Using Approximation Hardness to Achieve Dependable Computation*

Mike Burmester[1], Yvo Desmedt[1,2], Yongge Wang[3]

[1] Department of Mathematics, Royal Holloway – University of London, Egham, Surrey TW20 OEX, UK, m.burmester@rhbnc.ac.uk.
[2] Center for Cryptography, Computer and Network Security, Department of EE & CS, University of Wisconsin – Milwaukee, P.O. Box 784, WI 53201 Milwaukee, USA, desmedt@cs.uwm.edu.
[3] Department of EE & CS, University of Wisconsin – Milwaukee, P.O. Box 784, WI 53201 Milwaukee, USA, wang@cs.uwm.edu.

Abstract. Redundancy has been utilized to achieve fault tolerant computation and to achieve reliable communication in networks of processors. These techniques can only be extended to computations solely based on functions in one input in which redundant hardware or software (servers) are used to compute intermediate and end results. However, almost all practical computation systems consist of components which are based on computations with multiple inputs. Wang, Desmedt, and Burmester have used AND/OR graphs to model this scenario. Roughly speaking, an AND/OR graph is a directed graph with two types of vertices, labeled $\wedge$-*vertices* and $\vee$-*vertices*. In this case, processors which need all their inputs in order to operate could be represented by $\wedge$-vertices, whereas processors which can choose one of their "redundant" inputs could be represented by $\vee$-vertices. In this paper, using the results for hardness of approximation and optimization problems, we will design dependable computation systems which could defeat as many malicious faults as possible. Specifically, assuming certain approximation hardness result, we will construct k-connected AND/OR graphs which could defeat a ck-active adversary (therefore a ck-passive adversary also) where $c > 1$ is any given constant. This result improves a great deal on the results for the equivalent communication problems.

1 Introduction

Redundancy has been utilized to achieve reliability, for example to achieve fault tolerant computation and to achieve reliable communication in networks of processors. One of the primary objectives of a redundant computation system is

* Research supported by DARPA F30602-97-1-0205. However the views and conclusions contained in this paper are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advance Research Projects Agency (DARPA), the Air Force, of the US Government.

to tolerate as many faults (accidental or malicious) as possible. Hence, one of the crucial requirements in designing redundant computation systems is to use the least resources (redundancy) to achieve dependable computation against the most powerful adversaries. It has been proven (see, e.g., Hadzilacos [15], Dolev [10], Dolev, Dwork, Waarts, and Yung [11], and Beimel and Franklin [4]) that in the presence of a k-passive adversary (respectively k-active adversary) the processors in a network can communicate reliably if and only if the network is $k+1$-connected (respectively $2k+1$-connected).

All these works mentioned above assume processors with one type of input, while in practice it is often the case that processors need more than one type of inputs. For example, for the national traffic control system, we need data from the aviation, rail, highway, and aquatic vehicles, conduits, and support systems by which people and goods are moved from a point-of-origin to a destination point in order to support and complete matters of commerce, government operations, and personal affairs. In addition, each component of the traffic control system is again a system consisting of computations with multiple inputs, e.g., the processors of the aviation control system need data from several sources such as the airplane's speed, current position, etc., to determine the airplane's next position. Wang, Desmedt, and Burmester [22] have used AND/OR graphs to model this scenario. Originally AND/OR graphs have been used in the context of artificial intelligence to model problem solving processes (see [17]). Roughly speaking, an AND/OR graph is a directed graph with two types of vertices, labeled $\wedge$-*vertices* and $\vee$-*vertices*. The graph must have at least one input (source) vertex and one output (sink) vertex. In this case, processors which need all their inputs in order to operate could be represented by $\wedge$-vertices, whereas processors which can choose (using some kind of voting procedure) one of their "redundant" inputs could be represented by $\vee$-vertices. A solution graph, which describes a valid computation of the system, is a minimal subgraph of an AND/OR graph with the following properties: If an $\wedge$-vertex is in the solution graph then all of its incoming edges (and incident vertices) belong to the solution graph; If an $\vee$-vertex is in the solution graph then exactly one of its incoming edges (and the incident vertex) belongs to the solution graph. Wang, Desmedt, and Burmester [22] showed that it is **NP**-hard to find vertex disjoint solution graphs in an AND/OR graph (though there is a polynomial time algorithm for finding vertex disjoint paths in networks of processors with one type of inputs). This result shows that in order to achieve dependable computation, the computation systems (networks of processors) must be designed in such a way that it is easy for the honest stations/agents to find the redundant information in the systems. A similar analysis as for the case of networks of processors with one type of inputs shows that in the presence of a k-passive adversary (respectively k-active adversary) the computation system modeled by an AND/OR graph is dependable if and only if the underlying graph (that is, the AND/OR graph) is $k+1$-connected (respectively $2k+1$-connected) and both the input vertices and the output vertex know the set of vertex disjoint solution graphs in the AND/OR graph. Later in

this paper, we will use $G_{\wedge\vee}$ to denote AND/OR graphs and G to denote standard undirected graphs unless specified otherwise.

