Resilient Authentication Using Path Independence

Michael K. Reiter and Stuart G. Stubblebine, *Member*, *IEEE*

Abstract—Authentication using a path of trusted intermediaries, each able to authenticate the next in the path, is a well-known technique for authenticating channels in a large distributed system. In this paper, we explore the use of multiple paths to redundantly authenticate a channel and focus on two notions of path independence—*disjoint* paths and *connective* paths—that seem to increase assurance in the authentication. We give evidence that there are no efficient algorithms for locating maximum sets of paths with these independence properties and propose several approximation algorithms for these problems. We also describe a service we have deployed, called PathServer, that makes use of our algorithms to find such sets of paths to support authentication in PGP applications.

Index Terms—Security, authentication, public key certificate, certificate path, independent paths.

———————————— ✦ ————————————

1 INTRODUCTION

AUTHENTICATING the source of a request in a computer system is a necessary ingredient to enforcing access controls. A request is received on some *channel*, such as a network, and determining the set of *principals* (e.g., users, processes, or computers) that could have initiated that request is called *authenticating* the channel (or request). Authentication in centralized computer systems is simplified by the presence of a central authority (the operating system) that controls all channels and knows what principals can initiate requests on what channels. In a distributed system there typically is no such central authority for this information. As the distributed system gets larger and more diverse, the difficulty of reliably authenticating a channel can increase substantially.

This difficulty arises in secure communication systems such as Privacy Enhanced Mail [11] and many systems that use the PGP [26] public key management and encryption tools. A message in these systems would typically include a *digital signature* to enable the recipient of the message to determine the user who sent the message. Following [13], the channel in this case is the public key that can be used to verify the signature on the message, and authenticating the channel means determining the principals that could have generated that signature. Because there is no global authority for this information, the user is asked to defer to a "path" of channels $c_1, \ldots, c_\ell$ (other public keys) such that:

1) The user believes it can authenticate c_1,
2) Each c_i, $i < \ell$, has uttered a statement (a certificate) regarding for what principal c_{i+1} speaks, and
3) c_ℓ has uttered a statement regarding for what principal the channel of interest to the user speaks.

• *M.K. Reiter is with Bell Laboratories, 600 Mountain Ave., Room 2A-342, Murray Hill, NJ 07974. E-mail: reiter@research.bell-labs.com.*
• *S.G. Stubblebine is with AT&T Labs—Research, 180 Park Ave., Room B235, Florham Park, NJ 07932. E-mail: stubblebine@research.att.com.*

If the user is willing to trust the statements of each channel on the path, then the user authenticates the target channel according to the statement that c_ℓ made about it. To our knowledge, using such paths for authentication was first proposed in [2] (for authentication based on shared keys) and, in addition to being used in the aforementioned systems, has been supported in [3], [8], [13], [25].

When a single path of channels is used to authenticate a target channel, the authentication is vulnerable to the compromise of any channel on that path. That is, if any c_i in the path provides a false statement regarding c_{i+1}, then there is no reason to believe that a proper semantics for the target channel is reached. One way to bolster assurance in the channel authentication is to limit the length of the path used, thus limiting the number of intermediate principals that must be trusted. A second way is to employ multiple paths and to authenticate the target channel using information obtained via each of these paths. This approach is inspired by prior work in nullifying Byzantine-faulty sources of information by consulting multiple "independent" sources of information and accepting as true the information returned by a majority of them (e.g., [21]). Here, our multiple sources of information are multiple paths of bounded length resulting in statements about the target channel.

In this paper, we explore what it means for multiple length-bounded paths to be *independent* in this context. We focus on two related notions of independence. In the first, a set of bounded paths are "independent" if they are pairwise disjoint, i.e., if no two paths share a common channel. We call this a set of *bounded disjoint paths*. Bounded disjoint paths are appealing because no channel is relied on multiple times in the authentication of the target channel. In the second, a set of bounded paths are "independent" if the removal of k channels is necessary to disconnect all of them. We call this a set of *bounded k-connective* paths. Bounded k-connective paths are robust to the compromise of any $k - 1$ channels: If some $k - 1$ channels are compromised and, thus, the statements they contribute are forgeries (and should be

disregarded), there is still a bounded path containing none of these compromised channels to the target channel. Note that a set of k bounded disjoint paths is a set of bounded k-connective paths, but, in general, a set of bounded k-connective paths will not be disjoint.

We have built a web service, called PathServer, that supports authentication of PGP public keys using our bounded disjoint paths and bounded connective paths paradigms. If bounded disjoint paths are requested, PathServer locates a set of such paths from the requesting principal (or, more precisely, a channel that is known to speak for it) to the target channel in our database of PGP certificates (a "keyring" in PGP parlance). If bounded connective paths are requested, PathServer returns a value k and a set of bounded k-connective paths from the requesting principal to the target channel. Though PathServer currently supports only PGP, our work can also be applied to other public key management systems (e.g., those based on X.509) as well as to systems that employ other types of channels (e.g., shared keys, protected physical links, or a combination of these [13]).

While user reaction to PathServer suggests that these independence concepts are useful in bolstering assurance in authentication, they also have limitations. First, our insistence on independent paths is an effort to avoid depending heavily on a few principals in the process of authenticating a target channel. However, since computer systems can identify principals only syntactically, in general it is outside the scope of a system to detect channels controlled by principals whose actions are closely correlated (e.g., two close friends). Thus, we are forced to settle for the aforementioned syntactic notions of independence and to appeal to the user for assistance in pruning potentially correlated paths further. Second, the complexities of finding a *maximum* set of bounded disjoint paths (i.e., a set of largest cardinality) and of finding the *maximum k* for which there exists a set of bounded k-connective paths provide strong evidence that neither can be performed in polynomial time. Specifically, the former is NP-hard and the latter is coNP-hard [6]. Moreover, the foremost practical instances of these problems that we are targeting (i.e., public key certification systems such as PGP) induce graphs of sufficient size to make this a severe limitation. We thus propose heuristics to approximate solutions to these problems.

The rest of this paper is structured as follows. We formalize our problems in Section 2. We present and evaluate our approximation algorithms for finding a maximum set of bounded-length disjoint paths in Section 3. We extend these algorithms to compute a set of bounded k-connective paths for an approximately maximum k in Section 4. In Section 5, we describe the PathServer application that we have built using the algorithms in this paper. We discuss related work in Section 6, and we conclude in Section 7.

2 PROBLEM STATEMENT

We borrow concepts and terminology from [13] to formalize our problem. Our system consists of a set of *principals* (e.g., people, machines, roles), some of which are *channels* (e.g., network addresses or encryption keys). Channels are the only principals that can make statements directly. For the

purposes of this paper, the only statements that we consider are statements of the form "c_1 **says** $c_2 \Rightarrow P$," where c_1 and c_2 are channels, P is a principal, and $\Rightarrow$ denotes the "speaks for" relation. Intuitively, $c_2 \Rightarrow P$ means that if a statement emanates from c_2 (i.e., c_2 "**says**" the statement), then the statement can be treated as if P said it. c_1 **says** $c_2 \Rightarrow P$ is then c_1's statement that this is true. We model our system with a directed graph $G = (V, E)$, where V is a finite set of channels (nodes) and E is a finite set of edges denoting statements of the form described above. The statement c_1 **says** $c_2 \Rightarrow P$, where $c_1, c_2 \in V$, is represented by an edge $c_1 \overset{P}{\to} c_2$ in E, which we often abbreviate by $c_1 \to c_2$ when P is not important. We represent multiple statements $c_1 \overset{P_1}{\to} c_2, \dots, c_1 \overset{P_j}{\to} c_2$ made by the same channel c_1 about the same channel c_2 by a single statement $c_1 \overset{P_1 \wedge \dots \wedge P_j}{\to} c_2$. This graph is perhaps most easily pictured in the context of a "web" of public keys. In this case, V would be a set of public keys and E would be a set of certificates. Nevertheless, the graph can be interpreted to include any channels and appropriate statements.

