marked appropriately. Thus, the messages observed are prefix-free.

It remains to show that whenever Alice and Bob generate two different communication sequences on two inputs x and x', then P_q observes two different probability distributions on x and x'. Assume that we observe c and c' on x and x', respectively. Then there exists some k such that $c[k] \neq c'[k]$. Without loss of generality, we assume that $c[k] < c'[k]$. Then the probability of sending $c[k]$ after $c[1], \ldots, c[k-1]$ on x' must be zero, since otherwise our protocol would favor $c[k]$ over $c'[k]$. Thus, the two probability distributions differ. □

Due to the construction, we obtain from a communication protocol acting in k rounds a private protocols that needs at most $\lceil \frac{k+1}{2} \rceil$ phases. Analogously, we obtain from a k-phase private protocol a communication protocol that needs at most $2k-1$ rounds.

From the above lemmas, we immediately get the following theorem.

Theorem 4.4 *Let $f : \mathbb{B}^{m_1} \times \mathbb{B} \times \mathbb{B}^{m_2} \to \mathbb{B}$ be a function, $f(x, z, y)$ and let $a \in \mathbb{B}$ be arbitrary.*

- *Assume that Alice knows x and a and Bob knows y and a. Assume that $f_{q \leftarrow a}$ can be computed by a communication protocol with protocol partition number C. Then f can be computed on a graph consisting of two blocks of size $m_1 + 1$ and $m_2 + 1$, where x and y are distributed among the first and second block, respectively, and the common bridge player P_q knows a. This can be done with $s_G(q, a) \leq C$.*

- *Consider a graph consisting of two blocks of size $m_1 + 1$ and $m_2 + 1$. The bits x and y are distributed among the nodes of the first and second block, respectively, while the common bridge node knows a. If f can be computed with $s_G(q, a) \leq C$, the $f_{q \leftarrow a}$ can be computed by a communication protocol with protocol partition number bounded by C.*

From the above theorem and Lemma 4.1, we get the following corollary.

Corollary 4.5 *Let $f : \mathbb{B}^m \times \mathbb{B} \times \mathbb{B}^n \to \mathbb{B}$ be a function, $f(x, z, y)$. Let $a \in \mathbb{B}$ be arbitrary. If, in the models described in the above theorem, $f_{q \leftarrow a}$ can be computed with communication complexity c, then f can be computed with $s_G(q, a) \leq c$. If f can be computed with $s(q, a) = \lambda$, then $f_{q \leftarrow a}$ can be computed with communication complexity bounded by $3 \cdot \lambda$.*

4.2 Multi-Party with Referee

In this section we generalize our previous results to multi-party communication. We generalize Lemmas 4.2 and 4.3 by showing that similar bounds hold if we compare the information source of a bridge player that is connected to more than two blocks with the size of a communication protocol that makes use of a referee. Through the section we assume that graph G of n nodes consist of k blocks sharing the bridge node P_q. Let $m_1, \ldots, m_k$, with $1 + m_1 + m_2 + \ldots + m_k = n$, be the sizes of connected subgraphs $G_1, \ldots, G_k$ obtained from G after removing P_q and let $f : \mathbb{B}^n \to \mathbb{B}$ be an arbitrary function. We will relate $s_G(q, a)$ for f and the optimum partition number CP^{R} for function $f_{q \leftarrow a}$, for any $a \in \mathbb{B}$. In the model of private computation the input bits are distributed among n players and the input bits in a communication protocol are distributed among the k parties. We identify the input $\vec{v}_i$ of the ith party with the m_i input bits known by players of G_i. We assume that the value of the input bit x_q is known by all parties.

Lemma 4.6 *For $a \in \mathbb{B}$ we have $s_G(q, a) \le \mathrm{CP}^{\mathrm{R}}(f_{q \leftarrow a})$.*

Proof: Let $\mathcal{B}$ be a deterministic communication protocol for $f_{q \leftarrow a}$. We construct a protocol that is private with respect to all players except for bridge players. Furthermore, we show that the size of the information source of P_q is bounded by $\mathrm{CP}^{\mathrm{R}}(\mathcal{B})$.

Since $\mathcal{B}$ is deterministic, there exist k functions $T_i : \mathbb{B}^{m_i} \times \mathbb{B}^\star \to \mathbb{B}^\star$ for $i \in [k]$ and a function $B : \mathbb{B}^\star \to [k]$ such that the messages exchanged in successive rounds between R and $A_1, \ldots, A_k$ according to $\mathcal{B}$ can be computed by evaluating T_i and B as follows:

$$w_j := \begin{cases} \lambda & \text{if } j = 0\,, \\ w_{j-1} T_{B(w_{j-1})}(x_{B(w_{j-1})}, w_{j-1}) & \text{if } j > 0\,. \end{cases}$$

$B(w_{j-1})$ determines the party A_i the referee wants to talk to in round j after receiving the communication string w_{j-1}. $T_i(x_i, w_{j-1})$ determines the corresponding communication string exchanged in round j between R and $A_{B(w_{j-1})}$.

The function B can always be evaluated by the bridge player P_q and $T_{B(w_{j-1})}$ can be computed on the $B(w_{j-1})$-th block and the players that are reachable from the players in the $B(w_{j-1})$-th block without passing P_q such that only P_q knows the result of the computation and no internal player learns anything. By iterating these computations, P_q can generate the complete communication sequence and finally compute the result.

The distribution of the communication seen by P_q is uniquely determined by the communication sequence of the communication protocol, since it does not depend on the random strings of the players. From this observation, the lemma follows. □

Next we show how we can simulate the computation of a protocol $\mathcal{A}$ by a communication protocol $\mathcal{B}$ with a referee. The simulation works analogously to the simulation in Lemma 4.3. Additionally, the simulation allows us to specify one distinguished communication string w to be used, if this string has positive probability for the input of $\mathcal{A}$.

For a communication string w we define the weighted lexicographic order $\leq_{\text{lex}}^{w}$ as follows. Let c_1, c_2 be arbitrary communication strings. Without loss of generality we assume that any protocol sends as a last message a unique message indicating the end of the computation. Hence, particularly neither w can be a prefix of c_i nor c_i a prefix of w. Denote by t, ℓ_1, ℓ_2 the numbers of blocks of w, c_1, and c_2, respectively. Let $\ell = \min\{t, \ell_1, \ell_2\}$. Define

$$c_1 \leq_{\text{lex}}^{w} c_2 \iff \begin{cases} c_1 = w & \text{or} \\ \exists i \leq \ell & [\forall j \leq i : \text{block}_j(c_1) = \text{block}_j(w) = \text{block}_j(c_2)] \text{ and} \\ & [\text{block}_{i+1}(c_1) = \text{block}_{i+1}(w) \neq \text{block}_{i+1}(c_2) \text{ or} \\ & \ \text{block}_{i+1}(c_1) \neq \text{block}_{i+1}(w) \neq \text{block}_{i+1}(c_2) \text{ or} \\ & \ \text{block}_{i+1}(c_1) \leq_{\text{lex}} \text{block}_{i+1}(c_2)]\ , \end{cases}$$

where $\leq_{\text{lex}}$ is the common lexicographical ordering of two strings. Note, the communication string w is always the minimum string with respect to the order $\leq_{\text{lex}}^{w}$.

