Private computation

K-connected versus 1-connected networks

J. Cryptology (2006) 19: 341–357
DOI: 10.1007/s00145-005-0329-x

Journal of
CRYPTOLOGY

Private Computation: k-Connected versus 1-Connected Networks*

Markus Bläser
Institut für Theoretische Informatik,
ETH Zürich,
8092 Zürich, Switzerland
mblaeser@inf.ethz.ch

Andreas Jakoby, Maciej Liśkiewicz, and Bodo Manthey
Institut für Theoretische Informatik,
Universität zu Lübeck,
Ratzeburger Allee 160, 23538 Lübeck, Germany
{jakoby,liskiewi,manthey}@tcs.uni-luebeck.de

Communicated by Matthew Franklin

Received 25 April 2003 and revised October 14, 2004
Online publication 27 May 2005

Abstract. We study the role of connectivity of communication networks in private computations under information theoretical settings in the honest-but-curious model. We show that some functions can be 1-privately computed 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 1-privately computed on non-2-connected networks.

Furthermore, we present a technique for simulating 1-private protocols that work on arbitrary (complete) networks on k-connected networks. For this simulation, at most $(1 - k/(n - 1)) \cdot L$ additional random bits are needed, where L is the number of bits exchanged in the original protocol and n is the number of players.

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-privately, namely $\lceil (n - 2)/(k - 1) \rceil - 1$ random bits for networks consisting of n players.

Key words. Private computation, Secure multi-party computation, Secure function evaluation, Connectivity, Parity, Randomness.

* A preliminary version appeared in *Proceedings of the* 22*nd Annual International Cryptology Conference* (*CRYPTO* 2002), pp. 194–209, volume 2442 of Lecture Notes in Computer Science, Springer, Berlin, 2002. The work by Markus Bläser was done while at the Institut für Theoretische Informatik, Universität zu Lübeck, Germany. Maciej Liśkiewicz is on leave from Instytut Informatyki, Uniwersytet Wroclawski, Poland. The last author's birth name was Bodo Siebert and he was supported by DFG Research Grant Re 672/3.

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 of the other players that cannot be derived from the function value and its own secret. An example for such a computation is the "secret ballot problem": The members of a committee wish to decide whether the majority votes for yes or no. However, after the vote nobody should know anything about the opinions of the other committee members, not even about the exact number of votes for yes and no, except for whether the majority has voted for yes or no. To come to a decision, any two members can talk to each other in private. If however the members are distributed in a network, then only those members that are connected by a link can talk to each other. In this work we investigate the influence of the underlying network on the ability to perform private computations.

Let f be an n-ary Boolean function and let $x_1, \ldots, x_n$ be bits distributed among n players. A protocol for computing $f(x_1, \ldots, x_n)$ is called t-private if after executing the protocol all players know $f(x_1, \ldots, x_n)$, but no group of at most t players learns anything about the bits of the other players except for what they can deduce from the function value and their own bits.

Depending on the computational power of the players we distinguish between cryptographically secure privacy and information theoretically secure privacy. In the first case we assume that no player is able to gain any information about the input bits of the other players within polynomial time [28], [29]. In the second case we do not restrict the computational power of the players. This notion of privacy (sometimes called unconditional privacy) has been introduced by Ben-Or et al. [3] and Chaum et al. [8]. Private computation has been examined with two different types of players. Malicious players (also called Byzantine players) may arbitrarily deviate from the protocol in order to jam the correctness or the privacy constraint [19], [28], [29]. Honest-but-curious players follow the protocol precisely but are allowed to "gossip" afterwards [19], [23].

We are concerned with 1-privacy in the information theoretically secure setting with honest-but-curious players.

1.1. *Previous Results*

Private computation has been the subject of a considerable amount of research. In the information theoretically secure model, all n-ary Boolean functions can t-privately be computed if $t < n/2$ in the case of honest-but-curious players and if $t < n/3$ in the case of malicious players [3], [8]. In the cryptographically secure model, this holds for $t \leq n$ in the case of honest-but-curious players and $t < n/2$ in the case of malicious players [19], [28] (assuming that trapdoor functions exist). Canetti and Ostrovsky [7] proved that in the cryptographically secure setting, it can be tolerated that all parties deviate from the protocol under the restriction that most parties do not risk being detected by other parties. (Malicious players do not care about being detected.)

Traditionally, one investigates the number of rounds and random bits as complexity measures for private protocols. According to Canetti et al. [6], the quantification of the

amount of randomness needed in cryptographically secure privacy is not meaningful, since it can be reduced using pseudorandom generators [4], [21].

The following papers deal with information theoretical private computation with honest-but-curious players: Chor and Kushilevitz [12] have studied the number of rounds necessary for privately computing the sum modulo an integer. This function has also been investigated by Blundo et al. [5] and Chor et al. [10]. The number of random bits needed for privately computing the parity function has been examined by Kushilevitz and Mansour [25] and Kushilevitz and Rosén [27]. Gál and Rosén [16] 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 for privately computing a function have also been given by Bar-Ilan and Beaver [2] and by Kushilevitz [24]. Gál and Rosén [17] have proved an upper and lower bound for the number of random bits needed for t-privately computing parity.