The problem at hand is for a principal to authenticate some channel $t \in V$ of interest, called the *target*. For simplicity, we assume that the principal has sole control of some channel $s \in V$, called the *source*, and that any statements that the principal is willing to utter regarding other channels are represented by edges in E emanating from s. We assume that the principal has access to all of G. As motivated in Section 1, it is our thesis that multiple independent paths from s to t, each of at most some specified length, can help the principal to authenticate t. More precisely, a *path* from s to t in G is a sequence of edges $s \to c_1 \to \dots \to c_\ell \to t$ for some $\ell \geq 0$, where each $c_i \notin \{s, t\}$ and where $i \neq j$ implies $c_i \neq c_j$. The *length* of a path $s \to c_1 \to \dots \to c_\ell \to t$ is ℓ, and a path is *b-bounded* if its length is at most b. In our first interpretation of "independent," we employ disjoint paths. More precisely, two paths from s to t, say $s \to c_1^1 \to \dots \to c_{\ell_1}^1 \to t$ and $s \to c_1^2 \to \dots \to c_{\ell_2}^2 \to t$, are *disjoint* if $c_i^1 \neq c_j^2$ for all i, $1 \leq i \leq \ell_1$, and all j, $1 \leq j \leq \ell_2$. In this case, our problem becomes the following:

Bounded Disjoint Paths (BDP):
   Given: A directed graph G, distinguished nodes s and t, and a path bound b.
   Problem: Find a maximum set of mutually disjoint b-bounded paths from s to t.

To capture our second notion of "independence," we say that a set D of paths from s to t, where $s \to t \notin D$, is k-*connective* if the smallest subset of $V \backslash \{s, t\}$ that intersects every path in D is of size k. That is, the paths in D are k-connective if it is necessary to remove k nodes (other than s and t) to disconnect them all. The *b-connectivity* from s to t is the maximum k for which a set of b-bounded k-connective paths from s to t exists (with the edge $s \to t$ removed if it exists). Our second problem is then:

Bounded Connective Paths (BCP):

   Given: A directed graph G, distinguished nodes s and t, and a path bound b.

   Problem: Find the b-connectivity from s to t, say k, and a set of b-bounded k-connective paths from s to t.

Note that if $s \not\rightarrow t$, then any k disjoint b-bounded paths from s to t are k-connective, but, in general, a set of b-bounded k-connective paths from s to t are not disjoint. For comparisons of these concepts on undirected graphs, see [5], [15].

Solutions to BDP and BCP can be useful in supporting authentication of a target channel. However, it is up to individual users' policies to determine exactly how they are used. Given a set of disjoint or connective paths

$$s \rightarrow c_1^1 \rightarrow \cdots \rightarrow c_{\ell_1}^1 \overset{P_1}{\rightarrow} t$$
$$s \rightarrow c_1^2 \rightarrow \cdots \rightarrow c_{\ell_2}^2 \overset{P_2}{\rightarrow} t$$
$$\vdots$$
$$s \rightarrow c_1^j \rightarrow \cdots \rightarrow c_{\ell_j}^j \overset{P_j}{\rightarrow} t$$

the requesting principal might authenticate t by, e.g., requiring consensus among the paths, i.e., that $P_1 = P_2 = \ldots = P_j$. If there exist P_i, $P_{i'}$ that are different, then this indicates a discrepancy in what different paths reported about t that must be resolved by the requesting principal's policy (e.g., adopting a P_i common to $k + 1$ disjoint paths overcomes k compromised channels).

3 BOUNDED DISJOINT PATHS

We now turn to an algorithmic consideration of the problems we presented in Section 2, beginning with Bounded Disjoint Paths (BDP). BDP has been previously studied from a complexity-theoretic point of view, and has been proved to be NP-hard [9].[1] Thus, there is little hope of finding an efficient solution to BDP, and we turn to finding approximation algorithms for this problem. By an "approximation algorithm," we intuitively mean an efficient algorithm that usually comes close to the actual answer; a more careful definition and discussion can be found in [6]. The only prior work of which we are aware on approximation algorithms for BDP is due to Ronen and Perl [20]. They proposed an algorithm and showed empirically that it performs well on small random undirected graphs of 50 nodes. Their algorithm runs in $O(b^2 n^2 m)$ time and $O(b^2 nm)$ space with a path bound b on a graph with n nodes and m edges.

We describe a class of algorithms in this section that is much simpler than that in [20] and offers superior time and space complexity. In one instantiation, our algorithm runs in $O(nm)$ time and $O(n + m)$ space. Another runs in $O(bnm + bn^2 \log(bn))$ time and $O(bn + m)$ space. In order to motivate

---

1. More precisely, BDP remains NP-hard for any fixed $b \geq 4$, but can be solved in $O(m\sqrt{n})$ time on a graph with n nodes and m edges if $b < 4$ using maximum matching and maximum flow techniques [9]. The related problem of finding a requested number of disjoint paths of minimum *total* length can be solved in polynomial time [22].

our algorithms, we first present another algorithm that runs in $n^{O(b)}$ time and space and, thus, is exponential in b. While we introduce this first algorithm primarily for motivational purposes, it can be argued to be "efficient" in the following senses. First, if P $\neq$ NP, then there is no algorithm for solving BDP exactly that is polynomial in n since BDP remains NP-hard for any fixed $b \geq 4$ [9]. Put another way, a user that always chooses a fixed bound $b \geq 4$ will observe polynomial growth in the running time of this algorithm as a function of the graph size, whereas there is no known algorithm that can solve BDP exactly for a fixed $b \geq 4$ and provide polynomial growth as a function of graph size if P $\neq$ NP. Second, in most cases, the size of b that users desire is reasonably small.

Let $\mathbf{bdp}(G, b, s, t)$ denote the cardinality of a maximum set of disjoint paths from s to t of length at most b in graph $G = (V, E)$. Each of the algorithms A that we present here produce a set with cardinality $A(G, b, s, t)$ of disjoint paths from s to t of length at most b, where

1) $A(G, b, s, t) \leq \mathbf{bdp}(G, b, s, t)$, and
2) if $\mathbf{bdp}(G, b, s, t) > 0$, then $A(G, b, s, t) > 0$.

The first of these properties (which is also required by the definition of an approximation algorithm [6]) indicates that our algorithms are *fail-secure*, in the sense that they will never return a set of paths from a source channel s to a target channel t that exaggerates the actual maximum set of disjoint paths from s to t. In addition to the above properties, each algorithm uses heuristics to search for a *maximum* set of paths. In Section 3.3, we give empirical evidence that our algorithms perform well on one type of interesting graph. However, this data also indicates that the error of our algorithms is not constant, but rather increases as a function of problem size. The following theorem provides a small amount of justification.

THEOREM 1. *If* P $\neq$ NP, *then no polynomial approximation algorithm A for BDP can guarantee* $\mathbf{bdp}(G, b, s, t) - A(G, b, s, t) \leq K$ *for a fixed constant K.*

PROOF. Suppose, for a contradiction, that there is such an algorithm A and constant K. We use A to construct a polynomial time algorithm for solving BDP exactly. Consider a problem instance $(G = (V, E), b, s, t)$, and assume without loss of generality that $s \not\rightarrow t$ and that K is an integer. The algorithm constructs a new problem instance (G', b, s, t) where G' consists of $K + 1$ "copies" of G with the exception that s and t are represented in G' only once. That is, the nodes for G' are

$$V' = \{s, t\} \cup \left[\bigcup_{c \in V \setminus \{s, t\}} \{c[1], \ldots c[K + 1]\} \right]$$

and the edge set E' is defined by

$$E' \ni \begin{cases} c_1[j] \rightarrow c_2[j] & (1 \leq j \leq K + 1) & \text{if } c_1, c_2 \notin \{s, t\} \\ & & \text{and } c_1 \rightarrow c_2 \in E \\ s \rightarrow c[j] & (1 \leq j \leq K + 1) & \text{if } s \rightarrow c \in E \\ c[j] \rightarrow t & (1 \leq j \leq K + 1) & \text{if } c \rightarrow t \in E. \end{cases}$$

G' can be constructed in polynomial time, since K is fixed.

1. Set $I = \emptyset$ and $U = V$.
2. Let v be a node in U with the minimum degree in the subgraph induced by U. Set $I = I \cup \{v\}$ and $U = U \setminus (\{v\} \cup \{u \in U : (v, u) \in E\})$.
3. If $U = \emptyset$, then halt and return I. Otherwise, go to 2.

Fig. 1. Johnson's approximation algorithm for MIS on undirected graph $G = (V, E)$.

