Private Computation – k-connected versus 1-connected Networks

Markus Bläser, Andreas Jakoby, Maciej Liśkiewicz*, and Bodo Siebert**

Institut für Theoretische Informatik
Universität zu Lübeck
Wallstraße 40, 23560 Lübeck, Germany
blaeser/jakoby/liskiewi/siebert@tcs.mu-luebeck.de

Abstract. We study the role of connectivity of communication networks in private computations under information theoretic settings. It will be shown that some functions can be computed by private protocols even if the underlying network is 1-connected but not 2-connected. Then we give a complete characterisation of non-degenerate functions that can be computed on non-2-connected networks.

Furthermore, a general technique for simulating private protocols on arbitrary networks will be presented. Using this technique every private protocol can be simulated on arbitrary k-connected networks using only a small number of additional random bits.

Finally, we give matching lower and upper bounds for the number of random bits needed to compute the parity function on k-connected networks.

1 Introduction

Consider a set of players, each knowing an individual secret. The goal is to compute a function depending on these secrets such that after the computation none of the players knows anything about the secrets that cannot be derived from the result of the function. 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 for no. But the ballot should be proprietary, i.e. after the vote nobody should know anything about the opinion of the other committee members or about the exact number of yes- or no-votes. The only thing known after the computation is whether the majority votes for yes or for no. To exchange data we allow that the committee member can talk to each other in private.

More formally, the players exchange messages to compute the value of a function. But no player should learn anything about the concrete input values of the other players. Depending on the computational power of the players we distinguish between cryptographically secure privacy and privacy in an information theoretic sense. In the first case we assume that no player is able to recompute

* On leave from Instytut Informatyki, Uniwersytet Wrocławski, Poland.
** Supported by DFG research grant Re 672/3.

any information about the input within polynomial time (see e.g. [5, 15, 21, 22]). In the second case we do not restrict the computational power of the players (see e.g. [3, 6]). Hence, this notion of privacy is much stronger than in the cryptographic setting. In this paper we use the information theoretic approach.

Private computation has been the subject of a considerable amount of research. Traditionally, one investigates the number of rounds and random bits as complexity measures for private protocols. Chor and Kushilevitz [10] have studied the number of rounds necessary to compute the sum modulo an integer. This function has also been investigated by Blundo et al. [4] and Chor et al. [8]. The number of random bits needed to compute the parity function, i.e. the sum modulo 2, has been examined in [17, 19]. Gál and Rosén [14] have shown that the parity function cannot be computed by any private protocol in $o(\log n/\log d)$ rounds using d random bits. They have also given an almost tight randomness-round-tradeoff for private computations of arbitrary Boolean functions depending on their sensitivity. Bounds on the maximum number of rounds needed in the worst-case to compute a function by a private protocol are given by Bar-Ilan and Beaver [2] and by Kushilevitz [16].

The number of random bits necessary to compute a Boolean function by a private protocol is closely related to its circuit size. Kushilevitz et al. [18] have shown that every function that can be computed with linear circuit size can also be computed by a private protocol with only a constant number of random bits. Using this result one can show that the majority function can be computed by a private protocol using a constant number of random bits and simultaneously a linear number of exchanged bits between players (for the circuit complexity of majority see e.g. [20]).

So far we have assumed that players do not attempt to cheat. Depending on the way players attempt to acquire information about the input of the other players we distinguish between dishonest players and players that can work in teams (e.g. [3, 5, 6, 12]). The goal in this approach is to investigate the number of dishonest players or players in a team that are necessary to learn anything about the input of the remaining players. Chor and Kushilevitz [9] have shown that Boolean functions with one bit output can be computed with teams either of size at most $\left\lfloor \frac{n-1}{2} \right\rfloor$ or of any size up to n. For extensions, see [7, 8].

All papers mentioned above do not restrict the communication capabilities of the players. In other words, they use complete graphs as underlying communication networks. However, most realistic parallel architectures have a restricted connectivity and nodes of bounded degree. Franklin and Yung [13] have been the first who studied the role of connectivity in private computations. They have presented a protocol for k-connected bus networks. This protocol can simulate one communication step of a private protocol that was originally written for a complete graph. To simulate such a communication step, their protocol uses $O(n)$ additional random bits.

In this paper we investigate the number of random bits needed to compute functions by private protocols on k-connected networks. The consideration of k-connected networks instead of complete networks seems to be quite realistic for practical applications. We present a new simulation technique that allows us to reduce the number of random bits by taking the connectivity of the network into account. Furthermore, we show that the parity function can be computed by a private protocol on every k-connected network with $\lceil \frac{n-2}{k-1} \rceil - 1$ random bits. On the other hand, we will present k-connected networks where $\lceil \frac{n-2}{k-1} \rceil - 1$ random bits are necessary.

Furthermore, we investigate networks that are not 2-connected and present non-trivial functions that can still be computed by private protocols on such networks. We introduce the notion of a dominated function and prove that a function can be computed by a private protocol on non-2-connected networks if and only if the function is dominated. This result can be generalised to the case where the players can work in teams. Such a computation is not possible if some of the players are dishonest.

The paper is organised as follows. In the next section we define some notations and give a formal definition of private computation. In Section 3 we present a new technique to simulate private protocols on k-connected networks. Furthermore, we present a simple non-trivial function that can be computed by a private protocol on a non-2-connected network. In Section 4 we investigate the number of random bits needed to compute the parity function on arbitrary k-connected networks. Finally, in Section 5 we investigate non-2-connected networks and give a structural property that precisely determines whether a function can be computed on a non-2-connected network.