The number of random bits necessary for privately computing a Boolean function is closely related to its circuit size. Kushilevitz et al. [26] have shown that every function can be computed with linear circuit size if and only if it can be privately computed with a constant number of random bits.

Chor and Kushilevitz [11] have characterised the class of Boolean functions that can be t-privately computed for some $t \geq n/2$. Any such function can already be n-privately computed.

Chor et al. [9], [10] have extended the field of private computation to functions defined over finite domains. Kilian et al. [22], [23] have introduced a notion of reduction and completeness in private computation.

All the 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 [15] have been the first who studied the role of connectivity in private computations. They have presented a protocol for k-connected bus networks that simulates communication steps of a private protocol that was originally written for a complete graph. To simulate a single communication step, their protocol uses $O(n)$ additional random bits. Franklin and Wright [14] have examined which functions are still privately computable, if the players are malicious and the network connectivity is low.

1.2. *Our Results*

As mentioned above, we are concerned with 1-privacy in the information theoretically secure setting with honest-but-curious players. In the following we use the term "private" for "1-private".

In this paper we investigate the number of random bits needed to compute functions by private protocols on k-connected networks. We present a simulation of private protocols designed for arbitrary networks on arbitrary k-connected networks (for $k \geq 2$) in Section 3. For this simulation, only $(1 - k/(n-1)) \cdot \min\{L, (k-2)/(k-1) \cdot (n^2 -$

$n) + L/(k-1)\}$ additional random bits are needed, where L is the total number of bits sent in the original protocol.

In Section 4 we study the parity function to a greater extent. For every k-connected graph with $k \geq 2$, we design a private protocol for computing the parity function that uses only $\lceil (n-2)/(k-1) \rceil - 1$ random bits. This considerably reduces the number of random bits compared with the general simulation technique of Section 3 for the specific case of the parity function. This result is tight: There are k-connected graphs on which every private protocol needs that many random bits to compute the parity function.

All of the above results hold for $k \geq 2$. In Section 5 we investigate graphs that are not 2-connected. Our first insight is the following: The parity function over $n > 2$ bits cannot be computed by a private protocol on any network that is not 2-connected. This can be generalised to a large class of non-degenerate functions. An n-ary Boolean function is called *non-degenerate* if it depends on all of its n input bits. It turns out that there are functions that can be privately computed, even if the underlying network is not 2-connected. An example is the following non-degenerate function $f : \{0,1\}^{2n+1} \to \{0,1\}$ (for $n \geq 2$):

$$f(x, y, z) = \left(x \wedge \bigwedge_{i=1}^{n} y_i \right) \vee \left(\overline{x} \wedge \bigwedge_{i=1}^{n} z_i \right).$$

Here, x is a single bit and both y and z are bit strings of length n. We construct a communication network G for f as follows: Let G_y and G_z be complete networks with n players each. Then connect another player P_x with all players in both G_y and G_z. The network obtained is not 2-connected. Using a slight modification of the protocol presented by Kushilevitz et al. [26], one can privately compute the subfunctions

$$f_y(x, y) = x \wedge \bigwedge_{i=1}^{n} y_i \quad \text{and}$$
$$f_z(x, z) = \overline{x} \wedge \bigwedge_{i=1}^{n} z_i$$

on the networks G_y with P_x and G_z with P_x, respectively. Overall, the protocol is private as will be shown in Section 5.

We fully characterise the class of non-degenerate functions that can be privately computed on non-2-connected networks. It turns out that the above example is fairly representative: Each such function has this `if-then-else` structure. The corresponding non-2-connected network consists of two 2-connected components of appropriate sizes.

2. Preliminaries

For $n \in \mathbb{N}$ let $[n] = \{1, \ldots, n\}$. A graph G is called k-connected if, after deleting an arbitrary subset of at most $k-1$ nodes, the resulting node-induced graph remains connected. Equivalently, for any two nodes u and v of G, there are at least k pairwise node-disjoint paths between u and v. A *block* of G is a maximum node-induced subgraph of G that is 2-connected.

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 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 theoretical sense, i.e. with unbounded computational power, except for the information he can deduce from his own bit and the result. Such a protocol is called private. (Recall that private in this paper means 1-private and that we are considering honest players.)

Definition 2.1. Let C_i be a random variable of the communication string seen by player P_i and let R_i be his random string. 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 $x_i = y_i$, the following conditions hold:

1. for all r, $\Pr(R_i = r \mid x) = \Pr(R_i = r \mid y)$, and
2. for all r with $\Pr(R_i = r \mid x) > 0$ and for all c,

$$\Pr(C_i = c \mid R_i = r, x) = \Pr(C_i = c \mid R_i = r, y).$$

(The probabilities are taken over the random strings of all other players.) A protocol $\mathcal{A}$ is *private* if it is private with respect to all players.