Note that $\mathbf{bdp}(G', b, s, t) = (K + 1)\mathbf{bdp}(G, b, s, t)$. Moreover, an exact solution to BDP on the instance (G, b, s, t) can be obtained by running A on (G', b, s, t) and taking the largest subset of paths that A selected from any single copy of G, since A can find less than a maximum set of paths on at most K copies of G. □

A further characterization of approximation limitations for the Bounded Disjoint Paths problem is a topic for future research. We now turn to presenting our algorithms.

3.1 Independent Set

The first approximation algorithm for BDP that we present was influenced by work on approximation algorithms for a different problem, called Maximum Independent Set.

Maximum Independent Set (MIS):
Given: An undirected graph $G = (V, E)$.
Problem: Find a set $V' \subseteq V$ of largest cardinality such that no two nodes in V' are joined by an edge in E.

Any set $V' \subseteq V$ such that no two nodes in V' are joined is said to be an *independent set*. Such a set V' of largest cardinality is said to be a *maximum* independent set.

MIS is a well-known NP-hard problem (see [6]). In [10], Johnson presented a simple approximation algorithm for this problem; the algorithm is detailed in Fig. 1. Intuitively, it constructs an approximately maximum independent set by repeating the following step: Find the node v with the smallest degree (i.e., that has the fewest neighbors), add v to the independent set, and delete v and all neighboring nodes from the graph. Choosing the node with the smallest degree minimizes the number of candidate nodes eliminated by each choice of node to include in the independent set.

This approximation algorithm for MIS suggests the following approximation algorithm for BDP: Given an instance (G, b, s, t) of BDP, we construct an undirected graph $\hat{G}$ whose nodes denote paths from s to t of length at most b in G, and where two nodes in $\hat{G}$ are connected if and only if the paths they represent in G are not disjoint. Since there is a one-to-one correspondence between independent sets in $\hat{G}$ and sets of disjoint paths from s to t of length at most b in G, we can employ Johnson's algorithm on $\hat{G}$ to find an approximate solution to the BDP problem on G. It is instructive to note that, by applying Johnson's algorithm to $\hat{G}$, we are choosing paths from s to t in G that intersect the fewest other paths from s to t.

The proof of [10, Theorem 3.1] shows that the algorithm in Fig. 1 is guaranteed to find an independent set of size at least $\lfloor \log_k \hat{n} \rfloor$ in any undirected graph $\hat{G} = (\hat{V}, \hat{E})$, where

1. Set $U = V$ and
$$D = \begin{cases} \{s \to t\} & \text{if } s \to t \in E \\ \emptyset & \text{otherwise} \end{cases}$$
2. Find a path $p = s \to c_1 \to \ldots \to c_\ell \to t$, $1 \leq \ell \leq b$, in the subgraph induced by U such that $\Phi(p)$ is the minimum over all such paths. If no such path exists, then halt and return D.
3. Set $D = D \cup \{p\}$ and $U = U \setminus \{c_1, \ldots, c_\ell\}$, and go to 2.

Fig. 2. Approximation algorithm for BDP on instance $(G = (V, E), b, s, t)$.

$\hat{n} = |\hat{V}|$ and k is the smallest integer such that $\hat{V}$ can be partitioned into k independent sets (and, thus, the size of the maximum independent set is at least $\lceil \hat{n}/k \rceil$). We obtain the analogous result for BDP as a corollary, i.e., where $\hat{n}$ is the number of paths from s to t of length at most b. While this guarantee is weak, the algorithm seems to perform much better in practice, as we show in Section 3.3. The dominant cost in this algorithm is constructing $\hat{G}$, which requires $n^{O(b)}$ time and space if $G = (V, E)$ and $|V| = n$. As we argued previously, growth that is exponential in b is not necessarily a limiting factor for the applications that we are considering. Nevertheless, in the following section, we explore algorithms whose complexity grows polynomially in both n and b. In the rest of the paper, we refer to the algorithm of this section as *Independent Set*.

3.2 Approximating *Independent Set*

The algorithms in this section can be viewed as algorithms that approximate the behavior of the *Independent Set* algorithm of the previous section. Recall the intuition behind that algorithm: At each step, choose the path from s to t that intersects the fewest other paths. The main cost in that algorithm is determining how many other paths that each path intersects; this is precisely the information contained in the undirected graph $\hat{G}$. So, a natural direction to speeding up this approach is to avoid this determination explicitly and to use other information to indicate at each step the path that is likely to intersect the fewest other paths.

Given an instance $(G = (V, E), b, s, t)$ of BDP, our algorithms then will proceed to efficiently find a path from s to t of length at most b that we have reason to believe will intersect the fewest other paths from s to t of length at most b. We will add this path to the set of disjoint paths we are generating, delete it and all incident edges from the graph, and repeat. For the moment, we abstract the function we use to choose a path as an evaluation function $\Phi(p)$ on paths p; i.e., we choose the path p that minimizes $\Phi(p)$. Thus, our algorithm executes as shown in Fig. 2.

In this paper, we consider the following evaluation functions for locating a path that is "likely" to intersect the fewest other paths.

1) *Length:* In each iteration of the algorithm, choose the path from s to t of shortest length (in the range $[1, \ldots, b]$), i.e., $\Phi(p)$ is the length of p. Intuitively, shorter paths have fewer nodes to share with other paths and, thus, should be likely to intersect fewer other paths. The

TABLE 1
GRAPHS USED IN TESTING

graph		G_5	G_{10}	G_{15}	G_{20}	G_{25}	G_{30}	G_{40}
nontrivial nodes		1,105	1,846	2,477	3,011	3,487	3,881	4,636
nontrivial edges		768	1,518	2,265	2,978	3,687	4,406	5,809
connected	$b = 5$	1,498	6,906	20,639	45,341	73,299	117,081	204,776
node pairs	$b = 10$	1,652	12,324	50,515	92,994	170,771		
	$b = 15$	1,652	14,263	54,522	95,115			
multiply	$b = 5$	10	242	2,010	5,692	9,509	17,847	33,079
connected	$b = 10$	12	430	6,240	14,184	22,649		
node pairs	$b = 15$	12	463	6,726	14,514			

shortest path from s to t can be found in $O(m)$ time using breadth-first search, where $m = |E|$.

2) *Degree:* For a path $p = s \rightarrow c_1 \rightarrow \ldots \rightarrow c_\ell \rightarrow t$, the *degree* of p is defined as

$$\deg(p) = \sum_{1 \leq i \leq \ell} \deg(c_i),$$

where the $\deg(c_i)$ denotes the degree of (i.e., the number of edges incident on) c_i. In each iteration of the algorithm, choose the path from s to t of length at most b with the smallest degree. Thus, $\Phi(p) = \deg(p)$. Intuitively, paths with lower degree offer fewer opportunities for other paths to cross them. The path of length at most b from s to t with the smallest degree can be found in $O(bm + bn \log(bn))$ time where $n = |V|$ and $m = |E|$, using a variation of Dijkstra's shortest path algorithm [4].

3) *Random:* Prior to executing the algorithm of Fig. 2, assign a random weight $w(c)$ to each $c \in V$. Define the weight of a path $p = s \rightarrow c_1 \rightarrow \ldots \rightarrow c_\ell \rightarrow t$ as

$$w(p) = \sum_{1 \leq i \leq \ell} w(c_i).$$

Then, let $\Phi(p) = w(p)$. There is little intuition as to why this choice of Φ should yield a path that intersects few other paths, and it is included primarily as a point of comparison for our empirical evaluation in Section 3.3. The path p minimizing $\Phi(p)$ can be found in $O(bm + bn \log(bn))$ time, using a similar variation of Dijkstra's shortest path algorithm.

Unlike the *Independent Set* algorithm of Section 3.1, the algorithm of Fig. 2, combined with any of the choices of $\Phi(p)$ described above, can offer no nontrivial guarantee of the cardinality of the set of disjoint paths that it will locate. This is because, for any of these choices for Φ, it is possible to construct classes of graphs that will foil this algorithm (in the case of *Random*, almost all of the time), causing it to return a set of disjoint paths of cardinality at most one for some s and t regardless of the actual number of disjoint paths there are from s to t. The advantage of this algorithm, however, is its efficiency. Since each execution of Step 3 removes at least one node from U, Step 2 can be executed at most $n = |V|$ times. Thus, the algorithm instantiated with $\Phi(p)$ being the length of p (*Length*) runs in $O(nm)$ time. If $\Phi(p) = \deg(p)$ (*Degree*) or $\Phi(p) = w(p)$ (*Random*), then the algorithm runs in $O(bnm + bn^2 \log(bn))$ time.