2 Preliminaries

2.1 Notations

For $i, j \in \mathbb{N}$ define $[i] := \{1, \ldots, i\}$ and $[i..j] := \{i, \ldots, j\}$. Throughout this paper, we will often use the following string operations. Let $x = x[1]x[2] \ldots x[n] \in \{0,1\}^n$ be a string of length n. Then, for $I \subseteq [n]$ and $\alpha \in \{0,1\}^{|I|}$, $x\lceil_{I \leftarrow \alpha}$ is defined as follows:

$$z \;=\; x\lceil_{I\leftarrow\alpha} \;:\Longleftrightarrow\; \forall i \in [n] \;:\; z[i] \;=\; \begin{cases} x[i] & \text{if } i \notin I \\ \alpha[j] & \text{if } i \in I \text{ and } i \text{ is the } j\text{th smallest} \\ & \text{element in } I\,. \end{cases}$$

For sets $I_1, I_2, \ldots, I_k \subseteq [n]$ and strings $\alpha_1, \alpha_2, \ldots, \alpha_k \in \{0,1\}^*$ with $|\alpha_i| = |I_i|$ we define

$$x\lceil_{I_1,I_2,\ldots,I_k \leftarrow \alpha_1,\alpha_2,\ldots,\alpha_k} \;:=\; (x\lceil_{I_1\leftarrow\alpha_1})\lceil_{I_2,\ldots,I_k\leftarrow\alpha_2,\ldots,\alpha_k}\,.$$

Let $\overline{x}$ denote the bitwise negation of x, i.e. $\forall i \in [n] \;:\; \overline{x}[i] = \overline{x[i]}$. For a function $f : \{0,1\}^n \to \{0,1\}$, a set of indices $I \subseteq [n]$, and a string $\alpha \in \{0,1\}^{|I|}$ define the partially restricted function $f\lceil_{I\leftarrow\alpha} : \{0,1\}^{n-|I|} \to \{0,1\}$ as the function obtained from f by assigning the values given by α to the positions in I, i.e.

$$\forall x \in \{0,1\}^{n-|I|} \;:\; f\lceil_{I\leftarrow\alpha}(x) \;:=\; f(0^n\lceil_{I,J\leftarrow\alpha,x})\,,$$

where $J = [n] \setminus I$. Finally, for a string $x \in \{0,1\}^n$ and a set $I \subseteq [n]$ define $x[I] \in \{0,1\}^{|I|}$ as follows:

$$\forall j \leq |I| \ : \ \ (x[I])[j] = x[i] \ :\Longleftrightarrow \ i \text{ is the } j\text{th smallest element in } I\ .$$

A graph is called k-connected if, after deleting an arbitrary subset of at most $k-1$ nodes, the resulting node-induced graph remains connected.

2.2 Private Computation

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. 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 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 theoretic sense except for the information it can deduce from its 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_i 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 every pair of input vectors x and y with $f(x) = f(y)$ and $x[i] = y[i]$, for every c_i, and for every random string R_i* provided *to P_i,*

$$\Pr[C_i = c_i \mid R_i, x] \ = \ \Pr[C_i = c_i \mid R_i, y]\ ,$$

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 *synchronous* if the communication takes place in rounds and each message consists of a single bit. We call a synchronous protocol *oblivious* if the number of bits that player P_i sends to P_j in round t depends only on i, j, and t but not on the input and the random strings. Furthermore, we do not bound the computational resources of the players. We assume that all of them are honest, i.e. the computation and the interactions between players are determined only by the protocol.

For a synchronous oblivious protocol $\mathcal{A}$ let $L(P_i, P_j, \mathcal{A})$ be the number of bits sent from P_i to P_j in $\mathcal{A}$ and

$$L(\mathcal{A}) \ := \ \textstyle\sum_{i\in[n]} \sum_{j\in[n]} L(P_i, P_j, \mathcal{A})\ .$$

We distribute the given input bits among the nodes of the graph. For convenience, we call the node that gets the bit $x[i]$ player P_i. The players P_i and P_j can communicate directly if and only if they are connected by an edge in the graph.

3 Private Computation on k-connected Networks

Most known private protocols are written for specific networks. A simulation of such a private protocol on a different network can be done in such way that each player of the new network simulates a player of the original network step-by-step. Hence, we have to find a way to realize the communication steps between all players that are not directly connected. Franklin and Yung [13] have presented a strategy to simulate a transmission of one single bit on a hypergraph by using $O(n)$ additional random bits. Thus, the whole simulation presented there requires $O(m + nL(\mathcal{A}))$ random bits where m is the number of random bits used by the original protocol. If we consider 2-connected graphs we can simulate each communication step between two players P_i and P_j by one additional random bit r as follows: Assume P_i has to send bit b to P_j. Then P_i chooses two disjoint paths to P_j and sends r to P_j along the one path and $r \oplus b$ along the other path. In this way, $O(m + L(\mathcal{A}))$ random bits are sufficient.

To reduce the number of random bits even more we consider the following problem:

Definition 2 (Max-Neighbour-Embedding). *Let $G = (V, E)$ be a graph with edge weights $\sigma : E \to \mathbb{N}$ and $G' = (V', E')$ a graph with $|V| = |V'|$. Let $\pi : V \to V'$ be a bijective mapping. Then the performance of π is defined as*

$$\rho(\pi) \ := \sum_{\substack{\{u,v\} \in E \ \ and \\ \{\pi(u),\pi(v)\} \in E'}} \sigma(\{u,v\}) \, .$$

The aim is to find a bijection $\pi : V \to V'$ that maximizes $\rho(\pi)$ over all bijections.

By a reduction from the 3-Dimensional-Matching-Problem, it can be shown that the decision problem corresponding to finding an optimal bijection is $\mathcal{NP}$-hard. The Max-Neighbour-Embedding-problem is $\mathcal{NP}$-hard even if both graphs have maximum degree 4.

In the following lemma we estimate the performance for the case that G' is k-connected.

Lemma 1. *Let $G = (V, E)$ be an undirected graph with n nodes and edge weights σ. Let $G' = (V', E')$ be a k-connected graph with n nodes. Then we have*

$$\max_{\substack{\pi : V \to V' \\ \pi \ is \ bijective}} \rho(\pi) \quad \geq \quad \frac{k}{n-1} \sum_{e \in E} \sigma(e) \, .$$

Proof. By the definition above, there is no difference between edges with weight 0 and nonexistent edges. Therefore, we treat nonexistent edges like edges with weight 0 and restrict ourselves to the case that G is a complete graph.

The graph G' is k-connected. Thus, every node in V' has degree at least k.

Let Π be a random bijection from V to V'. Since every node in V' has degree at least k, the probability that two arbitrary nodes u and v are neighbours under

Π, i.e. $\{\Pi(u), \Pi(v)\} \in E'$, is at least $\frac{k}{n-1}$. Thus, the edge $e = \{u, v\} \in E$ yields weight $\sigma(e)$ with probability at least $\frac{k}{n-1}$ and its expected weight is at least $\frac{k}{n-1} \cdot \sigma(e)$. Hence, the expected performance $\rho(\Pi)$ fulfils

$$\mathbb{E}(\rho(\Pi)) \geq \sum_{e \in E} \tfrac{k}{n-1} \cdot \sigma(e) = \tfrac{k}{n-1} \cdot \sum_{e \in E} \sigma(e) .$$

Therefore, there exists a bijection with performance at least $\frac{k}{n-1} \cdot \sum_{e \in E} \sigma(e)$. $\square$

A bijection that fulfils the requirements of the above lemma can be computed in polynomial time using the method of conditional expectation (see e.g. Alon et al. [1]).

