Constructing Disjoint Paths for Secure Communication

Amitabha Bagchi, Amitabh Chaudhary, Michael T. Goodrich, and Shouhuai Xu

Dept. of Information & Computer Science,
University of California,
Irvine, CA 92697-3425, USA
{bagchi,amic,goodrich,shxu}@ics.uci.edu

Abstract. We propose a bandwidth-efficient algorithmic solution for perfectly-secure communication in the absence of secure infrastructure. Our solution involves connecting vertex pairs by a set of k edge-disjoint paths (a structure we call a k-*system*) where k is a parameter determined by the connectivity of the network. This structure is resilient to adversaries with bounded eavesdropping capability. To ensure that bandwidth is efficiently used we consider connection requests as inputs to the k-Edge Disjoint Path Coloring Problem (k-EDPCOL), a generalization of the Path Coloring Problem, in which each vertex pair is connected by a k-system, and each k-system is assigned a color such that two overlapping k-systems do not have the same color. The objective is to minimize the number of colors. We give a distributed and competitive online algorithm for k-EDPCOL. Additionally, since security applications are our focus we prove that a malicious adversary which attacks the algorithm during the process of construction of a k-system cannot learn anything more than if it had attacked the k-system once it was built.

1 Introduction

Secure communication over a public network is one of the most important issues in deploying distributed computing. Various solutions have been proposed to this problem, yet those solutions typically assume the existence of a trusted third party or a public key infrastructure. For large-scale applications or Internet-wide distributed computing this assumption about the existence of either a trusted third party or a public key infrastructure often lead to both technical and non-technical complications. Therefore alternatives that facilitate secure communications without relying on such assumptions could potentially be extremely useful. In this paper, we explore one such alternative and show that network properties can be used to provide infrastructureless secure communications.

In a pioneering work, Dolev et. al. [14] showed how high connectivity can be used for perfectly secure transmission in the presence of bounded adversaries. In that work the authors used the notion of *wires*: disjoint paths connecting sender to receiver. The simple expedient of breaking up data into several shares and sending them along these disjoint paths makes it difficult for an adversary with bounded eavesdropping capability to intercept a transmission or tamper with it. They established bounds on the number of wires required for the kind of security they wished to provide, and described protocols using those wires. They did not, however, make any attempt to actually *construct* these wires. The main contribution of our paper is in formalizing the algorithmic setting for this construction.

We define a k-*system*: a set of k edge disjoint paths joining a vertex pair. In realizing the wires of [14] we give an algorithm for constructing k-systems between requesting pairs.

We use the theoretical abstraction of a uniform capacity optical network. Each edge has uniform capacity divided into a number of wavelengths, also known as *colors* in routing terminology. A path from source to sink is routed using the same wavelength on all the edges it traverses. See [29] for an elaboration of this setting. For simplicity of analysis we require that all the k paths of a given k-system use the same wavelength.

To ensure that each connection gets enough bandwidth we ensure that the total number of wavelengths sharing the edges' capacity is minimized. We define a generalization of the Minimum Path Coloring Problem (MPCP) [29] called k-EDPCOL (in Section 4) for this purpose and build k-systems to provide a competitive and distributed solution to this problem.

We ensure that the communication complexity of constructing these k-systems is as low as possible. Given that the applications we propose for our k-systems are in the realm of security, we show that a disruptive adversary operating during the running of our algorithm will not be able to learn more than if it had attacked a correctly constructed k-system.

The second contribution of this paper is to show the applications of k-systems to secure communications. Our k-systems can be used to realize the wires required in [14] so that perfectly secure communications can be achieved. In [14], in fact, the wires refer to *vertex disjoint paths*, to allow processor as well as edge faults. Our k-systems, however, work when the faults are limited to edges.[1]

As another application to allow computationally secure communications, we introduce the notion, and construction, of *Identity Based Key Distribution* (IBKD). An IBKD scheme allows a pair of parties to conduct secure communications without relying on any trusted third party or public key infrastructure.

Organization of the paper. In the next section we formally define paths systems and our adversarial models. In Section 3 we discuss how the k-systems produced by our algorithm can be used for secure key distribution and secure transmission. In Section 4 we provide some terminology and definitions and provide a context for our work. In Section 5 we outline the algorithm and analyze it. Finally we conclude in Section 6 with a discussion of the issues raised by our work and the open problems remaining.

[1] Note that finding vertex disjoint paths in graphs is considered to be a significantly harder problem than finding edge disjoint paths. See [22] and [23] for details.

