Privacy in Non-Private Environments*

Markus Bläser[1**], Andreas Jakoby[2], Maciej Liśkiewicz[2***], and Bodo Manthey[2†]

[1] Institut für Theoretische Informatik, ETH Zürich, Switzerland
mblaeser@inf.ethz.ch
[2] Institut für Theoretische Informatik, Universität zu Lübeck, Germany
jakoby/liskiewi/manthey@tcs.uni-luebeck.de

Abstract. We study private computations in information-theoretical settings on networks that are not 2-connected. Non-2-connected networks are "non-private" in the sense that most functions cannot privately be computed on them. We relax the notion of privacy by introducing lossy private protocols, which generalize private protocols. We measure the information each player gains during the computation. Good protocols should minimize the amount of information they lose to the players. Throughout this work, privacy always means 1-privacy, i.e. players are not allowed to share their knowledge. Furthermore, the players are honest but curious, thus they never deviate from the given protocol.
By use of randomness by the protocol the communication strings a certain player can observe on a particular input determine a probability distribution. We define the loss of a protocol to a player as the logarithm of the number of different probability distributions the player can observe. For optimal protocols, this is justified by the following result: For a particular content of any player's random tape, the distributions the player observes have pairwise fidelity zero. Thus the player can easily distinguish the distributions.
The simplest non-2-connected networks consists of two blocks that share one bridge node. We prove that on such networks, communication complexity and the loss of a private protocol are closely related: Up to constant factors, they are the same.
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.
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.

* The full version of this work appeared as Rev. 1 of Report 03-071, ECCC, 2003.
** Work done while at the Institut für Theoretische Informatik, Universität zu Lübeck.
*** On leave from Instytut Informatyki, Uniwersytet Wrocławski, Poland.
† Supported by DFG research grant RE 672/3.

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 other secrets except for what he is able to deduce from his own secret and the function value. This is the aim of *private computation* (also called *secure multi-party computation*). To compute the function, the players can send messages to each other using secure links.

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

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

Any Boolean function can privately (in the following we identify privately with 1-privately) be computed on any 2-connected network. Unfortunately, there are many Boolean functions, even simple ones like parity or disjunction, 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 are important when considering non-2-connected networks. For all non-bridge players we can guarantee that they do not learn anything except for what they can deduce from their own bit and the function value. Thus, the bridge players are the only players that are able to learn something more. The question is now, how much the bridge players need to learn such that the function can be computed. The simplest setting is a network of two blocks with one bridge node in common. (A block is a maximal 2-connected subnetwork.) This reminds one of communication complexity with a man in the middle: Alice (one block) and Bob (another block) want to compute a function depending on their input while preventing Eve (the bridge node) from learning anything about their input. Unfortunately, Eve listens to the only communication channel between Alice and Bob. In terms of communication complexity, this problem had been examined by Modiano and Ephremedis [13, 14] and Orlitsky and El Gamal [17] under cryptographic security. In contrast, we deal with information-theoretical security, i.e. the computational power of the players is unrestricted. Furthermore, we are not interested in minimizing communication but in minimizing the information learned by any player. It turns out that there is a close relation between communication and privacy, at least in this special case.

1.1 Previous Results

Private computation was introduced by Yao [20]. He considered the problem under cryptographic assumptions. Private Computation with information-theoretical security has been introduced by Ben-Or et al. [3] and Chaum et al. [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. 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] used directed hypergraphs for communication and described those networks on which every Boolean function can privately be computed.

While all Boolean functions can privately be computed on any undirected 2-connected network, Bläser 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. 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 [20] 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 2.2). Here and in the following, private always means 1-private. Throughout this work, we restrict ourselves to non-2-connected (in the sense of non-2-vertex-connected) networks that are still 2-edge-connected. Every block in such a network has size at least three and private computation within such a block is possible. We measure the information any particular player gains during the execution of the protocol in an information-theoretical sense. This is the *loss* of the protocol to the player. The players are assumed to be honest but curios. This means that they always follow the protocol but try to derive as much information as possible.