What happens if we want to tolerate more powerful adversaries? Adding more channel is costly, so we suggest a simpler solution: designing the AND/OR graph in such a way that it is hard for the adversary to find a vertex separator of the maximum set of vertex disjoint solution graphs (that is, find at least one vertex on each solution graph in the maximum set of vertex disjoint solution graphs), whence the adversary does not know which processors to block (or control). In order to achieve this purpose, we need some stronger results for approximation and optimization problems. There have been many results (see, e.g., [1, 21] for a survey) for hardness of approximating an **NP**-hard optimization problem within a factor c from "below". For example, it is hard to compute an independent set[1] V' of a graph $G(V, E)$ (note that here G is a graph in the standard sense instead of being an AND/OR graph) with the property that $|V'| \geq \frac{k}{c}$ for some given factor c, where k is the size of the maximum independent set of G. But for our problem, we are more concerned with approximating an **NP**-hard optimization problem from "above". For example, given a graph $G(V, E)$, how hard is it to compute a vertex set V' of G with $|V'| \leq ck$ such that V' *contains* an optimal independent set of G, where k is the size of the optimal independent set of G? We show that this kind of approximation problem is also **NP**-hard. Then we will use this result to design dependable computation systems such that with k redundant computation paths we can achieve dependable computation against a ck-active (Byzantine style) adversary (therefore against a ck-passive adversary also), where $c > 1$ is any given constant. This result improves a great deal on the equivalent communication problems (see our discussion on related works below).

The organization of this paper is as follows. We first prove in Section 2 the following result: For any given constant $c > 1$, it is **NP**-hard to compute a vertex set V' of a given graph $G(V, E)$ with the properties that $|V'| \leq ck$ and V' contains an optimal independent set of $G(V, E)$, where k is the size of the optimal independent set of $G(V, E)$. Section 3 surveys a model for fault tolerant computation and describes the general threats to dependable computation systems. In Section 4 we demonstrate how to use AND/OR graphs with trap-doors to achieve dependable computation against passive (and active) adversaries. In Section 5 we outline an approach to build AND/OR graphs with trap-doors. We conclude in Section 6 with remarks towards practical solutions and we present some open problems.

Related work

Achieving processor cooperation in the presence of faults is a major problem in distributed systems. Popular paradigms such as Byzantine agreement have been studied extensively. Dolev [10] (see also, Dolev, Dwork, Waarts, and Yung [11]) showed that a necessary condition for achieving Byzantine agreement is that the

[1] An independent set in a graph $G(V, E)$ is a subset V' of V such that no two vertices in V' are joined by an edge in E.

number of faulty processors in the system is less than one-half of the connectivity of the system's network (note that in order to achieve Byzantine agreement, one also needs that $n > 3k$ where n is the number of processors in the network and k is the number of faulty processors). Hadzilacos [15] has shown that even in the absence of malicious failures connectivity $k + 1$ is required to achieve agreement in the presence of k faulty processors. Beimel and Franklin [4] have shown that if authentication techniques are used, then Byzantine agreement is achievable only if the graph of the underlying network is $k+1$ connected and the union of the authentication graph and the graph of the underlying network is $2k + 1$ connected in the presence of k faulty processors. All these works assume processors with one type of inputs. Recently, Wang, Desmedt, and Burmester [22] have considered the problem of dependable computation with multiple inputs, that is, they considered the networks of processors where processors may have more than one type of inputs. While there is a polynomial time algorithm for finding vertex disjoint paths in networks of processors with one type of inputs, Wang, Desmedt, and Burmester's work shows that the equivalent problem in computation with multiple inputs is **NP**-hard.

Approximating an **NP**-hard optimization problem within a factor of $1 + \varepsilon$ means to compute solutions whose "cost" is within a multiplicative factor $1+\varepsilon$ of the cost of the optimal solution. Such solution would suffice in practice, if ε were close enough to 0. The question of approximability started receiving attention soon after **NP**-completeness was discovered [14, 20] (see [14] for a discussion). The most successful attempt was due to Papadimitriou and Yannakakis [18], who proved that MAX-3SAT (a problem defined by them) is complete for MAX-SNP (a complexity class defined by them), in other words, any approximability result for MAX-3SAT transfers automatically to a host of other problems. Among other results, they have shown that there is a constant $\varepsilon > 0$ such that it is **NP**-hard to compute a $\frac{k}{1+\varepsilon}$ size independent set of a given graph G, where k is the size of the maximum independent set of G. The results in [18] have been improved by many other authors, especially, after the emergence of the PCP theorem [2, 3], that is, $PCP(\log n, 1) = \mathbf{NP}$ (for a survey, see, e.g., [1, 21]). For example, Arora, Lund, Motwani, Sudan, and Szegedy have shown that it is **NP**-hard to n^{δ}-approximate an independent set for some $\delta > 0$. However, all these results are related to approximating the independent set from "below", that is, to compute an independence set whose size is smaller than the optimal independent set. We will show that it is easy to convert these results to the results of hardness of approximating an independent set from "above" instead of from "below" as done in [1, 21], that is, it is hard to delete some vertices from a given graph such that the resulting graph contains an optimal independent set of the original graph.

