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

Privacy in Non-Private Environments

Markus Bläser* Andreas Jakoby†
Maciej Liśkiewicz‡ Bodo Manthey§

Universität zu Lübeck
Institut für Theoretische Informatik
Wallstraße 40, 23560 Lübeck, Germany
blaeser/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 such networks. 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 it loses to the players.

The loss of a protocol to a player is the logarithm of the number of different probability distributions on the communication strings a 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.

Then we study 1-phase protocols, an analogue of 1-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 1-phase protocol that for symmetric functions simultaneously minimizes the loss at all nodes, where the minimum is taken over all 1-phase protocols.

*Supported by DFG research grant BL 511/5-1.
†Part of the work was done while visiting the International University of Bremen.
‡On leave from Instytut Informatyki, Uniwersytet Wrocławski, Poland.
§Supported by DFG research grants RE 672/3 and BL 511/5-1.

ISSN 1433-8092

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 conclude from his own secret and the function value. This is the aim of *private computation* (sometimes also called *secure multiparty 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 ballot 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 any group of at most t players cannot infer any knowledge 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, and conjunction, that cannot privately be computed if the underlying network is not 2-connected [5].

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 play an important role when considering networks that are not 2-connected. Indeed, the bridge players are the only players that are able to learn something more. For all other players we can guarantee that they do not learn anything except for what they can deduce from their own secret and the function value.

The remaining question is, how much the bridge players need to learn such that the function can correctly be computed. The simplest setting is a network of two blocks that have one bridge node in common. (A block in a connected network is a maximum subnetwork that is 2-connected.) This special case 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 actual 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 [13, 14] and Orlitsky and El Gamal [16]. While they have examined the problem under cryptographic security, we are interested in 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. However, 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 [19]. He considered the problem under cryptographic assumptions. Private Computation with unconditional, i.e. information-theoretical, security has been introduced by Ben-Or et al. [3] and Chaum et al. [6]. Kushilevitz et al. [12] proved that the class of Boolean functions that have a circuit of linear size is exactly the class 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 [10] and Chor et al. [7] considered private computations of integer-valued functions. They examined which functions can privately be computed by two players.

Franklin and Yung [9] investigated the role of the connectivity of the underlying network in private computations. They used directed hypergraphs for communication and described those networks on which every Boolean function can privately be computed.

While any Boolean function can privately be computed on any undirected 2-connected network, Bl¨aser et al. [5] 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. [6] 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 $\lfloor \frac{n-1}{2} \rfloor$-private, i.e. at most $\lfloor \frac{n-1}{2} \rfloor$ players collude. Chor and Kushilevitz [8] 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 [19] 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). Throughout this work, we restrict ourselves to 2-edge-connected networks. Every block in such a network has size at least three (i.e. the network is isthmus-free) 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. Lossy protocols can be divided into phases. Within any phase, a bridge player P may exchange messages only once with each block P belongs to. Phases correspond to rounds in communication complexity but they are defined locally for each bridge player. Of particular interest are 1-phase protocols, because they fit the distributed nature of our problem.

In the definition of lossy protocols, the loss of a protocol to a player is basically the logarithm of the number of different probability distributions on the communication strings a player can observe. We justify this definition in Section 3.3: 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. Thus in order to actually gain the 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 1-phase protocols. We start with networks that consist of d blocks that all share the same bridge player P. In a 1-phase protocol, P can communicate with each block only once. 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 minimal number of communication strings. In particular, for symmetric functions it is optimal to sort the components by increasing size.

Then we design a 1-phase protocol for computing functions f (Theorem 6). If f is symmetric, then this protocol has a remarkable feature: At any node, it has minimal loss of information. Hence, it simultaneously minimizes the loss for all nodes where the minimum is taken over all 1-phase protocols.

In Section 6, 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.

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 1-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 the input is distributed among the players; each player knows only a single bit of the input.

Kushilevitz [10] examined which integer-valued functions can privately be computed by two players. He showed that requiring privacy can result in exponentially larger communication costs. Furthermore, he showed that randomization does not help in this model, not even to improve on the number of rounds. Chor et al. [7] 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. Furthermore, they characterized the class of privately computable Boolean functions on countable domains. Neither Kushilevitz [10] nor Chor et al. [7] examine the problem how function 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 subject 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 cost for several functions. Moreover, they showed that sacrificing some privacy can reduce the number of messages required during the computation. More specifically: Every function can be computed by exchanging just two messages and there exist functions that require up to 2^{n+1} messages to be computed privately [10]. Bar-Yehuda et al. showed 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. However, while they examined how revealing information in the two-player scenario can help reducing the necessary amount of communication, we examine how much information has to be learned when one wants to compute a given function.

The counterpart of the two-party scenario in the distributed setting 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 is that in the latter we somewhat like a "man in the middle" that can learn more than any other player in either component.

2 Preliminaries

2.1 Notations

For $i, j \in \mathbb{N}$, let $[i] := \{1, \ldots, i\}$ and $[i..j] := \{i, \ldots, j\}$. 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 $x\lceil_{I\leftarrow\alpha}$ defined as follows: For $x \in \{0,1\}^n$, $I \subseteq [n]$, and $\alpha \in \{0,1\}^{|I|}$, $x\lceil_{I\leftarrow\alpha}$ is defined by

$$(x\lceil_{I\leftarrow\alpha})[i] = \begin{cases} x[i] & \text{if } i \notin I, \\ \alpha[j] & \text{if } i \in I \text{ and } i \text{ is the } j\text{th smallest element in } I, \end{cases}$$

for all $i \in [n]$. For a function $f : \{0,1\}^n \to \{0,1\}$, a set of indices $I \subseteq [n]$, and a string $\alpha \in \{0,1\}^{|I|}$, $f\lceil_{I\leftarrow\alpha} : \{0,1\}^{n-|I|} \to \{0,1\}$ denotes the function obtained from f by specialising the positions in I to the values given by α, i.e. for all $x \in \{0,1\}^{n-|I|}$,

$$f\lceil_{I\leftarrow\alpha}(x) = f((0^n\lceil_{I\leftarrow\alpha})\lceil_{J\leftarrow x}) \,,$$

where $J = [n] \setminus I$. 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 block of size two is called an *isthmus*. The blocks of an undirected loopless graph G partition the edges of G, i.e. each edge belongs to exactly one block. A graph is called *2-edge-connected* if after removal of one edge, the graph is still connected. A graph is 2-edge-connected if it is connected and has no isthmi. A node belonging to more than one block is called a cut node or *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 [18] 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 i is equipped with a random tape. 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 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_i provided to P_i,*

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

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.*

We call a protocol $\mathcal{A}$ *synchronous* if it works in rounds and in each round each player may either send one bit to one of its neighbours or receive one bit from one of its neighbours in the underlying communication network. The fact that each player either 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 fulfils this restriction by simulating this one round by several rounds and sending bits consecutively.) We encode each bit sent as a triple $\{0,1\} \times V^2$ where (b, P_{k_1}, P_{k_2}) indicates that P_{k_1} sends the bit b to P_{k_2}. The communication string c seen by player P_i is an element from $(\{0,1\} \times V^2 \cup \{\perp\})^*$. $c[j] = \perp$ means that P_i does neither send nor receive a bit in round j. If $c[j] = (b, P_{k_1}, P_{k_2})$, then either $k_1 = i$ or $k_2 = i$.