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

In the definition of lossy protocols, the loss of a protocol to a player is merely the logarithm of the number of different probability distributions on the communication strings a player can observe. We justify this definition in Section 3: For a protocol with minimum loss to a player P and any particular content of P's random tape, the support of any two probability distributions is disjoint. Thus, in order to gain information, P can distinguish the distributions from the actual communication he observes and does not need to sample.

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

Then we study 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 only once with each block he belongs to. However, the loss of the protocol may depend on the order in which P communicates with the blocks. In Section 5, we show that the order in which P should communicate with the blocks to minimize the loss equals the order in which d parties should be ordered on a directed line when they want to compute the function with minimum communication complexity. Particularly for symmetric functions, it is optimal to sort the components by increasing size. Then we design a 1-phase protocol (Theorem 9), which has the remarkable feature, that it achieves minimal loss at any node for symmetric functions. Hence, it simultaneously minimizes the loss for all nodes where the minimum is taken over all 1-phase protocols.

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

1.3 Comparison of Our Results with Previous Work

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

Kushilevitz [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 and that randomization does not help in this model. 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, and they characterized the class of privately computable Boolean functions on countable domains. Neither

Kushilevitz [10] nor Chor et al. [7] examined the problem how functions that cannot privately be computed can still be computed while maintaining as much privacy as possible.

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

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

2 Preliminaries

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. We often use the string operation $x_{i \leftarrow a}$ defined for any $i \in [n]$ and $a \in \{0,1\}$ by $x_1 \ldots x_{i-1}\, a\, x_{i+1} \ldots x_n$. For a function $f : \{0,1\}^n \to \{0,1\}$, an index $i \in [n]$, and $a \in \{0,1\}$, $f_{i \leftarrow \alpha} : \{0,1\}^{n-1} \to \{0,1\}$ denotes the function obtained from f by specialising the position i to the value given by a, i.e. for all $x = x_1 x_2 \ldots x_{n-1} \in \{0,1\}^{n-1}$,

$$f_{i \leftarrow a}(x) = f(x_1, \ldots, x_{i-1}, a, x_i, \ldots x_{n-1})\,.$$

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 $G|_U := (U, E|_U)$ be the graph induced by U. A subgraph $G|_U$ is called a *block*, if $G|_U$ is 2-connected and no proper supergraph $G|_U$ is 2-connected. A block of size two is called an *isthmus*. 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 *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 *symmetric*, if the function value depends only on the number of 1s in the input. See Wegener [19] for a survey on Boolean functions.

2.1 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 has a random tape. The players can send messages to other players using secure links where the link topology is 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. 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 all players.*

In the following, we use a strengthened definition of privacy: We allow only one player, say P_i, to know the result. The protocol has to be private with respect to P_i according to Definition 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 $\Pr[C_j = c|R_j, x] = \Pr[C_j = c|R_j, y]$. Thus, all other players do not learn anything. This definition does not restrict the class of functions computable by private protocols according to Definition 1. To achieve this additional restriction, P_i generates a random bit r. Then we use a private protocol for computing $r \oplus f(x)$.

2.2 Information Source

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

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

In the following, $c_1, c_2, c_3, \ldots$ denotes a fixed enumeration of all communication strings seen by any player during the execution of $\mathcal{A}$.

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 strings 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 *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 $n - 1$ [18]. 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.

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*

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

The loss of a protocol $\mathcal{A}$ is *bounded by* $\lambda \in \mathbb{N}$, if $\ell_\mathcal{A}(i, a, b) \leq \lambda$ for all i, a, and b. $\ell_G(i, a, b)$ is obtained by locally minimizing the loss to each player P_i over all protocols. It is a priori not clear whether there is one protocol with $\ell_G(i, a, b) = \ell_\mathcal{A}(i, a, b)$ for all i, a, b. We show that this is the case for symmetric functions and 1-phase protocols (as defined in Section 2.3).

We also use the size of the information source, defined by $s_\mathcal{A}(i, a, b, R_i) = |\mathcal{S}_\mathcal{A}(i, a, b, R_i)|$ and $s_\mathcal{A}(i, a, b) = \max_{R_i} s_\mathcal{A}(i, a, b, R_i)$ 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 omit the subscript $\mathcal{A}$. Let f be an n-ary Boolean function. For a network $G = (V, E)$ with $|V| = n$, we define $s_G(i, a, b) := \min_\mathcal{A} s_\mathcal{A}(i, a, b)$. 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). Players P_i that are bridge nodes are in general able to infer some information about the input.