Lemma 4.7 *For every $a \in \mathbb{B}$ and every protocol $\mathcal{A}$ for computing a function f there exists a communication protocol $\mathcal{B}$ computing $f_{q\leftarrow a}$ with*

$$\text{CP}^{\text{R}}(\mathcal{B}) \leq s_{\mathcal{A}}(q, a)\ .$$

Additionally, $\mathcal{B}$ has the following properties. Let $X = \{x \in \mathbb{B}^n : x_q = a\}$. Then $\mathcal{B}$ starting with parameters w and r_q, where w is an arbitrary communication string, and r_q is P_q's random bit string, simulates $\mathcal{A}$ such that

1. *if $\mu_x(w) > 0$ for some $x \in X$ then for every x', with $\mu_{x'}(w) > 0$, the communication string of $\mathcal{B}$ during the simulation on x' is the same as during the simulation on x;*

2. *for any $x, x' \in X$ if $\hat{c}$ is the minimum communication sequence with respect to the order $\leq_{\text{lex}}^{w}$ in both sets $\{c : \mu_x(c) > 0\}$ and $\{c : \mu_{x'}(c) > 0\}$ then the communication string of $\mathcal{B}$ during the simulation on x is the same as during the simulation on x'.*

Proof: Let $V_1, \ldots, V_k$ be a partition of all players except for P_q into subsets of maximum cardinality, such that for each set V_k and every pair $P_i, P_j \in V_k$ the player P_i is reachable from P_j without passing P_q. Let r_q be P_q's random bits and w be an arbitrary communication string. We construct a communication protocol by simulating $\mathcal{A}$ and searching the minimal communication sequence according to $\leq_{\text{lex}}^{w}$ for P_q that has positive probability.

Let c be the communication string observed by P_q for a fixed content $r_1, \ldots, r_n$ of the random tapes and input x. Every block sequence $\text{block}_j(c)$ of c is associated with a subset V_k and can deterministically be computed from the contents of the random tapes of the players in $V_k \cup \{P_q\}$, the input of these players, and $\mathrm{block}_i(c)$ with $i < j$. Analogously, the index d of the subset V_d that is associated to the block sequence $\mathrm{block}_{j+1}(c)$ can be determined from r_q, x_q, and the subsequences $\mathrm{block}_i(c)$ with $i \leq j$. Let $h(r_q, x_q, \mathrm{block}_1(c) \ldots \mathrm{block}_j(c))$ be the function that determines this index.

We say that a string c' is a valid prefix, if it can be extended to a communication string c, i.e. $c = c'u$ for some string u such that $\mu_x(c) > 0$ and P_q makes an alternation in c between c' and u.

Let $\mathcal{R}_i^0$ be the sets of all possible contents of the random tapes of the players in V_i and let $\alpha_0 = \lambda$ be the empty string. Furthermore, let x^i be the input of the players in V_i. We simulate the private protocol $\mathcal{A}$ as follows: Initially, referee R sends r_q and x_q to all parties $A_1, \ldots, A_k$. Then we do the following iteratively for $j = 1, 2, \ldots$:

1. R computes index $i_j = h(r_q, x_q, (\alpha_0, \ldots, \alpha_{j-1}))$ and sends $\alpha_0, \ldots, \alpha_{j-1}$ to the party A_{i_j}.

2. The party A_{i_j} determines the set H_j of all strings α such that α is a block and $\alpha_0, \ldots, \alpha_{j-1}, \alpha$ is a valid prefix of a communication string where the content of the random tapes of the players in V_{i_j} is in $\mathcal{R}_{i_j}^{j-1}$, the inputs of these players are given by x^{i_j}, the content of P_q's random tape is r_q, and P_q's input is x_q. A_{i_j} chooses $\alpha_j \in H_j$ such that for any $\alpha \in H_j$

$$\alpha_0, \ldots, \alpha_{j-1}, \alpha_j \leq_{\text{lex}}^{w} \alpha_0, \ldots, \alpha_{j-1}, \alpha$$

and $\mathcal{R}_{i_j}^{j} \subseteq \mathcal{R}_{i_j}^{j-1}$ as the set of all possible contents of the random tapes of the players in V_{i_j} such that the prefix of the communication string observed by P_q is $\alpha_1, \ldots, \alpha_{j-1}, \alpha_j$. Finally, A_{i_j} sends α_j to the referee R.

3. Each party $A_k \neq A_{i_j}$ chooses $\mathcal{R}_k^j = \mathcal{R}_k^{j-1}$.

To get a communication protocol, the parties $A_1, \ldots, A_\ell$, and R iteratively compute i_j, α_j, and $\mathcal{R}_i^j$ until R determines the end of the simulation. The correctness of this protocol follows from the correctness of the private protocol.

It remains to show that whenever $A_1, \ldots, A_k$, and R generate two different communication strings for two different inputs $x = (x^1, \ldots, x^k)$ and $x' = ((x')^1, \ldots, (x')^k)$, then the two corresponding inputs x and x' for the protocol $\mathcal{A}$ instantiate two different distributions μ_x and $\mu_{x'}$ where the lexicographically minimal string according to the ordering $\leq_{\text{lex}}^{w}$ with positive probability in μ_x differs from the corresponding string with positive probability in $\mu_{x'}$.

Let $c = \alpha_1, \ldots, \alpha_\ell$ be the communication string computed during the simulation on x and analogously let $c' = \alpha'_1, \ldots, \alpha'_{\ell'}$ be the communication string computed during the simulation on x'. Note that we can assume that neither c is a prefix of $\acute{c}$ nor c' is a prefix of c.

Let i_0 be minimal such that $\alpha_{i_0} \neq \alpha'_{i_0}$ and for all $i < i_0$ $\alpha_i = \alpha'_i$. In the following we assume that $c <_{\text{lex}}^{w} c'$. We distinguish two cases:

1. Assume that α_{i_0} is not a prefix of α'_{i_0}. By our construction of the substrings α_{i_0} and α'_{i_0}, it follows that if $\mu_{x'}(\alpha_1, \ldots, \alpha_{i_0}, u) > 0$ for some u such that there is an alternation between $\alpha_1, \ldots, \alpha_{i_0}$ and u, then our algorithm would prefer to use α_{i_0} on input x', too. Hence, $\mu_{x'}(\alpha_1, \ldots, \alpha_{i_0}, u) = 0$ for all such u. On the other hand, for $u = \alpha_{i_0+1}, \ldots, \alpha_\ell$ we have $\mu_x(\alpha_1, \ldots, \alpha_{i_0}, u) > 0$ and there is an alternation between $\alpha_1, \ldots, \alpha_{i_0}$ and u. Thus, the lexicographically minimal string according to the ordering $\leq^w_{\text{lex}}$ with positive probability in μ_x differs from the corresponding string with positive probability in $\mu_{x'}$.

2. Assume that α_{i_0} is a prefix of α'_{i_0}. Note that we have $i_0 < \ell$. Then
$$\alpha_1, \ldots, \alpha_{i_0}\alpha_{i_0+1} <^w_{\text{lex}} \alpha'_1 \ldots, \alpha'_{i_0}.$$
By our construction of c, it follows that if $\mu_{x'}(\alpha_1, \ldots, \alpha_{i_0}, \alpha_{i_0+1}, u) > 0$ for some u such that there is an alternation between $\alpha_1, \ldots, \alpha_{i_0}$ and u, then our algorithm prefers to use $\alpha_{i_0}, \alpha_{i_0+1}$ on x', too. Hence $\mu'_x(\alpha_1, \ldots, \alpha_{i_0}, \alpha_{i_0+1}, u) = 0$ for all such u. On the other hand, for $u = \alpha_{i_0+2}, \ldots, \alpha_\ell$ it is true that $\mu_x(\alpha_1, \ldots, \alpha_{i_0}, \alpha_{i_0+1}, u) > 0$ and there is an alternation between $\alpha_1, \ldots, \alpha_{i_0}$ and u. Thus, the lexicographically minimal string according to the ordering $\leq^w_{\text{lex}}$ with positive probability in μ_x differs from the corresponding string with positive probability in $\mu_{x'}$.

□

From Lemmas 4.6 and 4.7, we get the following theorem.

Theorem 4.8 *For $a \in \mathbb{B}$ we have $s_G(q, a) = \mathrm{CP}^{\mathrm{R}}(f_{q \leftarrow a})$.*

5 One-Phase Protocols

We start our study of one-phase protocols with considering networks that consist of one bridge player who is incident with d blocks. For the case that the order in which the bridge player communicates with the blocks is fixed for all inputs, we show a relationship between the size of the information source of one-phase protocols and communication size of multi-party one-way protocols. We prove that for some Boolean functions there exists no fixed order that minimizes the loss of information of one-phase protocols. On the other hand we prove that for every symmetric Boolean function one-phase protocols can minimize the loss of information when the bridge player sorts the blocks by increasing size. Then we present a simple one-phase protocol on arbitrarily connected network that is optimal for every symmetric function.