If the number of random bits each player uses is independent of the input, we can omit the first condition since in this case all these probabilities are equal. However, if the number of random bits a player uses depends on the input of the other players and on their random bits, this player might be able to learn something from his random string if $\Pr(R_i = r \mid x) \neq \Pr(R_i = r \mid y)$.

In all the protocols presented here it is known in advance how many random bits a player uses. The lower bound for the number of random bits needed for computing parity however holds also for the more general case when this is not known in advance.

We call a protocol *oblivious* if the communication takes place in rounds, each message consists of a single bit, and the number of bits (which is then either zero or one) that 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. For an oblivious protocol $\mathcal{A}$ let $L(P_i, P_j, \mathcal{A})$ be the total number of bits sent from P_i to P_j in $\mathcal{A}$ and

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

We distribute the input bits among the nodes of the graph. For convenience, we call the node that gets bit x_i player P_i. The players P_i and P_j can communicate directly with each other 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 a 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 realise the communication steps between all players that are not directly connected. Franklin and Yung [15] 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 by them 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 first path and $r \oplus b$, the parity of r and b, along the second path. In this way, $m + L(\mathcal{A})$ random bits are sufficient. To reduce the number of random bits even further, we consider the following optimisation problem.

Definition 3.1 (Max-Neighbour-Embedding). Let $G = (V, E)$ be a complete graph with edge weights $\sigma : E \to \mathbb{N}$ and let $G' = (V', E')$ be a graph with $|V| = |V'|$. Let $\pi : V \to V'$ be a bijective mapping. Then the performance of π is defined as

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

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

By reduction from 3-Dimensional-Matching [18, SP1], it can be shown that the decision problem corresponding to Max-Neighbour-Embedding is $\mathcal{NP}$-hard, even if σ is $\{0, 1\}$-valued, the graph consisting of the weight one edges of G has maximum degree four, and G' has maximum degree four. In the following lemma we estimate the performance for the case that G' is k-connected.

Lemma 3.2. *Let $G = (V, E)$ be a graph with n nodes and edge weights σ. Let $G' = (V', E')$ be a k-connected graph with n nodes. Then we have*

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

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

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

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

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. [1]).

In the simulation described below, the graph G is the network for which a given protocol was designed. The edge weights are the number of bits exchanged over each edge (with weight zero if there is no edge in the original network). The graph G' is the k-connected network on which we want to simulate the protocol.

The main idea is that for all nodes P_i and P_j in a k-connected graph, we have k node-disjoint paths connecting these two nodes. Thus, we can simulate $k-1$ bits sent from P_i to P_j as follows: First, P_i sends a random bit to P_j on one path. Then he uses this random bit to encode $k-1$ bits sent along the other $k-1$ paths.

Theorem 3.3. *Every oblivious private protocol $\mathcal{A}$ using m random bits can be simulated with $m + (1 - k/(n-1)) \cdot \min\{L(\mathcal{A}), (k-2)/(k-1) \cdot (n^2 - n) + L(\mathcal{A})/(k-1)\}$ random bits on every k-connected graph.*

Proof. Let $G = (V, E)$ be the network used in protocol $\mathcal{A}$ and let $G' = (V', E')$ be the k-connected network. To simulate $\mathcal{A}$ we first choose a bijection between the players in G and the players in G'. For every edge $\{P_i, P_j\} \in E$ let $\sigma(\{P_i, P_j\}) = L(P_i, P_j, \mathcal{A}) + L(P_j, P_i, \mathcal{A})$. In Lemma 3.2 we have seen that there exists a bijection $\pi : V \to V'$ with performance $\rho(\pi) \geq (k/(n-1))L(\mathcal{A})$. Using this bijection, at least $k/(n-1) \cdot L(\mathcal{A})$ 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 without additional random bits.

For the remaining $(1 - k/(n-1)) \cdot L(\mathcal{A})$ 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 he 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_ℓ for each block B_ℓ. He sends r_ℓ 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.

$\sum_{i \in [n], j \in [n] \setminus \{i\}} \lceil L(P_i, P_j, \mathcal{A})/(k-1) \rceil \leq (k-2)/(k-1) \cdot (n^2 - n) + L(\mathcal{A})/(k-1)$ holds, since we round at most $n^2 - n$ fractions with denominator $k-1$. (This is a worst-case estimate. Given a concrete protocol, additional knowledge about the distribution of the bits on the links may be used to get a better bound.) However, we never need more than $(1 - k/(n-1)) \cdot L(\mathcal{A})$ bits altogether. Both observations together imply the bound proposed. □

4. Computing Parity on k-Connected Networks

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

Note that in the proof of the following lemma, we make no assumptions about how many random bits any player uses or that the number of random bits is known in advance.