2 Paths Systems and Adversaries

Since the connectivity of the underlying network is a foundational aspect of our work we begin this section by defining it formally:

Definition 1. *An* edge cut *of a graph G is a set $S \subseteq E(G)$ such that $G' = (V, E \backslash S)$ is not connected. A graph G is k-*connected *if every edge cut has at least k edges. The* connectivity *of G is the maximum k such that G is k-connected.*

Path Systems and k-systems. For a given sender s and receiver t, a *path system* is a set $\mathcal{P}$ of paths between s and t in the underlying communication network. A *k-path system* is a path system with $|\mathcal{P}| = k$. If all the paths in $\mathcal{P}$ are edge disjoint, we call it an *edge disjoint path system.* A *k-edge disjoint path system* is an edge disjoint path system with $|\mathcal{P}| = k$. In the following we refer to a k-edge disjoint path system simply as a *k-system.* Our focus in this paper is on algorithms for construction of k-systems.

Adversaries and Compromised paths. An *α-passive adversary*, also called a *listening* or *eavesdropping* adversary, is one that listens to messages on a fixed set L of edges in the communication network with $|L| \leq \alpha$. The objective of a passive adversary is to *learn* the confidential communication between s and t.[2]

An *(α, β)-active adversary*, also called a *malicious* adversary, is one that listens to messages on a fixed set L of edges, and in addition, disrupts, alters, and generates messages on a fixed set $D \subseteq L$ of edges in the communication network, with $|L| \leq \alpha$ and $|D| \leq \beta$. An active adversary may behave arbitrarily in both phases: the establishment of a k-system, and the future communication over it. The objective of an active adversary is, in addition to learning any confidential communication, to prevent t from correctly receiving confidential communication from s.

A *compromised edge* is an edge on which an active or passive adversary is operating. We assume that, in general, an edge is used for two way communication and an adversary can compromise communication in both directions. A *compromised path* is a path in which at least one edge has been compromised.

3 Security Applications of Edge Disjoint Paths

In this section we investigate some interesting security applications of k-systems. In general, we are interested in protocols for secure communication over k-systems in the presence of active and passive adversaries. Here, we present two applications, one each from the two categories: (1) those that require perfect security (see Section 3.1), and (2) those that require just computational security (see Section 3.2).

[2] Informally, "to learn" means that the adversary improves its ability to guess what the messages are. See [14] for a formal definition.

3.1 As a Building Block for Perfectly Secure Message Transmission

Perfectly secure message transmission is a fundamental problem in distributed computing. It is known that every function of n inputs can be efficiently and securely computed by a complete network of n participants even in the presence of $t < \frac{n}{3}$ Byzantine faults [6, 12]. In general communication networks, researchers have been striving to pursue alternatives for perfectly secure message transmission. The classic result, which simultaneously achieves perfect secrecy, perfect resilience, and worst-case time linear in the diameter of the network, is due to [14]. Specifically, using the protocol of [14] for perfectly secure message transmissions, the completeness theorem can be naturally extended to general networks of necessary connectivity $2t+1$. This has recently been extended (for instance) to accommodate more powerful communication capability [17] and non-threshold adversary capability [25]. Note that these schemes typically require multiple rounds of communications.

In [14], it is assumed that a set of wires (e.g., *vertex-disjoint paths* or *edge-disjoint paths*) are given as input to their algorithms. As a consequence, the k-system produced by our algorithms presented in Section 5, can be directly used as the input to their algorithms. Of course, security is naturally translated into the context of tolerating an adversary that is only able to corrupt up to certain number of edges.

3.2 Applications to Identity-Based Key Distribution

Identity-based cryptography is a notion introduced by Shamir [31]. Such schemes have the advantages (over traditional schemes such as those based on public key infrastructures) that secure communications can be established as long as a sender knows the identity of a receiver; this is a perfect analogy of the real-world mail system. There have been secure identity-based signature schemes [16] and secure public key cryptosystems [9]. Although it has never been explicitly stated, identity-based symmetric key cryptography (with the same functionality similar to Shamir's) has also been intensively investigated (see [7, 8, 26] and their follow-ons). However, all of the above mentioned (public key and symmetric key) schemes assume the existence of a trusted third party that predistributes certain secrets to the participants, and the existence of a set of system-wide cryptographic parameters. As we shall see, our k-systems facilitate identity-based key distribution (IBKD) without relying on any trusted third party or the existence of system-wide cryptographic parameters. In the sequence, we first present a definition for IBKD and then suggest some practical constructions.

Definition 2. (Identity-Based Key Distribution, IBKD) *An* IBKD *scheme consists of three probabilistic polynomial-time algorithms:* **Init**, **Send**, *and* **Receive**. *Let $\mathcal{S}$ denote a sender and $\mathcal{R}$ denote a receiver.*

Init. *This process may include the following steps.*

1. *$\mathcal{S}$ chooses a security parameter κ and generates a fresh secret K and a cryptographic setting $\mathbb{E}$ (e.g., algorithm identification).*

2. *$\mathcal{S}$ chooses an appropriate secret sharing scheme to split K into a set of shares $(K_1, K_2, \cdots, K_k)$ and possibly some accompanying information T.*
3. *The sender runs the algorithm presented in Section 5 to produce a k-system $\mathcal{P} = \{p_1, \ldots, p_k\}$.*