Theorem 1. *Every oblivious private protocol $\mathcal{A}$ using m random bits can be simulated on every k-connected graph by using $m + (1 - \frac{k}{n-1}) \cdot \min\{L(\mathcal{A}),\ n^2 + \frac{L(\mathcal{A})}{k-1}\}$ random bits.*

Proof. Let $G = (V, E)$ be the network used in protocol $\mathcal{A}$ and $G' = (V', E')$ be the k-connected network for protocol $\mathcal{A}'$. To simulate $\mathcal{A}$ we first choose a bijection between the players in G and the players in G'. For every edge $\{P_i, P_j\} \in E$ define $\sigma(\{P_i, P_j\}) := L(P_i, P_j, \mathcal{A}) + L(P_j, P_i, \mathcal{A})$. In Lemma 1 we have seen that there exists a bijection $\pi : V \to V'$ with performance $\rho(\pi) \geq \frac{k}{n-1} L(\mathcal{A})$. Using this bijection, at least $\frac{k}{n-1} L(\mathcal{A})$ bits of the total communication in $\mathcal{A}$ are sent between players that are also neighbours in G'. Thus, this part of the communication can be simulated directly and without additional random bits.

For the remaining $(1 - \frac{k}{n-1}) L(\mathcal{A})$ bits we proceed as follows: Let P_i and P_j be two players that are not directly connected in G'. Then P_i partitions the bits it will send to P_j into blocks $B_1, \ldots, B_{\lceil L(P_i, P_j, \mathcal{A})/(k-1) \rceil}$ of size at most $k - 1$. Furthermore, P_i chooses k node-disjoint paths from P_i to P_j. P_i uses a separate random bit $r[\ell]$ for each block B_ℓ. It sends $r[\ell]$ along the first path and $b \oplus r[\ell]$ for each $b \in B_\ell$ along the remaining paths, each bit on a separate path. $\square$

We have seen that every function that can be computed by a private protocol on some network can also be computed by a private protocol on an arbitrary 2-connected network. On the other hand, there exist functions that cannot be computed by a private protocol, if the underlying network is not 2-connected.

Proposition 1. *The parity function over $n > 2$ bits cannot be computed by a private protocol on any network that is not 2-connected.*

The above theorem can be generalised to a large class of non-degenerate functions. This will be done in Section 5. There we give a characterisation for the class of non-degenerate functions that can be computed by private protocols on networks that are not 2-connected.

Definition 3. *A function $f : \{0,1\}^n \to \{0,1\}$ is called **non-degenerate** if for every $i \in [n]$ we have $f\lceil_{\{i\} \leftarrow 0} \neq f\lceil_{\{i\} \leftarrow 1}$.*

In other words, a non-degenerate function depends on all of its input bits. It turns out that there are functions that can be computed by a private protocol, even if the underlying network is not 2-connected.

Proposition 2. *There are non-degenerate functions that can be computed by a private protocol on networks that are not 2-connected.*

Consider the following non-degenerate function $f : \{0,1\}^{2n+1} \to \{0,1\}$:

$$f(z,x,y) := (z \wedge \bigwedge_{i=1}^{n} x[i]) \vee (\overline{z} \wedge \bigwedge_{i=1}^{n} y[i]) \ .$$

Here, z is a single bit and both x and y are bit strings of length n. We construct a communication network G for f as follows: Let G_x and G_y be complete networks with n players each. Then connect another player P_z with all players in both G_x and G_y. Obviously, the obtained network is not 2-connected. Using a slight modification of the protocol presented by Kushilevitz et al. [18] one can compute the subfunctions

$$f_x(z,x) \ := \ z \wedge \bigwedge_{i=1}^{n} x[i] \quad \text{and} \quad f_y(z,y) \ := \ \overline{z} \wedge \bigwedge_{i=1}^{n} y[i]$$

by a private protocol on the networks G_x with P_z and G_y with P_z, respectively. After the computation has been completed, P_z is the only player that knows the results of both subfunctions. Due to symmetry we consider the case that $z = 1$. Then $f_y(z,y) = 0$ and therefore, since f_y has been computed by a private protocol, P_z does not learn anything about y. Furthermore, P_z does not learn anything about x what he has not already known before the computation started.

4 Computing Parity on *k*-connected Networks

It is well known that the parity function of n bits can be computed on a cycle by using only one random bit. On the other hand, using our simulation discussed in Section 3 one gets an upper bound of n random bits for general 2-connected networks. The aim of this section is to close this gap. We present a private protocol for parity that uses $\lceil \frac{n-2}{k-1} \rceil - 1$ random bits and show that there are k-connected networks on which parity cannot be computed with less than $\lceil \frac{n-2}{k-1} \rceil - 1$ random bits.

Lemma 2. *There exist k-connected networks with $n \geq 2k$ players on which the parity function cannot be computed by a private protocol with less than $\lceil \frac{n-2}{k-1} \rceil - 1$ random bits.*

Proof. We consider the bipartite graph $K_{k,n-k}$ (which is obviously k-connected) and show that every private protocol that computes the parity function on this network needs at least $\lceil \frac{n-2}{k-1} \rceil - 1$ random bits. Let $\{P_1, P_2, \ldots, P_k\}$ and $\{P_{k+1}, P_{k+2}, \ldots, P_n\}$ be the two sets of nodes of $K_{k,n-k}$. Recall that for each $i = 1, \ldots, k$ and $j = k+1, \ldots, n$ we have an edge $\{P_i, P_j\}$ in $K_{k,n-k}$ and that there are no other edges. Now assume to the contrary that there exists a private protocol $\mathcal{A}$ on $K_{k,n-k}$ using less than $\lceil \frac{n-2}{k-1} \rceil - 1$ random bits.