3.3 Empirical Results

Motivated by our PathServer application (see Section 5), we performed tests on the approximation algorithms described in Sections 3.1 and 3.2 to evaluate their accuracy on a number of different graphs. In order to measure their accuracy, for each test graph and for every ordered pair of nodes in the graph, we computed the number of disjoint paths from the first node to the second, both in actuality and according to each approximation algorithm. As the basis for the graphs in our tests, we used the PGP keyring held at the MIT PGP Key Service (pgp-public-keys@pgp.mit.edu) as of 21 November 1995. This keyring induces a graph consisting of 13,896 nontrivial edges (i.e., edges of the form $c_1 \rightarrow c_2$ for $c_1 \neq c_2$) and 7,529 nontrivial nodes (i.e., nodes with incident, nontrivial edges). Due to the size of this graph, it was not possible to evaluate the accuracy of our approximation algorithms on the entire graph. Doing so would require us to compute the *actual* number of disjoint paths between pairs of nodes, which is an exponential computation that far exceeds our resources for a graph of this size.

In an effort to evaluate the accuracy of our algorithms despite this hurdle and also to learn how our algorithms performed as a function of graph size, we used various subgraphs induced by selecting (nontrivial) edges randomly from the total graph. In the rest of this section, let G_x denote the subgraph that resulted by selecting each edge from the whole graph with probability $\frac{x}{100}$. Some statistics for graphs we used are shown in Table 1. "Connected node pairs" are the number of node pairs (s, t) such that **bdp**$(G_x, b, s, t) \geq 1$. "Multiply connected node pairs" are the number of node pairs (s, t) such that **bdp**$(G_x, b, s, t) \geq 2$.

The results of our tests are shown in Table 2. This table characterizes the error of each algorithm. For each graph $G \in \{G_5, G_{10}, G_{15}, G_{20}, G_{25}, G_{30}, G_{40}\}$, each path bound $b \in \{5, 10, 15\}$ ($b \in \{5, 10\}$ for G_{25} and $b = 5$ for G_{30}, G_{40}), and each algorithm $A \in \{\textit{Independent Set, Length, Degree, Random}\}$ we computed the following values:

err: among all pairs (s, t) such that **bdp**$(G, b, s, t) \geq 2$, the fraction for which $A(G, b, s, t) \neq$ **bdp**(G, b, s, t) (recall that **bdp**$(G, b, s, t) = A(G, b, s, t)$ if **bdp**$(G, b, s, t) \leq 1$);

avg: for all (s, t) such that $A(G, b, s, t) \neq$ **bdp**(G, b, s, t), the average value of **bdp**$(G, b, s, t) - A(G, b, s, t)$;

max: for all (s, t) such that $A(G, b, s, t) \neq$ **bdp**(G, b, s, t), the maximum value of **bdp**$(G, b, s, t) - A(G, b, s, t)$.

Equations for each of these values are given at the bottom of Table 2. Note that these measures pertain only to those

TABLE 2
ACCURACY RESULTS FOR MULTIPLY CONNECTED PAIRS

		Independent Set			Length			Degree			Random		
		err	avg	max	err	avg	max	err	avg	max	err	avg	max
G_5	$b=5$	.000	$\perp$	0	.000	$\perp$	0	.000	$\perp$	0	.000	$\perp$	0
	$b=10$	.000	$\perp$	0	.000	$\perp$	0	.000	$\perp$	0	.000	$\perp$	0
	$b=15$	.000	$\perp$	0	.000	$\perp$	0	.000	$\perp$	0	.000	$\perp$	0
G_{10}	$b=5$	.000	$\perp$	0	.008	1.000	1	.004	1.000	1	.016	1.000	1
	$b=10$	.000	$\perp$	0	.025	1.000	1	.032	1.000	1	.051	1.000	1
	$b=15$	.000	$\perp$	0	.021	1.000	1	.030	1.000	1	.036	1.000	1
G_{15}	$b=5$	.000	$\perp$	0	.095	1.000	1	.067	1.000	1	.089	1.000	1
	$b=10$	.000	1.000	1	.152	1.004	2	.123	1.003	2	.160	1.007	2
	$b=15$	.002	1.000	1	.076	1.004	2	.053	1.003	2	.091	1.006	2
G_{20}	$b=5$	.001	1.000	1	.140	1.011	2	.100	1.005	2	.132	1.011	2
	$b=10$	.005	1.000	1	.168	1.018	2	.119	1.009	2	.165	1.019	2
	$b=15$	.008	1.000	1	.109	1.008	2	.079	1.003	2	.119	1.019	2
G_{25}	$b=5$	.002	1.000	1	.164	1.017	2	.101	1.009	2	.156	1.017	2
	$b=10$	.009	1.025	2	.167	1.017	2	.112	1.011	2	.168	1.028	3
G_{30}	$b=5$	.002	1.000	1	.192	1.037	3	.105	1.012	2	.183	1.030	3
G_{40}	$b=5$	.005	1.006	2	.237	1.064	3	.115	1.028	3	.224	1.062	3

$M = \{(s,t) : A(G,b,s,t) \neq \mathsf{bdp}(G,b,s,t)\}$

$\mathsf{err} = |M|/|\{(s,t) : \mathsf{bdp}(G,b,s,t) \geq 2\}|$

$\perp$ = undefined (division by zero)

$\mathsf{avg} = [\sum_{(s,t)\in M} \mathsf{bdp}(G,b,s,t) - A(G,b,s,t)]/|M|$

$\mathsf{max} = \max_{(s,t)\in M}\{\mathsf{bdp}(G,b,s,t) - A(G,b,s,t)\}$

pairs of nodes that are multiply connected which, according to Table 1, is the vast minority of node pairs. On the remaining vast majority of node pairs, each of the algorithms is guaranteed to return a true maximum set of disjoint paths (of cardinality zero or one).

While Table 2 is inconclusive, some trends seem to emerge; see Fig. 3. First, and not surprisingly, *Independent Set* seems to be more accurate than any of the other algorithms. Second, *Degree* seems to become more accurate than *Length* and *Random* as the graph size increases. Setting aside *Random* (it is slower than *Length* and no more accurate), it appears that we can rank the algorithms on accuracy in the order *Independent Set, Degree, Length*. On the other hand, these algorithms are ranked in terms of performance in exactly the opposite order (and our empirical observations support this ordering), with *Independent Set* becoming costly quickly as a function of *b*. For PathServer (see Section 5), we therefore typically use the *Degree* algorithm as a good balance between accuracy and interactive performance. Another observation that we can make from Table 2 is that when our *Length, Degree*, and *Independent Set* algorithms erred, they usually missed a maximum set of disjoint paths by only one (see the "**avg**" columns). If this apparent stability of the error magnitude continues as the graph grows, it is conceivable that we could predict with high probability the error of our algorithms for a given graph size. This would be an interesting contrast to Theorem 1 proved at the beginning of this section, stating that no absolute bound on error could be guaranteed. Further tests are required, however, before we can draw any such conclusions.

More generally, we caution the reader that the numbers of Table 2 are no guarantee of good accuracy for all graphs, or even for all graphs induced by certification systems. We consider an important open problem to be the discovery of efficient approximation algorithms for BDP for which some nontrivial accuracy can be proven. An equally important goal is to find algorithms (perhaps our own) that can be empirically shown to provide accurate results on a wider range of graphs that are characteristic of those we expect to see in practice. Since the characteristics of such graphs are yet to be conclusively identified, however, even generating

such test graphs remains an open problem. We hope that work such as [17] will shed light on this issue.

4 BOUNDED CONNECTIVE PATHS

To our knowledge, the Bounded Connective Paths (BCP) problem of Section 2 has not been considered from the algorithmic and complexity-theoretic viewpoints in the past. Given its close relationships to BDP [5], [15], we might be inclined to think that analogs of the results and techniques for BDP in the previous section could be developed for BCP. In fact, this is somewhat true and, in this section, we summarize these results and techniques.

