Cocaine delivery via driverless vehicles OR Scaling physical-proximity attacks on driverless vehicles using structured graph analysis

Ryan Shah and Shishir Nagaraja
University of Strathclyde

Abstract—Driverless vehicles are expected to form the foundation of future connected transport infrastructure. A key weakness of connected vehicles is their vulnerability to physical-proximity attacks such as sensor saturation attacks. It is natural to study whether such attacks can be used to disrupt swarms of autonomous vehicles used as part of a large fleet providing taxi and courier delivery services. In this paper, we start to examine the strategic options available to attackers and defenders (autonomous-fleet operators) in such conflicts. We propose the adversarial Traveling Salesperson Problem, and analyse the salesperson’s efficiency in the face of attackers launching coordinated physical-proximity attacks on the courier’s vehicle. We find that currently attackers have the upper hand in most cases and are able to carry out crippling denial-of-service attacks on fleets, by leveraging the inherent deficiencies of road networks identified by techniques from graph analysis. Experimental results on ten cities using real-world courier traces shows that most cities will require upgraded infrastructure to defend driverless vehicles against denial-of-service attacks. We found several hidden costs that impact equipment designers and operators of driverless vehicles — not least, that entire sections of roadways may need to be redesigned for robustness against attacks.

I. INTRODUCTION

The area of driverless vehicles has seen rapid developments in the last few years. Substantial industrial investment in driverless technology has been made in the wake of recent advances in sensing and computational control systems. While the transformative impact of such automation has been recognised, the trust implications of their deployment have yet to be adequately discussed.

As a working example, we consider a driverless (fully autonomous) vehicle used to courier packages to customers. The vehicle stops at the delivery address and alerts the recipient, who walks over to the vehicle, and types in a PIN to retrieve her package. Such driverless cars are expected to be used to deliver packaged items by online retailers in the future. Other applications include driverless taxis and restaurant deliveries where there quality-of-service especially timing considerations are even more important. These might not sound like an example with significant trustworthiness requirements until one is told that the transport logistics of an online super-retailer could be entirely based on a fleet of driverless vehicles.

As another example, consider a courier who has been tasked with delivering bags of cocaine by a dealer. The courier sets up secret appointments to drop off the shipments at predetermined locations. The courier must select a route visiting all the locations and returning back to the starting point, whilst ensuring minimum expense and maximising safety of the goods. The courier is at risk of attacks from rival dealers who may wish to disrupt delivery, steal goods, or worse.

While driverless vehicles were conceived nearly a century ago, it was not until the application of statistical machine learning combined with control automation, that the ideas crystallised into reality. Therefore it is natural to ask whether it is possible to apply adversarial statistics to also disrupt fleets of driverless vehicles at scale. In the context of fleets, we are concerned about the availability of driverless as the key security property, followed by authenticity of control, integrity, and lastly confidentiality of information. Our key concern is availability, because it is the most easily influenced property — an attacker who jams the optical and acoustic channels can induce an emergency stop resulting in a denial-of-service attack. The question however is, can this be done at scale? Our experiments, carried out within the first systematic study of the area, suggest that this may indeed be the case.

Driverless vehicles collect sensory input via lidars, radar, visual-range cameras, and ultrasound sensors — all of which are vulnerable to signal saturation attacks [16]. Directional ultrasound acoustic jammers consist of an array of powered ultrasound transducers whose output is focused into a narrow beam with a distance range of a few tens of meters [21]. These were developed with (a short range) for medical imaging and (medium range) for sound entertainment systems [18]. Such precision engineering tools can be repurposed as attack tools to shine an acoustic spotlight to stealthily saturate ultrasound sensors. To further compromise safety, an attacker can combine signal saturation with illusion attacks [16]. Illusion attacks cause the information available to the car prior to jamming to be undependable thus causing the vehicle to execute an unsafe stop.

While disabling a single driverless vehicle might not impact the bottom line, doing so at scale certainly will. It is therefore natural to investigate whether individual attacks can be scaled into a service denial attack on an entire fleet of driverless vehicles. We can formulate this as a game-theoretic problem comprising hiders and seeker playing a zero sum game. Each hider travels from a source location to a number of destination locations on a graph, while the seekers attempt to intercept the hider on one or more edges. A successful intercept causes a delay and an aggregation of delays causes a loss in delivery service times. This leads us to a novel variant of the Travelling Salesperson Problem (TSP) where the salesperson (hider) is at risk of being attacked with a probability distribution over the set of locations. Unlike classical TSP, the adversarial variant motivates a change in the optimisation function of the salesperson. Instead of merely minimising aggregate path length, the salesperson needs to consider the actions of the attacker (seeker) who aims to maximally disrupt their activity by launching service denial attacks at chosen locations or between locations. In contrast with classical TSP,
goal is to visit all the locations along the shortest possible route whilst minimising the attack risk and returning to the starting location. We build a framework for analysing the TSP problem using tools from game theory. We carry out a systematic analysis of the problem space showing that constructing optimal attacks is an NP hard problem. Finally, we evaluate a number of approximate algorithms for attacks and defences using real-world and synthetic datasets across ten different cities to confirm that it is possible to scale physical-plane attacks to disrupt swarms of driverless cars.