Let $R = (R_1, \ldots, R_n)$ be the contents $R_1, \ldots, R_n$ of all random tapes. For a string $x \in \{0,1\}^n$ and $i \in [n]$, let $\mathcal{C}_i(x, R)$ be a full description of the communication received by P_i during the computation of $\mathcal{A}$ with R on the input x. Moreover, let

$$\mathcal{C}(x) = \{\langle c_1, c_2, \ldots, c_k \rangle \mid \ \exists R \, \forall i \in [k] \ \ c_i = \mathcal{C}_i(x, R)\} \, .$$

We consider computations of $\mathcal{A}$ on inputs

$$X = \{x \mid x[1] = x[2] = \ldots = x[k] = 0 \quad \text{and} \quad \textstyle\bigoplus_{i=1}^{n} x[i] = 0\} \, .$$

Then for any $x \in X$ and any communication c_1 we define

$$\mathcal{C}(c_1, x) = \{\langle c_2, \ldots, c_k \rangle \mid \langle c_1, c_2, \ldots, c_k \rangle \in \mathcal{C}(x)\} \, .$$

From the fact that $\mathcal{A}$ is private it follows:

Claim. $\exists c_1 \, \forall x \in X \quad \mathcal{C}(c_1, x) \neq \emptyset$.

Indeed, because x is a valid input for the protocol $\mathcal{A}$, there exists at least one tuple $\langle c_1, \ldots, c_k \rangle$ in $\mathcal{C}(x)$. Hence, there exists at least one c_1 with $\mathcal{C}(c_1, x) \neq \emptyset$. On the other hand, if for some $y \in X$ the set $\mathcal{C}(c_1, y)$ is empty then one can conclude that $\mathcal{A}$ is not private.

Note that $|X| = 2^{n-k-1}$ and that for every $x, y \in X$ and $i \in [k]$ we have $\bigcup_R \mathcal{C}_i(x, R) = \bigcup_R \mathcal{C}_i(y, R)$. Furthermore, using a bound on the number of different communication strings from Kushilevitz and Rosén [19] it follows that $|\bigcup_R \mathcal{C}_i(x, R)| < 2^{\frac{n-k-1}{k-1}}$. Hence, we have $|\bigcup_{x \in X} \mathcal{C}(c_1, x)| < 2^{n-k-1}$, because $\mathcal{A}$ uses less than $\frac{n-k-1}{k-1}$ random bits. Therefore, by the pigeon hole principle and the above claim we obtain

$$\exists c_1, c_2, \ldots, c_k \, \exists x, y \in X \quad x \neq y \ \text{ and } \ \langle c_2, \ldots, c_k \rangle \in \mathcal{C}(c_1, x) \cap \mathcal{C}(c_1, y) \, .$$

This means that there are two different input string $x, y \in X$ such that on both strings the players $P_1, \ldots, P_k$ receive $c_1, \ldots, c_k$, respectively. Let i, with $k + 1 \leq i \leq n$, be a position where x and y differ, i.e. $x[i] \neq y[i]$. Let $R = \langle R_1, \ldots, R_n \rangle$ and $R' = \langle R'_1, \ldots, R'_n \rangle$ be the contents of the random tapes such that $c_i = \mathcal{C}_i(x, R) = \mathcal{C}_i(y, R')$ for all $1 \leq i \leq k$.

It is easy to see that during a computation of $\mathcal{A}$ with random string $R'' = \langle R_1, \ldots, R_{i-1}, R'_i, R_{i+1}, \ldots R_n \rangle$ on the input $x\lceil_{\{i\} \leftarrow y[i]}$ the players $P_1, P_2, \ldots, P_k$ receive again communication strings $c_1, c_2, \ldots, c_k$, respectively. Hence, for this input they give the same result as for x – a contradiction. $\square$

Now we show that this bound is best possible. To obtain a private protocol that computes the parity function with $\lceil \frac{n-2}{k-1} \rceil - 1$ random bits we use the result from Egawa, Glas, and Locke [11] that every k-connected graph G with minimum degree at least d and with at least $2d$ vertices has a cycle of length at least $2d$ through any specified set of k vertices. From this result we get the following observations:

Proposition 3. *Let $G = (V, E)$ be a k-connected graph with $k \leq |V| - 1$. Then for any subset $V' \subseteq V$ with $|V'| = k + 1$ there exists a simple path containing all nodes in V'.*

Proposition 4. *Let $G = (V, E)$ be a k-connected graph with $k \leq |V|$. Then for every subset $V' \subseteq V$ with $|V'| = k$ there exists a simple cycle containing all nodes in V'.*

Proposition 5. *Let $G = (V, E)$ be a k-connected graph. Then G has a simple path of length at least* $\min\{2k + 1, |V|\}$.

To compute the parity function by a private protocol on an arbitrary k-connected network G, we proceed as follows:

1. Mark all nodes in G red. Set $z[i] := x[i]$ for each player P_i.
2. Choose a path in G of length $2k+1$. According to Proposition 5 such a path always exists. The first player P_i in the path generates a random bit r. Then P_i computes $r \oplus z[i]$ and sends the result to the next player in the path. Finally, P_i sets $z[i] := r$.
   Each internal player P_j on the path receives a bit b from its predecessor in the path, computes $b \oplus z[j]$, sends this bit to its successor, and changes its colour to black.
   The last player P_ℓ on the path receives a bit b from its predecessor and computes $z[\ell] := z[\ell] \oplus b$.
   After this step $2k - 1$ players have changed their colour.
3. We repeat the following step $\lceil \frac{n-3k+1}{k-1} \rceil$ times.
   Choose $k+1$ red nodes and a path in G containing all these nodes. According to Proposition 3 such a path always exists. We can assume that the start and the end node of the path are among the $k + 1$ given players, hence both are red. Then the first player P_i on this path generates a random bit r, computes $r \oplus z[i]$, and sends the result to the next player in the path. Finally, P_i sets $z[i] := r$.
   Each internal player of the path P_j receives a bit b from its predecessor in the path. If P_j is a black player, it sends b to its successor. If P_j is a red player, it computes $b \oplus z[j]$, sends this bit to its successor, and changes its colour to black.
   The last player P_ℓ on the path receives a bit b from its predecessor and computes $z[\ell] := z[\ell] \oplus b$.
   After this step $k-1$ players have changed their colour. Hence, after $\lceil \frac{n-3k+1}{k-1} \rceil$ iterations of this step we have at least
   $$\left\lceil \tfrac{n-3k+1}{k-1} \right\rceil \cdot (k-1) + 2k - 1 \;\geq\; n - k$$
   black players. Thus, at most k are red.
4. Choose a cycle in G containing all red nodes. According to Proposition 4 such a cycle always exists. Let P_{i_0} be a red player. Then P_{i_0} generates a random bit r, computes $r \oplus z[i_0]$, and sends the result to the next player in the cycle.