Let $\mathbf{bcp}(G, b, s, t)$ denote the *b*-connectivity from *s* to *t* in *G*. With regards to complexity, we show below that computing whether $\mathbf{bcp}(G, b, s, t) \geq k$ for some given *k* is coNP-complete, i.e., that computing whether $\mathbf{bcp}(G, b, s, t) < k$ is NP-complete. As with NP-completeness, the fact that computing whether $\mathbf{bcp}(G, b, s, t) \geq k$ is coNP-complete is widely believed to imply that $\mathbf{bcp}(G, b, s, t)$ cannot be computed in polynomial time [6].

THEOREM 2. *Determining whether $\mathbf{bcp}(G, b, s, t) < k$ is NP-complete for any fixed $b \geq 4$.*

PROOF. The proof is by transformation from the following problem, called Maximum 2-Satisfiability, which was proven NP-complete in [7]. In the statement of this problem, a *literal* is a variable or its negation (e.g., *x* or $\bar{x}$), and a *clause* is a disjunction of literals.

Maximum 2-Satisfiability (2SAT):
Given: A set *U* of variables, a collection *C* of clauses over *U* such that each clause in *C* consists of two literals, and a positive integer $k \leq |C|$.
Problem: Is there a truth assignment for *U* that simultaneously satisfies at least *k* of the clauses in *C*?

We prove the result only for $b = 4$; the extension to larger fixed *b* is straightforward. Given an instance (U, C, k) of 2SAT, we construct a graph *G* with distinguished nodes *s* and *t* and positive integer k' such that there is a set of $k' - 1$ nodes whose removal eliminates
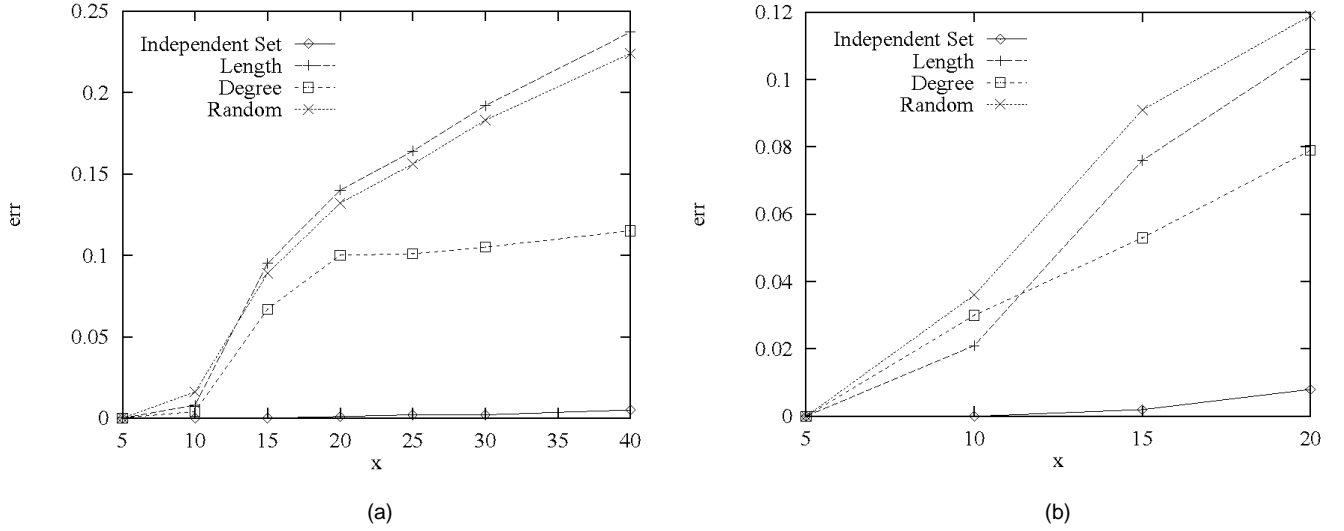
Fig. 3. Accuracy for each graph G_x. (a) $b = 5$. (b) $b = 15$.

all paths from s to t of length four or less if and only if there is a truth assignment to the variables of U that satisfies at least k of the clauses in C. Let $c = |C|$ and $n = |U|$. We construct the graph as follows. For each variable x, the subgraph shown in Fig. 4a is included within G. An additional subgraph is added (superimposed) per clause, where the subgraph depends on the number of negative literals in the clause. If there is one negative literal in the clause, say $x \vee \overline{y}$, then the subgraph shown in Fig. 4b is added. If there are zero negative literals in the clause, say $x \vee y$, then the subgraph shown in Fig. 4c is added. Finally, if there are two negative literals in the clause, say $\overline{x} \vee \overline{y}$, then the subgraph shown in Fig. 4d is added. In the last two cases, i.e., zero or two negative literals, we say the clause is *monotonic*. Let m denote the number of monotonic clauses.

We claim that there is a truth assignment for U that satisfies at least k of the clauses in C if and only if all paths from s to t of length at most $b = 4$ can be eliminated by removing $cn + m + c - k$ nodes (i.e., $k' = cn + m + c - k + 1$). First, suppose that there is a truth assignment for U that satisfies at least k of the clauses in C. For each variable $x \in U$, if x is true, then remove nodes $x_1, \ldots, x_c$ from G; otherwise, remove $\overline{x}_1, \ldots, \overline{x}_c$. In total, this removes cn nodes. Now, partition the clauses into the satisfied monotonic C_{sm}, unsatisfied monotonic C_{um}, satisfied nonmonotonic C_{sn}, and unsatisfied nonmonotonic C_{un}. For each member of C_{sm} of the form $x \vee y$, if x is true, then remove $f_{x \vee y}$ and, otherwise, remove $g_{x \vee y}$. For each element of C_{sm} of the form $\overline{x} \vee \overline{y}$, if x is false, then remove $l_{\overline{x} \vee \overline{y}}$ and, otherwise, remove $g_{\overline{x} \vee \overline{y}}$. Thus, C_{sm} contributes $|C_{sm}|$ removals. For each element of C_{um}, say $x \vee y$ (resp., $\overline{x} \vee \overline{y}$), remove both $f_{x \vee y}$ and $g_{x \vee y}$ (resp., $l_{\overline{x} \vee \overline{y}}$ and

$g_{\overline{x} \vee \overline{y}}$) for a total of $2|C_{um}|$ removals. Finally, for each element of C_{un}, say $x \vee \overline{y}$, remove $d_{x \vee \overline{y}}$, for a total of $|C_{un}|$ removals. Summing these removals, we get

$$cn + |C_{sm}| + 2|C_{um}| + |C_{un}| =$$
$$cn + (|C_{sm}| + |C_{um}|) + (|C_{um}| + |C_{un}|) \leq$$
$$cn + m + (c - k).$$

It is simple to verify by inspection of Fig. 4 that we have disconnected all paths from s to t of length at most four.

Now, suppose that it is possible to eliminate all paths from s to t of length at most four by removing $cn + m + c - k$ nodes from G. In order to eliminate all such paths, either all nodes $x_1, \ldots, x_c$ or all nodes $\overline{x}_1, \ldots, \overline{x}_c$ must be removed for each variable x, which accounts for a total of cn removals. For each monotonic clause, say $x \vee y$ (resp., $\overline{x} \vee \overline{y}$), it is necessary to remove at least one of $d_{x \vee y}$, $e_{x \vee y}$, $f_{x \vee y}$, and $g_{x \vee y}$ (resp., $d_{\overline{x} \vee \overline{y}}$, $e_{\overline{x} \vee \overline{y}}$, $l_{\overline{x} \vee \overline{y}}$, and $g_{\overline{x} \vee \overline{y}}$) to eliminate all paths of length four from s to t. Thus, we now can characterize where $cn + m$ of the removals must be. For each variable x, set x to true if all of $x_1, \ldots, x_c$ are removed and to false if all of $\overline{x}_1, \ldots, \overline{x}_c$ are removed. (Not all of both $x_1, \ldots, x_c$ and $\overline{x}_1, \ldots, \overline{x}_c$ could be removed, as this would imply $cn + m + c > cn + m + c - k$ removals in total.) If there are fewer than k clauses satisfied, then more than $c - k$ additional removals would be required to eliminate all paths from s to t of length at most four, namely one per unsatisfied clause in the subgraph corresponding to the clause. Thus, at least k clauses must be satisfied.