Send. *For a given $\mathcal{P} = \{p_1, \ldots, p_k\}$, $\mathcal{S}$ sends $(\mathbb{E}, K_i, T)$ to the receiver via path p_i, where $1 \leq i \leq k$.*

Receive. *After receiving $\{(\mathbb{E}_i, K_i, T_i)\}_{1\leq i\leq k}$, $\mathcal{R}$ executes according to the $\mathbb{E}_i$'s and T_i's to reconstruct the secret K.*

Definition 3. *We call an* IBKD *scheme (α, β)-secure, if an (α, β)-adversary is (1) unable to learn any information about the secret K, and (2) unable to prevent the receiver $\mathcal{R}$ from reconstructing the secret K.*

Concrete Constructions. For concreteness, we outline two IBKD schemes that are secure against a computationally-bound adversary (this is an adversary we confront in the real-world). The first instantiation is based exactly on Shamir's secret sharing [30]. When Alice intends to conduct secure communication with Bob, she simply executes according to the specification with $T = \emptyset$. Note that the original Shamir secret sharing scheme is at best $(\frac{k-1}{3}, \frac{k-1}{3})$-secure [27], where secrecy is in an information-theoretical sense[3]. The second instantiation is an extension to Shamir's secret sharing. It is an $(\frac{k-1}{2}, \frac{k-1}{2})$-secure IBKD scheme. The trick is that we can let T be certain checksum so that the receiver $\mathcal{R}$ can differentiate invalid incoming messages from valid ones. As a simple implementation (see, e.g., [19]), one could let $T = (f(K_1), \cdots, f(K_1))$ where f is an appropriate one-way function.

4 Path Coloring and Bandwidth Maximization

The network setting we assume is a uniform capacity network provisioned with optical fibre links and optical switches. Each link can provide bandwidth B on C different wavelengths i.e. each link can accomodate at most C different paths through it at a maximum rate of B bits per second for each path. Since we assume that every request entering the system has to be serviced, it is likely that the number of wavelengths required is more than C. In this case it might be be necessary to use time division multiplexing to share a given edge between several k-systems, thereby reducing the effective bandwidth available to each. In general, therefore, it is desirable to keep the number of different wavelengths down to the minimum possible.

Formally speaking the problem we wish to solve is stated as follows: Given an undirected graph $G = (V, E)$ and a (multi) set of request pairs $T = \{(s_i, t_i) | s_i \neq t_i, s_i, t_i \in V\}$. For each demand pair (s_i, t_i), find k edge disjoint paths (called a *k-system*) that connect s_i and t_i. Assign a color to each k-system such that

[3] Perfect secrecy here is achieved at the price of a single round of communication. However, the secret itself will be used for communication that is secure in a computational sense.

no two k-systems which share an edge have the same color. The objective is to minimize the number of colors used. We call this the k-Edge Disjoint Path Coloring problem (k-EDPCOL).

Note that the requirement that each of the k paths in a k-system be colored the same color is inessential to the basic project of constructing k-systems. We impose this requirement to simplify the competitive analysis of the algorithm. In a real-world setting it might be advisable to remove this constraint.

k-EDPCOL is a generalization of the well known Minimum Path Coloring Problem (MPCP) [28] and was first defined in [4]. There is a rich and interesting literature on the MPCP. Due to space constraints we omit a discussion of it here, referring the reader to the full version of this paper [3].

4.1 Flow Number

We begin by reintroducing the *Flow Number*, a network measure introduced in [24], which allows for more precise bounds for MPCP. One important property of this parameter which we would like to mention at the outset is that the Flow Number can be computed exactly in polynomial time which gives it a major advantage over other routing parameters like the expansion and the Routing Number.

Before we introduce the flow number, we need some notation. In a *concurrent multicommodity flow problem* there are k commodities, each with two terminal nodes s_i and t_i and a demand d_i. A *feasible* solution is a set of flow paths for the commodities that obey capacity constraints but need not meet the specified demands. An important difference between this problem and the unsplittable flow problem is that the commodity between s_i and t_i can be routed along multiple paths. The *(relative) flow value of a feasible solution* is the maximum f such that at least $f \cdot d_i$ units of commodity i are simultaneously routed for each i. The *max-flow* for a concurrent multicommodity flow problem is defined as the maximum flow value over all feasible solutions. For a path p in a solution, the *flow value of* p is the amount of flow routed along it. A special class of concurrent multicommodity flow problems is the *product multicommodity flow problem* (PMFP). In a PMFP, a nonnegative weight $\pi(u)$ is associated with each node $u \in V$. There is a commodity associated with every pair of nodes (u, v) whose demand is equal to $\pi(u) \cdot \pi(v)$.