5.1 Orderings

It is easy to see that in any 2-party communication protocol, the party that starts sending messages to the other party is independent of the input. Analogously, we can show the following lemma.

Lemma 5.1 *Let G be a connected network with one bridge player P_q and let $\mathcal{A}$ be a one-phase protocol on G. Then the block P_q starts to exchange messages with is independent of the actual input x_i of all other players $P_i \neq P_q$.*

Nevertheless, the communication order of P_q can depend on the input of P_q.

A natural extension of the two-party scenario for one-way communication is a scenario in which the parties use a directed chain for communication. Hence, we consider parties $A_1, \dots, A_d$ that are connected by a directed chain, i.e. A_i can only send messages to A_{i+1}. For a communication protocol $\mathcal{B}$ on G and $i \in [d]$ let $S_i^{\mapsto}(\mathcal{B})$ be the number of possible communication sequences on the subnetwork of $A_1, \dots, A_i$. Each communication protocol $\mathcal{B}$ can be modified without increasing $S_i^{\mapsto}(\mathcal{B})$ ($i \in [d]$) in the following way: Every party A_i first sends the messages it has received from A_{i-1} to A_{i+1} followed by the messages it has to send according to $\mathcal{B}$. In the following we restrict ourselves to communication protocols of this form.

If the network G consists of d blocks B_i with $i \in [d]$ and one bridge player P_q, we consider a chain of d parties $A_1, \dots, A_d$. For a σ-ordered one-phase protocol $\mathcal{A}$, we assume that the enumeration of the blocks reflects the ordering σ. Analogously to our simulation in Section 4, we have to determine the input bits of the parties in the chain according to the input bits of the players in the protocol. In the following we will assume that A_i knows the input bits of the players in B_i. Thus, each party A_i has to know the input bit $x[q]$ of the bridge player P_q. Therefore, we will investigate the restricted function $f_{q \leftarrow a}$ whenever we analyze the communication size of a communication protocol.

For a σ-ordered protocol $\mathcal{A}$ define

$$\mathcal{S}_{\mathcal{A}}^{[i]}(q, a, b, r_q) := \{\hat{\mu}_x \mid x[q] = a \text{ and } f(x) = b\},$$

where

$$\hat{\mu}_x(\hat{c}_k) := \sum_{c \text{ with } \widehat{c}_k = \text{block}_1(c) \dots \text{block}_i(c)} \Pr[C_q = c \mid r_q, x]$$

and $\widehat{c}_1, \widehat{c}_2, \widehat{c}_3, \dots$ is a fixed enumeration of all strings describing the communication of P_q in the first i block sequences.

Let $\mathcal{I}_i$ be the set of input positions known by the players in B_i except for P_q. Then for a fixed input $a \in \mathbb{B}$ of player P_q define $\mathcal{Y}_u^0 = \mathbb{B}^\star$ and

$$\begin{aligned}
\mathcal{Y}_0^i &:= \{x \in \mathcal{Y}_u^{i-1} \mid (f_{q \leftarrow a})_{\mathcal{I}_i \leftarrow x[\mathcal{I}_i]} \equiv 0\}\ , \\
\mathcal{Y}_1^i &:= \{x \in \mathcal{Y}_u^{i-1} \mid (f_{q \leftarrow a})_{\mathcal{I}_i \leftarrow x[\mathcal{I}_i]} \equiv 1\}\ , \text{ and} \\
\mathcal{Y}_u^i &:= \mathbb{B}^{n-1} \setminus \textstyle\bigcup_{j=1}^{i} (\mathcal{Y}_0^j \cup \mathcal{Y}_1^j)
\end{aligned}$$

Lemma 5.2 *For $a \in \mathbb{B}$ let $\mathcal{B}$ be a d-party one-way communication protocol computing $f_{q \leftarrow a}$ on a chain network. Then there exists a σ-ordered one-phase protocol $\mathcal{A}$ computing f such that for all $i \in [d-1]$ and for every content r_q of P_q's random tape*

$$S_i^{\mapsto}(\mathcal{B}) \;=\; |\mathcal{S}_{\mathcal{A}}^{[i]}(q, a, 0, r_q) \cup \mathcal{S}_{\mathcal{A}}^{[i]}(q, a, 1, r_q)|\ .$$

Proof: We use a simulation analogously to the simulation in Lemma 4.2. According to our observations above we can conclude for any input $x \in \mathbb{B}^n$ with $x[q] = a$:

- If $x \in \bigcup_{j=1}^{i} \mathcal{Y}_0^j$, then the resulting distribution is in $\mathcal{S}_{\mathcal{A}}^{[i]}(q, a, 0, r_q)$ but not in $\mathcal{S}_{\mathcal{A}}^{[i]}(q, a, 1, r_q)$.
- If $x \in \bigcup_{j=1}^{i} \mathcal{Y}_1^j$, then the resulting distribution is in $\mathcal{S}_{\mathcal{A}}^{[i]}(q, a, 1, r_q)$ but not in $\mathcal{S}_{\mathcal{A}}^{[i]}(q, a, 0, r_q)$.
- If $x \in \mathcal{Y}_u^i$, then the resulting distribution is in $\mathcal{S}_{\mathcal{A}}^{[i]}(q, a, 1, r_q) \cap \mathcal{S}_{\mathcal{A}}^{[i]}(q, a, 0, r_q)$.

Each possible communication sequence on the subnetwork of $A_1, \ldots, A_{i+1}$ results in exactly one distribution in $\mathcal{S}_{\mathcal{A}}^{[i]}(q, a, 0, r_q) \cup \mathcal{S}_{\mathcal{A}}^{[i]}(q, a, 1, r_q)$. Thus, the lemma is proved. □

Lemma 5.3 *Let $\mathcal{A}$ be a σ-ordered one-phase protocol for computing f on a network as described above. Then for every $a \in \mathbb{B}$ and every content r_q of P_q's random tape there exists a one-way communication protocol $\mathcal{B}$ for computing $f_{q\leftarrow a}$ such that for all $i \in [d-1]$*

$$S_i^{\mapsto}(\mathcal{B}) \;\leq\; |\mathcal{S}_{\mathcal{A}}^{[i]}(q, a, 0, r_q) \cup \mathcal{S}_{\mathcal{A}}^{[i]}(q, a, 1, r_q)| \;.$$

Proof: Analogously to our simulation in Lemma 4.3 the parties A_i compute the lexicographically minimal block sequences describing the communication of P_q with the players in B_i on input $x[\mathcal{I}_i]$ and $x[q]$ where the block sequences for the communication of P_q with the players of the blocks $B_1, \ldots, B_{i-1}$ are determined by the communication string on the subnetwork of $A_1, \ldots, A_i$. Each distribution gives at most one communication sequence on the subnetwork on $A_1, \ldots, A_{i+1}$. The lemma follows directly. □

The simulations above give us even more.

Proposition 5.4 *Let $a \in \mathbb{B}$ and $\mathcal{B}$ be a communication protocol as described above for computing $f_{q\leftarrow a}$ on a chain network. Then there exists a σ-ordered one-phase protocol $\mathcal{A}$ for computing f such that for all $b \in \mathbb{B}$, every $j \in [d-1]$, and every content r_q of P_q's random tape the following holds:*

> *If we restrict the inputs to $x \in \mathbb{B}^{n-1}$ with $f_{q\leftarrow a}(x) = b$, the number of possible communication sequences on the subnetwork $A_1, \ldots, A_{i+1}$ is given by $|\mathcal{S}_{\mathcal{A}}^{[i]}(q, a, b, r_q)|$.*

Furthermore, let $\mathcal{A}$ be a σ-ordered one-phase protocol for computing f on a network as described above. Then for every $a \in \mathbb{B}$, every content r_q of P_q's random tape, and every $b \in \mathbb{B}$ there exists a one-way communication protocol $\mathcal{B}$ for computing $f_{q\leftarrow a}$ such that the following properties hold for all $i \in [d-1]$:

If we restrict the inputs to $x \in \mathbb{B}^{n-1}$ with $f_{q\leftarrow a}(x) = b$, the number of possible communication sequences on the subnetwork of $A_1, \ldots, A_{i+1}$ is bounded from above by $|\mathcal{S}^{[i]}_{\mathcal{A}}(q, a, b, r_q)|$.