2 Optimization and approximation

In this section we present some graph theoretic results which will be used in later sections. First we remind the reader of the graphs defined in the transformation

from 3SAT to Vertex Cover[2] in Garey and Johnson [14, pp. 54–56] and we give such kind of graphs a special name.

Definition 1. *Let n and m be two positive integers. A graph $G(V, E)$ is called an* $n+m$-SAT-graph *if there are $n+m$ subgraphs $L_1(V_{L_1}, E_{L_1})$, ..., $L_n(V_{L_n}, E_{L_n})$, $T_1(V_{T_1}, E_{T_1})$, ..., $T_m(V_{T_m}, E_{T_m})$ of G with the following properties:*

1. *$V = (\cup_{i=1}^{n} V_{L_i}) \cup (\cup_{i=1}^{m} V_{T_i})$.*
2. *For each $i \leq n$, $|V_{L_i}| = 2$ and E_{L_i} consists of the one edge connecting the two vertices in V_{L_i}.*
3. *For each $i \leq m$, T_i is a triangle, which is isomorphic to the undirected graph $T = (V_T, E_T)$ where $V_T = \{v_1, v_2, v_3\}$ and $E_T = \{(v_1, v_2), (v_2, v_3), (v_3, v_1)\}$.*
4. *There is a function $f : (\cup_{i=1}^{m} V_{T_i}) \to (\cup_{i=1}^{n} V_{L_i})$ such that the edge set of G is $E = (\cup_{i=1}^{n} E_{L_i}) \cup (\cup_{i=1}^{m} E_{T_i}) \cup \{(v, f(v)) : v \in \cup_{i=1}^{m} V_{T_i}\}$.*

The following results are straightforward from the definitions.

Lemma 1. *Given an $n+m$-SAT-graph $G(V, E)$, the following conditions hold.*

1. *The size of an independent set of G is at most $n+m$.*
2. *The size of a vertex cover of G is at least $n+2m$.*

The following result is proved in [14, pp. 54–56].

Lemma 2. *(see [14]) Given a 3SAT formula C with n variables and m clauses, there is an $n+m$-SAT-graph $G(V, E)$ with the following properties:*

1. *C is satisfiable if and only if there is an independent set of size $n+m$ in $G(V, E)$.*
2. *C is satisfiable if and only if there is vertex cover of size $n+2m$ in $G(V, E)$.*

Corollary 1. *It is* **NP***-hard to decide whether there is an independent set of size $n+m$ in an $n+m$-SAT-graph.*

In addition to the problem of deciding whether an $n+m$-SAT-graph has an independent set of size $n+m$, we are also interested in the following approximation problem: for some constant $\varepsilon > 0$ and each $n+m$-SAT-graph G, can we compute in polynomial time an independent set of size $k/(1+\varepsilon)$ in G, where k is the size of the maximum independent set of G? Papadimitriou and Yannakakis [18] (see also, [13, 2]) have proved the following result (note that their original result is for general graphs though their proof is for $n+m$-SAT-graphs).

Definition 2. *For a rational number $\varepsilon > 0$, an algorithm is said to* compute $(1+\varepsilon)$-approximation *to the maximum independent set if given any graph G its output is an independent set of G with size at least $k/(1+\varepsilon)$ where k is the size of the maximum independent set of G.*

[2] A vertex cover of a graph $G(V, E)$ is a subset V' of V such that every edge in E is incident to a vertex in V'.

Theorem 1. *(see [18, 13]) There is a constant $\varepsilon > 0$ such that approximating an independent set of an $n + m$-SAT-graph $G(V, E)$ within a factor $1 + \varepsilon$ is* **NP***-hard.*

Arora, Lund, Motwani, Sudan, and Szegedy [2] have proved the following stronger result.

Theorem 2. *(see [2]) There is a constant $\delta > 0$ such that approximating an independent set of a graph G within a factor n^δ is* **NP***-hard, where n is the number of vertices in G.*

Note that Theorem 2 is only for general graphs. The following variants of Theorems 1 and 2 are useful for our discussions.

Theorem 3. *(see [18, 13]) There is a constant $\varepsilon > 0$ and a polynomial time algorithm to construct for each 3SAT clause C an $n + m$-SAT-graph G with the following properties:*

1. *If C is satisfiable, then G has an independent set of size $n + m$.*
2. *If C is not satisfiable, then $k < \frac{n+m}{1+\varepsilon}$, where k is the size of the maximum independent set in G.*

Proof. It follows from the proof of Theorem 1. □

Theorem 4. *(see [2, 5]) There is a constant $\delta > 0$ and a series of pairs of positive integers $(s_1, c_1), (s_2, c_2), \ldots$ such that $\frac{c_n}{s_n} \geq n^\delta$ for large enough n and from each 3SAT clause C we can construct in a polynomial time a graph G with the following properties:*

1. *If C is satisfiable, then $k \geq c_n$, where k is the size of the maximum independent set in G and n is the number of vertices in G.*
2. *If C is not satisfiable, then $k \leq s_n$, where k is the size of the maximum independent set in G and n is the number of vertices in G.*

Proof. It follows from the proof of Theorem 2. □

Given a graph $G(V, E)$, an edge set $E' \subseteq E$ is said to be *independence eligible* if there is an independent set $V' = \{u :$ there is a $v \in V$ such that the unordered pair $(u, v) \in E'\}$ of size $|E'|$ in G. Note that given an independence eligible edge set E', it is easy to compute an independent set of size $|E'|$ (by a standard algorithm of computing a satisfying assignment of a 2SAT formula).