2.3 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. During such an alternation, information can flow from one block to another. We partition a communication sequence $c = d_1 d_2 \ldots$ of P_q into a minimal number of disjoint subsequences $(d_1,\ldots,d_{i_1}),(d_{i_1+1},\ldots,d_{i_2}),\ldots$ 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 and define $\mathrm{block}_j(c) := (d_{i_{j-1}+1},\ldots,d_{i_j})$ 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 already has communicated with previously in the phase.

A protocol $\mathcal{A}$ is a *k-phase protocol for a bridge node* P_q if for every input string and contents of all random tapes, 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 $\ell_G^k(i,a,b)$ and $s_G^k(i,a,b)$ 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 $s_G(i,a,b,\sigma) := \min\{s_{\mathcal{A}}(i,a,b) \mid \mathcal{A} \text{ is } \sigma\text{-ordered for } f \text{ on } G\}$.

2.4 Communication Protocols

For comparing the communication complexity with the loss of private protocols, we need the following definitions. 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 $\mathcal{B}$.

$\mathrm{CC}(\mathcal{B})$ 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, we define $\mathrm{CC}(f) = \min_{\mathcal{B}\text{ for }f}\mathrm{CC}(\mathcal{B})$ 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. The complexity is at most doubled.

We also consider multi-party communication with a referee. Let $f : \{0,1\}^{m_1} \times \{0,1\}^{m_2} \times \ldots \times \{0,1\}^{m_k} \to \{0,1\}$ 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 $\vec{x}_1, \ldots, \vec{x}_k$, the referee cooperates with $A_1, \ldots, A_k$ as follows:

- Initially, $\vec{x}_1, \ldots, \vec{x}_k$ are distributed among $A_1, \ldots, A_k$, i.e. A_i knows $\vec{x}_i$. The referee R does not have any knowledge about the inputs.
- In successive rounds, R 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. Denote by $\mathrm{c}^R_{\mathcal{B}}(\vec{x}_1, \ldots, \vec{x}_k)$ the whole communication string of R after protocol $\mathcal{B}$ has been finished. More precisely, $\mathrm{c}^R_{\mathcal{B}}(\vec{x}_1, \ldots, \vec{x}_k)$ is a concatenation of messages sent (to or from R) on input $\vec{x}_1, \ldots, \vec{x}_k$ with additional stamps describing the sender and the receiver of each message. For $b \in \{0,1\}$ let

$$
\begin{aligned}
\mathcal{CS}^R_{\mathcal{B}}(b) &= \{\mathrm{c}^R_{\mathcal{B}}(\vec{x}_1, \ldots, \vec{x}_k) \mid \forall i \in [k] : \vec{x}_i \in \{0,1\}^{m_i} \text{ and } f(\vec{x}_1, \ldots, \vec{x}_k) = b\} \,, \\
\mathcal{CS}^R(\mathcal{B}) &= \mathcal{CS}^R_{\mathcal{B}}(0) \cup \mathcal{CS}^R_{\mathcal{B}}(1) \,, \\
\mathrm{CS}^R_{\mathcal{B}}(b) &= |\mathcal{CS}^R_{\mathcal{B}}(b)| \,, \qquad \mathrm{CS}^R(\mathcal{B}) = |\mathcal{CS}^R(\mathcal{B})| \,, \\
\mathrm{CS}^R(f,b) &= \min\nolimits_{\mathcal{B} \text{ for } f} \mathrm{CS}^R_{\mathcal{B}}(b) \,, \text{ and } \mathrm{CS}^R(f) = \min\nolimits_{\mathcal{B} \text{ for } f} \mathrm{CS}^R(\mathcal{B}) \,.
\end{aligned}
$$

3 The Suitability of the Model

We observe that it suffices to consider bridge players when talking about the loss of a protocol. More precisely, any protocol can be modified such that the loss to all internal players is zero, while the loss to any bridge player does not increase.

All Boolean functions can be computed by using only three players [3]. Thus, it is possible to compute functions privately within any block, since the networks we consider are isthmus-free. This holds even if some of the players know a subset of the input bits and the result consists of a binary string.