Let us now focus on the structure of the possible communication sequences of an optimal communication protocol on a chain. Such a protocol has to specify the subfunction

$$f_{i,x} \ := \ (f_{q\leftarrow a})_{\bigcup_{j=1}^{i} \mathcal{I}_j \leftarrow x[\bigcup_{j=1}^{i} \mathcal{I}_j]}$$

for any input x for any $i < d$ by the corresponding communication string on the link (A_i, A_{i+1}). As we have seen above, we do not increase the number of communication strings in the subnetwork $A_1, \ldots, A_{i+1}$, if the message sent by A_i specifies all subfunctions $f_{1,x}, \ldots, f_{i,x}$. Hence, the number of possible communication sequences on the network $A_1, \ldots, A_d$ is at least the number of different sequences $f_{1,x}, \ldots, f_{d-1,x}$ where we vary over the different inputs x.

The knowledge about these sequences must also be provided to the bridge player by the probability distribution of a σ-ordered one-phase protocol. Hence, for every fixed r_q and $b \in \mathbb{B}$ the number of distributions in $\mathcal{S}^{[d-1]}_{\mathcal{A}}(q, a, b, r_q)$ is at least the number of different sequences $f_{1,x}, \ldots, f_{d-1,x}$ for inputs x with $x[q] = a$ and $f(x) = b$. This implies the following lemma.

Lemma 5.5 *For $a \in \mathbb{B}$ let $\mathcal{B}$ be a communication protocol for computing $f_{q\leftarrow a}$ on a chain network. Then there exists a σ-ordered one-phase protocol $\mathcal{A}$ for computing f such that for all $i \in [d-1]$ and every content r_q of P_q's random tape*

$$S_i^{\mapsto}(\mathcal{B}) \ = \ |\mathcal{S}^{[i]}_{\mathcal{A}}(q, a, 0, r_q) \cup \mathcal{S}^{[i]}_{\mathcal{A}}(q, a, 1, r_q)| \ .$$

Furthermore, for any $b \in \mathbb{B}$ it holds: If we restrict the inputs to $x \in \mathbb{B}^{n-1}$ with $f_{q\leftarrow a}(x) = b$, the number of possible communication sequences on the subnetwork $A_1, \ldots, A_{i+1}$ is given by $|\mathcal{S}^{[i]}_{\mathcal{A}}(q, a, b, r_q)|$.

5.2 Quasi-Ordered Protocols

We can show that there exist functions, for which no ordered one-phase protocol minimizes the size of the bridge players' information source. Thus, we generalize the class of ordering that we consider to achieve such a property.

We call a protocol $\mathcal{A}$ *quasi-ordered* if for every $a, b \in \mathbb{B}$, for every content r_q for P_q's random tape, and for every distribution $\mu \in \mathcal{S}_{\mathcal{A}}(q, a, b, r_q)$ there exists a one-phase ordering σ such that every communication string c with $\mu(c) > 0$ the string c is σ-ordered. Note that this ordering is not necessarily the same for all inputs. However, given any input, the ordering is fixed.

Lemma 5.6 *Let G be a connected network with one bridge player P_q and d blocks. Then for every one-phase protocol $\mathcal{A}$ there exists a quasi-ordered one-phase protocol $\mathcal{A}'$ such that for all $a, b \in \mathbb{B}$ and every content r_q of P_q's random tape*

$$s_{\mathcal{A}}(q, a, b, r_q) \ \geq \ s_{\mathcal{A}'}(q, a, b, r_q) \, .$$

Proof: We prove this lemma by induction in the number of blocks of the graph. The lemma follows from Lemma 5.1 for every function and every connected network G with one bridge player P_q and 2 blocks.

Let us now assume that the claim holds for every function and every connected network with one bridge player P_q and $d-1$ blocks. Let G be a connected network with one bridge player P_q and d blocks, let f be the function we want to compute on G, and $\mathcal{A}$ be a one-phase protocol for computing f on G.

According to Lemma 5.1 the block where P_q starts to exchange messages is independent of the actual input $x[i]$ of all other players $P_i \neq P_q$. Thus the index i_1 of the block can be determined by $a, b \in \mathbb{B}$ and the content r_q of P_q's random tape. For every input $x \in \mathbb{B}^n$ with $x[q] = a$ and $f(x) = b$ the first block sequence has to determine the type of the subfunction $f_{1,x}$. If two input strings x, y with $x[q] = y[q] = a$ and $f(x) = f(y) = b$ have different subfunction $f_{1,x} \neq f_{1,y}$, then these inputs result in different distributions over the possible communication strings in the first block sequence as well. If for two inputs $x \neq y$ with $x[q] = y[q] = a$ and $f(x) = f(y) = b$ we have $f_{1,x} = f_{1,y}$ then there exists a protocol that uses the same distribution over the possible communication strings in the first block sequence for x and y.

Let $F_{1,x}$ be the set of all different subfunctions $f_{1,x}$ ($x \in \mathbb{B}$ with $x[q] = a$ and $f(x) = b$). Let $t = |F_{1,x}|$. For every subfunction $h \in F_{1,x}$ let x_i be an input with $h = f_{1,x_i}$ and $\hat{c}_i$ be a string that describes the communication between P_q and B_{i_1} on input x with positive probability. Let $\mathcal{A}_i$ be the part of the protocol in which $\mathcal{A}$ continues its computation after seeing c_i. Then the following inequality holds:

$$s_{\mathcal{A}}(q, a, b, r_q) \ \geq \ \sum_{j=1}^{t} s_{\mathcal{A}_j}(q, a, b, r_q) \, .$$

By the induction hypothesis for every $j \in [t]$ there exists a quasi-ordered protocol $\mathcal{A}'_j$ that computes the same function f_j on the same network as $\mathcal{A}_j$ and

$$s_{\mathcal{A}_j}(q, a, b, r_q) \ \geq \ s_{\mathcal{A}'_j}(q, a, b, r_q) \, .$$

On the other hand, for every input x the bridge player P_q can compute the subfunctions $f_{i,x}$ on the block B_{i_1} privately by a protocol $\mathcal{A}'_0$. Let $\mathcal{A}'$ be the quasi-ordered one-phase protocol that we get by combining the protocols $\mathcal{A}'_0, \mathcal{A}'_1, \ldots, \mathcal{A}'_t$. Then

$$s_{\mathcal{A}}(q, a, b, r_q) \ \geq \ \sum_{j=1}^{t} s_{\mathcal{A}_j}(q, a, b, r_q) \ \geq \ \sum_{j=1}^{t} s_{\mathcal{A}'_j}(q, a, b, r_q) \ = \ s_{\mathcal{A}'}(q, a, b, r_q) \, .$$

The claim follows, since both $\mathcal{A}$ and $\mathcal{A}'$ are one-phase protocols for computing the same function. □

5.3 Orderings for Symmetric Functions

If we restrict ourselves to symmetric Boolean functions f, we can show even more. Arpe et al. [1] have proved the following for symmetric Boolean functions with a fixed partition of the input bits: for all i, $S_i^{\mapsto}(\mathcal{B})$ can be minimized, if the number of bits known by the parties in the chain corresponds to the position of the party, i.e. the first party knows the smallest number of input bits, the second party knows the second smallest number, and so on.

This observation is also valid, if we count the number of communication sequences in a chain network for inputs x with $f(x) = 1$ and if we count the number of communication sequences in a chain network for inputs x with $f(x) = 0$. By combining these observations with Lemma 5.5, we obtain the following lemma.