Theorem 5. *Let ε be the constant in Theorem 3. Then it is* **NP***-hard to compute an edge set E' of a given $n + m$-SAT-graph G with the following properties:*

1. *$|E'| \leq (1 + \varepsilon)k$, where k is the size of a maximum independent set of G.*
2. *E' contains an independence eligible edge set E'' such that $|E''| = k$.*

Proof. It follows from Theorem 3. □

Theorem 6. *There is a constant $\varepsilon > 0$ such that it is* **NP***-hard to compute an edge set $E' \subseteq E$ of a graph $G(V, E)$, with the following properties:*

1. *$|E'| \leq kn^{\varepsilon}$, where k is the size of the maximum independent set of G and $n = |V|$.*
2. *E' contains an independence eligible edge set E'' such that $|E''| \geq \frac{k}{2}$.*

Proof. Let s_n, c_n and δ be the constants in Theorem 4. And let $\varepsilon = \frac{\delta}{2}$. We reduce the **NP**-complete problem 3SAT to the problem of this Theorem. For each 3SAT formula C, construct a graph $G(V, E)$ satisfying the conditions of Theorem 4. Let E' be an edge set satisfying the conditions of the Theorem. Then it suffices to show that if $|E'| \geq \frac{c_n}{2}$ then $k \geq c_n$ (therefore C is satisfiable) else $k \leq s_n$ (therefore C is not satisfiable). If $|E'| \geq \frac{c_n}{2}$ then, by the condition that $|E'| \leq kn^{\varepsilon}$, we have $\frac{c_n}{2} \leq kn^{\varepsilon}$. That is,

$$k \geq \frac{c_n}{2n^{\varepsilon}} > \frac{c_n}{n^{\delta}} \geq s_n.$$

Whence $k \geq c_n$. Otherwise $|E'| < \frac{c_n}{2}$, and

$$\frac{k}{2} \leq |E''| \leq |E'| < \frac{c_n}{2}.$$

That is, $k < c_n$. Whence $k \leq s_n$. □

Corollary 2. *There is a constant $\varepsilon > 0$ such that it is* **NP***-hard to compute a vertex set $V' \subseteq V$ of a graph $G(V, E)$ with the following properties:*

1. *$|V'| \leq kn^{\varepsilon}$, where k is the size of the maximum independent set of G and $n = |V|$.*
2. *V' contains an independent set V'' of $G(V, E)$ such that $|V''| \geq \frac{k}{2}$.*

3 General threats and models for dependable computations

General threats A simple attack to defend against is of a restricted adversary (called *passive adversary*) who is allowed only to monitor communication channels and to jam (denial of service) several processors in the computation system, but is *not* allowed to infiltrate/monitor the internal contents of any processor of the computation system. Of course, a more realistic adversary is the *active adversary* (Byzantine faults) that can monitor all communication between processors and which in addition is also trying to infiltrate the internal contents of several processors.

A passive adversary with the power of jamming up to k processors is called a *k-passive adversary*. An active adversary (Byzantine faults) may mount a more sophisticated attack, where he manages to comprise the security of several internal processors of the system, whereby he is now not only capable of monitoring the external traffic pattern and capable of jamming several processors but is also

capable of examining and modifying every message and data (that is, creating bogus messages and data) which passes through (or stored at) these infiltrated processors. Thus, we define a *k-active adversary*, an adversary that can monitor all the communication lines between processors and also manages to examine and to modify the internal contents of up to k processors of the system. (Similar definitions were considered in the literature, see, for example [7, 8, 12, 19] and references therein).

Achieving processor cooperation in the presence of faults is a major problem in distributed systems, and has been studied extensively (see, e.g, [4, 10, 11, 15]). All these works assume processors with one type of inputs. Recently, Wang, Desmedt, and Burmester [22] have considered the problem of dependable computation with multiple inputs, that is, they considered the networks of processors where processors may have more than one type of inputs. While there is a polynomial time algorithm for finding vertex disjoint paths in networks of processors with one type of inputs, Wang, Desmedt, and Burmester's work shows that the equivalent problem in computation with multiple inputs is **NP**-hard. In this paper, we will consider redundant computation systems with multiple inputs which can be modeled by AND/OR graphs which we now briefly survey.