In the following, we use a strengthened definition of privacy of protocols: We allow only one player, say P_i, to know the result computed. The protocol has to be private with respect to P_i according to Definition 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_j we require $\Pr[C_j = c|R_j, x] \;=\; \Pr[C_j = c|R_j, y]$. 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 1. Every function f in this class can be computed by a protocol $\mathcal{A}$ fulfilling the conditions above. To achieve these additional restrictions, player 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, then 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 fewer 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 infinite probability spaces instead of probability distributions on finite spaces. The concepts arising would be completely the same.)

Definition 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_i provided to P_i, define the* information source *of P_i on a, b, and R_i as*

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

where $\mu_x(c_k) := \Pr[C_i = c_k | R_i, x]$ and the probability is taken over the random variables $R_1, \ldots, R_{i-1}, R_{i+1}, \ldots, R_n$ of all other players.

Basically $\mathcal{S}_{\mathcal{A}}(i, a, b, R_i)$ 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 $f(x) = b$.

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

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

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 $n - 1$ [17]. 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 iff $\mathcal{A}$ 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 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*

$$\ell_{\mathcal{A}}(i,a,b) = \max_{R_i} \log |\mathcal{S}_{\mathcal{A}}(i,a,b,R_i)| \; .$$

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

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

The loss of a protocol $\mathcal{A}$ is called *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 will show that this is the case for symmetric functions and 1-phase protocols (as defined in Section 2.4).

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

$$\begin{aligned} s_{\mathcal{A}}(i,a,b,R_i) &= |\mathcal{S}_{\mathcal{A}}(i,a,b,R_i)| \quad \text{and} \\ s_{\mathcal{A}}(i,a,b) &= \max_{R_i} s_{\mathcal{A}}(i,a,b,R_i) \end{aligned}$$

for a given protocol $\mathcal{A}$. 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 will occasionally omit the subscript $\mathcal{A}$ and write $\ell(i,a,b)$ and $s(i,a,b)$.

Definition 4 *Let f be an n-ary Boolean function. We define for a network $G = (V,E)$ with $|V| = n$*

$$s_G(i,a,b) := \min_{\mathcal{A}} s_{\mathcal{A}}(i,a,b) \; .$$

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 P_i that are bridge nodes are in general able to infer some information about the input.

2.4 Phases in a Protocol

We say that a player 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.) That means, if $c[i_1 - h_1] = (b, P_{k_1}, P_{k_2})$, $c[i_1 + 1] = (b', P_{j_1}, P_{j_2})$, and $c[i_1 - \lambda] = \perp$ for all $0 \leq \lambda < h_1$, then P_{k_s} and P_{j_t} belong to different blocks. Here c is the communication string of P_q and $s,t \in \{1,2\}$ are chosen such that $k_s \neq q$ and $j_t \neq q$. During such an alternation, information can flow from one block to another.

We partition a communication sequence c of P_q into a minimal number of disjoint subsequences $c[1..i_1], c[(i_1+1)..i_2], \ldots$ such that each subsequence is alternation-free (i.e. P_q makes no alternation during the corresponding interval). To make such a partition unique assume that each subsequence (maybe except for the first one) starts with a non-empty message. We call these subsequences *block sequences* of c_i and define

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

with $i_0 = 0$. 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 has communicated with in the current phase.

Definition 5 *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 1-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.

Definition 6 *Let $f : \{0,1\}^{m_1} \times \{0,1\}^{m_2} \to \{0,1\}$ be a Boolean function and $\mathcal{B}$ be a two-party communication protocol for computing f. Let $y_1 \in \{0,1\}^{m_1}$ and $y_2 \in$*

$\{0,1\}^{m_2}$ *be two strings as input for the two parties. Then* $\mathrm{CC}_{\mathcal{B}}(y_1, y_2)$ *is the total number of bits exchanged by the two parties when executing* $\mathcal{B}$*.*

Furthermore, $\mathrm{CC}(\mathcal{B})$ *is the maximum number of bits exchanged by executing* $\mathcal{B}$ *on any input.*

Analogously, $\mathrm{CS}(\mathcal{B})$ *is the number of different communication strings that occur. (We simply concatenate the messages sent.) Finally,* $\mathrm{CC}(f) = \min_{\mathcal{B} \text{ for } f} \mathrm{CC}(\mathcal{B})$ *and* $\mathrm{CS}(f) = \min_{\mathcal{B} \text{ for } f} \mathrm{CS}(\mathcal{B})$.

$\mathrm{CC}(f)$ and $\mathrm{CS}(f)$ are the communication complexity and communication size, respectively, of the function f. $\mathrm{CC}(\mathcal{B})$ and $\mathrm{CS}(\mathcal{B})$ are the communication complexity and communication size for a certain protocol $\mathcal{B}$. The communication size is closely related to the number of leaves in a protocol tree, usually denoted by $C^P(\mathcal{B})$. In the definition of CS, we do not care about who has sent any bit, since we concatenate all messages. In a protocol tree however, each edge is labeled by the bit sent and by its sender. The bits on a path from the root to a leaf form a communication string. Usually, the messages sent in a communication protocol are assumed to be prefix-free. In this case, we can reconstruct the sender of any bit from the communication string. If this is not the case, then we can make a particular communication protocol prefix-free by replacing the messages sent in each round by prefix-free code words. There are prefix-free codes such that the length of each code-word is at most twice the length of the original message sent, for instance, $w \mapsto 1^{|w|-1}0w$. Thus, the communication complexity is at most doubled.

We also consider multi-party communication with a referee.

Definition 7 *Let* $f : \{0,1\}^{m_1} \times \ldots \times \{0,1\}^{m_k} \to \{0,1\}$ *be a function. Let* $A_1, \ldots, A_k$ *be* k *parties and* R *be a referee, all with unlimited computational power. For computing* f *on input* $x_1, \ldots, x_k$*, the referee cooperates with* $A_1, \ldots, A_k$ *as follows:*

- *Initially,* $x_1, \ldots, x_k$ *are distributed among* $A_1, \ldots, A_k$*, i.e.* A_i *knows* x_i*. The referee* R *does not have any knowledge about the inputs.*
- R*, in successive rounds, exchanges messages with* $A_1, \ldots, A_k$ *according to a communication protocol. In each round* R *can communicate (i.e. receive or send a message) only with a single party.*
- *After finishing the communications,* R *eventually computes the result of* f*.*

Let $\mathcal{B}$ *be a communication protocol for computing* f*. We denote by* $c^R_{\mathcal{B}}(x_1, \ldots, x_k)$ *the whole communication string of* R *after protocol* $\mathcal{B}$ *has been finished. More precisely,* $c^R_{\mathcal{B}}(x_1, \ldots, x_k)$ *is a concatenation of messages sent (to or from* R*) on input* $x_1, \ldots, x_k$ *with additional stamps describing the sender and the receiver of each message. For*

$b \in \{0,1\}$ *let*

$$\begin{array}{lcl} \mathcal{CS}^R_{\mathcal{B}}(b) & = & \{\, c^R_{\mathcal{B}}(x_1,\ldots,x_k) \mid \forall i \in [1..k] \,:\, x_i \in \{0,1\}^{m_i} \\ & & \qquad \textit{and } f(x_1,\ldots,x_k) = b \,\}\,, \\ \mathcal{CS}^R(\mathcal{B}) & = & \mathcal{CS}^R_{\mathcal{B}}(0) \cup \mathcal{CS}^R_{\mathcal{B}}(1)\,, \\ \mathrm{CS}^R_{\mathcal{B}}(b) & = & |\, \mathcal{CS}^R_{\mathcal{B}}(b) \,|\,, \textit{ and} \\ \mathrm{CS}^R(\mathcal{B}) & = & |\, \mathcal{CS}^R(\mathcal{B}) \,|\,. \end{array}$$

Finally, $\mathrm{CS}^R(f,b) = \min_{\mathcal{B} \textit{ for } f} \mathrm{CS}^R_{\mathcal{B}}(b)$ *and* $\mathrm{CS}^R(f) = \min_{\mathcal{B} \textit{ for } f} \mathrm{CS}^R(\mathcal{B})$.