Suppose we have a network $G = (V, E)$ with arbitrary non-negative edge capacities. For every node v, let the *capacity* of v be defined as

$$c(v) = \sum_{w:\{v,w\}\in E} c(v, w)$$

and the capacity of G be defined as $\Gamma = \sum_v c(v)$. Given a concurrent multicommodity flow problem with feasible solution $\mathcal{S}$, let the *dilation* $D(\mathcal{S})$ of $\mathcal{S}$ be defined as the length of the longest flow path in $\mathcal{S}$ and the *congestion* $C(\mathcal{S})$ of $\mathcal{S}$ be defined as the inverse of its flow value (i.e., the congestion tells us how many times the edge capacities would have to be increased in order to fully satisfy all the original demands, along the paths of $\mathcal{S}$). Let I_0 be a multicommodity flow problem in which each pair of nodes (v, w) has a commodity with demand $c(v) \cdot c(w)/\Gamma$. The *flow number* $F(G)$ of a network G is the minimum of $\max\{C(\mathcal{S}), D(\mathcal{S})\}$ over all feasible solutions $\mathcal{S}$ of I_0. When there is no risk of confusion, we simply write F instead of $F(G)$. Note that the flow number of a network is invariant to scaling of capacities.

The smaller the flow number, the better are the communication properties of the network. For example, $F(\text{line}) = \Theta(n)$, $F(\text{mesh}) = \Theta(\sqrt{n})$, $F(\text{hypercube}) = \Theta(\log n)$, $F(\text{butterfly}) = \Theta(\log n)$, and, $F(\text{expander}) = \Theta(\log n)$.

5 The Algorithm

In this section we describe our algorithm to construct k-systems for secure communication in optical networks, optimizing the bandwidth usage. The framework under which our algorithm is designed has the following features:

Requests come online. When a request, in the form of a sender-receiver node pair (s, t) that has to be connected by a k-system, comes it has to be satisfied before the next request comes [4]. Note that we do not assume any probability model for the requests.

Distributed computation. There is no central authority that accepts requests and constructs k-systems. Each request comes at the corresponding sender node s. The onus is on s to construct the k-system to connect with the receiver t. Initially the nodes do not have any global information about the network topology; they do, however, know who their immediate neighbors are. Thus s needs to send messages, first to its neighbors, and then through them to other nodes to learn the topology and construct the k-system. We assume that the protocol for constructing the k-systems is followed by all nodes, and that some information can be stored on links that facilitates this protocol.

Presence of an active adversary. The k-systems, after construction, are used for secure communication. In addition, we assume that an adversary is present even during the construction phase, and attempts to disrupt proper construction of k-systems.

Correspondingly, we describe our algorithm in three steps: Section 5.1 describes a centralized algorithm that has a bounded competitive ratio for any online request sequence; Section 5.2 details a distributed implementation of this algorithm; and finally, Section 5.3 tackles the question of resilience to an adversary attacking the algorithm while it runs.

5.1 The Online Algorithm and its Competitive Ratio

We run a *Bounded Greedy Algorithm* for k-EDPCOL.[5] A high level description of **BoundedGreedy** would be:

[4] This is the standard model for *online algorithms*. See [10] for a general introduction.

[5] The usage of bounded greedy algorithms for path routing problems was pioneered by Kleinberg in [22].

> For a given pair (s,t) find the lowest color class c such that a k-system with no more than $L = 24k^2F$ edges can be established between s and t that does not share an edge with any other k-system colored c — where F is the flow number of the communication network.

The algorithm works by looking for a non-intersecting k-system of bounded length in one color class after the next, establishing a new color class if it fails to do so. The value for L specified is necessary for the analysis of the competitive ratio.

The analysis of **BoundedGreedy** is a simple extension of the analysis of the bounded greedy algorithm for solving the k *Edge Disjoint Paths* problem (k-EDP) introduced in [5].[6] k-EDP is the admission control version of k-EDPCOL. Formally speaking: In the k-EDP we are given an undirected graph $G = (V, E)$ and a set of terminal pairs (or requests) T. The problem is to find a maximum subset of the pairs in T such that each chosen pair can be connected by k disjoint paths (k-systems) and, moreover, the paths for different pairs are mutually disjoint.

In [5] a bounded greedy algorithm was given for k-EDP and a competitive ratio of $O(k^3F)$ was proved which was subsequently improved to $O(k^2F)$ [4]. Our own **BoundedGreedy** is an iterated version of that algorithm.

We state the result here, omitting the proof.

Theorem 1. BoundedGreedy *run on set T with parameter $L = 24k^2 \cdot F$ is a $O(k^2F \cdot \log |T|)$ competitive algorithm for k-EDPCOL.*

Note that the algorithm requires knowledge of the flow number F of the communication network. This value can be pre-computed and stored at all nodes, and updated as an when the network topology changes significantly.

5.2 Distributed Implementation

The essential computation in **BoundedGreedy** is constructing the minimum sized k-system between s and t that can be colored c. To begin with, remove all edges that already have a k-system colored c. The rest of the problem can be easily formulated as a min cost flow problem, and any min cost flow algorithm can be used to solve it (see, e.g., [1]). We use an algorithm, based on Galil and Yu's work [18].[7]

In the distributed implementation of **BoundedGreedy** each sender node s finds a k-system to receiver t of size L, in the lowest color class possible. To prevent conflicts on edges, each edge e stores a list $colors(e)$ of the colors of all exisiting k-systems that use e. Initially, s does not know the topology of

[6] A similar connection has been observed independently by Aumann and Rabani [2] in the case of the path colouring problem and the edge disjoint paths problem.