Finally, determining whether **bcp**$(G, b, s, t) < k$ is in NP, since, given a set of $k - 1$ nodes, it is possible to verify in polynomial time that their removal eliminates all paths from s to t of length at most b. □
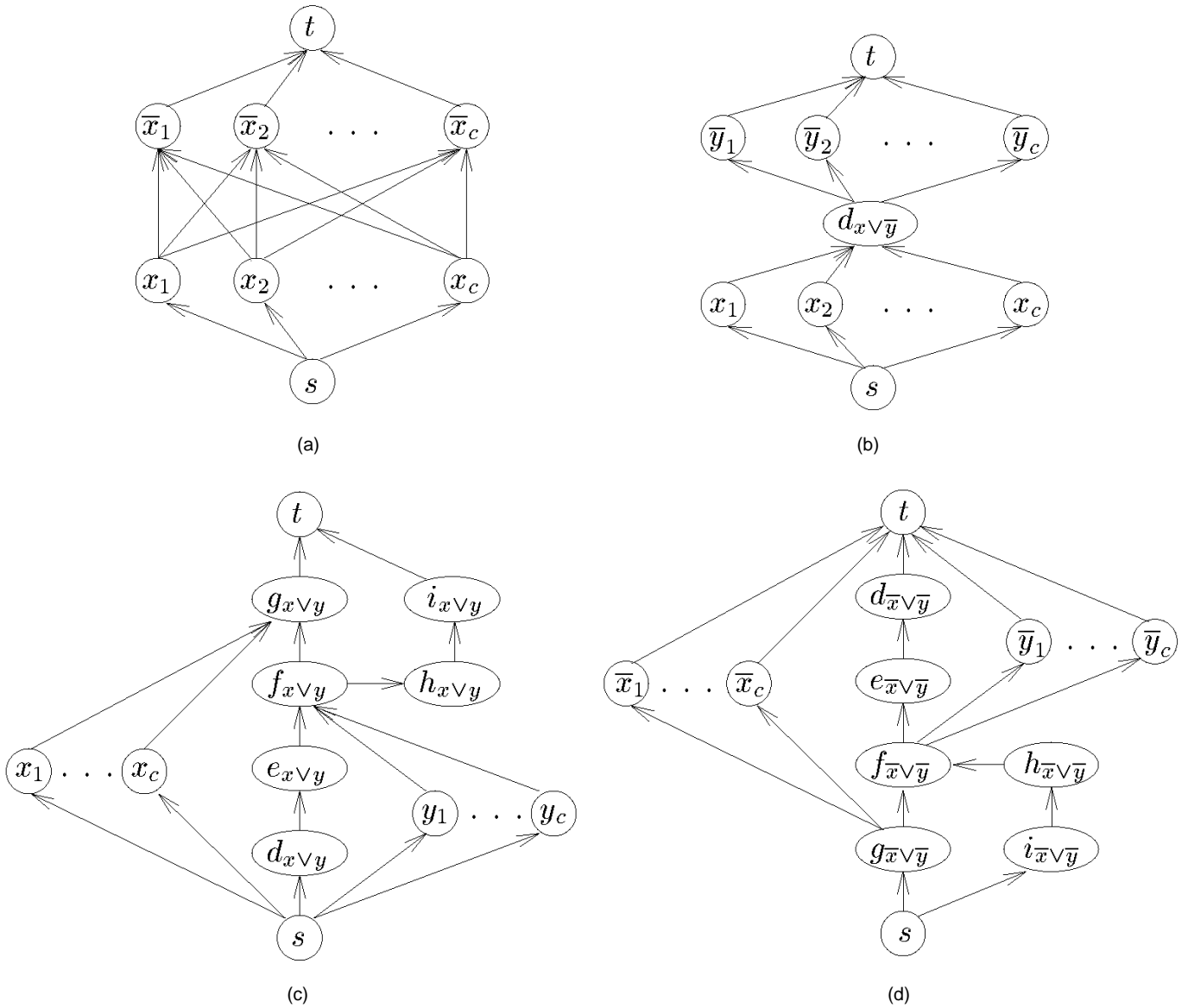
Fig. 4. Component graphs for proof of Theorem 2.

Theorem 2 also has other implications that we care about. Following the widely held belief that NP ≠ coNP, it implies that there is no polynomial-time algorithm for verifying that there is a set of k-connective b-bounded paths from s to t, even if k and the set of paths (or any other information) is given. For a service like PathServer, this means that clients that request bounded connective paths will either have to trust PathServer that the returned paths are k-connective (for the value k that it returns) or be prepared to perform a possibly exponential computation to verify this assertion. This is one of the main differentiators between bounded connective paths and bounded disjoint paths, the latter of which can be easily verified by clients.

Due to Theorem 2, we now turn to approximating **bcp**(G, b, s, t). Below, we describe the algorithm that we use to approximate a solution to BCP in PathServer. To describe it, we first introduce some concepts. For a set D of paths, the subgraph *induced by* D is the graph whose nodes and edges are those that occur on some path in D. The *(s, t, b)-*

closure of a set D of paths from s to t (or just the b-closure when s and t are understood) is the set of b-bounded paths from s to t in the subgraph induced by D.

Given G, b, s, and t, our algorithm returns a set D of b-bounded paths from s to t and the value k such that the b-closure of D is k-connective. Note that the subgraph induced by D is identical to that induced by its b-closure and, thus, D is indistinguishable from its b-closure when graphically displayed (as by PathServer; see Section 5). The set D of b-bounded paths is constructed in polynomial time, but finding k still takes time exponential in the size of the graph in the worst case. Fortunately, this exponential computation executes on the typically small subgraph induced by D and, so, it almost always completes with brief delay. The algorithm is outlined in Fig. 5. It begins by locating a set D of disjoint b-bounded paths from s to t using one of the algorithms described in Section 3. The algorithm then repeatedly augments D with other b-bounded paths from s to t. At each step, the augmenting path is chosen to minimize some criterion

1. Find a set D of disjoint b-bounded paths from s to t using one of the algorithms described in Section III. If $D = \emptyset$, then return $<0, D>$.
2. Find a path $p = s \rightarrow c_1 \rightarrow \ldots \rightarrow c_\ell \rightarrow t$, $1 \leq \ell \leq b$, in G such that $\Phi(p, D)$ is the minimum over all such paths.
3. Set $D = D \cup \{p\}$. If termination condition Ψ is not met, go to 2; otherwise go to step 4.
4. Compute k such that the (s, t, b)-closure of $D \setminus \{s \rightarrow t\}$ is k-connective, and return $<k, D>$.

Fig. 5. Approximation algorithm for BCP on instance ($G = (V, E)$, b, s, t).

$\Phi(p, D)$. The criterion that we presently use is based on the *path degree* pdeg(c, D) of node c with respect to D, which is the degree of c in the subgraph induced by D. Our criterion is to choose the path that minimizes the sum of the path degrees of its nodes, i.e., $\Phi(p, D) = \sum_{c \in p} \text{pdeg}(c, D)$. The repeated augmentation of D terminates on some condition Ψ. In our present implementation, this condition is met when $|D| = \frac{bd}{2}$, where d is the original size of D at the end of Step 1 in Fig. 5. This choice of Ψ is motivated by the fact that the b-connectivity from s to t is at most $\frac{b}{2}$ times the size of a maximum set of disjoint paths from s to t [15].

Once the condition Ψ is met, the algorithm determines k such that the (s, t, b)-closure of D is k-connective (ignoring the path $s \rightarrow t$ if it is present in D). This algorithm is essentially brute-force, iterating through sets of nodes and testing if a path in the b-closure of D would continue to exist if those nodes were removed. To optimize this algorithm, any path in D that is disjoint from all other paths in D is removed before the search begins, as each such path contributes exactly one to the final value of k. In the other paths, only nodes with in-degree or out-degree greater than one in the subgraph induced by D need be included in sets whose removal is tested.

At the time of this writing, we are less experienced with the BCP problem and this algorithm than we are with BDP and its approximation algorithms. Moreover, our initial experiments indicate that an analysis comparable to that for our BDP algorithms will require considerably more computational resources than we have available to us. Thus, we do not have sufficient data to report on the accuracy of this algorithm, or to determine if our conditions $\Phi(p, D)$ and Ψ can be improved. This algorithm seems to offer adequate performance in most cases, especially when b is small, but also threatens to become intractable as the graph grows larger. The design of better algorithms to approximate BCP, as well as approaches for evaluating algorithms on realistic graphs, is a topic for future work.

