Computational Complexity of Traffic Hijacking under BGP and S-BGP*

M. Chiesa[1], G. Di Battista[1], T. Erlebach[2], and M. Patrignani[1]

[1] Dept. of Computer Science and Automation, Roma Tre University
[2] Dept. of Computer Science, University of Leicester

Abstract. Harmful Internet hijacking incidents put in evidence how fragile the Border Gateway Protocol (BGP) is, which is used to exchange routing information between Autonomous Systems (ASes). As proved by recent research contributions, even S-BGP, the secure variant of BGP that is being deployed, is not fully able to blunt traffic attraction attacks. Given a traffic flow between two ASes, we study how difficult it is for a malicious AS to devise a strategy for hijacking or intercepting that flow. We show that this problem marks a sharp difference between BGP and S-BGP. Namely, while it is solvable, under reasonable assumptions, in polynomial time for the type of attacks that are usually performed in BGP, it is NP-hard for S-BGP. Our study has several by-products. E.g., we solve a problem left open in the literature, stating when performing a hijacking in S-BGP is equivalent to performing an interception.

1 Introduction and Overview

On 24th Feb. 2008, Pakistan Telecom started an unauthorized announcement of prefix 208.65.153.0/24 [13]. This announcement was propagated to the rest of the Internet, which resulted in the *hijacking* of YouTube traffic on a global scale. Incidents like this put in evidence how fragile is the *Border Gateway Protocol (BGP)* [10], which is used to exchange routing information between Internet Service Providers (ISPs). Indeed, performing a hijacking attack is a relatively simple task. It suffices to issue a BGP announcement of a victim prefix from a border router of a malicious (or unaware) *Autonomous System (AS)*. Part of the traffic addressed to the prefix will be routed towards the malicious AS rather than to the intended destination. A mischievous variation of the hijacking is the *interception* when, after passing through the malicious AS, the traffic is forwarded to the correct destination. This allows the rogue AS to eavesdrop or even modify the transit packets.

In order to cope with this security vulnerability, a variant of BGP, called S-BGP [8], has been proposed, that requires a PKI infrastructure both to validate the correctness of the AS that originates a prefix and to allow an AS to sign its announcements to other ASes. In this setting an AS cannot forge announcements that do not derive from announcements received from its neighbors. However,

* An extended version of the paper is available on the arXiv Web site.

	AS-paths of any length	Bounded AS-path length	Bounded AS-path length and AS degree
Origin-spoofing	NP-hard (Thm. 1)	P(Thm. 2)	P
S-BGP	NP-hard	NP-hard (Thm. 3)	P(Thm. 4)

Table 1: Complexity of finding a HIJACK strategy in different settings.

[4] contains surprising results: (i) simple hijacking strategies are tremendously effective and (ii) finding a strategy that maximizes the amount of traffic that is hijacked is NP-hard both for BGP and for S-BGP.

In this paper we tackle the hijacking and interception problems from a new perspective. Namely, given a traffic flow between two ASes, how difficult is it for a malicious AS to devise a strategy for hijacking or intercepting at least that specific flow? We show that this problem marks a sharp difference between BGP and S-BGP. Namely, while it is polynomial time solvable, under reasonable assumptions, for typical BGP attacks, it is NP-hard for S-BGP. This gives new complexity related evidence of the effectiveness of the adoption of S-BGP. Also, we solve an open problem [4], showing when every hijack in S-BGP results in an interception. Tab. 1 summarizes our results. Rows correspond to different settings for a malicious AS m. The origin-spoofing setting (Sect. 2) corresponds to a scenario where m issues BGP announcements pretending to be the owner of a prefix. Its degree of freedom is to choose a subset of its neighbors for such a bogus announcement. This is the most common type of hijacking attack to BGP [1]. In S-BGP (Sect. 3) m must enforce the constraints imposed by S-BGP, which does not allow to pretend to be the owner of a prefix that is assigned to another AS. Columns of Tab. 1 correspond to different assumptions about the Internet. In the first column we assume that the longest *valley-free* path (i.e. a path enforcing certain customer-provider constraints) in the Internet can be of arbitrary length. This column has a theoretical interest since the length of the longest path (and hence valley-free path) observed in the Internet remained constant even though the Internet has been growing in terms of active AS numbers during the last 15 years [7]. Moreover, in today's Internet about 95% of the ASes is reached in 3 AS hops [7]. Hence, the second column corresponds to a quite realistic Internet, where the AS-path length is bounded by a constant. In the third column we assume that the number of neighbors of m is bounded by a constant. This is typical in the periphery of the Internet. A "P" means that a Polynomial-time algorithm exists. Since moving from left to right the setting is more constrained, we prove only the rightmost NP-hardness results, since they imply the NP-hardness results to their left. Analogously, we prove only the leftmost "P" results.