Each other player P_j on the cycle receives a bit b from its predecessor. If P_j is a black player, it sends b to its successor. If P_j is a red player, it computes $b \oplus z[j]$, sends this bit to its successor, and changes its colour to black.
If P_{i_0} receives a bit b, it computes $b \oplus r$. The result of this step is the result of the parity function.

Let us now count the number of random bits used in the protocol above. In the second and in the last step we use one random bit. In the third step we need $\lceil \frac{n-3k+1}{k-1} \rceil$ random bits. Hence, the total number of random bits is

$$\left\lceil \tfrac{n-3k+1}{k-1} \right\rceil + 2 \;=\; \left\lceil \tfrac{n-2}{k-1} \right\rceil - 1\,.$$

It remains to show that the protocol is private and computes the parity function. The correctness follows from the fact that each input bit $x[i]$ is stored by exactly one red player and each random bit is stored by either none or two players that are red after each step. By storing a bit b we mean that a player P_i knows a value $z[i]$ that depends on b. Since P_{i_0} is the last red player, it knows the result of the parity function.

Every bit a player receives in the second and third step is masked by a separate random bit. Hence, none of these players can learn anything from these bits. The same holds for all players except for player P_{i_0} in the last step. So we have to analyse the bits sent and received by P_{i_0} more carefully. In the last step $z[i_0]$ is either $x[i_0]$, a random bit, or the parity of a subset of input bits masked by a random bit. In neither case P_{i_0} can learn anything about the other input bits from the bit it receives and the value of $z[i_0]$ except for what can be derived from the result of the function and $x[i_0]$.

Theorem 2. *Let G be an arbitrary k-connected network. Then the parity function of n bits can be computed by a private protocol on G using at most $\lceil \frac{n-2}{k-1} \rceil - 1$ random bits. Moreover, there exist k-connected networks for which this bound is best possible.*

For 2-connected networks, we obtain the following corollary.

Corollary 1. *Let G be an arbitrary 2-connected network of n players ($n \geq 4$). Then the parity function over n bits can be computed by a private protocol on the network G using $n-3$ random bits. Moreover, there exists 2-connected networks for which this bound is best possible.*

5 Private Computation on Non-2-connected Networks

In Section 3 we have claimed that the parity function cannot be computed by a private protocol on a network that is not 2-connected. On the other hand, we have presented a non-degenerate function that can be computed on a non-2-connected network. In this section, we study this phenomenon to a greater extend.

Throughout this section $f : \{0,1\}^n \to \{0,1\}$ denotes the function we want to compute. Furthermore, I_1, I_2, J_1, J_2 denote both subsets of input positions and indices of players.

We say that a pair (J_1, J_2) of two disjoint subsets $J_1, J_2 \subseteq [n]$ has the **flip-property** if there exist an input $x \in \{0,1\}^n$ and two strings $\alpha \in \{0,1\}^{|J_1|}$ and $\beta \in \{0,1\}^{|J_2|}$ with

$$f(x\lceil_{J_1,J_2\leftarrow\alpha,\beta}) \;\neq\; f(x\lceil_{J_1,J_2\leftarrow\overline{\alpha},\beta}) \;=\; f(x\lceil_{J_1,J_2\leftarrow\alpha,\overline{\beta}}) \,.$$

We call the strings α and β **flip-witnesses** for (J_1, J_2).

Lemma 3. *If a function $f : \{0,1\}^n \to \{0,1\}$ is non-degenerate, then for every partition $I_1, I_2 \subseteq [n]$ and every $i \in I_1$ and $j \in I_2$ we have: There exist subsets $J_1 \subseteq I_1$ and $J_2 \subseteq I_2$ with $i \in J_1, j \in J_2$ such that (J_1, J_2) has the flip-property.*

Loosely speaking, this lemma says that each non-degenerate function behaves on subsets of input positions in some sense like the parity function.

Proof. By contradiction, assume that the lemma does not hold for a particular partition $I_1, I_2 \subseteq [n]$ and two indices $i \in I_1$ and $j \in I_2$. From Def. 3 it follows that for every $i \in I_1$ and $j \in I_2$ there exist input strings $y, z \in \{0,1\}^n$ such that

$$f(y\lceil_{\{i\}\leftarrow 0}) \;\neq\; f(y\lceil_{\{i\}\leftarrow 1}) \quad\text{and}\quad f(z\lceil_{\{j\}\leftarrow 0}) \;\neq\; f(z\lceil_{\{j\}\leftarrow 1}) \,.$$

If the lemma does not hold, we can conclude that

$$f(y\lceil_{\{i\},\{j\}\leftarrow 0,0}) \;=\; f(y\lceil_{\{i\},\{j\}\leftarrow 0,1}) \;\neq\; f(y\lceil_{\{i\},\{j\}\leftarrow 1,0}) \;=\; f(y\lceil_{\{i\},\{j\}\leftarrow 1,1}) \,.$$

Otherwise, at least one of the following cases holds:

– $f(y\lceil_{\{i\},\{j\}\leftarrow 0,1}) \neq f(y\lceil_{\{i\},\{j\}\leftarrow 1,1})$ and $f(y\lceil_{\{i\},\{j\}\leftarrow 0,1}) = f(y\lceil_{\{i\},\{j\}\leftarrow 1,0})$. Choosing $J_1 = \{i\}$, $J_2 = \{j\}$, and $\alpha = \beta = 1$ satisfies the claim of the lemma.
– $f(y\lceil_{\{i\},\{j\}\leftarrow 0,0}) \neq f(y\lceil_{\{i\},\{j\}\leftarrow 0,1})$ and $f(y\lceil_{\{i\},\{j\}\leftarrow 0,0}) = f(y\lceil_{\{i\},\{j\}\leftarrow 1,1})$ Choosing $J_1 = \{i\}$, $J_2 = \{j\}$, $\alpha = 0$ and $\beta = 1$ satisfies the claim of the lemma.