5 THE PATHSERVER APPLICATION

The algorithms of Sections 3 and 4 are central to, and were motivated by, the PathServer application to which we have alluded. PathServer is a web-based service for finding bounded disjoint or connective paths from a source channel to a target channel. PathServer is currently implemented to work in the context of the PGP key management and encryption system, although it could be easily adapted to work with other types of public key management systems. PGP is one of the most popular civilian public key systems in the world today, due in no small part to the decentralized model of trust it supports. In PGP, users create signed certificates (statements) that bind semantics (e.g., a name and an e-mail address) to a public key. These statements, which taken together form a graph as described in Section 2, can be disseminated through personal communications, on electronic newsgroups, or, as is often the case, via a number of PGP servers spread across the world. Authentication of a message—i.e., of the public key (channel) that signed (stated) it—takes place as described in Section 1, with a user finding a path of channels by which it can authenticate the channel of interest. PGP allows a user to specify a bound on the length of paths she is willing to accept. PGP also provides primitive support for using multiple paths. More precisely, it provides interfaces to specify keys as being "completely" or "marginally" trusted for certification and for specifying how many completely trusted or marginally trusted signatures are required to authenticate a channel.

PGP, however, lacks the ability to search for all independent information about a key that is likely to be useful to the requesting party. Following our thesis that the appropriate information to provide is disjoint or connective paths of bounded length from a channel that the requesting principal trusts to the channel of interest, we have implemented PathServer to provide this information. Our service provides a World Wide Web interface by which a user can submit a path length bound, PGP key identifiers for a source key (e.g., her own) and a target key, and a choice of disjoint or connective paths, and will receive in real time a display of the requested paths. An example is shown in Fig. 6, which is the result of specifying disjoint paths of length at most eight with a source key identifier of C7A966DD and a target key identifier of A40B96D9. The service generates this information using a graph built from a database of PGP certificates, which our service updates periodically from other PGP key servers throughout the world.

It is important to note that PathServer need not be trusted (modulo certain caveats that were discussed in Section 4): A user can verify the information retrieved from our service by retrieving the appropriate certificates from any PGP database (including PathServer) and verifying for herself that the paths exist using the existing PGP program. Thus, the information retrieved from our service can merely be considered as hints to enable independent corroboration of the semantics associated with a given target key. If a user trusts PathServer and, thus, does not cryptographically verify the existence of the paths claimed by PathServer, then at least the user should connect to PathServer using an authenticated connection.

User response to PathServer indicates that PathServer is one instance of a broader class of services that might be useful for authenticating keys. We have received, and ourselves developed, numerous ideas for improving the service, some of which are listed below. Many such extensions are technically feasible, but pose challenges to maintaining a simple user interface.
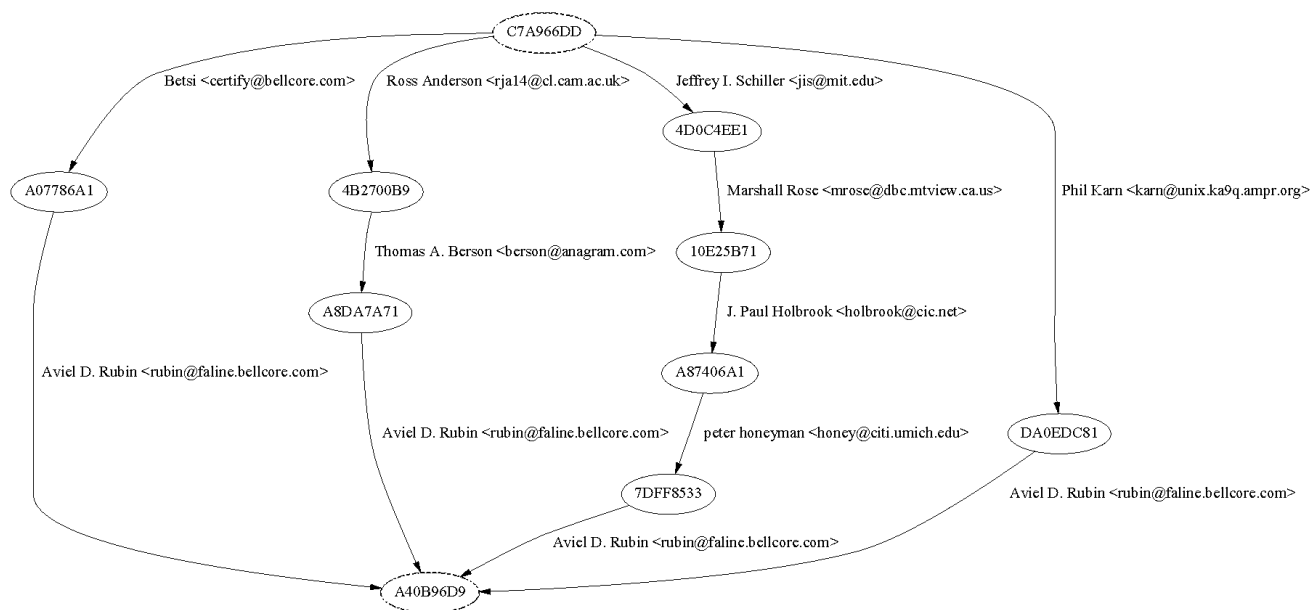
Fig. 6. An output of PathServer (disjoint paths, source key id = C7A966DD, target key id = A40B96D9).

Multiple Sources. The service could allow the user to specify multiple source nodes in a query, as would be appropriate if the user has numerous source keys it trusts. For example, for the case of bounded disjoint paths, the returned paths would begin at any one of the specified source nodes. The paths returned using multiple sources are not necessarily the same as would be obtained if individual queries were submitted, once for each source node, since a node might be common to paths returned in each of the individual queries.

Node and Edge Exclusion. The service could allow the user to choose nodes or edges to exclude in the algorithm for finding paths. This type of search could be used to implement a policy particular to individual edges or nodes. For example, one might want to exclude nodes representing weak keys (e.g., due to key length or signing algorithm) or one might want to exclude edges whose labels indicate undesirable organizational affiliations.

More Complex Policies. It might be possible to accommodate more complex exclusion policies than simple node and edge exclusion. For example, it might be desirable to exclude an edge from returned paths if the edge "conflicts" with another edge, i.e., if there exists another edge to the same node with an inconsistent label.

Graph Management Algorithms. The service could provide guidance to the user in the case that the paths returned by PathServer are insufficient for the user. For example, if the user desired seven disjoint paths from a source to a target but got back only four, then PathServer might recommend sets of edges (certificates) that, if added to the graph, would enable seven disjoint paths to be found from the source to the target.

Certificate Formats and Applications. PathServer could be extended to graphs containing certificate formats other than PGP certificates, such as X.509 certificates. This is particularly interesting if keys in X.509 certificates may cross certify other keys. Also, PathServer could be extended for use in other applications such as interorganizational workflow. For workflow applications, PathServer might employ many types of certificates, including certificates for identity, delegation, and authorization.

6 RELATED WORK

There has been much work on the problem of gaining increased assurance in the authentication provided by paths of channels. Much of this work has focused on assigning numerical measures of trustworthiness to paths or collections of paths (e.g., [23], [1], [16], [19], [14]). These efforts have recognized that shorter paths and multiple paths lend additional credibility to the authentication of a channel and the derived numerical measures tend to reflect these observations. Our work complements this research by providing algorithms and tools to efficiently locate as many independent paths as possible, which can serve as input to such evaluation functions. Moreover, since the initial conference publication of this work [18], the number of "independent" paths to a channel, as defined in the present paper, has itself been explored as a numerical measure of assurance for the authentication of that channel [19], [14]. In particular, Levien and Aiken [14] prove that this measure provides optimal or near-optimal resilience to key compromise of various forms.

Prior work on locating paths of channels typically focuses on finding a single path to a channel and either assumes a known "topology" regarding what channels make statements about others [2], [8], [13] or exhibits exponential worst-case complexity as a function of the number of channels and statements [23], [24], [25]. A contribution of our work is to look beyond a single path to locate a collection of paths with desired independence properties, without assuming a known topology on the relationships between

channels and often without suffering from exponential complexity.