1.1 A Model for BGP Routing

As in previous work on interdomain hijacking [4], we model the Internet as a graph $G = (V, E)$. A vertex in V is an *Autonomous System (AS)*. Edges in E are *peerings* (i.e., connections) between ASes. A vertex owns one or more *prefixes*,

i.e., sets of contiguous IP numbers. The routes used to reach prefixes are spread and selected via BGP. Since each prefix is handled independently by BGP, we focus on a single prefix π, owned by a destination vertex d.

BGP allows each AS to autonomously specify which paths are forbidden (*import policy*), how to choose the best path among those available to reach a destination (*selection policy*), and a subset of neighbors to whom the best path should be announced (*export policy*). BGP works as follows. Vertex d initializes the routing process by sending *announcements* to (a subset of) its neighbors. Such announcements contain π and the *path* of G that should be traversed by the traffic to reach d. In the announcements sent from d such a path contains just d. We say that a path $P = (v_n \ \ldots \ v_0)$ is *available* at vertex v if v_n announces P to v. Each vertex checks among its available paths that are not filtered by the import policy, which is the best one according to its selection policy, and then it announces that path to a set of its neighbors in accordance with the export policy. Further, BGP has a loop detection mechanism, i.e., each vertex v ignores a route if v is already contained in the route itself.

Policies are typically specified according to two types of relationships [6]. In a *customer-provider* relationship, an AS that wants to access the Internet pays an AS which sells this service. In a *peer-peer* relationship two ASes exchange traffic without any money transfer between them. Such commercial relationships between ASes are represented by orienting a subset of the edges of E. Namely, edge $(u, v) \in E$ is directed from u to v if u is a customer of v, while it is undirected if u and v are peers. A path is *valley-free* if provider-customer and peer-peer edges are only followed by provider-customer edges.

The Gao-Rexford [3] Export-all (*GR-EA*) conditions are commonly assumed to hold in this setting [4]. **GR1**: G has no directed cycles that would correspond to unclear customer-provider roles. **GR2**: Each vertex $v \in V$ sends an announcement containing a path P to a neighbor n only if path $(n\ v)P$ is valley-free. Otherwise, some AS would provide transit to either its peers or its providers without revenues. **GR3**: A vertex prefers paths through customers over those provided by peers and paths through peers over those provided by providers. **Shortest Paths**: Among paths received from neighbors of the same class (customers, peers, and provider), a vertex chooses the shortest ones. **Tie Break**: If there are multiple such paths, a vertex chooses according to some tie break rule. As in [4], we assume that the one whose next hop has lowest AS number is chosen. Also, as in [2], to tie break equal class and equal length simple paths $P_1^u = (u\ v)P_1^v$ and $P_2^u = (u\ v)P_2^v$ at the same vertex u from the same neighbor v, if v prefers P_1^v over P_2^v, then u prefers P_1^u over P_2^u. This choice is called *policy consistent* in [2] and it can be proven that it has the nice property of making the entire set of policies considered in this paper policy consistent. **NE policy**: a vertex always exports a path except when GR2 forbids it to do so.

Since we assume that the GR-EA conditions are satisfied, then a (partially directed) graph is sufficient to fully specify the policies of the ASes. Hence, in the following a *BGP instance* is just a graph.

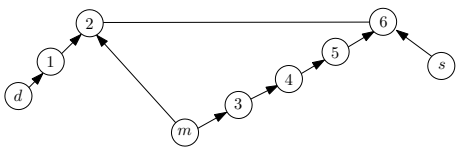


Fig. 1: A network for Examples 1 and 2.

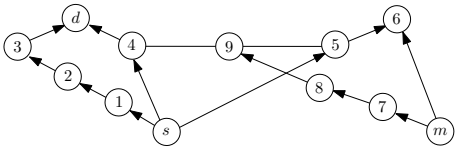


Fig. 2: A network for Example 3.

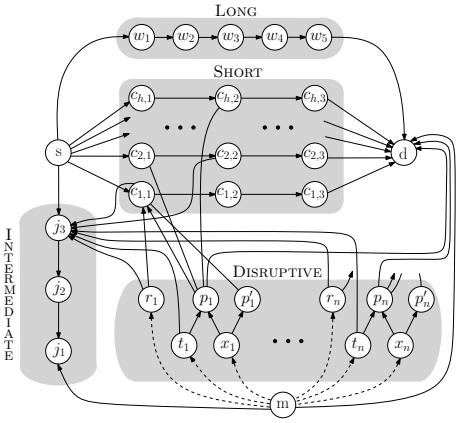


Fig. 3: Reduction of a constrained 3-SAT problem to the HIJACK problem when m has S-BGP cheating capabilities.