Since the referee has to compute the result we obviously have

$$\mathcal{CS}^R_{\mathcal{B}}(0) \,\cap\, \mathcal{CS}^R_{\mathcal{B}}(1) \;=\; \emptyset\,.$$

3 The Suitability of the Model

The aim of this section is to justify the definitions given in Section 2. First, we argue that it is sufficient to consider bridge players when talking about the loss of a protocol. Second, we present a protocol for computing arbitrary Boolean functions using three players. Thus, it is possible to compute functions privately within any block, since the networks we consider are isthmus-free. Finally, 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 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. [12] (see also Section 3.2) such that for any $i \in [n]$ and any round r 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 Three Players are Sufficient for Private Computations

Nearly no function can be computed by a private protocol, if only two players are involved [3]. In this section, we show that three players are sufficient for any Boolean function: Any circuit can privately be evaluated on a component that consists only of three players P_1, P_2, and P_3. This will be shown by modifying the protocol presented by Kushilevitz et al. [12].

Let $x_{i,1}, \ldots, x_{i,k_i}$ be the input bits of P_i ($i = 1, 2, 3$). We simulate the circuit gate by gate. Before starting the simulation, we proceed as follows:

1. Player P_1 generates $k_1 + k_2 + k_3$ random bits $r_{1,1}, \ldots, r_{1,k_1}$, $r_{2,1}, \ldots, r_{2,k_2}$, and $r_{3,1}, \ldots, r_{3,k_3}$ and sends $r_{2,1}, \ldots, r_{2,k_2}$ to P_2 and $r_{3,1}, \ldots, r_{3,k_3}$ to P_3.
2. P_i ($i = 1, 2, 3$) computes $x_{i,j} \oplus r_{i,j}$ for all $j \in [k_i]$.
3. P_1 sends $x_{1,j} \oplus r_{1,j}$ ($j \in [k_1]$) to both P_2 and P_3, P_2 sends $x_{2,j} \oplus r_{2,j}$ ($j \in [k_2]$) to P_3, and P_3 sends $x_{3,j} \oplus r_{3,j}$ ($j \in [k_3]$) to P_2.

Now P_1, P_2, and P_3 start to evaluate the circuit gate by gate.

For simulating a gate g, P_1 generates 5 new random bits: one random r_c to mask the output c of g and a random 2×2 matrix

$$Z := \begin{pmatrix} z_{0,0} & z_{0,1} \\ z_{1,0} & z_{1,1} \end{pmatrix}.$$

Let G be the matrix describing gate g, a and b be the input bits of g, and c be its output bit. Furthermore, let r_a and r_b be the random bits that mask a and b, respectively. The random bit r_c will mask the output c of g after the computation. At any stage of the

computation, both P_2 and P_3 know $r_a \oplus a$ and $r_b \oplus b$ while P_1 knows r_a and r_b. P_1 computes the bitwise XOR of these matrices:

$$\begin{aligned} F &= \begin{pmatrix} r_c & r_c \\ r_c & r_c \end{pmatrix} \oplus Z_c \oplus G \\ &= \begin{pmatrix} r_c \oplus z_{0,0} \oplus g(0,0) & r_c \oplus z_{0,1} \oplus g(0,1) \\ r_c \oplus z_{1,0} \oplus g(1,0) & r_c \oplus z_{1,1} \oplus g(1,1) \end{pmatrix} = \begin{pmatrix} f_{0,0} & f_{0,1} \\ f_{1,0} & f_{1,1} \end{pmatrix} . \end{aligned}$$

P_1 exchanges the rows of F, if $r_a = 1$ and exchanges the columns of F, if $r_b = 1$. He proceeds with Z analogously. Let F' and Z' be the resulting matrices, i.e.

$$F' = \begin{pmatrix} f'_{0,0} & f'_{0,1} \\ f'_{1,0} & f'_{1,1} \end{pmatrix} = \begin{pmatrix} f_{r_a,r_b} & f_{r_a,1-r_b} \\ f_{1-r_a,r_b} & f_{1-r_a,1-r_b} \end{pmatrix}$$

and

$$Z' = \begin{pmatrix} z'_{0,0} & z'_{0,1} \\ z'_{1,0} & z'_{1,1} \end{pmatrix} = \begin{pmatrix} z_{r_a,r_b} & z_{r_a,1-r_b} \\ z_{1-r_a,r_b} & z_{1-r_a,1-r_b} \end{pmatrix} .$$

Then P_1 sends Z' to P_2 and F' to P_3.

Now P_2 sends $z'_{r_a \oplus a, r_b \oplus b}$ to P_3 while P_3 sends $f'_{r_a \oplus a, r_b \oplus b}$ to P_2. Finally, P_2 and P_3 compute

$$c \oplus r_c = z'_{r_a \oplus a, r_b \oplus b} \oplus f'_{r_a \oplus a, r_b \oplus b} .$$

After this computation, both P_2 and P_3 know $c \oplus r_c$, which is the output of the gate masked with r_c. The matrix sent by P_1 has been masked with Z. Furthermore, P_2 knows Z' but does not know F', while P_3 knows F' but does not know Z'. Thus, neither P_2 nor P_3 is able to compute r_c; the computation is private with respect to both players. Although P_1 knows r_c, he does not know $c \oplus r_c$. Since he does not receive anything during the whole simulation the computation is private with respect to P_1. The privacy of the complete simulation follows by induction on the number of gates.

To compute the result of the function, we proceed as follows:

- If P_1 has to compute the output of the circuit, then P_2 sends the masked output bit to P_1. Thus, P_1 can compute the result.
- If P_i ($i \in \{2,3\}$) has to compute the output, then P_1 sends the random bit that masks the output bit to P_i. Therefore, P_i can compute the result.

3.3 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, $a, b \in \{0,1\}$, and R_q be the random string provided to P_q. We define

$$X = \{x \in \{0,1\}^n \mid x[i] = a \wedge f(x) = b\}$$

and for any communication string c_i

$$\psi(c_i) = \{x \in X \mid \mu_x(c_i) > 0\}.$$

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

Theorem 2 *If $s_G(q,a,b) > 1$, then for any protocol $\mathcal{A}$ and every communication string c_i that can be observed by P_q on input $x \in X$, $\psi(c_i)$ is a non-trivial subset of X, i.e. $\emptyset \neq \psi(c_i) \subsetneq X$, and there exist at least $s_G(q,a,b)$ different such sets. Hence, from seeing c_i on $x \in X$, P_q always gains some information and there are at least $s_G(q,a,b)$ different peaces 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* measures the similarity of μ and μ' and is defined by

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

Theorem 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 2 follows directly from the Lemmas 1 and 2 below. Theorem 3 follows from Lemma 3.