Thus, the lower bound holds also for the more general case where the number of random bits each player uses can depend on the input and the other player's random tapes.

Lemma 4.1. *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 $\lceil (n-2)/(k-1) \rceil - 1$ random bits.*

Proof. We consider the bipartite graph $K_{k,n-k}$, which is k-connected, and show that every private protocol that computes the parity function on this network needs at least $\lceil (n-2)/(k-1) \rceil - 1$ 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 $K_{k,n-k}$. For every $i = 1, \ldots, k$ and $j = k+1, \ldots, n$ we have an edge $\{P_i, P_j\}$ in $K_{k,n-k}$. Now assume to the contrary that there exists a private protocol $\mathcal{A}$ on $K_{k,n-k}$ using less than $\lceil (n-2)/(k-1) \rceil - 1$ random bits.

Let $r = \langle r_1, \ldots, r_n \rangle$ 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 execution of $\mathcal{A}$ with random bits r on input x. Moreover, let

$$\begin{aligned} \mathcal{C}(x) &= \{\langle c_1, c_2, \ldots, c_k \rangle \mid \exists r \, \forall i \in [k] : \; c_i = \mathcal{C}_i(x, r)\} \quad \text{and} \\ \mathcal{C}_i(x) &= \{c \mid \exists r : \; c = \mathcal{C}_i(x, r)\}. \end{aligned}$$

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

$$X = \left\{ x \mid x_1 = x_2 = \cdots = x_k = 0 \text{ and } \bigoplus_{i=1}^{n} x_i = 0 \right\}.$$

For every $x \in X$ and every communication c_1 we define

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

Claim 4.2. $\exists c_1 \, \forall x \in X : \; \mathcal{C}(c_1, x) \neq \emptyset$.

Proof. Let $x \in X$. 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 $\mathcal{C}(c_1, x) \neq \emptyset$. If for some $y \in X$ the set $\mathcal{C}(c_1, y)$ is empty, then this violates the privacy constraint. □

We also need the following claim, which follows from work by Kushilevitz and Rosén [27]. For the sake of completeness we give a proof though.

Claim 4.3. *Let d be the maximum number of random bits used. Then for all $i \in [k]$, we have $|\bigcup_{x \in X} \mathcal{C}_i(x)| \leq 2^d$.*

Proof. We start by considering any fixed $x \in X$ and show that $|\mathcal{C}_i(x)| \leq 2^d$. We view the execution of $\mathcal{A}$ such that in each round first P_1, then $P_2, \ldots$, and finally P_n performs his computation. This can be done, since these computations do not depend on each other. Viewing the computation this way, only one random bit is read at any time. The

claim follows from the following observations: Any random bit has two outcomes, the player who reads the next random bit is determined by the previous random bits and x, and the players read at most d random bits.

Finally, we have $\mathcal{C}_i(x) = \mathcal{C}_i(y)$ for all $x, y \in X$, since P_i must not be able to distinguish x and y. □

Since the number of random bits used by the protocol is less than $(n-k-1)/(k-1)$, we have $|\bigcup_{x\in X} \mathcal{C}_i(x)| < 2^{(n-k-1)/(k-1)}$. Hence, we have

$$\left|\bigcup_{x\in X} \mathcal{C}(c_1, x)\right| \leq \prod_{j=2}^{k} \left|\bigcup_{x\in X} \mathcal{C}_j(x)\right| < 2^{n-k-1}.$$

Since $|X| = 2^{n-k-1}$ and by Claim 4.2, we get

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

This means that there are two different strings $x, y \in X$ such that on either string 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 $x_i \neq y_i$. 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 $1 \leq i \leq k$.

During a computation of protocol $\mathcal{A}$ on input $x_1 \cdots x_{i-1} y_i x_{i+1} \cdots x_n$ with random strings $\langle r_1, \ldots, r_{i-1}, r'_i, r_{i+1}, \ldots, r_n\rangle$ the players $P_1, P_2, \ldots, P_k$ again receive the communication strings $c_1, c_2, \ldots, c_k$. This is because the graph is bipartite and P_i can only communicate with $P_1, \ldots, P_k$. Hence, for this input they compute the same result as for x, a contradiction. □

Now we show that this bound is best possible. To obtain a private protocol that computes the parity function with $\lceil (n-2)/(k-1)\rceil - 1$ random bits, we use the following result by Egawa et al. [13].