Finally, in optimal protocols, the probability distributions observed by any player have pairwise fidelity 0. Thus, any player can easily distinguish the different probability distributions he observes.

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 $X := \{x \in \{0,1\}^n \mid x_q = a \wedge f(x) = b\}$ and, for any communication string c, $\psi(c) := \{x \in X \mid \mu_x(c) > 0\}$, where $\mu_x(c) = \Pr[C_q = c | R_q, x]$. For every communication string c that can be observed by P_q on some input $x \in X$, P_q can deduce that $x \in \psi(c)$. If $s_{\mathcal{A}}(q,a,b) = s_G(q,a,b) = 1$, then we have either $\psi(c) = X$ or $\psi(c) = \emptyset$. Thus P_q does not learn anything in this case.

Theorem 4. *If $s_G(q,a,b) > 1$, then for any protocol $\mathcal{A}$ and every communication string c that can be observed by P_q on $x \in X$, $\psi(c)$ is a non-trivial subset of X, i.e. $\emptyset \neq \psi(c) \subsetneq X$, and there exist at least $s_G(q,a,b)$ different such sets.*

Hence, from seeing c on $x \in X$, P_q always gains some information and there are at least $s_G(q,a,b)$ different pieces of information that can be extracted by P_q on inputs from X. To prove this, we show that for each distribution we can find one representative string that can be used in the communication protocol.

The next result says that $s_G(q,a,b)$ is a tight lower bound on the number of pieces of information: the lower bound is achieved when performing an optimal protocol on G. Let μ and μ' be two probability distributions over the same set of elementary events. The *fidelity* is a measure for the similarity of μ and μ' (see e.g. Nielsen and Chuang [15]) and is defined by $F(\mu,\mu') = \sum_c \sqrt{\mu(c) \cdot \mu'(c)}$.

Theorem 5. *If $\mathcal{A}$ is an optimal protocol for P_q on a and b, i.e. $s_{\mathcal{A}}(q,a,b) = s_G(q,a,b)$, then for all random strings R_q and all probability distributions $\mu \neq \mu'$ in $\mathcal{S}_{\mathcal{A}}(q,a,b,R_q)$ we have $F(\mu,\mu') = 0$.*

4 Communication Complexity and Private Computation

In this section, we investigate the relations between deterministic communication complexity and the minimum size of an information source in a network with one bridge node. To distinguish protocols in terms of communication complexity and protocols in terms of private computation, we will call the former communication protocols. From the relation between C^P and CC (see e.g. Kushilevitz and Nisan [11, Sec. 2.2]), we get $\frac{1}{2}\log(\mathrm{CS}(f)) \leq \mathrm{CC}(f) \leq 3 \cdot \log(\mathrm{CS}(f))$. 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. Alice and Bob correspond to the first and second block, respectively, while both know the bridge player's bit.

Theorem 6. *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 $6\lambda + O(1)$.*

We can generalize the results obtained for the relation two-party communication and private computation to obtain similar results for the relation of multi-party communication with a referee and private computation as follows: For $a, b \in \{0,1\}$ we have $s_G(q,a,b) = \mathrm{CS}^R(f_{q\leftarrow a}, b)$.

5 1-Phase Protocols

We start our study of 1-phase protocols with considering networks that consist of one bridge player who is incident with d blocks. For the case that the order in which the bridge player communicates with the blocks is fixed for all inputs, we show a relationship between the size of the information source of 1-phase protocols and communication size of multi-party 1-way protocols. Furthermore, 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 networks that is optimal for every symmetric function.

5.1 Orderings

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: d parties $A_1, \ldots, A_d$ are connected by a directed chain, i.e. A_i can only send messages to A_{i+1}. For a communication protocol $\mathcal{B}$ on G and $i \in [d]$ let $S_i^{\hookrightarrow}(\mathcal{B})$ 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 $S_i^{\hookrightarrow}(\mathcal{B})$ in the following way: Every party A_i first sends the messages it has received from A_{i-1} to A_{i+1} followed by the messages it has to send according to $\mathcal{B}$. In the following we restrict ourselves to communication protocols of this form.