[7] We refer the reader to Galil and Yu's paper for a discussion of the relationship of their algorithm to the previous work done in similar settings by Jewell [21], Iri [20], and Busaker and Gowen [11]; and by Tomizava [32], and Edmonds and Karp [15].

the network, except for its neighbors. It learns the topology by sending explicit messages to the nodes it knows at any given point of time. We take care to ensure that these messages are kept to a minimum and that we consider only that part of the network which is absolutely essential for our needs.

Before delving into the internal workings of the algorithm we need some definitions.

Definition 4. *Let $H = \langle V, E\rangle$ be a directed network with edge capacity function $g : E \mapsto \Re^+$ and edge cost function $\lambda : E \mapsto \Re$.*

- *The cost function is said to be* skew symmetric *if $\forall(i,j) \in E : \lambda_{ij} = -\lambda_{ji}$.*
- *A* k-flow *from node s to t is a function $f : E \mapsto \Re^+$, such that, $\forall(i,j) \in E$: $0 \leq f_{ij} \leq g_{ij}$, and $\forall i \in V, i \neq s,t, \sum_{(i,j)\in E} f_{ij} = \sum_{(j,i)\in E} f_{ji}; \sum_{(s,j)\in E} f_{sj} = \sum_{(j,s)\in E} f_{js} + k; \sum_{(i,t)\in E} f_{it} = \sum_{(t,i)\in E} f_{ti} + k$. The* cost *of this flow is $cost(f) = \sum_{(i,j)\in E} \lambda_{ij} \cdot f_{ij}$. A* minimum cost k-flow *is one that minimizes $cost(f)$.*
- *Given flow f, the* residual capacity function *r is defined as $r_{ij} = g_{ij}+f_{ji}-f_{ij}$. The* residual network *$H(f)$ induced by f is one consisting only of edges with positive residual capacities, with capacity function r and the same cost function λ.*

An unweighted undirected graph G can be transformed into a directed network H with edge capacities and a skew symmetric cost function by replacing each edge $(i,j) \in V(G)$ with a directed subgraph $\langle V_{i,j}, E_{i,j}\rangle$, where $V_{i,j} = \{i, j, l_{ij}, l_{ji}\}$ and $E_{i,j}$ consists of four pairs of opposite edges between i and l_{ij}, l_{ij} and j, j and l_{ji}, l_{ji} and i. Edges in directed cycle i, l_{ij}, j, l_{ji} have capacity 1 and cost 1, and edges in directed cycle i, l_{ji}, j, l_{ij} have capacity 0 and cost -1.

We also define the following node potential function which will help us bound the time complexity of our min cost flow algorithm.

Definition 5. *A node potential is a function $\pi : V \mapsto \Re$. The reduced cost function λ^π is defined as $\lambda^\pi_{ij} = \lambda_{ij} + \pi(i) - \pi(j)$.*

BoundedGreedy is described in Figure 1. The communication network is represented by G. To find a min cost k flow in G we construct a shortest path tree τ in the residual network $H(f)$ using λ^π as edge length. Finding a shortest path requires significant knowledge of the network. Since the sender's knowledge is only local to begin with, and augmenting this knowledge incurs a communication cost, we compute the tree using a lazy version of the classic Dijkstra's algorithm. **LazyDijkstra** is described in Figure 2 (we have omitted some bookkeeping details for a simpler presentation). During its execution our knowledge about the graph may increase. On these occasions, computing the value of the potential π for the new nodes is easy, since it is easy to prove the following claim that allows us to update π using distances from a previous iteration.

Claim. In **BoundedGreedy**, the potential $\pi(u)$ of any node u in the beginning of iteration b is equal to its shortest distance from s in the residual network at the beginning of iteration $b-1$, using λ^π as the edge length.

Once a node has been visited, its information is stored at s. This is so even if the node is not useful in the current iteration because of color conflicts. It is easy to see that a node is visited only when it is essential for the computation of the shortest path to t.

Time and Communication Complexity. In [18], Galil and Yu show that Dial's implementation [13] of Dijkstra's algorithm finds a k-flow in $O(km)$ time, where m is the number of edges in G. If we ignore time for communication, this is true in our case too. So, barring communication time, **BoundedGreedy** runs in time $O(c \cdot km)$, where c is the number of colors used, and m is the number of edges in the network examined. Note that the size of the network examined is dependent on the edges that have been taken by the existing k-systems, and so cannot be characterized easily. Luckily, we can give a better bound on this value if we introduce a simple addition, described next.
A slightly improved algorithm. Let $\mathcal{P} = \{p_1, \ldots, p_k\}$ be the k-system obtained by decomposing the k-flow. Suppose we maintain $\mathcal{P}$ through all the k iterations. At any point, if the number of edges in $\mathcal{P}$ exceeds L, we know that we need to go to the next color class. If we add this check, only nodes that are within a distance of L from s will ever be considered. Denote the subgraph of these nodes by $N_s(L)$. The time bound is now $O(c \cdot k \cdot |E(N_s(L)|)$.