Lemma 1 *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 $\psi(\hat{c}) = X$. Then using Lemma 9 for $w = \hat{c}$, we can construct a communication protocol $\mathcal{B}$ for computing $f\lceil_{\{q\}\leftarrow a}$ such that $\mathcal{CS}^R_{\mathcal{B}}(b) = 1$. By Lemma 8 we have $s_G(q,a,b) \leq \mathrm{CS}^R(f\lceil_{\{q\}\leftarrow a}, b) \leq 1$. ∎

Lemma 2 *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 $c_1, c_2, c_3, \ldots$. By applying Lemma 9 for $w = \gamma^\tau$, we obtain the communication protocol $\mathcal{B}$ for computing $f\lceil_{\{q\}\leftarrow a}$ such that $|\mathcal{CS}^R_{\mathcal{B}}(b)| = |\mathcal{M}|$. By Lemma 8 we have $s_G(q,a,b) \leq \mathrm{CS}^R(f\lceil_{\{q\}\leftarrow a}, b) \leq |\mathcal{M}|$.

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 $\mu_x(\hat{c}_i) > 0$ iff $\mu_{x'}(\hat{c}_i) > 0$. 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 *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 $\mu_i(\hat{c}), \mu_j(\hat{c}) > 0$. Using Lemma 9 for $w = \hat{c}$ we construct a communication protocol $\mathcal{B}$ for $f\lceil_{\{q\}\leftarrow a}$ such that $\mathrm{CS}^R_{\mathcal{B}}(b) \leq m-1$. By Lemma 8 we have $s_G(q,a,b) \leq \mathrm{CS}^R(f\lceil_{\{q\}\leftarrow a}, b) \leq \mathrm{CS}^R_{\mathcal{B}}(b) \leq m-1$. ∎

4 Communication Complexity and Private Computation

4.1 Two-Party Model

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.

The communication complexity and the communication size are closely related.

Lemma 4

$$\tfrac{1}{2}\log(\mathrm{CS}(f)) \leq \mathrm{CC}(f) \leq 3 \cdot \log(\mathrm{CS}(f)) + O(1) .$$

The proof follows from the relation between C^P and CC (see e.g. Kushilevitz and Nisan [11, Sec. 2.2]). Making a communication protocol prefix-free yields the extra factor $\frac{1}{2}$.

Now we investigate the relations between communication size and the size of an information source on graphs that consist of two blocks sharing one bridge node P_q.

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 y_1 of Alice with the input bits known by players of the first block of the network and the input y_2 of Bob with the input bits known by the players of the second block. The input bit known by the bridge player P_q is known by both Alice and Bob. Let x be the input string for the players in the network. We consider the problem for $x[q] = 0$ and $x[q] = 1$ separately.

Lemma 5 *For $a \in \{0,1\}$ we have*

$$\max\{s_G(q,a,0),\ s_G(q,a,1)\} \leq \mathrm{CS}(f\lceil_{\{q\}\leftarrow a}) \,.$$

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

The protocol $\mathcal{B}$ is deterministic. Thus, for any input y_1, y_2 there exist two functions $A : \{0,1\}^{m_1} \times \{0,1\}^\star \to \{0,1\}^\star$ and $B : \{0,1\}^{m_2} \times \{0,1\}^\star \to \{0,1\}^\star$ such that the sequence of bits exchanged by Alice and Bob in $\mathcal{B}$ can be computed by evaluating A and B:

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

By introducing a third symbol, we can modify the protocol such that the results of all evaluations of A and B for every input $y_1 \in \{0,1\}^{m_1}$ and $y_2 \in \{0,1\}^{m_2}$ have equal length and the number of rounds is the same for all inputs. This can be done without increasing the size of the communication protocol. In the private protocol constructed, this third symbol can be simulated by sending no bit and waiting for one round.

A can privately be computed on the first block such that only P_q knows the result of the computation. Analogously, B can be computed on the second block. 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. ∎

Lemma 6 *For $a \in \{0,1\}$ we have*

$$\mathrm{CS}(f\lceil_{\{q\}\leftarrow a}) \leq s_G(q,a,0) + s_G(q,a,1) \,.$$

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 of the information source of P_q 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 be the communication string observed by P_q for a fixed content of the random tapes $R_1, \ldots, R_n$ and input x. For odd j, $\mathrm{block}_j(c)$ can deterministically be computed from $R_1, \ldots, R_q$, $x[1..q]$, and $\mathrm{block}_i(c)$ with $i < j$. Analogously for even j, $\mathrm{block}_j(c)$ can be computed from $R_q, \ldots, R_n$, $x[q..n]$, and $\mathrm{block}_i(c)$ with $i < j$.

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 say that a string $c' \in (\{0,1\} \times V^2 \cup \{\perp\})^*$ is a valid prefix, if it can be extended to a communication string c (i.e. $c = c'u$ for some $u \in (\{0,1\} \times V^2 \cup \{\perp\})^*$) such that $\mu_x(c) > 0$ and P_q makes an alternation in c between c' and u. (Here μ_x are probabilities as defined in Definition 2.) We simulate the private protocol iteratively as follows:

1. a_i is the lexicographically minimal block sequence (in which P_q communicates with the first block) such that $a_1 b_1 \ldots a_{i-1} b_{i-1} a_i$ is a valid prefix, if the content of the random tapes $R_1, \ldots, R_{q-1}$ is in $\mathcal{R}_A^{i-1}$, the inputs of $P_1, \ldots, P_q$ are given by $x[1..q]$ and the substrings of the previous block sequences are given by $a_1, b_1, \ldots, a_{i-1}, b_{i-1}$. (The symbol $\perp$ is considered smaller than any tuple when considering lexicographical orderings.)

2. $\mathcal{R}_A^i \subseteq \mathcal{R}_A^{i-1}$ is the set of all possible contents of the random tapes of $P_1, \ldots, P_{q-1}$ with the property that the prefix of the communication string observed by player P_q is $a_1 b_1 \ldots a_{i-1} b_{i-1} a_i$.

3. b_i is the lexicographically minimal block sequence (in which P_q communicates with the second block) such that $a_1 b_1 \ldots a_i b_i$ is valid, if the content of the random tapes $R_{q+1}, \ldots, R_n$ is in $\mathcal{R}_B^{i-1}$, the inputs of $P_q, \ldots, P_n$ are given by $x[q..n]$, and the substrings of the previous block sequences are given by $a_1, b_1, \ldots, a_i$.

4. $\mathcal{R}_B^i \subseteq \mathcal{R}_B^{i-1}$ is the set of possible contents of the random tapes of $P_{q+1}, \ldots, P_n$ such that the prefix of the communication string observed by P_q is $a_1 b_1 \ldots a_i b_i$.

To get a communication protocol, Alice and Bob iteratively compute a_i, b_i, $\mathcal{R}_A^i$, and $\mathcal{R}_B^i$ and exchange a_i and b_i. The correctness of this protocol follows from the correctness of the private protocol.

It remains to show that whenever Alice and Bob generate two different communication sequences for two different input pairs y_1, y_2 and y_1', y_2', then the two corresponding inputs x and x' for the private protocol $\mathcal{A}$ instantiate two different distributions μ and μ' (where $\mu := \mu_x$ and $\mu' = \mu_{x'}$).

We have to consider the following cases:

1. $a_i = a_i'$ and $b_i = b_i'$ for all $i \leq \min\{k, k'\}$. Then $k \neq k'$. W.l.o.g. we assume that $k < k'$.