If the network G consists of d blocks B_i with $i \in [d]$ and one bridge player P_q we consider a chain of d parties $A_1, \ldots, A_d$. For a σ-ordered 1-phase protocol $\mathcal{A}$, we assume that the enumeration of the blocks reflects the ordering σ. We have to determine the input bits of the parties in the chain according to the input bits of the players in the protocol. In the following we will assume that A_i knows the input bits of the players in B_i. Thus, each party A_i has to know the input bit x_q of the bridge player P_q. Therefore, we will investigate the restricted function $f_{q\leftarrow a}$ whenever we analyse the communication size of a communication protocol.

For a σ-ordered protocol $\mathcal{A}$ define $\mathcal{S}_{\mathcal{A}}^{[i]}(q,a,b,R_q) = \{\hat{\mu}_x \mid x_q = a \wedge f(x) = b\}$, where $\hat{\mu}_x(\hat{c}_k)$ denotes the sum of the probabilities $\Pr[C_q = c \mid R_q, x]$ over all c with $\widehat{c}_k = \mathrm{block}_1(c) \ldots \mathrm{block}_i(c)$ 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.

Lemma 7. *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 $f_{q\leftarrow a}$ such that for all $i \in [d-1]$, we have*

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

Let us now focus on the structure of the possible communication sequences of an optimal communication protocol on a chain. In such a protocol, the message sent from A_i to A_{i+1} has to specify the subfunction obtained by specifying the input bits of the first i blocks according to their input. Hence, the number of possible communication sequences on the network $A_1, \ldots, A_d$ is at least the number of different sequences of subfunctions that can be obtained in this way.

The knowledge about these sequences must also be provided to the bridge player. Hence, for every fixed R_q and $b \in \{0,1\}$ the number of distributions in $\mathcal{S}_{\mathcal{A}}^{[d-1]}(q, a, b, R_q)$ 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 8. *For $a \in \{0,1\}$, let $\mathcal{B}$ be a communication protocol for computing $f_{q \leftarrow a}$ on a chain network. Then there exists a σ-ordered 1-phase protocol $\mathcal{A}$ for f such that for all $i \in [d-1]$ and every content R_q of P_q's random tape, we have*

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

Furthermore, for any $b \in \{0,1\}$: If we restrict the inputs to $x \in \{0,1\}^{n-1}$ with $f_{q \leftarrow a}(x) = b$, the number of possible communication sequences on the subnetwork $A_1, \ldots, A_{i+1}$ is $|\mathcal{S}_{\mathcal{A}}^{[i]}(q, a, b, R_q)|$.

We can show that there exist functions, for which no ordered 1-phase protocol minimizes the size of the bridge players' information source. Thus, we generalize the class of ordering that we consider to achieve such a property.

We call a protocol $\mathcal{A}$ *quasi-ordered* if for every $a, b \in \{0,1\}$, 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. Note that this ordering is not necessarily the same for all inputs. However, given any input, the ordering is fixed.

We can prove that among all 1-phase protocols for a given function, there always exists a quasi-ordered protocol that minimizes the loss to P_q.

5.2 Orderings for Symmetric Functions

For symmetric Boolean functions, 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, $S_i^{\mapsto}(\mathcal{B})$ is minimal, 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 also holds, if we count the number of communication sequences in a chain network for inputs x with $f(x) = 1$ and the number of communication sequences in a chain network for inputs x with $f(x) = 0$. Together with Lemma 8, we obtain the following: 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 $|\mathcal{S}^{[i]}_{\mathcal{A}}(q,a,b,R_q)| \leq |\mathcal{S}^{[i]}_{\mathcal{A}'}(q,a,b,R_q)|$.

This result can 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.

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

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

For computing a symmetric function f we use the following protocol. Let σ be an increasing communication order. Then for an input x every bridge player q_i computes a sequence of strings $\vec{y}_1, \ldots, \vec{y}_{k_i-1}$ as follows: Let $X_j = \bigcup_{e \in [j]} V[v_{\sigma_{q_i}(e)}]$ and $\ell_j = |X_j|$. Then $\vec{y}_j \in \{0,1\}^{\ell_j}$ such that for all $j \leq k_i - 1$ the function obtained from f by specialising the positions in X_j to $\vec{y}_j$ is equal to the function obtained from f by specialising the positions to x_{X_j}, where x_I for $I \subseteq [n]$ denotes the input bits with indices in I. Finally, a node of the block that corresponds to the root of T_G computes the result $f(x)$. This can be implemented such that no player gains any additional information except for $\vec{y}_1, \ldots, \vec{y}_{k_i-1}$ learned by the bridge nodes q_i.