Lemma 5.7 *Let G be a connected network with one bridge player P_q and d blocks. Let σ be a one phase ordering that enumerates the blocks of G according to their size. Then for every ordered one-phase protocol $\mathcal{A}'$ there exists a σ-ordered one-phase protocol $\mathcal{A}$ such that for all $a, b \in \mathbb{B}$, for all $i \leq d - 1$, and every content r_q of P_q' random tape*

$$|\mathcal{S}_{\mathcal{A}}^{[i]}(q, a, b, r_q)| \;\leq\; |\mathcal{S}_{\mathcal{A}'}^{[i]}(q, a, b, r_q)| \,.$$

On the other hand, after finishing the computation steps with the players of the first $d - 1$ blocks, P_q can start a protocol for computing the final function value by exchanging messages with the players of the last block. According to the definition of protocols in 2-connected graphs, no player can learn anything about the inputs of the other players that cannot be derived from its own input and the result of the function. This observation implies that for all $a, b \in \mathbb{B}$ and every content r_q of P_q' random tape

$$|\mathcal{S}_{\mathcal{A}}^{[d-1]}(q, a, b, r_q)| \quad = \quad |\mathcal{S}_{\mathcal{A}}(q, a, b, r_q)| \,.$$

To prove that a one-phase protocol $\mathcal{A}$ that uses an order like in Lemma 5.7 is optimal with respect to the size of the information source of the bridge player P_q, it remains to show that the information source of such a protocol is also smaller than the information source of every non-ordered one-phase protocols $\mathcal{A}'$.

Lemma 5.8 *Let G be a connected network with one bridge player P_q and d blocks. Let σ be a one-phase ordering that enumerates the blocks of G according to their size. Then for every one-phase protocol $\mathcal{A}'$ there exists a σ-ordered one-phase protocol $\mathcal{A}$ such that for all $a, b \in \mathbb{B}$*

$$s_{\mathcal{A}}(q, a, b) \;\leq\; s_{\mathcal{A}'}(q, a, b) \,.$$

Proof: If $\mathcal{A}'$ is an ordered protocol then the claim follows directly from Lemma 5.7.

In the following we will assume, that π is a one-phase ordering that enumerates the blocks of G according to their size and fulfills the following additional property:

If G has a two blocks of the same size, then the block are ordered in π according to there indices.

Note that given a ordering of the blocks, this ordering is well-defined.

By contradiction let us assume, that there exists a non-ordered one-phase protocols $\mathcal{A}'$ such that for every ordered one-phase protocols $\mathcal{A}$ we have

$$s_{\mathcal{A}}(q,a,b) \;>\; s_{\mathcal{A}'}(q,a,b)\;.$$

By Lemma 5.6 we can assume that $\mathcal{A}'$ is quasi-ordered.

For any one-phase communication string c of the bridge player P_q let $\Delta(c)$ denote the number block sequences $\mathrm{block}_i(c)$ in c such that the suffix

$$\mathrm{block}_i(c)\ldots\mathrm{block}_d(c)$$

violates the ordering of π, i.e. there are two blocks B_j, B_k such that

- according to the ordering π, B_j is ranged before B_k,
- each block has its corresponding block sequences in the suffix $\mathrm{block}_i(c)\ldots\mathrm{block}_d(c)$ of c, and
- according to the ordering of this suffix, B_k is ranged before B_j.

We call $\Delta(c)$ the degree of disorder of c. For a distribution μ all communication strings of P_q, we define

$$\Delta(\mu) \;:=\; \max_{c \text{ with } \mu(c)>0} \Delta(c)\;.$$

For a quasi-ordered protocol the orderings for all communication strings c with $\mu(c) > 0$ are identical.

Finally for a one-phase protocol $\mathcal{A}$, $a,b \in \mathbb{B}$, and a content r_q of P_q's random tape define

$$\begin{aligned}\Delta_{\mathcal{A}}(a,b,r_q) &:= \sum_{\mu\in\mathcal{S}_{\mathcal{A}}(q,a,b,r_q)} \Delta(\mu) \quad \text{and}\\ \Delta_{\mathcal{A}}(a,b) &:= \sum_{r_q} \Delta_{\mathcal{A}}(a,b,r_q)\;.\end{aligned}$$

We call $\Delta_{\mathcal{A}}(a,b)$ the degree of disorder of the protocol $\mathcal{A}$.

Let $\mathcal{A}'$ be a quasi-ordered one-phase protocols $\mathcal{A}'$ such that

- $\mathcal{A}'$ has a minimum degree of disorder $\Delta_{\mathcal{A}}(a,b)$ over all quasi-ordered one-phase protocols fulfilling Equation 1 and
- for every ordered one-phase protocols $\mathcal{A}$ we have

$$s_{\mathcal{A}}(q,a,b) \;>\; s_{\mathcal{A}'}(q,a,b)\;. \tag{1}$$

We will now show, that such an optimal quasi-ordered one-phase protocols $\mathcal{A}'$ does not exist.

Let r_q be a possible content of P_q's random tape such that $\Delta_{\mathcal{A}'}(a, b, r_q) > 0$ and $\mu_x \in \mathcal{S}_{\mathcal{A}'}(q, a, b, r_q)$ be a distribution such that $\Delta(\mu_x)$ is maximal. Furthermore, let $\sigma_x = B_{i_1}, \ldots, B_{i_d}$ be the ordering of μ_x and choose k maximal, such that there exists a block B_{i_j} with $j > k$ and B_{i_j} is ranged before B_{i_k} in π.

Note that for each quasi-ordered one-phase protocol the first block of each communication string is always fixed. The index of the second block depends only on the type of the subfunction of f when we fix the input bits of the players in the first block. This subfunction is called $f_{1,x}$. In general the index of the ℓth block depends only on the type of the sequence $f_{1,x}, \ldots, f_{\ell-1,x}$, where $f_{j,x}$ is the subfunctions of f where we fix the input bits of the players in the first j blocks.

Let $\mathcal{A}''$ be the part of the protocol of $\mathcal{A}'$ that determines the behavior of P_q after receiving the information $f_{1,x}, \ldots, f_{i_k-1,x}$ from the first $i_k - 1$ blocks. Note that $\mathcal{A}''$ computes $f_{i_k-1,x}$ on the subgraph of G that consists of the blocks $B_{i_k}, \ldots, B_{i_d}$ only. Since μ_x has a maximum value of the degree of disorder the protocol $\mathcal{A}''$ is ordered.

Then we have

$$\begin{aligned} s_{\mathcal{A}'}(q, a, b, r_q) &= |\{\mu_y \mid f_{1,x}, \ldots, f_{i_k-1,x} \text{ differs from } f_{1,y}, \ldots, f_{i_k-1,y}\}| \\ &\quad + s_{\mathcal{A}''}(q, a, b, r_q) , \end{aligned}$$

where μ_y describes the probability distribution over the communication strings on input y. Since $\mathcal{A}''$ is ordered and $f_{i_k-1,x}$ is a symmetric function, we can apply Lemma 5.7 and modify $\mathcal{A}'$ by replacing $\mathcal{A}''$ with an ordered one-phase protocol $\mathcal{A}''_o$ for $f_{i_k-1,x}$ that communicates with the blocks $B_{i_k}, \ldots, B_{i_d}$ according to their size. Note that the resulting protocol $\mathcal{A}'_o$ for f is still quasi-ordered and by Lemma 5.7 we can chose $\mathcal{A}''$ such that

$$s_{\mathcal{A}''}(q, a, b, r_q) \geq s_{\mathcal{A}''_o}(q, a, b, r_q)$$

and

$$\Delta_{\mathcal{A}'}(a, b, r_q) \; \geq \; \Delta_{\mathcal{A}'_o}(a, b, r_q) \; .$$

This contradicts our assumption that $\mathcal{A}'$ has a minimum degree of disorder over all quasi-ordered one-phase protocols fulfilling Equation 1. □

5.4 An Optimal One-Phase Protocol for Symmetric Functions

The result of the previous section can also be generalized to networks with more than one bridge player. Let $G_1, \ldots, G_k$ be the connected subgraphs obtained by deleting the bridge player P_q with $|G_i| \leq |G_{i+1}|$. We say that P_q works in increasing order, if it starts communicating with G_1, then with G_2 and so on. We call a one-phase protocol $\mathcal{A}$ *increasing-ordered*, if every bridge player works in increasing order. This generalizes the ordering of $\mathcal{A}$ chosen in Lemma 5.8.