On input x', the private protocol cannot stop after P_q has seen $a_1, b_1, \ldots, a_k, b_k$. Hence, $\mu'(a_1 b_1 \ldots a_k b_k) = 0$ but $\mu(a_1 b_1 \ldots a_k b_k) > 0$. Therefore, the distributions μ and μ' are different.

The case that the shorter sequence ends with a_k (instead of b_k) is treated in the same manner.

2. There exists some $i_0 \leq \min\{k, k'\}$ such that $a_{i_0} \neq a'_{i_0}$ or $b_{i_0} \neq b'_{i_0}$. Let i_0 be minimal such that $a_{i_0} \neq a'_{i_0}$ or $b_{i_0} \neq b'_{i_0}$. In the following we assume that $a_{i_0} <_{\text{lex}} a'_{i_0}$. The case that $a_{i_0} = a'_{i_0}$ and $b_{i_0} \neq b'_{i_0}$ follows analogously.

   From the construction of the substrings a_{i_0} and a'_{i_0}, we have the following: If $\mu'(a_1 b_1 \ldots a_{i_0} u) \neq 0$ for some u such that there is an alternation between $a_1 b_1 \ldots a_{i_0}$ and u, then our algorithm would prefer to use a_{i_0} on input x', too. Hence, $\mu'(a_1 b_1 \ldots a_{i_0} u) = 0$ for all such u. On the other hand, there exists some u such that $\mu(a_1 b_1 \ldots a_{i_0} u) \neq 0$ and there is an alternation between $a_1 b_1 \ldots a_{i_0}$ and u. Thus, the distributions μ and μ' are different.

∎

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.

The proofs above give us even more.

Lemma 7 *For any function f and any protocol $\mathcal{A}$ for computing f there exists a communication protocol $\mathcal{B}$ for computing $f\lceil_{\{q\} \leftarrow a}$ ($a \in \{0,1\}$) such that*

$$\mathrm{CS}(\mathcal{B}) = |\mathcal{S}_{\mathcal{A}}(q, a, 0) \cup \mathcal{S}_{\mathcal{A}}(q, a, 1)| \ .$$

Vice versa, for any communication protocol $\mathcal{B}$ for computing $f\lceil_{\{q\} \leftarrow a}$ there exists a protocol $\mathcal{A}$ for computing f such that the same equality holds.

Theorem 4 *If a function f has communication complexity c then there exists a protocol for computing f with loss bounded by $2c$. On the other hand, if f can be computed by a protocol with loss bounded by λ, then the communication complexity of f is bounded by $6\lambda + O(1)$.*

Proof: If f has communication complexity c, then $\mathrm{CS}(f) \leq 2 \cdot 2^c$ by Lemma 4. Particularly $\mathrm{CS}(f\lceil_{\{q\} \leftarrow a}) \leq 2 \cdot 2^c$. Therefore, $\ell_G(q, a, b) \leq 2c$ for all $b \in \{0,1\}$ by Lemma 5. On the other hand, if there is a protocol for computing f whose loss is bounded by λ, then $\mathrm{CS}(f\lceil_{\{q\} \leftarrow a}) \leq 2^{2\lambda}$ by Lemma 6. Thus $\mathrm{CC}(f\lceil_{\{q\} \leftarrow a}) \leq 6\lambda + O(1)$ by Lemma 4. Now the result follows from $\mathrm{CC}(f) \leq \max\{\mathrm{CC}(f\lceil_{\{q\} \leftarrow a}) \mid a \in \{0,1\}\} + O(1)$. The latter inequality holds, since Alice and Bob only have to agree on the value of $x[q]$. ∎

4.2 Multi-Party with Referee

In this section we generalize our previous results to multi-party communication. We generalize Lemmas 5 and 6 in two directions. First, we show 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 with a referee. Second, we show that these bounds still hold if we restrict the set of communication strings allowed.

Lemma 8 *For $a, b \in \{0,1\}$ we have*

$$s_G(q, a, b) \leq \mathrm{CS}^R(f\lceil_{\{q\}\leftarrow a}, b) \; .$$

Proof: Let $\mathcal{B}$ be a deterministic communication protocol for $f\lceil_{q\leftarrow a}$. 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 $\mathrm{CS}^R_{\mathcal{B}}(b)$ for every $b \in \{0,1\}$.

Since $\mathcal{B}$ is deterministic, there exist k functions $T_i : \{0,1\}^{m_i} \times \{0,1\}^\star \to \{0,1\}^\star$ for $i \in [1..k]$ and a function $B : \{0,1\}^\star \to [1..k]$ 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 6. In addition to the simulation in Lemma 9 it allows us to address one distinguished communication sequence to be used, if this sequence has positive probability for the input of $\mathcal{A}$. For a communication sequence c we define the weighted lexicographic order $\leq^c_{\mathrm{lex}}$: For every pair of communication sequences c_1, c_2

let $\ell = \min\{|c_1|, |c_2|, |c|\}$. Define

$$c_1 \leq^c_{\text{lex}} c_2 \quad :\Longleftrightarrow \quad \begin{cases} c_1 = c & \text{or} \\ \exists i \leq \ell : \quad c_1[1..i] = c[1..i] \neq c_2[1..i] & \text{or} \\ \exists i \leq \ell : \quad c_1[1..i] = c[1..i] = c_2[1..i] & \\ \qquad \wedge\, c_1[i+1] \neq c[i+1] \neq c_2[i+1] & \\ \qquad \wedge\, c_1[i+1..|c_1|] \leq_{\text{lex}} c_2[i+1..|c_2|] \,, & \end{cases}$$

where $\leq_{\text{lex}}$ is the lexicographical ordering of two strings as follows: Let $b_1, b_2 \in (\{0,1\} \times V^2)^*$ be two communication sequences such that each of the sequences describes the communication of the bridge player P_q with players of one block. Then

$$b_1 <_{\text{lex}} b_2 :\Longleftrightarrow \quad \begin{cases} b_1 \text{ is the empty string and } |b_2| \geq 1, \\ \text{the block index of } b_1 \text{ is smaller than the block index of } b_2, \text{ or} \\ b_1 \text{ is lexicographically smaller than } b_2. \end{cases}$$

For a pair of communication sequences $c_1, c_2 \in (\{0,1\} \times V^2)^*$ let $\text{block}_1(c_i), \ldots, \text{block}_{d_i}(c_i)$ with $i \in \{1,2\}$ be the sequence of block sequences of c_i. Then define

$$c_1 \leq_{\text{lex}} c_2 \quad :\Longleftrightarrow \quad \exists i \leq \min\{d_1, d_2\}\, \forall k \leq i \; : \; \begin{array}{l} \text{block}_k(c_1) \;=\; \text{block}_k(c_2) \\ \wedge\, \text{block}_k(c_1) \leq_{\text{lex}} \text{block}_k(c_2) \,. \end{array}$$

Using the weighted lexicographic order $\leq^c_{\text{lex}}$, the communication sequence c is always the minimum string.

Lemma 9 *For every $a, b \in \{0,1\}$, every protocol $\mathcal{A}$ computing a function f, every content of P_q's random tape R_q, and every sequence $w \in (\{0,1\} \times V^2)^*$ there exists a communication protocol $\mathcal{B}$ computing $f^{\lceil}_{\{q\} \leftarrow a}$ with*

$$\text{CS}^R_{\mathcal{B}}(b) \leq s_{\mathcal{A}}(q, a, b, R_q) \,.$$

Moreover,

$$|\, \mathcal{CS}^R_{\mathcal{B}}(b) \,| \;=\; |\, \{\, c \mid \exists \mu \in \mathcal{S}_{\mathcal{A}}(q, a, b, R_q) \;:\; \mu(c) > 0 \text{ and } \forall c' \in (\{0,1\} \times V^2)^* \;:\; \mu(c') > 0 \;\Rightarrow\; c \leq^w_{\text{lex}} c' \,\} \,| \,.$$

Proof: Let $V_1, \ldots, V_\ell$ 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. We construct a communication protocol by simulating $\mathcal{A}$ and searching the lexicographically minimal communication sequence according to $\leq^w_{\text{lex}}$ for P_q that has positive probability.