II. SECURITY GAME

A. Description

An adversarial TSP game is modeled using a graph \( G(V,E) \) with vertex set \( V \) and edge set \( E \). The salesperson visits a number of destination vertices. The set of destinations within the tour are not under the courier’s control but the courier has some flexibility in choosing the sequence of nodes to visit within well-defined time constraints on a per-node basis (courier delivery window). The salesperson’s goal is designing a tour such that the delivery time on each destination on the tour is satisfied whilst minimising costs (distance/fuel). The attacker’s goal is to maximise disruption by intercepting the salesperson on one or more graph edges.

The salesperson’s pure strategies are all the possible paths starting from origin through all the destination nodes. The salesperson’s goal is to minimise the time taken to complete the tour. The attacker tries to intercept the salesperson by placing \( k \) attack resources on edges in the graph. The attacker’s pure strategies are all possible combinations of attack resources to edges, thus \( |E|C_k \) strategies. Each successful interception induces a constant delay \( M \) which corresponds to the amount of time required to ‘rescue’ the driverless vehicle and tow it beyond the attack location. The attacker’s goal is to maximise the amount of delays induced in a tour in order to maximise the number of late deliveries.

For a given tour, let \( Z \) be the set of nodes visited by the salesperson. Let \( P \) be the set of simple (loop free) paths starting from \( O \) and containing destinations \( z_1, ..., z_t \). Each path \( p \in P \) is a subset of vertex set \( V \). We define a distance function \( D_p : p \rightarrow \mathbb{R} \) defined as the sum of the edge weights traversed by path \( p \). The salesperson wishes to minimise delivery time \( T \), while the attacker aims to choose one or more edges to attack the salesperson. The threat model is that the salesperson’s tour \( Z \) is public though the specific path \( P_j \) taken to complete the tour is not (as this could be determined locally by the driverless vehicle itself). In the real-world tour \( Z \) will not be publicised by the salesperson, an attacker could have partial information about a tour, for instance, by making a purchase online and organising delivery to a location of their choosing or by gaining information from an insider attack. We will assume that the attacker has full knowledge of the tour. However, as we shall see later this capability is not key to the attack’s success. We can reason about the attacks and defense strategies in this space using a zero-sum interdiction game [20], [6].

Let \( X \) be the set of allocation of attack resources on \( E \), \( X = X_1, X_2, ..., X_n \), and let \( x \) represent the mixed strategy of the attacker over \( X \), and \((x)\) the corresponding optimal strategy. Each allocation \( X_i \) is a set of binary numbers such that \( X_{ie} = 1 \) if an attacker resource is deployed on edge \( e \in E \), otherwise \( X_{ie} = 0 \). The salesperson has the choice of path set \( P \) to complete a tour where \( P = P_1, P_2, ..., P_m \). The \( i \)th tour path \( P_i = p_{i0} \forall e, P_{ie} \in 0.1 \). Let \( s \) be the mixed strategy of the salesperson over \( P \), then the expected utility of the salesperson can be discussed in terms of the expected time taken to complete the tour:

\[
T(P, X) = \sum_{i \in m, j \in n} P_i X_{ij} (R(D_p + M) + (1 - R)D_p)
\]

The first term corresponds to the situation where the attacker successfully intercepts a vehicle with constant probability \( R \) (for simplicity we assume all vehicles have the same compromise probability) and causes a delay of \( M \) units of time. The second term corresponds to the alternate situation where the attack is unsuccessful or the attacker’s strategy fails to intercept the salesperson. The salesperson’s aim is to find the optimal mixed strategy \( s \) over possible paths \( P \), which gives the least expected delays time for the worst case of attacker choice of resource allocation \( x \). Thus we can formulate the adversarial TSP as a Min-max problem as follows:

\[
T = \min_{P, m} \max_{X, P, s} T(P, X)
\]

Wood [20] showed that the above min-max problem can be solved by formulating it as a linear programming problem and applying a standard LP solver.

B. Attack strategies – Mincuts and NP-hardness

The attacker’s goal is to deploy \( k \) edge resources over edgest set \( E \) using an optimal strategy (\( s \)) which maximises the probability that any tour is intercepted regardless of the choice of paths by the salesperson.