For a graph G let $\mathcal{G} = \{G_1 = (V_1, E_1), \ldots, G_h = (V_h, E_h)\}$ be the set of blocks and $\mathcal{Q} = \{q_1, \ldots, q_k\}$ be the set of bridge nodes of G. Every graph G induces a tree $T_G = (V_G, E_G)$ defined as follows: $V_G = V_\mathcal{Q} \cup V_\mathcal{G}$ with $V_\mathcal{Q} = \{u_1, \ldots, u_k\}$ and $V_\mathcal{G} = \{v_1, \ldots, v_h\}$ and $E_G = \{\{u_i, v_j\} \mid q_i \in V_j\}$.

For every one-phase communication order $\sigma = (\sigma_{q_1}, \ldots, \sigma_{q_k})$ and every bridge node q_i the order σ_{q_i} defines an ordering of the nodes $v_j \in V_\mathcal{G}$ adjacent to the tree-node u_i. Let $G_{\sigma_{q_i}(1)}, \ldots, G_{\sigma_{q_i}(k_i)}$ denote the ordering of blocks adjacent to q_i with respect to σ_{q_i} and $\mathrm{root}_\sigma(u_i) := v_{\sigma_{q_i}(k_i)}$. If σ is an increasing communication order, then there exists a single tree-node $v_j \in V_\mathcal{G}$, such that $v_j = \mathrm{root}_\sigma(u_i)$ for all $u_i \in V_\mathcal{Q}$ adjacent to v_j. Let us call this node the root of T_G. For a tree-node $w \in V_G$ let $T_G[w]$ denote the subtree of T_G rooted by w and let $V[w]$ denote the nodes of G located in the blocks G_j with $v_j \in T_G[w]$.

For computing a symmetric function f we use the following protocol. Let σ be an increasing communication order. Then for an input x every bridge player q_i computes a sequence of strings $y_1, \ldots, y_{k_i-1}$ as follows: Let $X_j = \bigcup_{e\in[j]} V[v_{\sigma_{q_i}(e)}]$ and $\ell_j = |X_j|$. Then $y_j \in \mathbb{B}^{\ell_j}$ such that for all $j \leq k_i - 1$ the function obtained from f by specializing the positions in X_j to y_j is equal to the function obtained from f by specializing the positions to x_{X_j}, where x_I for $I \subseteq [n]$ denotes the input bits with indices in I. Finally, a node of the block that corresponds to the root of T_G computes the result $f(x)$. This can be implemented such that no player gains any additional information except for the strings $y_1, \ldots, y_{k_i-1}$ learned by the bridge nodes q_i.

Theorem 5.9 *Let G be a 2-edge-connected network and f be a symmetric Boolean function. Then there exists an increasing-ordered one-phase protocol $\mathcal{A}$ for f on G such that for every one-phase protocol $\mathcal{A}'$ for f on G, for every player P_i, and for all $a, b \in \mathbb{B}$, we have*

$$s_\mathcal{A}(i, a, b) \ \leq\ s_{\mathcal{A}'}(i, a, b)\ .$$

Proof: To prove this theorem we will present a protocol for computing f on G that simultaneously minimizes the size of the information source of each player of G. Thus, the protocol is optimal with respect to the size of the information source of each player, if the function is symmetric and the network is 2-edge-connected.

Let G be a network and P_q be any bridge node in G. B_1, …, B_{d_q} are the blocks incident with P_q and $G_i = (V_i, E_i)$ is the connected subgraph of G that contains B_i after deleting P_q. Finally, let $\mathcal{I}_i$ be the set indices of the players in $V_i \cup \{q\}$ and $\#_q = \left|\bigcup_{i=1}^{d_q-1} \mathcal{I}_i\right|$. We assume that G_i covers the players of B_i (except for P_q) and $|\mathcal{I}_i| \leq |\mathcal{I}_{i+1}|$ for $1 \leq i < d_q$. Recall, that $x[\mathcal{I}_1]$, …, $x[\mathcal{I}_{d_q}]$ are the actual inputs for $\mathcal{I}_1$, …, $\mathcal{I}_{d_q}$, respectively.

For easier notion let $\mathcal{I}_0 = \emptyset$ and $x[\mathcal{I}_0]$ be the empty string. The protocol for P_q proceeds in d_q stages as follows:

1. In the first $d_q - 1$ stages the protocol P_q computes $f_{i,x} = f_{\bigcup_{j=1}^{i}\mathcal{I}_j \leftarrow x[\bigcup_{j=1}^{i}\mathcal{I}_j]}$ iteratively for $1 \leq i < d_q$ on B_i. Therefore, P_q chooses an arbitrary string

$\alpha_i \in \mathbb{B}^{|\bigcup_{j=1}^{i-1}\mathcal{I}_j|}$ such that $f_{\bigcup_{j=1}^{i-1}\mathcal{I}_j \leftarrow \alpha_i} = f_{i-1,x}$ and cooperates with the players in B_i as a player with input α_i.

2. In the last stage, P_q chooses an arbitrary string $\alpha_{d_q} \in \mathbb{B}^{|\bigcup_{j=1}^{d_q-1}\mathcal{I}_j|}$ such that

$$f_{\bigcup_{j=1}^{d_q-1}\mathcal{I}_j \leftarrow \alpha_{d_q}} = f_{d_q-1,x}$$

and cooperates with the players in B_{d_q} as a player with input α_{d_q}. We distinguish three cases:

(a) If $|\mathcal{I}_{d_q}| \leq \#_q$, then P_q privately computes $f_{d_q,x} = f_{d_q-1,x_{\mathcal{I}_{d_q}\leftarrow x[\mathcal{I}_{d_q}]}}$ on B_{d_q}.

(b) If $|\mathcal{I}_{d_q}| > \#_q$, $\#_q = \max\{\#_{q'} \mid P_{q'}$ is a bridge player$\}$, and $q < q'$ for all bridge player $P_{q'}$ with $\#_{q'} = \#_q$, then P_q privately computes $f_{d_q,x} = f_{d_q-1,x_{\mathcal{I}_{d_q}\leftarrow x[\mathcal{I}_{d_q}]}}$ on B_{d_q}.

(c) Otherwise, P_q proceeds in B_{d_q} as a non-bridge player with input α_{d_q}.

Now we prove that the size of the information source of every player is minimal. Every non-bridge player does not learn anything, not even the function value. Hence, the protocol is lossless with respect to any non-bridge player and it remains considering the bridge players. The only information a bridge player P_q can derive from the messages exchanged with the players of its incident blocks B_i with $1 \leq i \leq d_q - 1$ are the subfunctions $f_{i,x}$. This sequence gives the minimum communication size $S_i^{\mapsto}$ in a communication protocol on a chain where the parties are ordered according to the ordering chosen by our protocol. If the function computed is symmetric, we can apply Lemmas 5.5 and 5.8 to show that the ordering of the blocks for computing the sequence is optimal with respect to the size of the information source. □

Corollary 5.10 *The protocol presented in this section is optimal for one-phase computations of symmetric functions with respect to the size of the information source.*

6 A Phase Hierarchy

In this section we show that there are functions for which the size of the information source of some player for a $(k-1)$-phase protocol is exponentially larger than for a k-phase protocol. The natural candidate for proving such results is the pointer jumping function p_j: Our network G has two blocks A and B, one of size $n \log n$ and the other of size $n \log n + 1$, sharing one bridge player P_i. For simplicity we assume that A and B are complete subgraphs. The input bits represent two lists of n pointers, each of length $\log n$ bits. The input bit of P_i belongs to the list of the smaller component. Starting with some predetermined pointer of A, the task is to follow these pointers, find the jth pointer and output the parity of the bits of the jth pointer. We get the following upper bound for k-phase protocols. (Recall that k-phase protocols can simulate $2k-1$ rounds, since each phase except for the first one can simulate two communication rounds.)