The above check also gives a bound on the communication complexity, defined as the sum of the number of hops made by all the messages sent. Messages are sent only to nodes in $N_s(L)$. Each such node is at distance at most L. So the complexity is bounded by $O(L \cdot |N_s(L)|)$.

5.3 Resilience to a Active Adversary

The **BoundedGreedy** algorithm, as described, is resilient to a passive listening adversary, but not to an active disrupting adversary. A disrupting adversary controlling only a few edges may succeed in deceiving the sender s in constructing an incorrect k-system, or prevent it from constructing a k-system in color class c when it is possible. This is because, even though a k-system is itself used to provide communication through k edge disjoint paths, the algorithm does not specify the paths used by the messages like `LIST_NEIGHBORS` and `COLOR_EDGE` it sends. All these messages may pass through a small number of edges that may all be under the control of the adversary.

It is not necessary that, to be usable, the k-path system constructed should necessarily consist of k edge disjoint paths. In particular, we can say:

Theorem 2. *Every protocol for secure communication over a k-system in the presence of an (α, β)-active adversary, can be used for secure communication over a k-path system $\mathcal{P}$ in the presence of an (α, β)-active adversary, if $\mathcal{P}' \subseteq \mathcal{P}$ of at least $k - \beta$ paths are edge disjoint and the adversary is limited to listening on at most $\alpha - \beta$ edges in the paths in $\mathcal{P}'$.*

We omit the proof due to lack of space. Now, if detection of an unusable k-system is the only issue, it can be easily done by a slight modification to the last part of the algorithm. When a message is sent to nodes u and v to color edge $(u, v) \in p(b)$, we specify that it takes the specific path $p(b)$. We also ensure that the replies are sent back along this very path. If any node reports an error, we discard the k-system as unusable. We claim that if the adversary can compromise at most β edges, at least $k - \beta$ paths in the k-system are edge disjoint and not compromised.

The above idea can be extended to ensure that the algorithm finds a usable k-system in color class c if one exists. Following an idea introduced earlier, let $\mathcal{P} = \{p_1, \ldots, p_k\}$ be the k-system obtained by decomposing the k-flow. Modify the algorithm to maintain the current $\mathcal{P}$ at every stage. Further, in LazyDijkstra, we abandon the notion of a *visited* node. To every node u extracted from Q we send a `LIST_NEIGHBORS` message. Care is taken that this message goes along the path p in $\mathcal{P}$ that u would be a part of, and that the reply comes back along the same path. Lastly, if at any stage, the paths in $\mathcal{P}$ change such that the path to a node u in $\mathcal{P}$ is different from what was used by the `LIST_NEIGHBORS` message sent to it, we confirm our information about u by sending the message (and receiving a reply) again through the new path. Call this the ModifiedLazyDijkstra algorithm.

Claim. A k-path system found by **BoundedGreedy** using ModifiedLazyDijkstra, in the presence of an (α, β)-adversary has at least $k - \beta$ paths that are edge disjoint and the adversary is limited to listening on at most $\alpha - \beta$ edges on these paths. Further, such a k-system can always be found as long as a $(k + \beta)$-system exists.

Proof Sketch. Let the k-system found be $\mathcal{P}$. Each path $p \in \mathcal{P}$ is edge disjoint with every other path in $\mathcal{P}$. Further, all information about p is obtained by using only edges in p. This implies that with a single edge the disrupting adversary can deceive with respect to at most a single path in $\mathcal{P}$. Thus, with β edges, it can deceive with respect to at most β paths. Further, it can listen on at most $\alpha - \beta$ edges on the remaining $k - \beta$ paths.

Similarly, if a $(k + \beta)$-system exists, the adversary can deceive with respect to at most β paths in it. The rest of the paths form a k-system, which will be discovered by the "unhindered" algorithm. $\square$

6 Conclusions and Open Problems

Our primary concern here is to develop an efficient algorithm for a security problem. However, by using already extant network resources for our purpose we propose a new kind of resource efficiency and backward compatibility.

Our work also has important implications for network design. It can provide input as to what kind of parameters need to be tuned when networks are provisioned. For example, building a network with a low Flow Number might be very useful since the average latency and the competitive ratio of our algorithm are directly related to this quantity.

A number of areas for future investigation are thrown up by our work. It would be interesting to see what kind of algorithmic solutions can be obtained for constructing vertex disjoint path systems. We feel that security applications can give a new lease of life to research in this area. A useful investigation would be to see if we can adjust the algorithm to produce customized k-systems. For example, Alice may want $k = 11$ and Bob may only need $k = 5$. Such capability will enable the users to achieve security that is proper to their applications at a reasonable cost.