Lemma 4.4 [13]. *Let G be a k-connected graph, $k \geq 2$, with minimum degree d and at least $2d$ vertices. Let V' be an arbitrary set of k vertices of G. Then G has a cycle of length at least $2d$ that contains every vertex of V'.*

Lemma 4.5. *Let $G = (V, E)$ be a k-connected graph with $|V| \geq 2k$ and $k \geq 2$. Then for every subset $V' \subseteq V$ with $|V'| = k$, there exists a simple cycle of length at least $2k$ containing all nodes in V'.*

Proof. Since G is k-connected, every node has degree at least k. Thus, G contains a simple cycle of length at least $2k$ running through all nodes in V' by Lemma 4.4. □

Lemma 4.6. *Let $G = (V, E)$ be a k-connected graph, $k \geq 2$, with $|V| \geq 2k$. Then for every subset $V' \subseteq V$ with $|V'| = k+1$, there exists a simple path containing all nodes in V'.*

Proof. By Lemma 4.5, G contains a cycle C running through k of the nodes in V'. If the last node v of V' is also on C, we simply delete one edge of C and are done. Otherwise, since G is connected there is a path from v to a node u of C such that each internal node of this path is not in C. By deleting one edge of C incident with u, we obtain the desired path. □

Lemma 4.7. *Let $G = (V, E)$ be a k-connected graph, $k \geq 2$, with $|V| \geq 2k+1$. Then G has a simple path with at least $2k+1$ nodes.*

Proof. By Lemma 4.5, G has a cycle C of length at least $2k$. If this length is strictly greater than $2k$, we delete one of its edges and are done. Otherwise, there is a node v not in C. Since G is connected there is a path from v to a node u of C such that each internal node of this path is not in C. By deleting one edge of C incident with u, we obtain the desired path. □

Now we present a protocol for computing parity on arbitrary k-connected networks G. We first assume that G has at least $2k+1$ nodes. Basically, our protocol works as follows. Each player is either red or black. Initially, all players are red. A player is red as long as he holds some (input or random) bit that has not contributed to parity yet. Otherwise, he is black. Using Lemmas 4.5–4.7, we find paths or cycles containing a certain number of red players, who then contribute their bits. For each such path or cycle, we need one random bit.

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 Lemma 4.7 such a path exists. The first player P_i in the path generates a random bit r. Then P_i computes $r \oplus z_i$, sends the result to the next player in the path, and sets $z_i := r$.

   Each internal player P_j on the path receives a bit b from his predecessor in the path, computes $b \oplus z_j$, sends this bit to his successor, and changes his colour to black.

   The last player P_ℓ on the path receives a bit b from his 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 $\lceil (n-3k+1)/(k-1) \rceil$ times.

   Choose $k+1$ red nodes and a path in G containing all these nodes. According to Lemma 4.6 such a path 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 $r \oplus z_i$, sends the result to the next player in the path, and sets $z_i := r$.

   Each internal player of the path P_j receives a bit b from his predecessor in the path. If P_j is a black player, he sends b to his successor. If P_j is red, he computes $b \oplus z_j$, sends this bit to his successor, and changes his 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, at least $k-1$ players have changed their colour. Hence, after $\lceil (n-3k+1)/(k-1) \rceil$ iterations of this step we have at least

$$\left\lceil \frac{n-3k+1}{k-1} \right\rceil \cdot (k-1) + 2k - 1 \geq n - k$$

black players. Thus, at most k are red.

4. Choose a cycle in G containing all red nodes. According to Lemma 4.5 such a cycle 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 black, he sends b to its successor. If P_j is red, he computes $b \oplus z_j$, sends this bit to his successor, and changes his colour to black.

   If P_{i_0} receives a bit b, he computes $b \oplus r$. The result of this step is the result of the parity function.

Let us 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 $\lceil (n-3k+1)/(k-1) \rceil$ random bits. Hence, the total number of random bits is

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

It remains to show that the protocol is private and computes the parity function. 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, he knows the result of the parity function.

Every bit received by some player in the second and third steps 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 does P_{i_0} learn anything about the other input bits from the bit he 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 4.8. *Let G be an arbitrary k-connected network, $k \geq 2$, with n nodes such that $n \geq 2k$. Then the parity of n bits can be computed by a private protocol on G using at most $\lceil (n-2)/(k-1) \rceil - 1$ random bits. If $n < 2k$, then the parity can be computed with one random bit.*

For all $k \geq 2$ and $n > k$, there exists a k-connected network on n nodes for which this bound is best possible.

Proof. The case $n \geq 2k+1$ has already been demonstrated. If $n \leq 2k$, then every node in G has degree at least $n/2$. Thus, G contains a Hamiltonian cycle due to Dirac's theorem, see, e.g. [20], and parity can be computed using one random bit.

The lower bound follows from Lemma 4.1 for $n \geq 2k$. For $n < 2k$, only one random bit is needed, which is optimal. □

5. Private Computation on Non-2-Connected Networks

In this section we characterise the class of non-degenerate Boolean functions that can be privately computed on networks that are 1-connected but not 2-connected.

A function $f : \{0,1\}^n \to \{0,1\}$ is non-degenerate if for all $1 \leq i \leq n$, there are $x, y \in \{0,1\}^n$ that differ only at the ith position and $f(x) \neq f(y)$.