Definition 3. *(see [22]) An* AND/OR graph $G_{\wedge\vee}(V_\wedge, V_\vee, INPUT, output; E)$ *is a directed graph with a set* $V_\wedge$ *of* $\wedge$-vertices*, a set* $V_\vee$ *of* $\vee$-vertices*, a set* $INPUT$ *of* input vertices*, an* output vertex $output \in V_\vee$*, and a set of* directed edges E*. The vertices without incoming edge are input vertices and the vertex without outgoing edge is the output vertex.*

It should be noted that the above definition of AND/OR graphs is different from the standard definition in artificial intelligence (see, e.g., [17]), in that the directions of the edges are opposite. The reason is that we want to use the AND/OR graphs to model redundant computation systems.

Assume that we use the AND/OR graph to model a fault tolerant computation. So, information (for example, mobile codes) must flow from the input vertices to the output vertex. And a valid computation in an AND/OR graph can be described by a *solution graph* (the exact definition will be given below). However, if insider vertices may be faulty or even malicious, then the output vertex cannot trust that the result is authentic or correct. Firstly we assume that there is only one k-passive adversary at any specific time. The theory of fault tolerant computation (see, Hadzilacos [15]), trivially adapted to the AND/OR graph model, tells us that if there are $k+1$ vertex disjoint paths (solution graphs) of information flow in the AND/OR graph then the vertex *output* will always succeed in getting at least one copy of the results. Secondly we assume that there is one k-active adversary at any specific time. Then the theory of fault tolerant computation (see, e.g., Dolev [10], Dolev et al. [11], and Beimel and Franklin [4]) tells us that if there are $2k+1$ vertex disjoint paths (solution graphs) of information flow in the AND/OR graph then the vertex *output* will always succeed in getting at least $k+1$ identical results computed from the input vertices through vertex disjoint solution graphs, if *output* knows the layout of the graph. This

implies that if *output* knows the layout of the graph then it can use a majority vote to decide whether the result is correct or not. It follows that in order to achieve dependable computation with redundancy, it is necessary to find a set of vertex disjoint solution graphs in a given AND/OR graph.

Definition 4. *(see [22]) Let $G_{\wedge\vee}(V_\wedge, V_\vee, INPUT, output; E)$ be an AND/OR graph. A* solution graph *$P = (V_P, E_P)$ is a minimum subgraph of $G_{\wedge\vee}$ satisfying the following conditions.*

1. *$output \in V_P$.*
2. *For each $\wedge$-vertex $v \in V_P$, all incoming edges of v in E belong to E_P.*
3. *For each $\vee$-vertex $v \in V_P$, there is exactly one incoming edge of v in E_P.*
4. *There is a sequence of vertices $v_1, \ldots, v_n \in V_P$ such that $v_1 \in INPUT, v_n = output$, and $(v_i \rightarrow v_{i+1}) \in E_P$ for each $i < n$.*

Moreover, two solution graphs P_1 and P_2 are vertex disjoint *if $(V_{P_1} \cap V_{P_2}) \subseteq (INPUT \cup \{output\})$. An AND/OR graph is called* k-connected *if the following conditions are satisfied.*

1. *There are k vertex disjoint solution graphs in $G_{\wedge\vee}$.*
2. *There do not exist $k+1$ vertex disjiont solution graphs in $G_{\wedge\vee}$.*

In order for an adversary to attack the computation system, s/he does not need to find all vertex disjoint solution graphs in an AND/OR graph. For a passive adversary, s/he can choose to jam one vertex on each solution graph to corrupt the system. An active adversary needs to find one half of the vertices of a *vertex separator* (defined in the following).

Definition 5. *Let $G_{\wedge\vee}$ be a k-connected AND/OR graph, and $P = \{P_1, \ldots, P_k\}$ be a maximum set of vertex disjoint solution graphs in $G_{\wedge\vee}$. A set $S = \{v_1, \ldots, v_k\}$ of vertices in $G_{\wedge\vee}$ is called a* vertex separator *of P if for each solution graph $P_i \in P$ $(i = 1, \ldots, k)$, $v_i \in V_{P_i}$.*

Remark: The problem of finding a vertex separator in an AND/OR graph is **NP**-hard which will be proved in Section 5.

The question we are addressing in this paper is how to design AND/OR graphs with less vertex disjoint solution graphs to achieve dependable computation against more powerful passive or active adversaries.

4 Dependable computation with trap-doors

In this section, we show how to design dependable computation systems with trap-doors such that the following condition is satisfied:

- The computation system modeled by a k-connected AND/OR graph is robust against a k'-active adversary (therefore robust against a k'-passive adversary also) where $k' \leq ck$ and $c > 1$ is any given constant.