Acknowledgements. The authors thank Moti Yung for helping clarify the relationship between this work and [14], Petr Kolman and Ankur Bhargava for helpful discussions on k-EDPCOL, and Tanu Malik for proof reading and suggestions.

References

1. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows.* Prentice Hall, New Jersey, 1993.
2. Y. Aumann and Y. Rabani. Improved bounds for all-optical routing. In *Proc. of the 6th ACM-SIAM Symposium on Discrete Algorithms*, pages 567–576, 1995.
3. A. Bagchi, A. Chaudhary, M. T. Goodrich, and S. Xu. Constructing disjoint paths for secure communication (full version). Available online at `http://www.ics.uci.edu/~bagchi/pub.html`.
4. A. Bagchi, A. Chaudhary, and P. Kolman. Short length Menger's theorem and reliable optical routing. In *Proc. of the 15th Annual Symp. on Parallel Algorithms and Architectures*, pages 246–247, 2003.
5. A. Bagchi, A. Chaudhary, P. Kolman, and C. Scheideler. Algorithms for fault-tolerant routing in circuit-switched networks. In *Proc. 14th ACM Symp. on Parallel Algorithms and Architectures*, pages 265–274, 2002.
6. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computing (extended abstract). In *Proc. of 20th Annual Symposium on the Theory of Computing*, pages 1–10, 1988.
7. R. Blom. An optimal class of symmetric key generation systems. In *Proc. of EUROCRYPT '84*, pages 335–338, 1984.
8. C. Blundo, A. D. Santis, A. Herzberg, S. Kutten, U. Vaccaro, and M. Yung. Perfectly-secure key distribution for dynamic conferences. In *CRYPTO'92*, pages 471–486, 1992.
9. D. Boneh and M. Franklin. Identity-based encryption from Weil pairing. In *Proc. of CRYPTO 2001*, pages 213–229, 2001.
10. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, 1998.
11. R. G. Busaker and P. J. Gowen. A procedure for determining minimal-cost flows by doubling scaling. Technical Report ORO Technical Report 15, Operational Research Office, Johns Hopkins University, Baltimore, MD, 1961.
12. D. Chaum, C. Crepeau, and I. Damgard. Multiparty unconditionally secure protocols. In *Proc. of 20th Annual Symposium on the Theory of Computing*, pages 11–19, 1988.
13. R. Dial. Algorithm 360: Shortest path forest with topological ordering. *Comm. ACM*, pages 632–633, 1969.
14. D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *JACM*, 40(1):17–47, 1993.

15. J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19:248–264, 1972.
16. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proc. of CRYPTO '86*, pages 186–194, 1986.
17. M. Franklin and R. Wright. Secure communication in minimal connectivity models. In *Proc. of EUROCRYPT '98*, pages 346–360, 1998.
18. Z. Galil and X. Yu. Short length versions of Menger's theorem. In *Proc. of the 27th Annual ACM Symposium on Theory of Computing*, pages 499–508, 1995.
19. L. Gong. Increasing availability and security of an authentication service. *IEEE J. Selected Areas in Communications*, 11(5):657–662, 1993.
20. M. Iri. A new method for solving transportation-network problems. *Journal of the Operations Research Society of Japan*, 3:27–87, 1960.
21. W. S. Jewell. Optimal flow through networks. Technical Report Interim Technical Report 8, Operation Research Center, MIT, Cambridge, MA, 1958.
22. J. Kleinberg. *Approximation Algorithms for Disjoint Paths Problems*. PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1996.
23. S. G. Kolliopoulos and C. Stein. Approximating disjoint-path problems using greedy algorithms and packing integer programs. In *Proc. of the 6th International Integer Programming and Combinatorial Optimization Conference (IPCO)*, pages 153–168, 1998.
24. P. Kolman and C. Scheideler. Improved bounds for the unsplittable flow problem. In *Proc. of the 13th ACM-SIAM Symposium on Discrete Algorithms*, pages 184–193, 2002.
25. M. Kumar, P. Goundan, K. Srinathan, and C. Pandu Rangan. On perfectly secure communication over arbitrary networks. In *Proc. of 21st Annual ACM Symposium on the Principles of Distributed Computing*, pages 193–202, 2002.
26. T. Leighton and S. Micali. Secret-key agreement without public-key cryptography (extended abstract). In *Proc. of CRYPTO '93*, pages 456–479, 1993.
27. R. McEliece and D. Sarwate. On sharing secrets and Reed-Solomon codes. *Comm. ACM*, 24(9):583–584, 1981.
28. Y. Rabani. Path coloring on the mesh. In *Proc. of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 400–409, 1996.
29. P. Raghavan and E. Upfal. Efficient routing in all-optical networks. In *Proc. of the 26th Annual Symposium on the Theory of Computing*, pages 133–143, 1994.
30. A. Shamir. How to share a secret. *Comm. ACM*, 22(11):612–613, 1979.
31. A. Shamir. Identity-based cryptosystems and signature schemes. In *Proc. of CRYPTO '84*, pages 47–53, 1984.
32. N. Tomizava. On some techniques useful for solution of transportation network problems. *Networka*, 1:173–194, 1972.