Let c be the communication string observed by P_q for a fixed content of the random tapes $R_1, \ldots, R_n$ 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 $h(R_q, x[q], \mathrm{block}_1(c) \dots \mathrm{block}_j(c))$ be the function that determines this index.

We say that a string $c' \in (\{0,1\} \times V^2)^*$ is a valid prefi x, if it can be extended to a communication string c, i.e. $c = c'u$ for some $u \in (\{0,1\} \times V^2)^*$ 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 $a_0 = \lambda$ 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 $A_1, \dots, A_k$. Then iteratively for $j = 1, 2, \dots$

1. R computes the index $i_j = h(R_q, x[q], a_0 \dots a_{j-1})$ and $a_0 \dots a_{j-1}$ to the party A_{i_j}.

2. The party A_{i_j} determines the set H_j of all strings a such that $a_0 \dots a_{j-1}a$ is a valid prefi x of a communication sequence 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 $a_j \in H_j$ such that for any $a \in H_j$

$$a_0 \dots a_{j-1}a_j \leq^w_{\mathrm{lex}} a_0 \dots a_{j-1}a$$

   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 prefi x of the communication string observed by P_q is $a_1 \dots a_{j-1}a_j$. Finally, A_{i_j} sends a_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 $A_1, \dots, A_\ell$, and R iteratively compute i_j, a_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 $A_1, \dots, A_\ell$, and R generate two different communication sequences for two different inputs $x_1, \dots, x_\ell$ and $x'_1, \dots, x'_\ell$, 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 $\leq^w_{\mathrm{lex}}$ with positive probability in μ_x differs from the corresponding string with positive probability in $\mu_{x'}$.

We distinguish the following cases:

1. $a_i = a'_i$ for all $i \leq \min\{\ell, \ell'\}$. Then $\ell \neq \ell'$. W.l.o.g. we assume that $\ell < \ell'$. Then either

(a) w is a prefix of $a_1 \dots a_\ell$ or $w = a_1 \dots a_\ell$: Then on input x', the bridge player P_q continues to exchange messages with some other players after seeing $a_1 \dots a_k$. The protocol cannot stop on x' at this point in time. Hence, $\mu_{x'}(a_1 \dots a_k) = \mu_{x'}(w) = 0$ but $\mu_x(a_1 \dots a_k) = \mu_x(w) > 0$. Therefore, the minimal communication sequence with positive probability in μ_x differs from the corresponding string with positive probability in $\mu_{x'}$.

(b) $a_1 \dots a_\ell$ is a prefix of w of length m with $m < |w|$. Then $a_1 \dots a_\ell w[m+1]$ is not a valid prefix of a communication sequence on input x. Hence, $a_1 \dots a_\ell$ is the minimal communication sequence with positive probability in μ_x.

According to $a'_1 \dots a'_{\ell'}$ we consider two cases:

i. $a'_1 \dots a_{\ell'} w[m+1]$ is a valid prefix of some communication sequence on input x'. Therefore $a'_1 \dots a'_\ell w[m+1]$ is a prefix of $a'_1 \dots a'_{\ell'}$ and $a_1 \dots a_\ell <^w_{\text{lex}} a'_1 \dots a'_{\ell'}$.

ii. $a'_1 \dots a_{\ell'} w[m+1]$ is not a valid prefix of a communication sequence on input x' and the protocol cannot stop on x' after P_q has seen $a'_1 \dots a_{\ell'}$. Hence, for every communication sequence c with $\mu_{x'}(c)$ (and in particular for $c = a'_1 \dots a'_{\ell'}$) we have $c <^w_{\text{lex}} a_1 \dots a_\ell$.

2. There exists some $i_0 \leq \min\{\ell, \ell'\}$ such that $a_{i_0} \neq a'_{i_0}$. Let i_0 be minimal such that $a_{i_0} \neq a'_{i_0}$. In the following we assume that $a_1 \dots a_\ell <^w_{\text{lex}} a'_1 \dots a'_{\ell'}$. We distinguish the case that a_{i_0} is not a prefix of a'_{i_0} and the case that a_{i_0} is a prefix of a'_{i_0}:

- Let us first assume that a_{i_0} is not a prefix of a'_{i_0}. Then by our construction of the substrings a_{i_0} and a'_{i_0}, it follows that if $\mu_{x'}(a_1 \dots a_{i_0} u) \neq 0$ for some u such that there is an alternation between $a_1 \dots a_{i_0}$ and u, then our algorithm would prefer to use a_{i_0} on input x', too. Hence, $\mu'_x(a_1 \dots a_{i_0} u) = 0$ for all such u. On the other hand, for $u = a_{i_0+1} \dots a_\ell$ we have $\mu_x(a_1 \dots a_{i_0} u) \neq 0$ and there is an alternation between $a_1 \dots a_{i_0}$ 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 $\mu_{x'}$.
- Let us now assume that a_{i_0} is a prefix of a'_{i_0}. If $i_0 = \ell$, that means $a_1 \dots a_{i_0}$ gives a complete communication string on input x then the claim follows analogously to the first case.

  If $i_0 < \ell$, then we have $a_1 \dots a_{i_0} a_{i_0+1} <^w_{\text{lex}} a'_1 \dots a'_{i_0}$. By our construction of the substrings $a_{i_0} a_{i_0+1}$ and a'_{i_0}, it follows that if $\mu_{x'}(a_1 \dots a_{i_0} a_{i_0+1} u) \neq 0$ for some u such that there is an alternation between $a_1 \dots a_{i_0}$ and u, then our algorithm prefers to use $a_{i_0} a_{i_0+1}$ on input x', too. Hence, we have $\mu'_x(a_1 \dots a_{i_0} a_{i_0+1} u) = 0$ for all such u. On the other hand, for $u = a_{i_0+2} \dots a_\ell$ we have $\mu_x(a_1 \dots a_{i_0} a_{i_0+1} u) \neq 0$ and there is an alternation

between $a_1 \dots a_{i_0}$ 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 $\mu_{x'}$.

∎

Summarizing Lemma 8 and 9 we get:

Theorem 5 *For $a, b \in \{0,1\}$ we have*

$$s_G(q, a, b) = \text{CS}^R(f\lceil_{\{q\}\leftarrow a}, b) \ .$$

5 1-Phase Protocols

5.1 Orderings

We start our study of 1-phase protocols with considering networks that consist of one bridge player who is incident with d blocks. If 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 1-phase protocols and communication size of multiparty 1-way protocols. We prove that for some Boolean functions there exists no fixed order that minimizes the loss of information of 1-phase protocols. On the other hand we prove that for every symmetric Boolean function 1-phase protocols can minimize the loss of information when the bridge player sorts the blocks by increasing size. Then we present a simple 1-phase protocol on arbitrarily connected network that is optimal for every symmetric function.