1.2 Understanding Hacking Strategies

We consider the following problem. A BGP instance with three specific vertices, d, s, and m are given, where such vertices are: the AS originating a prefix π, a source of traffic for π, and an attacker, respectively. All vertices, but m, behave correctly, i.e., according to the BGP protocol and GR-EA conditions. Vertex m is interested in two types of attacks: *hijacking* and *interception*. In the hijacking attack m's goal is to attract to itself at least the traffic from s to d. In the interception attack m's goal is to be traversed by at least the traffic from s to d.

In Fig. 1 $(2, 6)$ is peer-to-peer and the other edges are customer-provider. Prefix π is owned and announced by d. According to BGP, the traffic from s to d follows (s 6 2 1 d). In fact, 2 selects (1 d). Vertex 6 receives a unique announcement from d (it cannot receive an announcement with (5 4 3 m 2 1 d) since it is not valley-free). By cheating, (**Example 1**) m can deviate the traffic from s to d attracting traffic from s. In fact, if m pretends to be the owner of π and announces it to 2, then 2 prefers, for shortest-path, (2 m) over (2 1 d). Hence, the traffic from s to d is received by m following (s 6 2 m). A hijack!

Observe that m could be smarter (**Example 2**). Violating GR2, it can announce (2 1 d) to 3. Since each of 3, 4 and 5 prefers paths announced by customers (GR3), the propagation of this path is guaranteed. Therefore, 6 has two available paths, namely, (2 1 d) and (5 4 3 m 2 1 d). The second one is preferred because 5 is a customer of 6, while 2 is a peer of 6. Hence, the traffic from s to d is received by m following path (s 6 5 4 3 m). Since after passing through m the traffic reaches d following (m 2 1 d) this is an interception.

Fig. 2 allows to show a negative example (**Example 3**). According to BGP, the traffic from s to d follows (s 4 d). In fact, s receives only paths (4 d) and (1 2 3 d), both from a provider, and prefers the shortest one. Suppose that m wants to hijack and starts just announcing π to 6. Since all the neighbors of s are providers, s prefers, for shortest path, (4 d) over (5 6 m) (over (1 2 3 d)

over (4 9 8 7 m)) and the hijack fails. But m can use another strategy. Since (s 5 6 m) is shorter than (s 1 2 3 d), m can attract traffic if (4 d) is "disrupted" and becomes not available at s. This happens if 4 selects, instead of (d), a path received from its peer neighbor 9 (m may announce that it is the originator of π also to 7). However, observe that if 4 selects path (4 9 8 7 m) then 5 selects path (5 9 8 7 m) since it is received from a peer and stops the propagation of (s 5 6 m). Hence, s still selects path (s 1 2 3 d) and the hijack fails.

In order to cope with the lack of any security mechanism in BGP, several variations of the protocol have been proposed by the Internet community. One of the most famous, S-BGP, uses both origin authentication and cryptographically-signed announcements in order to guarantee that an AS announces a path only if it has received this path in the past.

The attacker m has more or less constrained *cheating capabilities*. 1. With the *origin-spoofing* cheating capabilities m can do the typical BGP announcement manipulation. I.e., m can pretend to be the origin of prefix π owned by d, announcing this to a subset of its neighbors. 2. With the *S-BGP* cheating capabilities m must comply with the S-BGP constraints. I.e.: (a) m cannot pretend to be the origin of prefix π; and (b) m can announce a path $(m\ u)P$ only if u announced P to m in the past. However, m can still announce paths that are not the best to reach d and can decide to announce different paths to different neighbors. In Example 2, m has S-BGP cheating capabilities.

In this paper we study the computational complexity of the HIJACK and of the INTERCEPTION problems. The HIJACK problem is formally defined as follows. **Instance**: A BGP instance G, a source vertex s, a destination vertex d, a manipulator vertex m, and a cheating capability for m. **Question**: Does there exist a set of announcements that m can simultaneously send to its neighbors, according to its cheating capability, that produces a stable state for G where the traffic from s to d goes to m? The INTERCEPTION problem is defined in the same way but changing "the traffic from s to d goes to m" to "the traffic from s to d passes through m before reaching d".

BGP policies can be so complex that there exist configurations that do not allow to reach any stable routing state (see, e.g., [5]). A routing state is *stable* if there exists a time t such that after t no AS changes its selected path. If the GR-EA conditions are satisfied [3], then a BGP network always converges to a stable state. However, there is a subtle issue to consider in attacks. As we have seen in the examples, m can deliberately ignore the GR-EA conditions. Anyway, the following lemma makes it possible, in our setting, to study the HIJACK and the INTERCEPTION problem ignoring stability related issues.

Lemma 1. *Let G be a BGP instance and suppose that at a certain time a manipulator m starts announcing steadily any set of arbitrary paths to its neighbors. Routing in G converges to a stable state.*

The existence of a stable state (pure Nash equilibrium) in a game where one player can deviate from a standard behavior has been proved, in a different setting in [2]. Such a result and Lemma 1 are somehow complementary since

the export policies they consider are more general than Export-All, while the convergence to the stable state is not guaranteed (even if such a stable state is always reachable from any initial state).