Analogously, one can show that

$$f(z\lceil_{\{i\},\{j\}\leftarrow 0,0}) \;=\; f(z\lceil_{\{i\},\{j\}\leftarrow 1,0}) \;\neq\; f(z\lceil_{\{i\},\{j\}\leftarrow 0,1}) \;=\; f(z\lceil_{\{i\},\{j\}\leftarrow 1,1}) \,.$$

W.l.o.g. assume that $y[i] \neq z[i]$ and $y[j] \neq z[j]$. If $f(y) = f(z)$, we flip the bits $y[j]$ and $z[j]$. Since $f(y)$ does not depend on $y[j]$, we have $f(y) \neq f(z)$. We choose

$$Y_1 \;:=\; \{\; k \in I_1 \mid y[k] \neq z[k] \;\} \quad \text{and } Y_2 \;:=\; \{\; k \in I_2 \mid y[k] \neq z[k] \;\} \,.$$

Let $Y_1 := \{i_1, \ldots, i_{|Y_1|}\}$ with $i_1 < i_1 < \cdots < i_{|Y_1|}$ and $Y_2 := \{j_1, \ldots, j_{|Y_2|}\}$ with $j_1 < j_1 < \cdots < j_{|Y_2|}$. Define $\rho \in \{0,1\}^{|Y_1|}$ and $\sigma \in \{0,1\}^{|Y_2|}$ such that

$$\forall \ell \in [1, |Y_1|] \;:\; \rho[\ell] := y[i_\ell] \qquad\text{and}\qquad \forall \ell \in [1, |Y_2|] \;:\; \sigma[\ell] := y[i_\ell] \,.$$

Note that $y\lceil_{Y_1,Y_2 \leftarrow \overline{\rho},\overline{\sigma}} \;=\; z$.

Recall that $f(y) \neq f(z)$. To prove the claim we have to distinguish between the following three cases: $f(y) \neq f(y\lceil_{Y_2 \leftarrow \overline{\sigma}}) = f(z)$, $f(y) = f(y\lceil_{Y_1 \leftarrow \overline{\rho}}) \neq f(z)$, and $f(y) \neq f(y\lceil_{Y_1 \leftarrow \overline{\rho}}) = f(z)$. The last case can be reduced to the first case by exchanging y and z with each other.

1. If $f(y) \neq f(y\lceil_{Y_2 \leftarrow \overline{\sigma}}) = f(z)$, we choose

$$\alpha := y[i], \quad \beta := \sigma, \quad J_1 := \{i\}, \quad J_2 := Y_2, \quad \text{and} \quad x := y\,.$$

   From the definition of non-degenerate functions and the observation above we conclude that

$$\begin{aligned} y = x\lceil_{J_1,J_2 \leftarrow \alpha,\beta} \text{ and } J_1 = \{i\} &\Longrightarrow f(x\lceil_{J_1,J_2 \leftarrow \alpha,\beta}) \neq f(x\lceil_{J_1,J_2 \leftarrow \overline{\alpha},\beta})\,, \\ y\lceil_{J_2 \leftarrow \overline{\sigma}} = x\lceil_{J_1,J_2 \leftarrow \alpha,\overline{\beta}} &\Longrightarrow f(x\lceil_{J_1,J_2 \leftarrow \alpha,\beta}) \neq f(x\lceil_{J_1,J_2 \leftarrow \alpha,\overline{\beta}})\,. \end{aligned}$$

2. If $f(y) = f(y\lceil_{Y_2 \leftarrow \overline{\sigma}}) = f(z\lceil_{Y_1 \leftarrow \rho}) \neq f(z)$ then we choose

$$\alpha := \overline{\rho}, \quad \beta := z[j], \quad J_1 := Y_1, \quad J_2 := \{j\}, \quad \text{and} \quad x := z\,.$$

   It follows

$$\begin{aligned} z = x\lceil_{J_1,J_2 \leftarrow \alpha,\beta} \text{ and } J_2 = \{j\} &\Longrightarrow f(x\lceil_{J_1,J_2 \leftarrow \alpha,\beta}) \neq f(x\lceil_{J_1,J_2 \leftarrow \alpha,\overline{\beta}})\,, \\ z\lceil_{J_1 \leftarrow \rho} = x\lceil_{J_1,J_2 \leftarrow \overline{\alpha},\beta} &\Longrightarrow f(x\lceil_{J_1,J_2 \leftarrow \alpha,\beta}) \neq f(x\lceil_{J_1,J_2 \leftarrow \overline{\alpha},\overline{\beta}})\,. \end{aligned}$$

Hence, we can always find subsets $J_1 \subseteq I_1$ and $J_2 \subseteq I_2$ fulfilling the claim – a contradiction. □

For a given subset I_1 of input positions define the **flip-witness-set** for I_1

$$\begin{aligned} \text{f-set}(I_1) \;:=\; \{(\alpha, J_1) \mid\, & J_1 \subseteq I_1, \alpha \in \{0,1\}^{|J_1|} \\ & \text{and there exists } J_2 \subseteq [n] \setminus I_1, \beta \in \{0,1\}^{|J_2|} \\ & \text{such that } \alpha, \beta \text{ are flip-witnesses for } J_1, J_2\}\,. \end{aligned}$$

A set I_1 is **dominated** by an input position $k \in I_1$ if the following holds: For each pair of subsets $J_1 \subseteq I_1$ and $J_2 \subseteq [n] \setminus I_1$, such that (J_1, J_2) fulfils the flip-property, we have $k \in J_1$. A function is **ℓ-dominated** if there exists a set $I_1 \subseteq [n]$ of size ℓ that is dominated by some $k \in I_1$. A function f is called **dominated** if there exists $\ell > 1$ such that f is ℓ-dominated. Otherwise, f is called **non-dominated**.

Theorem 3. *Let f be a non-degenerate function and G be a network that can be separated into two networks G_1 and G_2 of size n_1 and n_2, respectively, by removing one bridge node from G. If f can be computed by a private protocol on G, then f is $(n_1 + 1)$- or $(n_2 + 1)$-dominated.*

Theorem 3 follows directly from Lemma 3 and the lemma below. Recall that for all $i \in [n]$ player P_i initially knows $x[i]$. Hence, we can obtain every possible allocation of players and input bits by permuting the enumeration of the players.

Lemma 4 (Fooling private protocols). *Let G be a network with n nodes. Assume that there exist $I_1, I_2 \subseteq [n]$ and $k \in [n]$, such that the following conditions hold:*

1. *$I_1, I_2 \neq \emptyset$ and $k \notin I_1 \cup I_2$, $I_1 \cap I_2 = \emptyset$,*
2. *for every path $W_{i,j}$ from P_i to P_j, with $i \in I_1$ and $j \in I_2$, $P_k \in W_{i,j}$, and*
3. *(I_1, I_2) has the flip-property.*

Then f cannot be computed on G by a private protocol.