The idea is to use the fact that it is **NP**-hard to approximate a vertex separator of an AND/OR graph from "above" (see Section 2 for details about approximating an **NP**-hard optimization problem from "above"). It follows that if one designs the AND/OR graph in such a way that the trusted participants can easily find vertex disjoint solution graphs in it (using some trap-doors), and the input vertices always initiate a computation through all solution graphs in the maximum set of vertex disjoint solution graphs, then dependable computation is possible. The benefit from using trap-doors in a computation system with multiple inputs is obvious. If we do not use trap-doors then, by extending the conventional fault tolerant computation theory (see, e.g., [4, 7, 10, 11, 15]), a k-connected AND/OR graph is only robust against k'-passive adversaries and only robust against k''-active adversaries respectively, when $k' < k$ and $k'' < \frac{k}{2}$. Since if the adversary has the power to jam k vertices in the AND/OR graph and s/he can find a vertex separator of size k, then s/he can jam all of the vertices in the vertex separator and corrupt the system. Indeed, if the adversary has the power to examine and modify messages and data in $\lfloor \frac{k}{2} \rfloor + 1$ processors, then the adversary may let the $\lfloor \frac{k}{2} \rfloor + 1$ faulty processors create and send faulty messages to the output processor claiming that they come from some bogus solution graphs. This will convince the output vertex to accept the bogus message since the majority messages are faulty. However, if we use trap-doors in the design of AND/OR graphs, then with high probability, a k-connected AND/OR graph is robust against k'-active adversaries (therefore against k'-passive adversaries) where $k' \leq ck$ and $c > 1$ is any given constant. The reason is that even though the adversary has the power to jam or control $k' > k$ vertices in the AND/OR graph, he does not know which vertices to corrupt, that is, the corrupted vertices (in his control) will appear on at least half of the k vertex disjoint solution graphs.

So one of the main problems is to design AND/OR graphs in which it is hard on the average case to approximate at least one half of a vertex separator from "above". In Section 5, we will outline an approach to generate such kind of AND/OR graphs. In the remaining part of this section we will demonstrate how to use these AND/OR graphs to achieve dependability.

Protocol I

1. Alice generates a k-connected AND/OR graph $G_{\wedge\vee}$ such that the graph $G_{\wedge\vee}$ can implement the desired computation and such that finding a ck size set of vertices which contains at least one half of the elements of a vertex separator is hard, where $c > 1$ is any given constant. (The details will be presented in Section 5).
2. Using a secure channel, Alice sends the input vertices the method of initiating a computation and sends the output vertex a maximum set of vertex disjoint solution graphs in $G_{\wedge\vee}$.
3. In order to carry out one computation, Alice initiates the computation through all solution graphs in the maximum set of vertex disjoint solution graphs.

4. When the output vertex gets all possible outputs, he compares the results from the k vertex disjoint solution graphs (note that the output vertex knows the maximum set of vertex disjoint solution graphs) and chooses the authentic result using a majority vote.

Note that our above protocol is not secure against a dynamic adversary who after observing one computation will change the vertices he controls. Indeed, it is an interesting open problem to design protocols which are secure against dynamic adversaries.

Now assume that Mallory is a k'-active adversary (or a k'-passive adversary) where $k' \leq ck$ for the constant $c > 1$, and $P = \{P_1, \ldots, P_k\}$ is a maximum set of vertex disjoint solution graphs in the AND/OR graph used in **Protocol I**. Since Mallory does not know how to find a k' size set of vertices which contains at least one half of the elements of a vertex separator for P (finding such a set is very hard), she does not know which vertices to corrupt so that she can generate at least $\lfloor \frac{k}{2} \rfloor + 1$ bogus messages to convince the output vertex to accept (or so that all these k solution graphs will be jammed), even though she has the power to corrupt $k' = ck$ vertices. It follows that the system is robust against a k'-active adversary (therefore robust against a k'-passive adversary also) where $k' \leq ck$.

5 AND/OR graphs with trap-doors

In this section, we outline an approach for constructing AND/OR graphs with trap-doors. We first show that it is **NP**-hard to approximate at least half of the elements of a vertex separator of an AND/OR graph from "above".

Theorem 7. *Given an AND/OR graph $G_{\wedge\vee}(V_\wedge, V_\vee, INPUT, output; E)$, it is* **NP***-hard to compute a vertex set $S' \subseteq (V_\wedge \cup V_\vee)$ with the following properties:*

1. *If $G_{\wedge\vee}$ is k-connected then $|S'| \leq ck$.*
2. *For some vertex separator S of $G_{\wedge\vee}$, $|S \cap S'| \geq \frac{k}{2}$.*

Proof. We reduce the problem of Theorem 6 to the problem of this Theorem.

For a given graph $G'(V', E')$, we construct an AND/OR graph $G''_{\wedge\vee}(V''_\wedge, V''_\vee, INPUT'', output''; E'')$ as follows. Assume that $V' = \{v_1, \ldots, v_n\}$. Let $INPUT'' = \{I_i, I_{i,j} : i, j = 1, \ldots n\}$, $V''_\vee = \{output\}$, $V''_\wedge = \{u_{i,j} : i, j = 1, \ldots n\} \cup \{u_i : i = 1, \ldots, n\}$, and E'' be the set of the following edges.