2 Checking if an Origin-Spoofing BGP Attack Exists

In this section, we show that, in general, it is hard to find an attack strategy if m has an origin-spoofing cheating capability (Theorem 1), while the problem turns to be easier in a realistic setting (Theorem 2).

We introduce some notation. A *ranking function* determines the level of preference of paths available at vertex v. If P_1, P_2 are available at v and P_1 is *preferred* over P_2 we write $P_1 <_\lambda^v P_2$. The *concatenation* of two nonempty paths $P = (v_k\ v_{k-1}\ \ldots\ v_i)$, $k \geq i$, and $Q = (v_i\ v_{i-1}\ \ldots\ v_0)$, $i \geq 0$, denoted as PQ, is the path $(v_k\ v_{k-1}\ \ldots\ v_{i+1}\ v_i\ v_{i-1}\ \ldots\ v_0)$. Also, let P be a valley-free path from vertex v. We say that P *is of class* 3, 2, or 1 if its first edge connects v with a customer, a peer, or a provider of v, respectively. We also define a function f^v for each vertex v, that maps each path from v to the integer of its class. Given two paths P and P' available at v if $f^v(P) > f^v(P')$ we say that the class of P *is better than* the class of P'. In stable routing state S, a path $P = (v_1\ \ldots\ v_n)$ is *disrupted at vertex* v_i by a path P' if there exists a vertex v_i of P such that v_i selects path P'. Also, if P' is preferred over $(v_i\ \ldots\ v_n)$ because of the GR3 condition, we say that path P *is disrupted by a path of a better class.* Otherwise, if P' is preferred over $(v_i\ \ldots\ v_n)$ because of the shortest-paths criterion, we say that P is disrupted by a path of the *same class.*

A hijacking can be obviously found in exponential time by a simple brute force approach which simulates every possible attack strategy and verifies its effectiveness. The following result in the case the Internet graph has no bound constraints may be somehow expected.

Theorem 1. *If the manipulator has origin-spoofing cheating capabilities, then problem* HIJACK *is NP-hard.*

Surprisingly, in a more realistic scenario, where the length of valley-free paths is bounded by a constant k, we have that in the origin-spoofing setting an attack strategy can be found in polynomial time ($n^{O(k)}$, where n is the number of vertices of G). Let N be the set of neighbors of m. Indeed, the difficulty of the HIJACK problem in the origin-spoofing setting depends on the fact that m has to decide to which of the vertices in N it announces the attacked prefix π, which leads to an exponential number of possibilities. However, when the longest valley-free path in the graph is bounded by a constant k, it is possible to design a polynomial-time algorithm based on the following intuition, that will be formalized below. Suppose m is announcing π to a subset $A \subseteq N$ of its neighbors and path $p = (z\ \ldots\ n\ m)$ is available at an arbitrary vertex z of the graph. Let n_1, n_2 be two vertices of $N \setminus A$. If p is disrupted (is not disrupted) by better class both when π is announced either to n_1 or to n_2, then p is disrupted (is not disrupted) by better class when π is announced to both n_1 and n_2. This implies

Algorithm 1 Algorithm for the HIJACK problem where m has origin-spoofing capabilities and the longest valley-free path in the graph is bounded.

```
1: Input: instance of HIJACK problem, m has origin-spoofing cheating capabilities;
2: Output: an attack pattern if the attack exists, fail otherwise;
3: let P_sm be the set of all valley-free paths from s to m;
4: for all p_sm in P_sm do
5:     let w be the vertex of p_sm adjacent to m; let A be a set of vertices and initialize
       A to {w}; let N be the set of the neighbors of m;
6:     for all n in N \ {w} do
7:         if there is no path p through (m, n) to a vertex x of p_sm such that f^x(p) >
           f^x(p_xm), where p_xm is the subpath of p_sm from x to m then
8:             insert n into A
9:     if the attack succeeds when m announces π only to the vertices in A then
10:        return A
11: return fail
```

that once m has a candidate path p^* for attracting traffic from s, it can check independently to which of its neighbors it can announce π without disrupting p^* by better class, which guarantees that a path from m to z longer than p cannot be selected at z.

In order to prove Theorem 2, we introduce the following lemmata that relate attacks to the structure of the Internet.