Lemma 10 *Let $\mathcal{B}$ be a 1-way communication protocol. Then the party that sends messages to the second party is independent of the actual input.*

Proof: Assume that there are strings $y_1, y_1' \in \{0,1\}^{m_1}$ and $y_2, y_2' \in \{0,1\}^{m_2}$ such that

- on input y_1, y_2 Alice sends messages to Bob and
- on input y_1', y_2' Bob sends messages to Alice.

Then on input y_1, y_2' both Alice and Bob send messages to each other. Hence, the protocol violates the restrictions of a 1-way communication protocol. ∎

Analogously, we can show the following lemma.

Lemma 11 *Let G be a connected network with one bridge player P_q and let $\mathcal{A}$ be a 1-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 $P_i \neq P_q$.*

Note that the communication order of P_q may depend on the input of P_q.

A natural extension of the two-party scenario for 1-way communication is a scenario in which the parties use a directed chain for communication. Hence, we consider parties $A_1, \ldots, A_d$ that are connected by a directed chain, i.e. A_i can only send messages to A_{i+1} for $i \in [d-1]$. 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 $A_1, \ldots, A_i$. Each communication protocol $\mathcal{B}$ can be modified without increasing $S_i^{\mapsto}(\mathcal{B})$ ($i \in [1..d]$) 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 will 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 will consider a chain of d parties $A_1, \ldots, A_d$. For a σ-ordered 1-phase protocol $\mathcal{A}$ we will 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 $f\lceil_{\{q\}\leftarrow a}$ whenever we analyse the communication size of a communication protocol.

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

$$\mathcal{S}_{\mathcal{A}}^{[i]}(q,a,b,R_q) \;:=\; \{(\widehat{p}_1(x),\widehat{p}_2(x),\widehat{p}_3(x),\ldots) \mid x[q]=a \text{ and } f(x)=b\,\},$$

where

$$\widehat{p}_k(x) := \sum_{c_j \text{ with } \widehat{c}_i \,=\, \mathrm{block}_1(c_j)\ldots\mathrm{block}_k(c_j)} \Pr[C_q = c_j \mid R_q, x]$$

and $\widehat{c}_1, \widehat{c}_2, \widehat{c}_3, \ldots$ 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 $a \in \{0,1\}$ of player P_q define

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

Lemma 12 *For $a \in \{0,1\}$ let $\mathcal{B}$ be a d-party 1-way communication protocol computing $f\lceil_{\{q\}\leftarrow a}$ on a chain network. Then there exists a σ-ordered 1-phase protocol $\mathcal{A}$ computing f such that for all $i \in [1..d-1]$ and for every content R_q of P_q's random tape*

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

Proof: We use a simulation analogously to the simulation in Lemma 5. According to our observations above we can conclude for any input $x \in \{0,1\}^n$ 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 13 *Let $\mathcal{A}$ be a σ-ordered 1-phase protocol for computing f on a network as described above. Then for every $a \in \{0,1\}$ and every content R_q of P_q's random tape there exists a 1-way communication protocol $\mathcal{B}$ for computing $f\lceil_{\{q\}\leftarrow a}$ such that for all $i \in [1..d-1]$*

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

Proof: Analogously to our simulation in Lemma 6 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 1 *Let $a \in \{0,1\}$ and $\mathcal{B}$ be a communication protocol as described above for computing $f\lceil_{\{q\}\leftarrow a}$ on a chain network. Then there exists a σ-ordered 1-phase protocol $\mathcal{A}$ for computing f such that for all $b \in \{0,1\}$, every $j \in [1..d-1]$, and every content R_q of P_q's random tape the following holds:*

> *If we restrict the inputs to $x \in \{0,1\}^{n-1}$ with $f\lceil_{\{q\}\leftarrow a}(x) = b$, 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 1-phase protocol for computing f on a network as described above. Then for every $a \in \{0,1\}$, every content R_q of P_q's random tape, and every $b \in \{0,1\}$ there exists a 1-way communication protocol $\mathcal{B}$ for computing $f\lceil_{\{q\}\leftarrow a}$ such that the following properties hold for all $i \in [1..d-1]$:

If we restrict the inputs to $x \in \{0,1\}^{n-1}$ with $f\lceil_{\{q\}\leftarrow a}(x) = b$, the number of possible communication sequences on the subnetwork of $A_1, \ldots, A_{i+1}$ is bounded 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

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

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 on the subnetwork $A_1, \ldots, A_{i+1}$, if the message sent by A_i specifi es 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 has also be provided to the bridge player by the probability distribution of a σ-ordered 1-phase protocol. Hence, for every fi xed R_q and $b \in \{0,1\}$ 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 14 *For $a \in \{0,1\}$ let $\mathcal{B}$ be a communication protocol for computing $f\lceil_{\{q\}\leftarrow a}$ on a chain network. Then there exists a σ-ordered 1-phase protocol $\mathcal{A}$ for computing f such that for all $i \in [1..d-1]$ and every content R_q for P_q's random tape*

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

Furthermore, for any $b \in \{0,1\}$ it holds: If we restrict the inputs to $x \in \{0,1\}^{n-1}$ with $f\lceil_{\{q\}\leftarrow a}(x) = b$, 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

Although we can show that there exist functions, for which no ordered 1-phase protocol minimizes the size of the bridge player's information source, we can prove such a property for quasi-ordered 1-phase protocols.

We call a protocol $\mathcal{A}$ *quasi-ordered* if for every $a, b \in \{0,1\}$, 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 1-phase ordering σ such that every communication string c with $\mu(c) > 0$ the string c is σ-ordered.

Lemma 15 *Let G be a connected network with one bridge player P_q and d blocks. Then for every 1-phase protocol $\mathcal{A}$ there exists a quasi-ordered 1-phase protocol $\mathcal{A}'$ such that for all $a, b \in \{0,1\}$ 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 11 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 1-phase protocol for computing f on G.

According to Lemma 11 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 $a, b \in \{0,1\}$ and the content R_q of P_q's random tape. For every input $x \in \{0,1\}^n$ 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 $\{f_1, \ldots, f_t\}$ be the set of all different subfunctions $f_{1,x}$ ($x \in \{0,1\}$ with $x[q] = a$ and $f(x) = b$). For every subfunction f_i let x_i be an input with $f_i = f_{1,x_i}$ 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 1-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 1-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 14, we obtain the following lemma.

Lemma 16 *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 1-phase protocol $\mathcal{A}'$ there exists a σ-ordered 1-phase protocol $\mathcal{A}$ such that for all $a, b \in \{0,1\}$, for all $i \leq d-1$, and every content R_q of P_q' random tape*

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

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 $a, b \in \{0,1\}$ 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 1-phase protocol $\mathcal{A}$ that uses an order like in Lemma 16 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 1-phase protocols $\mathcal{A}'$.

Lemma 17 *Let G be a connected network with one bridge player P_q and d blocks. Let σ be a 1-phase ordering that enumerates the blocks of G according to their size. Then for every 1-phase protocol $\mathcal{A}'$ there exists a σ-ordered 1-phase protocol $\mathcal{A}$ such that for all $a, b \in \{0,1\}$*

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

Proof: If $\mathcal{A}'$ is an ordered protocol then the claim follows directly from Lemma 16.

In the following we will assume, that π is a 1-phase ordering that enumerates the blocks of G according to their size and fulfils 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 1-phase protocols $\mathcal{A}'$ such that for every ordered 1-phase protocols $\mathcal{A}$ we have

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

By Lemma 15 we can assume that $\mathcal{A}'$ is quasi-ordered.