1. For each $i = 1, \ldots, n$, there is an edge $I_i \rightarrow u_i$.
2. For each pair $i, j = 1, \ldots, n$, there is an edge $I_{i,j} \rightarrow u_{i,j}$.
3. For each pair $i, j = 1, \ldots, n$, such that $(v_i, v_j) \in E'$, there are four edges $u_{i,j} \rightarrow u_i$, $u_{i,j} \rightarrow u_j$, $u_{j,i} \rightarrow u_i$, and $u_{j,i} \rightarrow u_j$.
4. For each i, there is an edge $u_i \rightarrow output''$.

It is clear that two solution graphs P_1 and P_2 in $G''_{\wedge\vee}$ which go through u_i and u_j respectively are vertex disjoint if and only if there is no edge (v_i, v_j) in E'. Hence there is a size k independent set in G' if and only if there are k vertex

disjoint solution graphs in $G''_{\wedge\vee}$. And from k vertex disjoint solution graphs in $G''_{\wedge\vee}$ one can compute in linear time a size k independent set in G'. Whence it is sufficient to show that from each vertex set S' satisfying the conditions of the Theorem, one can compute in polynomial time an edge set $E_{S'} \subseteq E'$ with the following properties:

1. If $G_{\wedge\vee}$ is k-connected (that is, if the optimal independent set in G' has size k) then $|E_{S'}| \leq ck$.
2. $E_{S'}$ contains an independence eligible edge set of size at least $\frac{k}{2}$.

The following algorithm will output an edge set $E_{S'}$ with the above properties. In the following S' is the vertex set satisfying the conditions of the Theorem.

- Let $E_{S'} = \emptyset$. For $i = 1, \ldots, ck$, we distinguish the following two cases:
  1. $s_i = u_j$ for some $j \leq n$. Let $E_{S'} = E_{S'} \cup \{(v_j, v'_j)\}$ where v'_j is any vertex in G' which is incident to v_j.
  2. $s_i = u_{j_1,j_2}$ for some $j_1, j_2 \leq n$. Let $E_{S'} = E_{S'} \cup \{(v_{j_1}, v_{j_2})\}$ if $(v_{j_1}, v_{j_2}) \in E'$ and $E_{S'} = E_{S'}$ otherwise.

By the property of S', it is clear that $E_{S'}$ has the required properties.

By Theorem 6, we have completed the proof of the Theorem. □

In the remaining part of this section, we outline how to construct AND/OR graphs with trap-doors.

Construction First generate a graph $G'(V', E')$ and a number k which satisfy the conditions of Theorem 3 (or Theorem 4). Secondly use the method in the proof of Theorem 7 to generate an AND/OR graph $G''_{\wedge\vee}$ with the property that it is hard to approximate at least half of the elements of a vertex separator of $G''_{\wedge\vee}$ from "above". The AND/OR graph $G_{\wedge\vee}$ is obtained by replacing all vertices $u_{i,j}$ of $G''_{\wedge\vee}$ with the AND/OR graph $G^1_{\wedge\vee}$, where $G^1_{\wedge\vee}$ is the AND/OR graph which can implement the desired computation. As a summary, the construction proceeds as follows.

$$\text{graph } G' \rightarrow \text{AND/OR graph } G''_{\wedge\vee} \overset{G^1_{\wedge\vee}}{\rightarrow} \text{AND/OR graph } G_{\wedge\vee}$$

6 Towards practical solutions

In the previous section, we considered the problem of designing AND/OR graphs with trap-doors. Specifically, we constructed AND/OR graphs which is robust against ck-active adversaries (therefore robust against ck-passive adversaries also). However, these constructions are inefficient and are only of theoretical interests. One of the most interesting open questions is how to efficiently generate hard instances of AND/OR graphs, especially, for arbitrary number k. If we do not require that c be an arbitrary given constant, then Theorem 5 can be used to construct AND/OR graphs which are more "efficient" (though still have

enormous complexity) than the AND/OR graphs constructed in the previous section and which are robust against $(1+\varepsilon)k$-passive adversaries where $\varepsilon < 1$ is a small positive rational number. However, in order to construct AND/OR graphs which are robust against ck-active adversaries for $c > \frac{1}{2}$, we have to use Theorem 6 in our construction. And the size of the graph G in Theorem 6 will be impractical if we want to make the security of the system to be at least as hard as an exhaustive search of a 1024-bit space.

We should also note that, in order to construct the AND/OR graphs in the previous section, we need to construct standard graphs which satisfy the conditions of Theorem 3 (or Theorem 4). That is, we need an algorithm to build graphs whose independent sets are hard to approximate in the average case (note that Theorem 7 only guarantees the worst-case hardness instead of average-case hardness). Whence it is interesting (and open) to prove some average-case hardness results for the corresponding problems.

In the following, we consider the problem of constructing practical average-case hard AND/OR graphs which are robust against $k + c$-passive adversaries, where c is some given constant. Our following construction is based on the hardness of factoring a large integer and we will not use the approximation hardness results.