Lemma 2. *Consider a valley-free path $p = (v_n \ \dots \ v_1)$ and consider an attack of m such that v_1 announces a path p_{v_1} to v_2 to reach prefix π and p is possibly disrupted only by same class. Vertex v_n selects a path $p_n \leq_\lambda^{v_n} pp_{v_1}$.*

Lemma 3. *Consider a successful attack for m and let p_{sm} be the path selected at s. Let p_{sd} be a valley-free path from s to d such that it does not traverse m and such that $p_{sd} <_\lambda^s p_{sm}$. Path p_{sd} is disrupted by a path of better class.*

Lemma 4. *Let $p = (v_n \ \dots \ v_1)$ be a valley-free path. Consider an attack where v_1 announces a path p_1 to v_2. Vertex v_n selects a path of class at least $f^{v_n}(p)$.*

Theorem 2. *If the manipulator has origin-spoofing cheating capabilities and the length of the longest valley-free path is bounded by a constant, then problem* HIJACK *is in P.*

Proof. We tackle the problem with Alg. 1. First, observe that line 9 tests if a certain set of announcements causes a successful attack and, in that case, it returns the corresponding set of neighbors to whom m announces prefix π. Hence, if Alg. 1 returns without failure it is trivial to see that it found a successful attack. Suppose now that there exists a successful attack a^* from m that is not found by Alg. 1. Let p_{sm}^* be the path selected by s in attack a^*. Let A^* be the set of neighbors of m that receives prefix π from m in the successful attack.

Consider the iteration of the Alg. 1 where path p_{sm}^* is analyzed in the outer loop. At the end of the iteration Alg. 1 constructs a set A of neighbors of m. Let a be an attack from m where m announces π only to the vertices in A.

First, we prove that $A^* \subseteq A$. Suppose by contradiction that there exists a vertex $n \in A^*$ that is not contained in A. It implies that there exists a valley-free path p through (m, n) to a vertex x of p^*_{sm} such that $f^x(p) > f^x(p_{xm})$, where p_{xm} is the subpath of p^*_{sm} from x to m. Since m announces π to n, by Lemma 4, we have that x selects a path p' of class at least $f^x(p)$, that is a contradiction since p^*_{sm} would be disrupted by better class. Hence, $A^* \subseteq A$.

Now, we prove that attack a is a successful attack for m. Consider a valley-free path p_{sd} from s to d that does not traverse m and is preferred over p^*_{sm}. By Lemma 3 it is disrupted by better class in attack a^*. By Lemma 4, since $A^* \subseteq A$, we have that also in a path p_{sd} is disrupted by better class. Let x be the vertex adjacent to s in p_{sd}. Observe that, vertex s cannot have an available path $(s\ x)p$ to d such that $(s\ x)p <^s_\lambda p^*_{sm}$, because $(s\ x)p$ must be disrupted by better class.

Moreover, consider path p^*_{sm}. Since in a^* path p^*_{sm} is not disrupted by better class by a path to d, by Lemma 4, there does not exist a path p'_{xd} from a vertex x of p^*_{sm} to d of class higher than p_{xm}, where p_{xm} is the subpath of p^*_{sm} from x to m. Hence, path p^*_{sm} cannot be disrupted by better class by a path to d. Also, observe that for each $n \in A$ there is no path p through (m, n) to a vertex x of p^*_{sm} such that $f^x(p) > f^x(p_{xm})$, where p_{xm} is the subpath of p_{sm} from x to m. Hence, p^*_{sm} can be disrupted only by same class. By Lemma 2, we have that s selects a path p such that $p \leq^s_\lambda p^*_{sm}$. Since path p cannot be a path to d, attack a is successful. This is a contradiction since we assumed that Alg. 1 failed.

Finally, since the length of the valley-free paths is bounded, the iterations of the algorithm where paths in P_{sm} are considered require a number of steps that is polynomial in the number of vertices of the graph. Also, the disruption checks can be performed in polynomial time by using the algorithm in [11]. □

3 S-BGP Gives Hackers Hard Times

We open this section by strengthening the role of S-BGP as a security protocol. Indeed, S-BGP adds more complexity to the problem of finding an attack strategy (Theorem 3). After that we also provide an answer to a conjecture posed in [4] about hijacking and interception attacks in S-BGP when a single path is announced by the manipulator. In this case, we prove that every successful hijacking attack is also an interception attack (Theorem 5).

Theorem 3. *If the manipulator has S-BGP cheating capabilities and the length of the longest valley-free path is bounded by a constant, then problem* HIJACK *is NP-hard.*