BoundedGreedy (t, L)

/* Initialize the graph with the local information available at s. */
$G \leftarrow \langle \{s\} \cup \{u | u$ is a neighbor of $s\}, \{(s,u) | u$ is a neighbor of $s\}\rangle$.
$c \leftarrow 0$.
connected $\leftarrow$ **false**.
/* Find a k-system that can be colored c. */
While(*connected* $=$ **false**)
  $G_{\overline{c}} \leftarrow G \setminus \{$edges that are part of an existing k-system colored $c\}$.

  /* Transform the graph into a form suitable for our min cost flow algorithm. */
  $H \leftarrow G_{\overline{c}}$
  $\forall e(i,j) \in E(H) : f_{ij} \leftarrow 0$.
  $\forall u \in V(H) : \pi(u) = 0$.
  $b \leftarrow 0$.
  /* Finding a min cost k flow from s to t. */
  While$(b < k)$
    /* Find a min cost augmenting path in the residual network. First construct shortest path tree using LazyDijkstra; this can potentially increase the information about G available at s. */
    $\tau \leftarrow$ **LazyDijkstra**$(H(f), \lambda^{\pi}, t, b, c)$.
    $\forall u \in V(H)$: let $d(u) \leftarrow$ distance of u from s in τ.
    If$(d(t) = \infty)$ restart outer while loop with $c \leftarrow c + 1$.
    Else **/* Pass a unit flow through the augmenting path.*/**
      Let $p'(b)$ be a shortest path in τ from s to t.
      $f \leftarrow f+$ unit flow in $p'(b)$ from s to t.
      $\forall u \in V(H) : \pi(u) \leftarrow \pi(u) + d(u)$.
      $b \leftarrow b + 1$.
    $\{p_1, \ldots, p_k\} \leftarrow$ edge disjoint paths formed by decomposing the flow in $\bigcup_b p'(b)$.
    If$(|\bigcup_b p(b)| \leq L)$
      $\forall (u,v) \in \bigcup_b p(b)$: Send message `COLOR_EDGE` to u and v requesting adding c to $colors(u,v)$.
      If$(\forall (u,v) \in \bigcup_b p(b)$: Coloring request granted) *connected* $\leftarrow$ **true**.
      Else $c \leftarrow c + 1$.
      /* If k-system cannot be used, or request denied, restart with next color. */

Fig. 1: The **BoundedGreedy** algorithm: Constructs a k-system, of size at most L, from sender node s to receiver node t in the minimum color class possible. The computation is performed at s.

LazyDijkstra($H(f), \lambda^{\pi}, t, b, c$)

$S \leftarrow s$.
/* Initialize the set S that contains vertices currently in τ. */
$Q \leftarrow V(H(f)) \setminus \{s\}$.

While($Q \neq \emptyset$)
/* Find shortest paths to as many nodes as possible. */
  $u \leftarrow$ node in Q with minimum λ^{π} distance from s (using edges only in and from S).
  If(u has not been visited before)
    If($t \in S$) **return**.
    Send message `LIST_NEIGHBORS` to u requesting the set $L = \{(w, colors(u, w)) | w$ is a neighbor of $u\}$.
    $\forall w$, w is a neighbor of u, and there is no visited node v such that w is a neighbor of v:
    /* Update graph G (and thereby H), set Q, and potential function π with the newly learned vertices. */
      $G \leftarrow \langle V(G) \cup \{w\}, E(G) \cup \{(u, w)\}\rangle$.
      If($c \notin colors(u, w)$):
        $Q \leftarrow Q \cup \{w\}$.
        $\pi(w) \leftarrow$ shortest distance of w from s, using λ as edge length, in the residual network at the start of the previous iteration, i.e., the network $H(\sum_{q=0}^{b-2}$ flow in $p'(q))$.
    Mark u as visited.
  $S \leftarrow S \cup \{u\}$.
  /* Include u in S and update shortest distance estimates. */
  $\forall w, w$ is a neighbor of u :
    If(distance$(s, w) >$ distance$(s, u) + \lambda\pi(u, w)$)
      distance$(s, w) \leftarrow$ distance$(s, u) + \lambda^{\pi}(u, w)$.

Fig. 2: The **LazyDijkstra** subroutine: $H(f)$ is the residual network of the graph currently known at s. The reduced cost function λ^{π} is used as edge length in $H(f)$. A shortest path tree τ is constructed from s to as many vertices in $H(f)$ as is possible without sending messages on the network. If, however, τ does not include receiver t, messages are sent to learn the graph necessary to include t in τ. b is the iteration index in **BoundedGreedy**. c is the color being considered.