Let f be a non-degenerate n-ary Boolean function. We say that a variable x_i *dominates* f if there is a partition Y, Z of the variables $\{x_1, \ldots x_n\}\backslash\{x_i\}$ with $Y, Z \neq \emptyset$, such that $f(x_1, \ldots, x_n)$ depends only on variables in Y if $x_i = 0$ and only on variables in Z if $x_i = 1$. (This partition is unique, since f is non-degenerate.) We call $bal_f(x_i) = \min\{|Y|, |Z|\}$ the *balance* of x_i in f. If f is dominated by a variable x, then we can reorder the variables of f and find g_0 and g_1 such that

$$f(x, y, z) = \begin{cases} g_0(y) & \text{if} \quad x = 0 \quad \text{and} \\ g_1(z) & \text{if} \quad x = 1. \end{cases}$$

For $k \leq n/2$, we denote by $B_{n,k}$ the class of all networks with n nodes and with exactly two blocks such that one block consists of $k+1$ and the other block consists of $n-k$ nodes. (The bridge node belongs to both components.)

Lemma 5.1. *Let f be a non-degenerate Boolean function. Then f cannot be both x- and y-dominated.*

Proof. Let f depend on x, y, and $z_1, \ldots, z_n$. Assume that f is dominated by x and y. Then for $x = 0$, f depends on variables $X_0 \subseteq \{y, z_1, \ldots, z_n\}$, and for $x = 1$, f depends on variables $X_1 = \{y, z_1, \ldots, z_n\}\backslash X_0$, since f is non-degenerate. Similarly, we have two disjoint sets Y_0 and Y_1 of variables with $Y_0 \cup Y_1 = \{x, z_1, \ldots, z_n\}$.

Without loss of generality we assume that $x \in Y_0$ and $y \in X_0$. Now consider $x = 1$. Since $y \notin X_1$, f does not depend on y when setting $x = 1$. Hence, f depends only on $X_1 = Y_0 \cap Y_1 = \emptyset$ for $x = 1$, because we assumed that f is y-dominated. Thus, x does not dominate f. □

Lemma 5.2. *Let f be a non-degenerate n-ary Boolean function, $n \geq 3$. Let G be a network in $B_{n,k}$. Assume that either*

1. *x does not dominate f or*
2. *x dominates f but $bal_f(x) \neq k$ or*
3. *x dominates f with partition Y and Z, $bal_f(x) = k$ but both blocks hold input bits from both Y and Z.*

Then f cannot be privately computed on G when the bridge player P_x holds x.

Proof. Let G_y and G_z be the two blocks of G. The vectors of input bits for G_y without P_x and G_z without P_x are y and z, respectively. In all three cases, $f(0, y, z)$ or $f(1, y, z)$

depends on both y and z. Without loss of generality assume that $f(0, y, z)$ depends on both y and z. Then there exist y' and z' such that $f(0, y', z)$ depends on z and $f(0, y, z')$ depends on y. Thus, there exist y'' and z'' such that

$$f(0, y', z'') = f(0, y'', z') \neq f(0, y', z').$$

Now consider any protocol for computing f on the given network. We fix some arbitrary content of P_x's random tape and $x = 0$.

In the following, m_y^t denotes a message received by P_x from G_y in round t. Analogously, m_z^t denotes a message received by P_x from G_z in round t. We assume that in any round first P_x receives m_y^t, then P_x receives m_z^t, and finally P_x sends messages to G_y and G_z. (Formally, this means splitting up one round into three.) Then m_y^t does not depend on m_z^t and m_z^t does not depend on m_y^{t+1}. Let $c_y^t = (m_y^1, \ldots, m_y^t)$ and $c_z^t = (m_z^1, \ldots, m_z^t)$. We call a certain c_y^t *undecided* if P_x can observe c_y^t both on input y' and y'' for G_y. Otherwise, we call c_y^t *decided*. For c_z^t, the terms decided and undecided are analogously defined. The intuition behind these terms is as follows: If c_y^t is decided, then P_x has learned that either y' or y'' is not G_y's input. On the other hand, if c_y^t is undecided, then we can change G_y's input from y' to y'' or vice versa and modify its random bits such that P_x does not perceive any differences. Clearly, c_y^0 and c_z^0 are undecided.

We start our protocol on y', z' as input for G_y, G_z. Now we prove two things: First, if both c_y^t and c_z^t are undecided for all t, then we can fool the protocol such that it computes a wrong function value. Second, if eventually c_y^t or c_z^t is decided, then the protocol is not private with respect to P_x.

Assume that c_y^t and c_z^t are undecided for all t. Our protocol eventually outputs $f(0, y', z')$ and c_y^t and c_z^t are still undecided. Then we can replace y' by y'' and adjust G_y's random bits such that P_x does not notice a difference. Thus, our protocol has computed $f(0, y', z') \neq f(0, y'', z')$, but $f(0, y'', z')$ would have been the right value.