Proof. We reduce from a version of 3-SAT where each variable appears at most three times and each positive literal at most once [9]. Let F be a logical formula in conjunctive normal form with variables $X_1 \dots X_n$ and clauses $C_1 \dots C_h$. We build a BGP instance G (see Fig. 3) consisting of 4 structures: INTERMEDIATE, SHORT, LONG, and DISRUPTIVE. The LONG structure is a directed path of length 6 with edges (s, w_1), (w_1, w_2), $\dots$, (w_4, w_5), and (w_5, d). The INTERMEDIATE structure consists of a valley-free path joining m and s. It has length 4 and it is composed by a directed path $(s\ j_3\ j_2\ j_1)$, and a directed edge (m, j_1). The SHORT structure

has h directed paths from s to d. Each path has length at most 4 and has edges $(s, c_{i,1})$, $(c_{i,1}, c_{i,2})$, $\ldots$, $(c_{i,v(C_i)}, d)$ $(1 \leq i \leq h)$, where $v(C_i)$ is the size of C_i. The DISRUPTIVE structure contains, for each variable X_i vertices, r_i, t_i, x_i, p_i and p'_i. Vertices, r_i, t_i, and x_i, are reached via long directed paths from m and are connected by (t_i, p_i), (x_i, p_i), (x_i, p'_i), (r_i, j_3), (p_i, j_3), and (p_i, d). Finally, suppose X_i occurs in clause C_j with a literal in position l. If the literal is negative the undirected edge $(p_i, c_{j,l})$ is added, otherwise, edges $(p_i, c_{j,l})$, $(r_i, c_{j,l})$, $(c_{j,l}, j_3)$, and undirected edge $(p'_i, c_{j,l})$ are added. An edge connects m to d. Vertices s, d, and m have source, destination, and manipulator roles, respectively.

Intuitively, the proof works as follows. The paths that allow traffic to go from s to m are only those passing through the DISRUPTIVE structure and the one in the INTERMEDIATE structure. Also, the path through the INTERMEDIATE structure is shorter than the one through the LONG structure, which is shorter than those through the DISRUPTIVE structure. If m does not behave maliciously, s receives only paths traversing the SHORT structure and the LONG structure. In this case s selects one of the paths in the SHORT structure according to its tie break policy. If m wants to attract traffic from s, then: (i) path $(j_3\ j_2\ j_1\ m\ d)$ must be available at s and (ii) all paths contained in the SHORT structure must be disrupted by a path announced by m. If (i) does not hold, then s selects the path contained in either the LONG structure or the SHORT structure. If (ii) does not hold, then s selects a path contained in the SHORT structure.

Our construction is such that the 3-SAT formula is true iff m can attract the traffic from s to d. To understand the relationship with the 3-SAT problem, consider the behavior of m with respect to variable X_1 (see Fig. 3) that appears with a positive literal in the first position of clause C_1, a negative literal in the first position of C_2 and a negative literal in the second position of C_h.

First, we explore the possible actions that m can perform in order to disrupt paths in the SHORT structure. Since m has S-BGP cheating capabilities, m is constrained to propagate only the announcements it receives. If m does not behave maliciously, m receives path (d) from d and paths P_{r_1}, P_{t_1}, and P_{x_1} from r_1, t_1, and x_1, respectively. These paths have the following properties: P_{r_1} contains vertex $c_{1,1}$ that is contained in the path of the SHORT structure that corresponds to clause C_1; paths P_{t_1} and P_{x_1} both contain vertex p_1 and do not contain vertex $c_{1,1}$ since p_1 prefers $(p_1\ d)$ over $(p_1\ c_{1,1}\ c_{1,2}\ c_{1,3}\ d)$.

Now, we analyze what actions are not useful for m to perform an attack. If m issues any announcement towards t_1 or r_1 the path traversing the INTERMEDIATE structure is disrupted by better class. Also, if m sends a path P_{r_1}, P_{t_1}, or P_{x_1} towards r_j, t_j, or x_j, with $j = 2, \ldots, n$, the path traversing the INTERMEDIATE structure is disrupted by better class. Also, if m sends $(m\ d)$ to x_1, then the path traversing the INTERMEDIATE structure is disrupted from $c_{1,1}$ by better class. If m sends P_{x_1} to x_1, then it is discarded by x_1 because of loop detection. In each of these cases m cannot disrupt any path traversing the SHORT structure without disrupting the path traversing the INTERMEDIATE structure. Hence, m can disrupt path in the SHORT structure without disrupting the path traversing the INTERMEDIATE structure only announcing P_{r_1} and P_{t_1} from m towards x_1.

If path P_{t_1} is announced to x_1, then p_1 discards that announcement because of loop detection and path $(s\ c_{1,1}\ c_{1,2}\ c_{1,3}\ d)$ is disrupted from p'_1 by better class. Also, the path through the INTERMEDIATE structure remains available because the announcement through p'_1 cannot reach j_3 from $c_{1,1}$, otherwise valley-freeness would be violated. Hence, announcing path P_{t_1}, corresponds to assigning true value to variable X_1, since the only path in the SHORT structure that is disrupted is the one that corresponds to the clause that contains the positive literal of X_1.