Proof. Assume that there exists such a protocol. Let M_i^t be a message sent by player P_i in round t and $T(\mathcal{A})$ be the maximum number of rounds of $\mathcal{A}$ for all inputs of length n and all random tapes. Obviously M_i^t is a function of the input string z and the random tapes R. Player P_i receives in round $t \leq T(\mathcal{A})$ the messages

$$C_i^t(z, R) := M_{i_1}^t(z, R), \ldots, M_{i_s}^t(z, R) ,$$

where $P_{i_1}, \ldots, P_{i_s}$ are all the players incident to player P_i. We denote the sequence $C_i^1(z, R)$, $C_i^2(z, R), \ldots, C_i^{T(\mathcal{A})}(z, R)$ by $C_i(z, R)$.

Now let k, I_1, I_2 fulfil conditions 1, 2, and 3 of the lemma and choose x, α, and β such that

$$f(x\lceil_{I_1,I_2 \leftarrow \alpha,\beta}) \neq f(x\lceil_{I_1,I_2 \leftarrow \alpha,\overline{\beta}}) = f(x\lceil_{I_1,I_2 \leftarrow \overline{\alpha},\beta}) .$$

Keep R fixed. Then consider $C_k(x\lceil_{I_1,I_2 \leftarrow \alpha,\overline{\beta}}, R)$, which is the sequence of messages received by the player k during the computation on $x\lceil_{I_1,I_2 \leftarrow \alpha,\overline{\beta}}$ with random bits R. Since the protocol is private and $k \notin I_1 \cup I_2$, there exists $R' = (R'_1, \ldots, R'_n)$, with $R_k = R'_k$, such that

$$C_k(x\lceil_{I_1,I_2 \leftarrow \overline{\alpha},\beta}, R') = C_k(x\lceil_{I_1,I_2 \leftarrow \alpha,\overline{\beta}}, R) . \tag{1}$$

Let $Y := \{\ell \mid \text{there is a path } W_{\ell,i} \text{ from } \ell \text{ to a node } i \in I_1 \text{ with } k \notin W_{\ell,i} \}$.

Obviously we have $I_1 \subseteq Y$ and $I_2 \cap Y = \emptyset$. Now let $R'' = (R''_1, \ldots, R''_n)$ be a content of random tapes defined as follows: for every $\ell \in Y$ let $R''_\ell := R_\ell$ and for every $j \in [n] \setminus Y$ let $R''_j := R'_j$. Note that $R''_k = R'_k = R_k$. From Equation (1) it follows that on input $x\lceil_{I_1,I_2 \leftarrow \alpha,\beta}$ and with random tapes R'' the protocol generates the following messages for every player $i \in [n]$ and every $t \geq 1$

$$M_i^t(x\lceil_{I_1,I_2 \leftarrow \alpha,\beta}, R'') = \begin{cases} M_i^t(x\lceil_{I_1,I_2 \leftarrow \alpha,\overline{\beta}}, R) & \text{if } i \in Y , \\ M_i^t(x\lceil_{I_1,I_2 \leftarrow \overline{\alpha},\beta}, R') & \text{if } i \in [n] \setminus Y . \end{cases}$$

Hence, given the input string $x\lceil_{I_1,I_2 \leftarrow \alpha,\beta}$ the protocol computes the same value as on the input string $x\lceil_{I_1,I_2 \leftarrow \overline{\alpha},\beta}$ and $x\lceil_{I_1,I_2 \leftarrow \alpha,\overline{\beta}}$ – a contradiction. □

Corollary 2. *A non-dominated non-degenerate function cannot be computed by a private protocol on a network that is not 2-connected.*

Examples of non-dominated non-degenerate functions are the parity function, the or function, and the majority function. Hence, these functions cannot be computed by private protocols on networks that are not 2-connected.

In the remainder of this section, we show that for every dominated function f there is a non-2-connected network on which f can be computed by a private protocol.

The following three lemmas can be proved similar to Lemma 3.

Lemma 5. *Assume that a set I_1 with $|I_1| \geq 2$ is dominated by an input position $k \in I_1$. Then every pair $(a, J_1) \in$ f-set(I_1) assigns the same value to $x[k]$.*

For $c \in \{0,1\}$, we call a set I_1 **(k,c)-dominated** if I_1 is dominated by k and for each pair $(\alpha, J_1) \in$ f-set(I_1), α assigns c to $x[k]$.

Lemma 6. *Assume that a set I_1 with $|I_1| \geq 2$ is (k,c)-dominated with $k \in I_1$ for some $c \in \{0,1\}$. Then for every $\alpha \in \{0,1\}^{|I_1|}$ with $\alpha[k] \neq c$, for every $w \in \{0,1\}^n$, $J_2 \subseteq [n] \setminus I_1$, and $\beta \in \{0,1\}^{|J_2|}$ we have*

$$f(w\lceil_{I_1,J_2 \leftarrow \alpha,\beta}) \;=\; f(w\lceil_{I_1,J_2 \leftarrow \alpha,\overline{\beta}}) \,.$$

By the previous lemma, we can conclude that for each set I_1 with $|I_1| \geq 2$ that is (k,c)-dominated with $k \in I_1$ there exists a function $f_1 : \{0,1\}^{|I_1|} \rightarrow \{0,1\}$ such that

$$f(x) \;=\; ((x[k] = c) \wedge f(x)) \vee ((x[k] \neq c) \wedge f_1(x[I_1])) \,.$$

This reduces the set of interesting variables to I_1 if $x[k] \neq c$. Let us now focus on input strings with $x[k] = c$.

Lemma 7. *Assume that a set I_1 with $|I_1| \geq 2$ is (k,c)-dominated with $k \in I_1$ for some $c \in \{0,1\}$. Then for every pair $w_1, w_2 \in \{0,1\}^n$ with $w_1[k] = w_2[k] = c$ and $w_1[i] = w_2[i]$ for all $i \in [n] \setminus I_1$ we have $f(w_1) = f(w_2)$.*

Thus, we can conclude that for each set I_1 with $|I_1| \geq 2$ that is (k,c)-dominated with $k \in I_1$, there exists a function $f_2 : \{0,1\}^{|I_2|} \rightarrow \{0,1\}$ such that $f(x) = ((x[k] \neq c) \wedge f_2(x[I_2])) \vee ((x[k] = c) \wedge f_1(x[I_1]))$. Summarising the above three lemmas we get the following result.