In the context of PGP, there have been efforts to gather statistics about the graph of channels (public keys) induced by PGP certificates worldwide [17]. This work focuses on characterizing the structure of the graph and, in particular, identifying its strongly connected components, determining mean and maximum shortest path distances between channels, and identifying channels in the graph that are central to its connectivity. Our work differs both in its goals—i.e., increasing assurance in authentication of any channel of interest (versus characterizing the structure of the graph)—and in its focus on locating independent paths to channels. Our PathServer service highlights both of these goals.

7 CONCLUSION

In this paper, we presented a foundation for resilient authentication of channels (e.g., cryptographic keys) in large scale systems. Resilience is achieved by employing multiple "independent" paths of channels to authenticate the target channel, thereby preventing the compromise of a limited number of channels from causing authentication to fail.

We presented two notions of "independence". In the first, a set of bounded paths are "independent" if they are pairwise disjoint, i.e., if no two paths share a common channel. In the second, a set of bounded paths are "independent" if the removal of k channels is necessary to disconnect all of them. Computing the number of bounded disjoint paths from the source to the target is known to be NP-hard, and here we showed that computing the maximum k such that the source and target are bounded k-connected is coNP-hard. Furthermore, practical instances of these problems that we hope to address induce graphs of sufficient size to make this a severe limitation. We presented heuristics to approximate solutions to these problems and evaluated their effectiveness on actual data.

Finally, we discussed the implementation of the Path-Server system. PathServer uses our heuristics to find both bounded disjoint and bounded connective paths from a trusted public key to a key in question. We also discussed extensions to PathServer that might be useful in practice.

A direction for future work is to identify better algorithms for finding bounded disjoint and bounded connective paths. David Johnson suggested computing a maximum flow (e.g., [12]) with capacity-constrained nodes for finding the number of disjoint paths from the source to the target. A maximum flow is not guaranteed to include only paths (or, for that matter, any paths) of length at most the specified path bound, even if run on a restricted graph consisting of only those nodes that are within the path bound from the source or target. It remains to be seen, however, whether this would be a problem in practice. Another promising direction is to find metrics of assurance for authentication that build on our notions of path independence yet take into account characteristics of individual channels on those paths.

REFERENCES

[1] T. Beth, M. Borcherding, and B. Klein, "Valuation of Trust in Open Networks," *Proc. Computer Security—ESORICS '94*, D. Gollman, ed., pp. 3-18, 1994.
[2] A.D. Birrell, B.W. Lampson, R.M. Needham and M.D. Schroeder, "A Global Authentication Service without Global Trust," *Proc. 1986 IEEE Symp. Security and Privacy*, pp. 223-230, Apr. 1986.
[3] International Telegraph and Telephone Consultative Committee (CCITT), *The Directory—Authentication Framework, Recommendation X.509*, 1988.
[4] E.W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, pp. 269-271, 1959.
[5] R.C. Entringer, D.E. Jackson, and P.J. Slater, "Geodetic Connectivity of Graphs," *IEEE Trans. Circuits and Systems*, vol. 24, no. 8, pp. 460-463, Aug. 1977.
[6] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* New York: W.H. Freeman and Company, 1979.
[7] M.R. Garey, D.S. Johnson, and L. Stockmeyer, "Some Simplified NP-Complete Graph Problems," *Theoretical Computer Science*, vol. 1, pp. 237-267, 1976.
[8] V.D. Gligor, S. Luan, and J.N. Pato, "On Inter-Realm Authentication in Large Distributed Systems," *Proc. 1992 IEEE Symp. Research in Security and Privacy*, pp. 2-17, May 1992.
[9] A. Itai, Y. Perl, and Y. Shiloach, "The Complexity of Finding Maximum Disjoint Paths with Length Constraints," *Networks*, vol. 12, pp. 277-286, 1982.
[10] D.S. Johnson, "Worst Case Behavior of Graph Coloring Algorithms," *Proc. Fifth Southeastern Conf. Combinatorics, Graph Theory, and Computing*, pp. 513-527, Feb. 1974.
[11] S. Kent, "Internet Privacy Enhanced Mail," *Comm. ACM*, vol. 36, no. 8, pp. 48-60, Aug. 1993.
[12] V. King, S. Rao, and R. Tarjan, "A Faster Deterministic Maximum Flow Algorithm," *Proc. Third ACM Symp. Discrete Algorithms*, pp. 157-164, 1992.
[13] B. Lampson, M. Abadi, M. Burrows and E. Wobber, "Authentication in Distributed Systems: Theory and Practice," *ACM Trans. Computer Systems*, vol. 10, no. 4, pp. 265-310, Nov. 1992.
[14] R. Levien and A. Aiken, "Attack-Resistant Trust Metrics for Public Key Certification," *Seventh USENIX Security Symp. Proc*, pp. 229-242, Jan. 1998.
[15] L. Lovász, V. Neumann-Lara, and M. Plummer, "Mengerian Theorems for Paths of Bounded Length," *Periodica Mathematica Hungarica*, vol. 9, no. 4, pp. 269-276, 1978.
[16] U. Maurer, "Modelling a Public-Key Infrastructure," *Computer Security—ESORICS '96*, E. Bertino, H. Kurth, G. Martella, and E. Montolivo, eds., 1996.
[17] N. McBurnett, "PGP Web of Trust Statistics," http://bcn.boulder.co.us/~neal/pgpstat/, 1996.
[18] M.K. Reiter and S.G. Stubblebine, "Path Independence for Authentication in Large-Scale Systems," *Proc. Fourth ACM Conf. Computer and Comm. Security*, pp. 57-66, Apr. 1997.
[19] M.K. Reiter and S.G. Stubblebine, "Toward Acceptable Metrics of Authentication," *Proc. 1997 IEEE Symp. Security and Privacy*, pp. 10-20, May 1997.
[20] D. Ronen and Y. Perl, "Heuristics for Finding a Maximum Number of Disjoint Bounded Paths," *Networks*, vol. 14, pp. 531-544, 1984.
[21] F.B. Schneider, "Implementing Fault-Tolerant Services Using the State Machine Approach: A Tutorial," *ACM Computing Surveys*, vol. 22, no. 4, pp. 299-319, Dec. 1990.

[22] J.W. Suurballe, "Disjoint Paths in a Network," *Networks*, vol. 4, pp. 125-145, 1974.
[23] A. Tarah and C. Huitema, "Associating Metrics to Certification Paths," *Proc. Computer Security—ESORICS '92*, pp. 175-189, 1992.
[24] R. Yahalom, B. Klein and T. Beth, "Trust Relationships in Secure Systems—A Distributed Authentication Perspective," *Proc. 1993 IEEE Symp. Research in Security and Privacy*, pp. 150-164, May 1993.
[25] R. Yahalom, B. Klein and T. Beth, "Trust-Based Navigation in Distributed Systems," *Computing Systems*, vol. 7, no. 1, pp. 45-73, 1994.
[26] P. Zimmerman, *The Official PGP User's Guide*. MIT Press, 1995.

Michael K. Reiter received the BS degree in mathematical sciences from the University of North Carolina at Chapel Hill in 1989 and the MS and PhD degrees in computer science from Cornell University in 1991 and 1993, respectively. From 1993 to 1998, he conducted research on security and fault-tolerance in distributed systems at AT&T Labs—Research (formerly AT&T Bell Labs). In 1998, he joined Bell Labs, Lucent Technologies as department head of the Secure Systems Research Department, where he is continuing his research program on computer security and fault tolerance.

Stuart G. Stubblebine received the BS degree in computer science and mathematics from Vanderbilt University in 1983, the MS degree from the University of Arizona in electrical engineering in 1988, and the PhD degree in electrical engineering from University of Maryland in 1992. From 1992 to 1994, he was a scientist at the University of Southern California's Information Sciences Institute and, in 1993, a research assistant professor in the Department of Computer Science. In 1994, he became a member of the technical staff at AT&T Bell Labs in Murray Hill, New Jersey. In 1996, he became a principal technical staff member at AT&T Labs–Research and remains an adjunct professor in the Computer Science Department of the University of Southern California.

Dr. Stubblebine's research interests are rooted in the design, analysis, and formal verification of cryptographic protocols. His recent accomplishments have been in the areas of private transactions, authentication and authorization protocols, protocol integrity, public key infrastructure, secure data structures, and recent-secure revocation in distributed systems.

Dr. Stubblebine is a member of the IEEE and ACM.