If path P_{r_1} is announced to x_1, then $c_{1,1}$ discards that announcement because of loop detection and both paths $(s\ c_{2,1}\ c_{2,2}\ c_{2,3}\ d)$ and $(s\ c_{h,1}\ c_{h,2}\ c_{h,3}\ d)$ are disrupted by better class from p_1. Also, the path through the INTERMEDIATE structure remains available because the announcement through p_1 cannot reach j_3 from $c_{2,1}$ or $c_{h,2}$, otherwise valley-freeness would be violated. Hence, announcing path P_{r_1}, corresponds to assigning false value to variable X_1, since the only paths in the SHORT structure that are disrupted are the ones that correspond to the clauses that contain a negative literal of X_1.

Hence, announcing path P_{t_1} (P_{r_1}) from m to x_1 corresponds to assigning the true (false) value to variable X_1. As a consequence, m can disrupt every path in the SHORT structure without disrupting the path in the INTERMEDIATE structure iff formula F is satisfiable. □

Theorem 4. *If the manipulator has S-BGP cheating capabilities and its degree is bounded by a constant, then problem* HIJACK *is in P.*

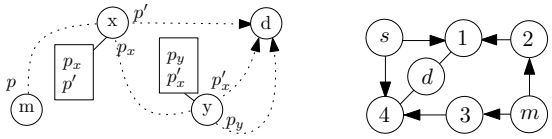
To study the relationship between hijacking and interception we introduce the following technical lemma.

Lemma 5. *Let G be a BGP instance, let m be a vertex with S-BGP cheating capabilities, and let $d \neq m$ be any vertex of G. Let N' be the set of the neighbors to whom m is announcing a path. Consider a vertex v admitting a class c valley-free path p to d such that either $p = (v\ \ldots\ n\ m\ \ldots\ d)$ to d, where $n \in N'$, or p does not contain m. Vertex v has an available path of class c or better to d, irrespective of the paths propagated by m to N'.*

Theorem 5. *Let N' be the set of the neighbors to whom a manipulator m with S-BGP cheating capabilities is announcing a path. If m announces the same path to all the vertices in N', then every successful hijacking attack is also a successful interception attack. If m announces different paths to different vertices, then the hijacking may not be an interception.*

Proof. We prove the following more technical statement that implies the first part of the theorem. Let G be a BGP instance, let m be a vertex with S-BGP cheating capabilities. Let p be a path available at m in the stable state S reached when m behaves correctly. Suppose that m starts announcing p to any subset of its neighbors. Let S' be the corresponding routing state. Path p remains available at vertex m in S'. The truth of the statement implies that m can forward the traffic to d by exploiting p.

Suppose for a contradiction that path p is disrupted in S' when m propagates it to a subset of its neighbors. Let x be the first vertex of p that prefers a different



(a) (b)

Fig. 4: Proof of Theorem 5. (a) The order of paths into the boxes represents the preference of the vertices. (b) The BGP instance used in the example.

path p_x (p is disrupted by p_x) in S' and let p' be the subpath of p from vertex d to x (see Fig. 4a). Observe that p is not a subpath of p_x as x cannot select a path that passes through itself. Since p_x is not available at x in S, let y be the vertex in p_x closest to d that selects a path p_y that is preferred over p'_x in S, where p'_x is the subpath of p_x from y to d.

We have two cases: either $f^x(p_x) > f^x(p')$ or $f^x(p_x) = f^x(p')$ (i.e., p_x is preferred to p' by better or by same class).

Suppose that $f^x(p_x) > f^x(p')$. By Lemma 5, since there exists a valley-free path p_x from x to d that does not traverse m, then x has an available path of class at least $f^x(p_x)$. Hence, x cannot select path p' in S, a contradiction.

Suppose that $f^x(p_x) = f^x(p')$. Two cases are possible: either p_y contains x or not. In the first case either $f^y(p_y) > f^y(p'_x)$ or $f^y(p_y) = f^y(p'_x)$. If $f^y(p_y) > f^y(p'_x)$, then we have that $f^y(p_y) \leq f^x(p') = f^x(p_x) \leq f^y(p'_x)$, a contradiction. If $f^y(p_y) = f^y(p'_x)$, we have that $|p'_x| < |p_x| \leq |p'| < |p_y|$. A contradiction since a longer path is preferred.