Construction Let N be a large number which is a product of two primes p and q. We will construct an AND/OR graph $G_{\wedge\vee}$ with the following property: given the number N and a vertex separator for $G_{\wedge\vee}$, one can compute efficiently the two factors p and q. Let $x_1, \ldots, x_t$ and $y_1, \ldots, y_t$ be variables which take values 0 and 1, where $t = \lfloor \log N \rfloor$. And let $(x_t \ldots x_1)_2$ and $(y_t \ldots y_1)_2$ to denote the binary representations of $\sum x_i 2^{i-1}$ and $\sum y_i 2^{i-1}$ respectively. Then use the relation

$$(x_t \ldots x_1)_2 \times (y_t \ldots y_1)_2 = N \tag{1}$$

to construct a 3SAT formula C with the following properties:

1. C has at most $O(t^2)$ clauses.
2. C is satisfiable and, from a satisfying assignment of C, one can compute in linear time a assignment of $x_1, \ldots, x_t, y_1, \ldots, y_t$ such that the equation (1) is satisfied. That is, from a satisfying assignment of C, one can factor N easily.

Now use Lemma 2 to construct an $n+m$-SAT-graph $G'(V', E')$ and a number $k = O(t^2)$ with the property that: from a size k independent set of G' one can compute in linear time a satisfying assignment of C. Lastly, use the method in the proof of Theorem 7 to generate an AND/OR graph $G_{\wedge\vee}$ with the property that, from a vertex separator of $G_{\wedge\vee}$, one can compute in linear time a size k independent set of G' (note that, instead of approximating a vertex separator, here we need to know a whole set of vertex separator). As in the proof of Theorem 7, from a vertex separator of $G_{\wedge\vee}$ one can easily compute a size k independence eligible edge set of G', from which one can compute in linear time a size k independent set of G' (using the method of computing a satisfying assignment of a 2SAT formula).

It is straightforward to see that the above constructed AND/OR graph $G_{\wedge\vee}$ is robust against $k + c$-passive adversaries if factoring N is hard, where c is any given constant.

References

1. S. Arora. *Probabilistic Checking of Proofs and Hardness of Approximation Problems*. PhD Thesis, CS Division, UC Berkeley, August, 1994.
2. S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. In: *Proceedings of 33rd IEEE Symposium on Foundations of Computer Science*, pp. 13–22, 1992.
3. S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of **NP**. In: *Proceedings of 33rd IEEE Symposium on Foundations of Computer Science*, pp. 2–13, 1992.
4. A. Beimel and M. Franklin. Reliable communication over partially authenticated networks. In. *Proceedings of the WDAG '97, Lecture Notes in Computer Science 1320*, pp. 245–259, Springer Verlag, 1997.
5. M. Bellare. Proof checking and approximation: towards tight results. In: *Complexity Theory Column 12, SIGACT News*. **27**(1), March 1996.
6. R. Boppana and M. Halldorsson. Approximating maximum independent sets by excluding subgraphs. *BIT*, **32**(2), pp. 180–196, 1992.
7. M. Burmester, Y. Desmedt, and G. Kabatianski. Trust and security: a new look at the Byzantine general problems. In: R. N. Wright and P. G. Neumann, Eds, *Network Threats, DIMACS, Series in Discrete Mathematics and Theoretical Computer Science,* AMS, Vol. 38, 1997.
8. R. Canetti, E. Kushilevitz, R. Ostrovsky, and A. Rosen. Randomness vs. fault-tolerance. In: *Proceedings of the PODC '97*, pp. 35–44, Springer Verlag, 1997.
9. D. Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. *Communication of the ACM*, Vol. 24, pp. 84–88, 1981.
10. D. Dolev. The Byzantine generals strike again. *J. of Algorithms*, **3**, pp. 14–30, 1982.
11. D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *J. of the ACM*, **40**(1), pp. 17–47, 1993.
12. S. Dolev and R. Ostrovsky. Efficient anonymous multicast and reception. In: Advances in Cryptology, Crypto'97, pp. 395–409, Springer Verlag, 1997.
13. U. Feige, S. Goldwasser, L. Lovasz, S. Safra, and M. Szegedy. Approximating clique is almost **NP**-complete. In: *Proceedings of 32nd IEEE Symposium on Foundations of Computer Science*, pp. 2–11, 1991.
14. M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of* **NP***-Completeness*. W. H. Freeman and Company, San Francisco, 1979.
15. V. Hadzilacos. *Issues of Fault Tolerance in Concurrent Computations*. PhD thesis, Harvard University, Cambridge, MA, 1984.
16. L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *J. of the ACM*, **32**(2), pp. 374–382, 1982.
17. N. J. Nilsson. *Principles of Artificial Intelligence*. Tioga, 1980.
18. C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences*, **43**, 425–440, 1991.
19. C. Rackoff and S. Simon. Cryptographic defense against traffic analysis. In: *Proceedings of the STOC 93*, pp. 672–681.

20. S. Sahni and T. Gonzalez. **P**-complete approximation problems. *J. of the ACM*, **23**, pp. 555–565, 1976.
21. M. Sudan. *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems.* PhD. thesis, U. C. Berkeley, 1992.
22. Y. Wang, Y. Desmedt, and M. Burmester. Hardness of dependable computation with multiple inputs. *Submitted.*