Theorem 9. *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 $a, b \in \{0,1\}$, we have $s_{\mathcal{A}}(i,a,b) \leq s_{\mathcal{A}'}(i,a,b)$.*

Thus, 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. 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 10. *For any protocol $\mathcal{A}$ for computing p_{2k-1}, we have $s_{\mathcal{A}}^{k-1}(i,a,b) = 2^{\Omega(n/(k \log k))}$ for all a, b. For p_{2k-1}, $s_G^k(i,a,b) = 2^{O(k \log n)}$ for all a, b.*

The lower bound follows from work by Nisan and Wigderson [16].

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 measure for the information that can be inferred by any player and discussed some general properties of protocols with respect to this measure. A natural question is finding optimal protocols for some concrete functions.

For threshold ($f_{n_0}(x_1, \dots, x_n) = 1$ iff $\sum_{i=1}^n x_i \geq n_0$) and counting modulo p ($g_p(x_1, \dots, x_n) = 1$ iff $\sum_{i=1}^n x_i \equiv 0 \pmod{p}$), 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. If we have blocks of less than $p-1$ nodes, there can be a slight difference in the size of P_q's information source depending on the order.

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 holds even in case of symmetric functions.

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: There are functions, for which no protocol exists that minimizes the loss to all players simultaneously.

It is open whether there exist functions and networks that do not allow to minimize the loss to each bridge player simultaneously. For such functions, we have to generalize our measure. Two simple examples one might want to examine is the sum of the loss to each player and the maximum loss to any player.

References

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

2. Reuven Bar-Yehuda, Benny Chor, Eyal Kushilevitz, and Alon Orlitsky. Privacy, additional information, and communication. *IEEE Transactions on Information Theory*, 39(6):1930–1943, 1993.
3. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of the 20th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 1–10. ACM Press, 1988.
4. Claude Berge. *Graphs*. North-Holland, 1991.
5. Markus Bläser, Andreas Jakoby, Maciej Liśkiewicz, and Bodo Siebert. Private computation — k-connected versus 1-connected networks. In M. Yung, editor, *Proc. of the 22nd Ann. Int. Cryptology Conf. (CRYPTO)*, volume 2442 of *Lecture Notes in Computer Science*, pages 194–209. IACR, Springer, 2002.
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. ACM Press, 1988.
7. Benny Chor, Mihály Geréb-Graus, and Eyal Kushilevitz. Private computations over the integers. *SIAM Journal on Computing*, 24(2):376–386, 1995.
8. Benny Chor and Eyal Kushilevitz. A zero-one law for boolean privacy. *SIAM Journal on Discrete Mathematics*, 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. ACM Press, 1995.
10. Eyal Kushilevitz. Privacy and communication complexity. *SIAM Journal on Discrete Mathematics*, 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. *Journal of Computer and System Sciences*, 58(1):129–136, 1999.
13. Eytan H. Modiano and Anthony Ephremides. Communication complexity of secure distributed computation in the presence of noise. *IEEE Transactions on Information Theory*, 38(4):1193–1202, 1992.
14. Eytan H. Modiano and Anthony Ephremides. Communication protocols for secure distributed computation of binary functions. *Information and Computation*, 158(2):71–97, 2000.
15. Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*, chapter 9. Cambridge University Press, 2000.
16. Noam Nisan and Avi Wigderson. Rounds in communication complexity revisited. *SIAM Journal on Computing*, 22(1):211–219, 1993.
17. Alon Orlitsky and Abbas El Gamal. Communication with secrecy constraints. In *Proc. of the 16th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 217–224. ACM Press, 1984.
18. Claude Elwood Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3, 4):379–423 & 623–656, 1948.
19. Ingo Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner, 1987.
20. Andrew Chi-Chih Yao. Protocols for secure computations. In *Proc. of the 23rd Ann. IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 160–164. IEEE Computer Society, 1982.