The second case ($f^x(p_x) = f^x(p')$ and p_y does not contain x) is more complex. We have that $|p'| \geq |p_x|$. Also, by Lemma 5, since p_y and p'_x do not pass through m, then y has an available path of class at least $\max\{f^y(p_y), f^y(p'_x)\}$. As y alternatively chooses p_y and p'_x we have that $f^y(p_y) = f^y(p'_x)$, which implies that $|p'_x| \geq |p_y|$. Denote by p_{xy} the subpath $(v_m \ \ldots v_0)$ of p_x, where $v_0 = y$ and $v_m = x$. Consider routing in state S. Two cases are possible: either $p_{xy}p_y$ is available at x or not. In the first case, since $|p'| \geq |p_x| = |p_{xy}p'_x| \geq |p_{xy}p_y|$, we have a contradiction because p' would not be selected in S. In the second case, we will prove that for each vertex $v_h \neq x$ in p_{xy} we have that $|p_h| \leq |(v_h \ \ldots \ v_0)p_y|$, where p_h is the path selected by v_h in S. This implies that $|(v_m \ v_{m-1})p_{m-1}| \leq |p_{xy}p_y| \leq |p_x| \leq |p'|$ and this leads to a contradiction. In fact, if $|(v_m \ v_{m-1})p_{m-1}| < |p'|$, then we have a contradiction because p' would not be selected in S. Otherwise, if $|(v_m \ v_{m-1})p_{m-1}| = |p'|$, we have that $|p_x| = |p'|$. Then, x prefers p_x over p' because of tie break. We have a contradiction since also $(v_m \ v_{m-1})p_{m-1}$ is preferred over p' because of tie break in S.

Finally, we prove that for each vertex $v_h \neq x$ in p_{xy} we have that $|p_h| \leq |(v_h \ \ldots v_0)p_y|$. This trivially holds for $v_0 = y$. We prove that if it holds for v_i then it also holds for v_{i+1}. If v_{i+1} selects $(v_{i+1} \ v_i)p_i$, then the property holds. Otherwise, $(v_{i+1} \ v_i)p_i$ is disrupted either by better class or by same class by a path p_{i+1}. In the first case, by Lemma 5, a path of a class better than $(v_{i+1} \ \ldots \ v_0)p'_x$ is available at v_{i+1} and so v_{i+1} cannot select $(v_{i+1} \ \ldots \ v_0)p'_x$ in S', a contradic-

tion. In the second case, we have that $|p_{i+1}| \leq |(v_{i+1}\ v_i)p_i| \leq |(v_{i+1}\ \ldots\ v_0)p_y|$. The second inequality comes from the induction hypothesis.

This concludes the first part of the proof. For proving the second part we show an example where m announces different paths to different neighbors and the resulting hijacking is not an interception. Consider the BGP instance in Fig. 4b. In order to hijack traffic from s, vertices 1 and 4 must be hijacked. Hence, m must announce $(m\ 3\ 4\ d)$ to 2 and $(m\ 2\ 1\ d)$ to 3. However, since $(3\ 4\ d)$ and $(2\ 1\ d)$ are no longer available at m the interception fails. □

4 Conclusions and Open Problems

Given a communication flow between two ASes we studied how difficult it is for a malicious AS m to devise a strategy for hijacking or intercepting that flow. This problem marks a sharp difference between BGP and S-BGP. Namely, while in a realistic scenario the problem is computationally tractable for typical BGP attacks it is NP-hard for S-BGP. This gives new evidence of the effectiveness of the adoption of S-BGP. It is easy to see that all the NP-hardness results that we obtained for the hijacking problem easily extend to the interception problem. Further, we solved a problem left open in [4], showing when performing a hijacking in S-BGP is equivalent to performing an interception.

Several problems remain open: 1. We focused on a unique m. How difficult is it to find a strategy involving several malicious ASes [4]? 2. In [12] it has been proposed to disregard the AS-paths length in the BGP decision process. How difficult is it to find an attack strategy in this different model?

References

1. IP hijacking (2012), http://en.wikipedia.org/wiki/IP_hijacking
2. Engelberg, R., Schapira, M.: Weakly-acyclic (internet) routing games. In: SAGT. pp. 290–301 (2011)
3. Gao, L., Rexford, J.: Stable internet routing without global coordination. In: Proc. SIGMETRICS (2000)
4. Goldberg, S., Schapira, M., Hummon, P., Rexford, J.: How secure are secure interdomain routing protocols? In: Proc. SIGCOMM (2010)
5. Griffin, T., Shepherd, F.B., Wilfong, G.: The stable paths problem and interdomain routing. IEEE/ACM Trans. on Networking 10(2), 232–243 (2002)
6. Huston, G.: Interconnection, peering, and settlements. In: Proc. INET (1999)
7. Huston, G.: AS6447 BGP routing table analysis report (2012), http://bgp.potaroo.net/as6447/
8. Kent, S., Lynn, C., , Seo, K.: Secure border gateway protocol (S-BGP) (2000)
9. Papadimitriou, C.M.: Computational complexity (1994)
10. Rekhter, Y., Li, T., Hares, S.: A Border Gateway Protocol 4 (BGP-4). RFC 4271
11. Sami, R., Schapira, M., Zohar, A.: Searching for stability in interdomain routing. In: Proc. INFOCOM (2009)
12. Schapira, M., Zhu, Y., Rexford, J.: Putting BGP on the right path: a case for next-hop routing. In: HotNets (2010)
13. Underwood, T.: Pakistan hijacks YouTube (2008), http://www.renesys.com/blog/2008/02/pakistan_hijacks_youtube_1.shtml