Theorem 6.1 *For p_{2k-1}, $s^k_G(i,a,b) = 2^{O(k \log n)}$ for all a, b.*

Proof: We get a lower bound via the relation between communication size and information source shown in Lemmas 4.2 and 4.3.

The players holding the bits of a particular pointer send their bits to P_i. (If A or B is not a complete graph, then the protocol can be modified such that it is private for the players other than P_i as follows: Each player sends his bit masked with a random bit on one path to P_i and the random bit on another path. This is possible since A and B are blocks. Furthermore, all other players of the block do the same, but with two random bits. This is done to prevent players from learning something by not getting a message.) Then P_i informs the players to which the received pointer points. The informed players send their bits to P_i and so on. After $2k-1$ iterations, P_i simply computes the parity of the last pointer received. In this way, P_i learns $O(k \log n)$ bits. In the worst-case, all pointers involved point from A to B and vice versa. In this case, the number of phases is k. □

Define CS^j and CC^j in the same manner as CS and CC, but by minimizing over j-round communication protocols instead of arbitrary communication protocols.

Theorem 6.2 *Let $\mathcal{A}$ be a protocol for computing p_{2k-1}. Then $s^{k-1}_{\mathcal{A}}(i,a,b) = 2^{\Omega(\frac{n}{k \log k})}$ for all a, b.*

Proof: By Lemmas 4.2 and 4.3 we have $s^{k-1}_{\mathcal{A}}(i,a,b) = \Theta(\mathrm{CS}^{2k-3}(p_{2k-1}))$. By the following Lemma 6.3, $\mathrm{CS}^{2k-3}(p_{2k-1}) \geq 2^{\Omega(\mathrm{CC}^{2k-3}(p_{2k-1})/k)}$. Now the result follows by the lower bound $\mathrm{CC}^{2k-2}(p_{2k-1}) = \Omega(n/\log k)$ for p_{2k-1} proved by Nisan and Wigderson [18]. □

Using more elaborate techniques, one should be able to get rid of this extra k. It remains to show the following lemma.

Lemma 6.3 *For any Boolean function f, we have $\log(\mathrm{CS}^j(f)) \geq \Omega(\mathrm{CC}^j(f)/j)$.*

Proof: Consider a protocol tree T for f with j rounds that has a minimal number of leaves. We modify T as follows: Consider the subtree S induced by nodes belonging to the first round. We can replace S by a balanced tree without changing the outcome of the protocol. Then we change all subtrees corresponding to the second round in the same manner and so forth. Call the resulting tree T'. By construction, T' has still j rounds and the number of leaves of T and T' are the same. But each subtree corresponding to a particular round is balanced.

Consider a longest path P in T' and let $h_1, \ldots, h_j$ be the length of the subpaths of P corresponding to the rounds $1, \ldots, j$, respectively. The number of leaves of T'

is at least $\sum_{i=1}^{j} 2^{h_i - 1}$, since we balanced all subtrees belonging to a particular round. In particular, $\mathrm{CS}^j(f) \geq \sum_{i=1}^{j} 2^{h_i-1}$. On the other hand, the height of T' is $h = h_1 + \ldots + h_j$. Therefore $h \geq \mathrm{CC}^j(f)$. The value $\sum_{i=1}^{j} 2^{h_i-1}$ attains its minimum if $h_1 = \ldots = h_j$. In this case $\sum_{i=1}^{j} 2^{h_i-1} = j \cdot 2^{h/j-1}$. Therefore, $\log \mathrm{CS}^j(f) \geq h/j - 1 + \log j$, which proves the claim. □

7 Conclusions and Open Problems

We have considered distributed protocols in "non-private" environments: networks that are connected but not 2-connected. Since private computation of arbitrary Boolean functions is impossible on such networks, we have introduced a new measure for the information that can be inferred from seeing particular communication strings and discussed some general properties of protocols with respect to this measure. A natural question that arises is finding optimal protocols for some concrete functions.

For common Boolean functions like e.g. threshold ($f_{n_0}(x_1, \ldots, x_n) = 1$ if and only if $\sum_{i=1}^{n} x_i \geq n_0$, particularly disjunction ($n_0 = 1$), conjunction ($n_0 = n$), and majority ($n_0 = \lceil \frac{n+1}{2} \rceil$)) and counting modulo p (i.e. $g_p(x_1, \ldots, x_n) = 1$ if and only if $\sum_{i=1}^{n} x_i \equiv 0 \pmod{p}$), we can prove that the information loss to any player does not depend on the ordering in which a one-phase protocol computes any of these functions, if each block has size at least n_0 and p, respectively.

Proposition 7.1 *Let a network be given on which we want to compute f_{n_0} or g_p. Each block of the network has size at least n_0 or $p-1$, respectively. Then the loss to each bridge player in an optimal one-phase protocol does not depend on the ordering in which the bridge players communicate with their incident blocks.*

Proof: It suffices to prove the proposition for networks consisting of a single bridge player P_q incident with k blocks ($k \geq 2$).

We start with considering g_p. Since any block has size at least $p-1$, there have to be p different strings P_q can receive from $k-1$ of its incident block (all but the one he communicates with last). This is independent of the ordering, in which he communicates with his incident blocks. The claim of the proposition follows immediately.

Now let us consider f_{n_0}. For the first block P_q communicates with, there are exactly $n_0 + 1$ possibilities that have to be distinguished: 0 ones, 1 one, …, $n_0 - 1$ ones, and n_0 ones (which means that the result is one independently of the input bits hold in the other blocks). If P_q receives $m \in [n_0] \cup \{0\}$, there remain $n_0 + 1 - m$ possibilities to distinguish and so on. None of the considerations depends on the ordering in which P_q communicates with his incident blocks, since any of these blocks has size at least n_0. The proposition follows. □

If we have blocks consisting of less than $p-1$ nodes, there can be a difference in the size of P_q's information source depending on the order. Consider a network consisting of one block of size, say, $k < p-1$ and another block of size $n-k \geq p-1$. Our aim is to find out, whether the number of ones is $0 \pmod p$. If P_q starts his communication with the smaller block, the size of his information source is clearly $k+1$. On the other hand, if he starts his communication with the larger block, the size of his information source is $k+2$: For any $0 \leq i \leq k$ we have a string saying "the result is 1 if there are exactly k ones in the smaller block" plus one string saying "result 1 cannot be achieved anymore". This observation can easily be modified for threshold functions.

In general, the size of the information source while communicating in one order can be exponentially larger than the size obtained by communication in another order. This is true, even if we restrict ourselves to symmetric functions.

Proposition 7.2 *There is a symmetric function f, a network G with one bridge player P_q, and two orderings σ and σ' such that $s_G(q,1,1,\sigma) = \Theta(\log s_G(q,1,1,\sigma'))$.*

Proof: For $\ell \in \mathbb{N}$, let $n = 2^\ell - 1$. For $x_1, \ldots, x_n \in \mathbb{B}$, let

$$y = \textstyle\sum_{i=1}^{n} x_i = \sum_{i=0}^{\ell-1} y_i 2^i \ .$$

For simplicity, we call the binary string of length that represents y again y. We split y into two parts: $a = y_{d-1} \ldots y_0$ and $z = y_{\ell-1} \ldots y_d$. We choose d maximal with $2^d \leq \ell - d$. Note that (by abusing notation) we also have $a = \sum_{i=0}^{d-1} y_i 2^i$. Then

$$f(x_1, \ldots, x_n) = y_{a+d+1} \ .$$

Thus, we use the lower part of the sum y of the inputs bits to address a bit in the higher part of y.