For a 1-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 1-phase protocol $\mathcal{A}$, $a, b \in \{0,1\}$, 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 1-phase protocols $\mathcal{A}'$ such that

- $\mathcal{A}'$ has a minimum degree of disorder $\Delta_{\mathcal{A}}(a,b)$ over all quasi-ordered 1-phase protocols fulfilling Equation 1 and
- for every ordered 1-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 1-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 1-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 behaviour 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 16 and modify $\mathcal{A}'$ by replacing $\mathcal{A}''$ with an ordered 1-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 16 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 1-phase protocols fulfilling Equation 1. ∎

5.4 An Optimal 1-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 1-phase protocol $\mathcal{A}$ *increasing-ordered*, if every bridge player works in increasing order. This generalizes the ordering of $\mathcal{A}$ chosen in Lemma 17.

Theorem 6 *Let G be a 2-edge-connected network and f be a symmetric Boolean function. Then for every 1-phase protocol $\mathcal{A}'$ computing f on G there exists an increasing-ordered 1-phase protocol $\mathcal{A}$ for f on G such that for every player P_i and for all $a, b \in \{0,1\}$*

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

Proof: To prove this claim 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 P_q privately computes $f_{i,x} = f\lceil_{\bigcup_{j=1}^{i} \mathcal{I}_j \leftarrow x[\bigcup_{j=1}^{i} \mathcal{I}_j]}$ iteratively for $1 \leq i < d_q$ on B_i. Therefore, P_q chooses an arbitrary string $\alpha_i \in \{0,1\}^{|\bigcup_{j=1}^{i-1} \mathcal{I}_j|}$ such that $f\lceil_{\bigcup_{j=1}^{i-1} \mathcal{I}_j \leftarrow \alpha_i} = f_{i-1,x}$ 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 $\alpha_{d_q} \in \{0,1\}^{|\bigcup_{j=1}^{d_q-1} \mathcal{I}_j|}$ such that

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

   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 $f_{d_q,x} = f_{d_q-1,x}\lceil_{\mathcal{I}_{d_q} \leftarrow x[\mathcal{I}_{d_q}]}$ on B_{d_q}.

   (b) If $|\mathcal{I}_{d_q}| > \#_q$, $\#_q = \max\{\#_{q'} \mid P_{q'} \text{ is a bridge player}\}$, and $q < q'$ for all bridge player $P_{q'}$ with $\#_{q'} = \#_q$, then P_q privately computes $f_{d_q,x} = f_{d_q-1,x}\lceil_{\mathcal{I}_{d_q} \leftarrow x[\mathcal{I}_{d_q}]}$ 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 14 and 17 to show that the ordering of the blocks for computing the sequence is optimal with respect to the size of the information source. ∎

Corollary 1 *The protocol presented in this section is optimal for 1-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 7 *For p_{2k-1}, $s_G^k(i,a,b) = 2^{O(k \log n)}$ for all a, b.*

Proof: We get a lower bound via the relation between communication size and information source shown in Lemmas 5 and 6.

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 8 *For any protocol $\mathcal{A}$ for p_{2k-1}, $s_{\mathcal{A}}^{k-1}(i,a,b) = 2^{\Omega(n/(k\log k))}$ for all a,b.*

Proof: By Lemmas 5 and 6 we have $s_{\mathcal{A}}^{k-1}(i,a,b) = \Theta(\mathrm{CS}^{2k-3}(p_{2k-1}))$. By the following Lemma 18, $\mathrm{CS}^{2k-3}(p_{2k-1}) \geq 2^{\Omega(\mathrm{CC}^{2k-3}(p_{2k-1})/k)}$. 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 [15]. ∎

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

Lemma 18 *For any binary function f, $\log \mathrm{CS}^j(f) \geq \Omega(\mathrm{CC}^j(f)/j)$.*

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, $\log \mathrm{CS}^j(f) \geq h/j - 1 + \log j$—the claim is proved. ∎

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. $g_p(x_1,\ldots,x_n) = 1$ iff $\sum_{i=1}^{n} x_i \equiv 0 \pmod p$), we can prove that the information loss to any player does not depend on the ordering in which a 1-phase protocol computes any of these functions, if each block has size at least n_0 and p, respectively.

Proposition 2 *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 1-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 3 *There exists a symmetric function f and a communication graph G with one bridge player P_q, such that there are two ordering σ and σ' with $s_G(i,1,1,\sigma) \in \Omega(\log s_G(i,1,1,\sigma'))$.*

Proof: For $\ell \in \mathbb{N}$, let $n = 2^\ell - 1$. For $x_1, \ldots, x_n \in \{0,1\}$, let

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

For simplicity, we call the binary string of length that represents y again y. We split y into two parts: $z_{\text{addr}} = y_{d-1} \ldots y_0$ 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 $z_{\text{addr}} = \sum_{i=0}^{d-1} y_i 2^i$. Then

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

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 $n \in \Theta\left(2^{2^d}\right)$) 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 1-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 1-phase protocols, the answer is negative: Consider the function f given by

$$f(x^0, x^1, y^0, y^1, z_1, z_2, z_3) = \begin{cases} 1 & \text{if } z_1 \oplus z_2 \oplus z_3 = 0 \text{ and } x^0 = y^0 \text{ or} \\ & z_1 \oplus z_2 \oplus z_3 = 1 \text{ and } x^1 = y^1 \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Here x^0, x^1, y^0, and 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: x^0 and x^1 are distributed within one block (B_X), y^0 and 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 1-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 x^0, x^1 and $x^{0\prime}, x^{1\prime}$ that are indistinguishable for the middle and the right component. Assume w.l.o.g. that $x^0 \neq x^{0\prime}$. Choose z_2 and z_3 such that $\ell = 0$. Then either for $y^0 = x^0$ or for $y^0 = x^{0\prime}$ 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.

An open problem is 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.

Another future research direction might be 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 *Proc. of the 14th Int. Symp. on Fundamentals of Computation Theory (FCT)*, volume 2751 of *Lecture Notes in Comput. Sci.*, pages 158–170. Springer, 2003.

[2] Reuven Bar-Yehuda, Benny Chor, Eyal Kushilevitz, and Alon Orlitsky. Privacy, additional information, and communication. *IEEE Trans. Inform. 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, 1988.

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

[5] Markus Bl¨aser, Andreas Jakoby, Maciej Liśkiewicz, and Bodo Siebert. Private computation — k-connected versus 1-connected networks. In *Proc. of the 22nd Ann. Int. Cryptology Conf. (CRYPTO)*, volume 2442 of *Lecture Notes in Comput. Sci.*, pages 194–209. Springer, 2002. A full version can be found as Technical Report 03-009, *Electron. Colloq. on Comput. Complex. (ECCC)*.

[6] 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, 1988.

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

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

[9] Matthew Franklin and Moti Yung. Secure hypergraphs: Privacy from partial broadcast. In *Proc. of the 27th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 36–44, 1995.

[10] Eyal Kushilevitz. Privacy and communication complexity. *SIAM J. Discrete Math.*, 5(2):273–284, 1992.

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

[12] Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. Characterizing linear size circuits in terms of privacy. *J. Comput. System Sci.*, 58(1):129–136, 1999.

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

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

[15] Noam Nisan and Avi Wigderson. Rounds in communication complexity revisited. *SIAM J. Comput.*, 22(1):211–219, 1993.

[16] 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, 1984.

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

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

[19] 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, 1982.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'