So consider the first t on which c_y^t or c_z^t is decided. Due to symmetry, we restrict ourselves to considering the first case. P_x has learned that y'' is no longer possible as input for G_y. Since P_x receives m_z^t after m_y^t, the current c_z^{t-1} is still undecided. Thus, we can replace z' with z''. When the protocol terminates, P_x knows the function value $f(0, y', z'') = f(0, y'', z')$. In addition, he knows that (y'', z') has not been the input. Thus, the protocol is not private. □

Lemma 5.3. *Let f be a non-degenerate n-ary Boolean function that is dominated by x with $bal_f(x) = 1$. Then f cannot be privately computed on any non-2-connected network.*

Proof. Due to Lemma 5.2, the only possibility for computing f is a network from $B_{n,1}$ with bridge node P_x. Let P_y be the other player of the block of size 2. Without loss of generality we assume that for $x = 0$, $f(x, y, z) = g(z)$, and for $x = 1$, $f(x, y, z)$ is either y or $\overline{y}$. We assume that $f(1, y, z) = y$.

We show how to compute the conjunction of two variables (namely x and y) privately. Consider f on some input z_0 with $g(z_0) = 0$. Then $f(x, y, z_0) = x \wedge y$. If P_x and P_y could compute f privately, then a single player would be able to simulate the behaviour of the large block on input z_0 and P_x on x while another player would be able to simulate

P_y on y. This would yield a protocol for privately computing $x \wedge y$, which is impossible for two players [24]. □

Theorem 5.4. *Let f be a non-degenerate n-ary Boolean function, $n \geq 3$, and let G be a connected network of n nodes. Then f cannot be privately computed on G, if one of the following conditions holds*:

1. *$G \in B_{n,k}$, but there is no variable x that dominates f with $bal_f(x) = k$.*
2. *G consists of more than two blocks.*
3. *f is x-dominated with $bal_f(x) = 1$.*

Proof. Items 1 and 3 follow immediately from Lemmas 5.2 and 5.3, respectively.

Now assume that G consists of more than two blocks. There are two possibilities: either all blocks share one bridge node or we have at least two bridge nodes. In both cases our aim is to apply Lemma 5.2. This is not directly possible since Lemma 5.2 only speaks about networks with two blocks. Note however that if one cannot privately compute a function on a given network H, then one cannot privately compute it on any subnetwork of H. Hence, if we cannot privately compute a function on a network H with two blocks, we cannot privately compute it on any network that is obtained by splitting up each of the two blocks into several new blocks.

First, we treat the case that there is only one bridge node P_x holding variable x. Since f is non-degenerate, for either $x = 0$ or $x = 1$ the function value depends on input bits of at least two blocks B_1 and B_2. Let G' be the network with two blocks such that one block is B_1 and the other block is the complete graph on the remaining nodes with the bridge node P_x. If f could be privately computed on G, then f could also be privately computed on G', but this contradicts Lemma 5.2.

Second, assume that there are two bridge nodes. If f could be privately computed on G, then there must be two variables x and x' that dominate f due to Lemma 5.2. (Here, we again unite blocks to end up with two blocks as above.) This contradicts Lemma 5.1. □

Many well-known Boolean functions like and, or, majority, and parity are not dominated and thus cannot be privately computed on non-2-connected networks.

Theorem 5.5. *Let f be a non-degenerate n-ary Boolean function, $n \geq 5$, that is dominated by x with $bal_f(x) = k > 1$. Then f can be privately computed on $B_{n,k}$.*

Proof. The protocol works the same as the one presented in Section 1.2. Let f be of the form

$$f(x, y, z) = \begin{cases} g_0(y) & \text{if} \quad x = 0 \quad \text{and} \\ g_1(z) & \text{if} \quad x = 1 \end{cases}$$

for some g_0 and g_1. Then $f(x, y, z) = (\overline{x} \wedge g_0(y)) \vee (x \wedge g_1(z))$. Assume that y contains k variables and z contains $n - k - 1$ variables. Let the bridge player P_x, which is part of both components, hold x. We share the k variables of y among the k remaining nodes

of the first component and the $n-k-1$ variables of z among the remaining nodes of the second component. Then we privately compute $(\overline{x} \wedge g_0(y))$ within the first block and $(x \wedge g_1(z))$ within the second block. This can be done since both blocks consist of at least three nodes. (Every Boolean function can be privately computed on a complete network of at least three players [3] and henceforth on any 2-connected network with at least three players.) Finally, P_x knows the result.

It remains to prove the protocol is private. It is clearly private with respect to all players except for P_x, since no players needs to learn anything about $(\overline{x} \wedge g_0(y))$ or $(x \wedge g_1(z))$. Let $x = 0$ ($x = 1$ follows analogously due to symmetry). Then $(x \wedge g_1(z)) = 0$ and thus P_x does not learn anything about z. Furthermore, P_x only learns $(\overline{x} \wedge g_0(y)) = g_0(y)$ about y, which is just $f(x, y, z)$. □

Note that when the conditions of Theorem 5.5 are not fulfilled, we can always apply Theorem 5.4. Furthermore, there is no function on three or four variables that can be privately computed on a non-2-connected network: either the function is not dominated or the balance is one.