The network we use will be quite simple. We have two blocks consisting of 2^d and $n - 2^d - 1$ nodes, respectively. (Note that $n = \Theta\big(2^{2^d}\big)$) Furthermore, we have a bridge player P_q which is part of both blocks. If P_q starts communication with the smaller block, we can easily achieve that the size of his information source is at most $2^d + 1$.

It remains to show that the size of P_q's information source is at least $2^{\ell-d-1}$, thus exponentially larger. If the size of his information source is smaller, there are at least two different indistinguishable input strings w and w' for the larger block with $\#w$ and $\#w'$ ones such that $\#w \equiv 0 \pmod 2^{d+1}$ and $\#w \equiv 0 \pmod 2^{d+1}$. Let v be an input string with $\#v$ ones for the smaller block such that the $d+1+\#v$'s bit of $\#w$ and $\#w'$ is different. The function value on w and w' together with v is different. But since w and w' are indistinguishable and v is fixed, the protocol computes the same result for either input, a contradiction. □

For one-phase protocols for symmetric Boolean functions, we have been able to minimize the number of bits a player learns for all players simultaneously. An obvious question concerns minimizing the loss of more than one bridge player simultaneously for general functions. For one-phase protocols, the answer is negative: Consider the function f given by

$$f(\vec{x}_0, \vec{x}_1, \vec{y}_0, \vec{y}_1, z_1, z_2, z_3) = \begin{cases} 1 & \text{if } z_1 \oplus z_2 \oplus z_3 = \xi \text{ and } \vec{x}_\xi = \vec{y}_\xi \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Here $\vec{x}_0$, $\vec{x}_1$, $\vec{y}_0$, and $\vec{y}_1$ are bit vectors of length n and z_1, z_2, and z_3 are single bits. We compute f on the following network: $\vec{x}_0$ and $\vec{x}_1$ are distributed within one block (B_X), $\vec{y}_0$ and $\vec{y}_1$ are distributed within another block (B_Y). z_1, z_2, and z_3 build a third block (B_Z), while z_1 is shared with the B_X and z_3 is shared with the B_Y.

In any one-phase protocol for f, either P_{z_1} or P_{z_3} learns at least $2n$ bits, the other one learns at least $n + 1$ bits.

Now we want to prove that this is optimal: The sum of bits learned by P_{z_1} and P_{z_3} is always at least $3n+1$. There are three different possibilities of one-way communication for this graph: all communication goes from left to right or from right to left or both blocks B_X and B_Y send to B_Z. Due to symmetry, we restrict ourselves to considering the first and third case.

We first consider the case that all communication goes from left to right. Assume that P_{z_1} learns less than $2n$ bits while $z_1 = 0$. Then there are at least two different inputs $\vec{x}_0, \vec{x}_1$ and $\vec{x}'_0, \vec{x}'_1$ that are indistinguishable for the middle and the right component. Assume w.l.o.g. that $\vec{x}_0 \neq \vec{x}'_0$. Choose z_2 and z_3 such that $\ell = 0$. Then either for $\vec{y}_0 = \vec{x}_0$ or for $\vec{y}_0 = \vec{x}'_0$ we get a wrong function value. We can argue similarly to prove that P_{z_3} has to learn $n + 1$ bits. Otherwise, either ℓ is unknown in B_Y or there are at least two different possible strings to compare with. In either case we obtain a contradiction.

Now we consider the case that both blocks B_X and B_Y send to the B_Z. We can argue similarly as in the previous case: If P_{z_1} learns less than $2n$ bits, there are at least two different inputs for B_X that cannot be distinguished. The same holds for the right component. Thus, both P_{z_1} and P_{z_3} must learn at least $2n$ bits each.

On the other hand, using two phases we can achieve the minimum loss of $n + 1$ bits to each bridge player. Compute ℓ and send it to both blocks B_X and B_Y. Then these blocks send x^ℓ and y^ℓ, respectively, to B_Z, which finally computes f.

It is open whether there exist functions and networks that do not allow to minimize the loss to each bridge player simultaneously. If such functions exist, it would be interesting trying to minimize some function depending on the information loss to each player instead of minimizing the loss to each player separately. One simple example one might want to examine is the sum of loss to each player.

Other future work is to generalize the model to t-privacy: How much information does any group of at most t players learn while computing the function.

References

[1] Jan Arpe, Andreas Jakoby, and Maciej Liśkiewicz. One-way communication complexity of symmetric boolean functions. In A. Lingas and B. J. Nilsson, editors, *Proc. of the 14th Int. Symp. on Fundamentals of Computation Theory (FCT)*, volume 2751 of *Lecture Notes in Computer Science*, pages 158–170. Springer, 2003.

[2] Reuven Bar-Yehuda, Benny Chor, Eyal Kushilevitz, and Alon Orlitsky. Privacy, additional information, and communication. *IEEE Transactions on Information Theory*, 39(6):1930–1943, 1993.

[3] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of the 20th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 1–10. ACM Press, 1988.

[4] Claude Berge. *Graphs*. North-Holland, 1991.

[5] Markus Bl¨aser, Andreas Jakoby, Maciej Liśkiewicz, and Bodo Manthey. Privacy in non-private environments. In P. J. Lee, editor, *Proc. of the 10th Int. Conf. on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, volume 3329 of *Lecture Notes in Computer Science*, pages 137–151. IACR, Springer, 2004.

[6] Markus Bl¨aser, Andreas Jakoby, Maciej Liśkiewicz, and Bodo Manthey. Private computation: k-connected versus 1-connected graphs. *Journal of Cryptology*, to appear. An extended abstract appeared in the Proceedings of the 22nd Ann. Int. Cryptology Conf. [7].

[7] Markus Bl¨aser, Andreas Jakoby, Maciej Liśkiewicz, and Bodo Siebert. Private computation — k-connected versus 1-connected networks. In M. Yung, editor, *Proc. of the 22nd Ann. Int. Cryptology Conf. (CRYPTO)*, volume 2442 of *Lecture Notes in Computer Science*, pages 194–209. IACR, Springer, 2002.

[8] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols. In *Proc. of the 20th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 11–19. ACM Press, 1988.

[9] Benny Chor, Mihály Geréb-Graus, and Eyal Kushilevitz. Private computations over the integers. *SIAM Journal on Computing*, 24(2):376–386, 1995.

[10] Benny Chor and Eyal Kushilevitz. A zero-one law for boolean privacy. *SIAM Journal on Discrete Mathematics*, 4(1):36–47, 1991.

[11] Matthew Franklin and Moti Yung. Secure hypergraphs: Privacy from partial broadcast. *SIAM Journal on Discrete Mathematics*, 18(3):437–450, 2004.

[12] Eyal Kushilevitz. Privacy and communication complexity. *SIAM Journal on Discrete Mathematics*, 5(2):273–284, 1992.

[13] Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[14] Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. Characterizing linear size circuits in terms of privacy. *Journal of Computer and System Sciences*, 58(1):129–136, 1999.

[15] Eytan H. Modiano and Anthony Ephremides. Communication complexity of secure distributed computation in the presence of noise. *IEEE Transactions on Information Theory*, 38(4):1193–1202, 1992.

[16] Eytan H. Modiano and Anthony Ephremides. Communication protocols for secure distributed computation of binary functions. *Information and Computation*, 158(2):71–97, 2000.

[17] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*, chapter 9. Cambridge University Press, 2000.

[18] Noam Nisan and Avi Wigderson. Rounds in communication complexity revisited. *SIAM Journal on Computing*, 22(1):211–219, 1993.

[19] Alon Orlitsky and Abbas El Gamal. Communication with secrecy constraints. In *Proc. of the 16th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 217–224. ACM Press, 1984.

[20] Claude Elwood Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3, 4):379–423 & 623–656, 1948.

[21] Ingo Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner, 1987.

[22] Andrew Chi-Chih Yao. Protocols for secure computations. In *Proc. of the 23rd Ann. IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 160–164. IEEE Computer Society, 1982.