Theorem 4. *Assume that a set I_1 with $|I_1| \geq 2$ is (k,c)-dominated with $k \in I_1$ for some $c \in \{0,1\}$. Let $I_2 = [n] \setminus I_1$. Then there are two functions $f_1 : \{0,1\}^{|I_1|} \rightarrow \{0,1\}$ and $f_2 : \{0,1\}^{|I_2|} \rightarrow \{0,1\}$ such that*

$$f(x) \;=\; ((x[k] = c) \wedge f_1(x[I_1])) \;\vee\; ((x[k] \neq c) \wedge f_2(x[I_2])) \,.$$

Note, that k, I_1, and I_2 are uniquely determined by the function f. Hence, every dominated function can be described by an if-then-else construction, i.e. it is of the form `if` $x[k] = c$ `then` $f_1(x[I_1])$ `else` $f_2(x[I_2])$.

Theorem 4 immediately implies that dominated functions can be computed on networks that are not 2-connected.

Theorem 5. *If f is ℓ-dominated with $\ell > 1$, then f can be computed by a private protocol on a network that consists of two 2-connected components with one node in common. One of the components has size ℓ and the other one size $n - \ell + 1$.*

Corollary 3. *Assume that f is a dominated function. Then there are non-2-connected networks on which f can be computed by a private protocol.*

Theorem 5 can be generalised to the case where we allow teams of players to work together. Assume that all members of a team belong to the component that computes, say, f_1. Then f is t-private if f_1 is t-private. If the members are distributed among both components, then this virtually decreases the team sizes for both components. f is t-private if both f_1 and f_2 are t-private.

6 Conclusions and Open Problems

We have investigated the relation between the connectivity of networks and the possibility of computing functions by private protocols on these networks. Special emphasis has been put on the amount of randomness needed.

We have presented a general simulation technique which allows us to transfer every oblivious private protocol on an arbitrary network into an oblivious private protocol on a given k-connected network of the same size, where $k \geq 2$. The new protocol needs $\left(1 - \frac{k}{n-1}\right) \cdot \min\left\{L, n^2 + \frac{L}{k-1}\right\}$ random bits more than the original protocol, where L is the total amount of bits sent in the original protocol. The obvious open question here is either to further reduce the number of extra random bits or to prove general lower bounds.

The parity function can be computed on a cycle using only one random bit and only one message per link. Thus, $1 + n - \frac{kn}{n-1}$ random bits are sufficient to compute the parity function on an arbitrary k-connected graph by a private protocol using our simulation. We have strengthened this bound by showing that on every k-connected graph, parity can be computed by an oblivious private protocol using at most $\left\lceil \frac{n-2}{k-1} \right\rceil - 1$ random bits. Furthermore, there exist k-connected networks for which this bound is sharp. The latter bound even holds for non-oblivious protocols.

While every Boolean function can be computed on a 2-connected network by a private protocol, this is no longer true for 1-connected networks. Starting from this observation, we have completely characterized the functions that can be computed by a private protocol on 1-connected networks.

Our simulation results focus on the extra amount of randomness needed. It would also be interesting to bound the number of rounds of the simulation in terms of the number of rounds of the original protocol and, say, the diameter of the new network.

Acknowledgement

We thank Adi Rosén for valuable discussions and hints to literature.

References

1. N. Alon, J. H. Spencer, and P. Erdös. *The Probabilistic Method*. John Wiley and Sons, 1992.
2. J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in a constant number of rounds of interaction. In *Proc. 8th Ann. Symp. on Principles of Distributed Comput. (PODC)*, pages 201–209. ACM, 1989.
3. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. 20th Ann. Symp. on Theory of Comput. (STOC)*, pages 1–10. ACM, 1988.
4. C. Blundo, A. de Santis, G. Persiano, and U. Vaccaro. Randomness complexity of private computation. *Comput. Complexity*, 8(2):145–168, 1999.
5. R. Canetti and R. Ostrovsky. Secure computation with honest-looking parties: What if nobody is truly honest? In *Proc. 31st Ann. Symp. on Theory of Comput. (STOC)*, pages 255–264. ACM, 1999.
6. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proc. 20th Ann. Symp. on Theory of Comput. (STOC)*, pages 11–19. ACM, 1988.
7. B. Chor, M. Geréb-Graus, and E. Kushilevitz. On the structure of the privacy hierarchy. *J. Cryptology*, 7(1):53–60, 1994.
8. B. Chor, M. Geréb-Graus, and E. Kushilevitz. Private computations over the integers. *SIAM J. Comput.*, 24(2):376–386, 1995.
9. B. Chor and E. Kushilevitz. A zero-one law for boolean privacy. *SIAM J. Discrete Math.*, 4(1):36–47, 1991.
10. B. Chor and E. Kushilevitz. A communication-privacy tradeoff for modular addition. *Inform. Process. Lett.*, 45(4):205–210, 1993.
11. Y. Egawa, R. Glas, and S. C. Locke. Cycles and paths through specified vertices in k-connected graphs. *J. Combin. Theory Ser. B*, 52:20–29, 1991.
12. M. Franklin and R. N. Wright. Secure communication in minimal connectivity models. *J. Cryptology*, 13(1):9–30, 2000.
13. M. Franklin and M. Yung. Secure hypergraphs: Privacy from partial broadcast. In *Proc. 27th Ann. Symp. on Theory of Comput. (STOC)*, pages 36–44. ACM, 1995.
14. A. Gál and A. Rosén. A theorem on sensitivity and applications in private computation. In *Proc. 31st Ann. Symp. on Theory of Comput. (STOC)*, pages 348–357. ACM, 1999.
15. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proc. 19th Ann. Symp. on Theory of Comput. (STOC)*, pages 218–229. ACM, 1987.
16. E. Kushilevitz. Privacy and communication complexity. *SIAM J. Discrete Math.*, 5(2):273–284, 1992.
17. E. Kushilevitz and Y. Mansour. Randomness in private computations. *SIAM J. Discrete Math.*, 10(4):647–661, 1997.
18. E. Kushilevitz, R. Ostrovsky, and A. Rosén. Characterizing linear size circuits in terms of privacy. *J. Comput. System Sci.*, 58(1):129–136, 1999.
19. E. Kushilevitz and A. Rosén. A randomness-rounds tradeoff in private computation. *SIAM J. Discrete Math.*, 11(1):61–80, 1998.
20. I. Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner, 1987.
21. A. C.-C. Yao. Protocols for secure computations. In *Proc. 23rd Ann. Symp. on Foundations of Comput. Sci. (FOCS)*, pages 160–164. IEEE, 1982.
22. A. C.-C. Yao. How to generate and exchange secrets. In *Proc. 27th Ann. Symp. on Foundations of Comput. Sci. (FOCS)*, pages 162–167. IEEE, 1986.