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 that 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 $(1-k/(n-1)) \cdot \min\{L, (k-2)/(k-1) \cdot (n^2-n) + L/(k-1)\}$ additional random bits, where L is the total number of bits sent in the original protocol. A future goal is either to reduce the number of additional random bits further 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. (Strictly speaking, an additional message per link is necessary to broadcast the result in the end. However, we do not need to use any random bits to encode this broadcast, hence we can assume that n bits are sent altogether.) Thus, $1+n-kn/(n-1) \leq n-k+1$ 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 $\lceil (n-2)/(k-1) \rceil - 1$ random bits. Furthermore, there exist k-connected networks for which this bound is tight.

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 characterised the functions that can be computed by a private protocol on non-2-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.

Acknowledgements

We thank Adi Rosén for fruitful discussions and hints to literature and Jan Arpe and the anonymous referees for valuable comments that helped improve the presentation.

References

[1] Noga Alon, Joel H. Spencer, and Paul Erdös. *The Probabilistic Method*, pages 223–232. Wiley, New York, 1992.

[2] Judit Bar-Ilan and Donald Beaver. Non-cryptographic fault-tolerant computing in a constant number of rounds of interaction. In *Proc. 8th Ann. ACM Symp. on Principles of Distributed Computing* (*PODC*), pages 201–209, 1989.

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

[4] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.

[5] Carlo Blundo, Alfredo de Santis, Giuseppe Persiano, and Ugo Vaccaro. Randomness complexity of private computation. *Comput. Complexity*, 8(2):145–168, 1999.

[6] Ran Canetti, Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. Randomness versus fault-tolerance. *J. Cryptology*, 13(1):107–142, 2000.

[7] Ran Canetti and Rafail Ostrovsky. Secure computation with honest-looking parties: What if nobody is truly honest? In *Proc. 31st Ann. ACM Symp. on Theory of Computing* (*STOC*), pages 255–264, 1999.

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

[9] Benny Chor, Mihály Geréb-Graus, and Eyal Kushilevitz. On the structure of the privacy hierarchy. *J. Cryptology*, 7(1):53–60, 1994.

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

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

[12] Benny Chor and Eyal Kushilevitz. A communication–privacy tradeoff for modular addition. *Inform. Process. Lett.*, 45(4):205–210, 1993.

[13] Yoshimi Egawa, Rainer Glas, and Stephen C. Locke. Cycles and paths through specified vertices in k-connected graphs. *J. Combin. Theory Ser. B*, 52:20–29, 1991.

[14] Matthew Franklin and Rebecca N. Wright. Secure communication in minimal connectivity models. *J. Cryptology*, 13(1):9–30, 2000.

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

[16] Anna Gál and Adi Rosén. A theorem on sensitivity and applications in private computation. *SIAM J. Comput.*, 31(5):1424–1437, 2002.

[17] Anna Gál and Adi Rosén. Lower bounds on the amount of randomness in private computation. In *Proc. 35th Ann. ACM Symp. on Theory of Computing* (*STOC*), pages 659–666, 2003.

[18] Michael R. Garey and David S. Johnson. *Computers and Intractability*: *A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, CA, 1979.

[19] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proc. 19th Ann. ACM Symp. on Theory of Computing* (*STOC*), pages 218–229, 1987.

[20] Frank Harary. *Graph Theory*. Addison-Wesley, Reading, MA, 1969.

[21] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[22] Joe Kilian. More general completeness theorems for secure two-party computation. In *Proc. 32nd Ann. ACM Symp. on Theory of Computing* (*STOC*), pages 316–324, 2000.

[23] Joe Kilian, Eyal Kushilevitz, Silvio Micali, and Rafail Ostrovsky. Reducibility and completeness in private computations. *SIAM J. Comput*., 29(4):1189–1208, 2000.
[24] Eyal Kushilevitz. Privacy and communication complexity. *SIAM J. Discrete Math*., 5(2):273–284, 1992.
[25] Eyal Kushilevitz and Yishay Mansour. Randomness in private computations. *SIAM J. Discrete Math*., 10(4):647–661, 1997.
[26] Eyal Kushilevitz, Rafail Ostrovsky, and Adi Rosén. Characterizing linear size circuits in terms of privacy. *J. Comput. System Sci*., 58(1):129–136, 1999.
[27] Eyal Kushilevitz and Adi Rosén. A randomness–rounds tradeoff in private computation. *SIAM J. Discrete Math*., 11(1):61–80, 1998.
[28] Andrew Chi-Chih Yao. Protocols for secure computations. In *Proc*. 23*rd Ann. IEEE Symp. on Foundations of Computer Science* (*FOCS*), pages 160–164, 1982.
[29] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proc*. 27*th Ann. IEEE Symp. on Foundations of Computer Science* (*FOCS*), pages 162–167, 1986.