We start with the observation that if a mincut of \( k \) edges (\( k \)-cut) exists in the graph, a tour must traverse the mincut since the destinations will lie in different partitions with high probability. Consider a \( k \)-cut that partitions a graph into several components of similar size. The likelihood of a node on the tour belonging to the same partition is defined as the predecessor node is denoted by \( p \), so with \( 1 - p \) the next node on the tour could belong to a different partition. The likelihood of a tour comprising all \( m \) nodes from the same partition is then given by \( f_p(j, m) \) where \( j \) is the number of consecutive tour nodes already in the same partition is defined as: \( f_p(j, m) = pf_p(j + 1, m) \). As \( m \) increases, \( \lim_{m} f_p(0, m) \rightarrow 0 \) for \( 0 < p \leq 1 \). As a concrete example, for \( p = 0.5 \), the likelihood of a 15-hop tour not traversing a mincut, is less than 65 per million tours (regardless of graph size), decreasing to almost zero per million tours for a tour of 20 nodes. Having established that any route chosen for a tour will include at least one edge of the mincut, with probability close to 1, what does that mean for attacker efficiency? The attacker can leverage this understanding to construct an optimal attack strategy that maximises her utility regardless of a salesperson’s strategy — to block all path choices of a salesperson by placing attack resources on every edge of the \( k \)-mincut. Further, the \( k \)-cut presents a theoretical limit on the number of disjoint paths available to the salesperson, as any route must involve at least one edge from the mincut. Note that this includes paths that are not shortest paths between a pair of consecutive nodes on the tour.

Our second observation is that, if the attacker can block all the paths a salesperson can use with \( k \) resources, then the set of edges must constitute a min-cut of the graph. By combining the above two
observations, we can state that a tour can be blocked by the attacker if and only if the attacker can block all possible salesperson paths with $k$ resources. This is essentially the set cover problem defined as follows. Given a set $U$, a collection $\mathcal{S} \subseteq 2^U$, and integer $k$, the $k$ vertex cover problem poses the following question: is there a coverset $C \subseteq \mathcal{S}$, s.t. $|C| = k$ and $\bigcup_{c \in C} = U$. The argument presented in the first observation above, satisfies the if condition and the second observation satisfies the only if condition. Thus the attacker’s problem is reduced to the vertex cover problem, which is known to be NP-hard.

C. Probabilistic attacks

a) Random edge removal (Baseline): The first attack strategy is the simplest of all, and is one of the most naive attacks. A location is chosen as a suitable site for launching service-denial attacks by choosing an edge from the corresponding graph uniformly at random. This models the case where an attacker has no other information to base their choice and must choose an attack location with no intelligence to hand.

b) Botgrep mincut detection: Thus far, we have established that the mincut size establishes the theoretical upper bound of defense. Therefore it is natural to consider mincut detection techniques as an attack strategy. The traditional description of a mincut from a graph theory perspective, is a partition of the graph into two disjoint subsets that are joined by a small (minimal) number of edges.

Botgrep [12] uses the relative mixing properties of subgraphs to identify edge cuts. Botgrep uses a special probability transition matrix to implement the random walks, where the transition probability between adjacent nodes $i, j \in V$ is $\min(\frac{1}{d_i}, \frac{1}{d_j})$, as opposed to $\frac{1}{d}$ from $i$ to $j$ in Markovian random walks, where $d_i$ is the degree of node $i$. Botgrep uses short random walks to instrument mixing time within a partition and to minimise the leakage of walks starting from a partition. It then applies the probabilistic model from SybilInfer to isolate edges which delineate the subset of the graph where mixing speed changes. Thus the output of Botgrep is various graph subsets with different mixing characteristics. The motivation for Botgrep is as follows. While the notion of a small-cut is a useful starting point, transport networks may not necessarily contain small-cuts that partition the graph into two or more components that are non-trivial in size. Thus a complimentary approach to small-cut detection is offered by the Botgrep technique which combines SybilInfer with machine learning to identify subgraphs with different mixing characteristics.

c) Infomap cutset detection: Another technique that leverages random walks is Infomap [14]. The intuition underlying Infomap is that the fraction of time spent visiting a node during a random walk can be used to uncover dense subgraphs and the cutsets separating them. Unlike Botgrep which uses short random walks, Infomap uses a few long random walks to sample the graph, and computes node centralities as a function of the number of visits during the random walk. This information is used to search for edge cutsets partitioning the graph using a deterministic greedy search algorithm.

D. Centrality attacks

A second class of attacks uses various measures of node centrality to identify important nodes and proceeds to execute service denial attacks on the node’s edges. The intuition underlying these attacks is that attackers often try to disconnect a network by destroying edges of important (central) nodes.

a) Degree centrality: The most obvious form of a node’s importance is the number of other nodes it is connected to. In this case, the attacker targets edges of high degree nodes by deploying attack resources on as many edges of the highest degree nodes whilst constrained by the attack budget.

b) Eigen centrality: A related intuition of a node’s importance is not just the number of neighbouring nodes to but the importance of the those nodes as well. A route that connects important drug routes is even more important. Accordingly, the eigen centrality of a node is algebraically computed as the sum of the centralities of neighbouring nodes, which are in turn connected to many others. The highest eigen centralities correspond to nodes located in dense partitions. Accordingly, the edge cutset comprises edges between the highest eigen-centrality nodes.

c) Betweenness centrality: The principal goal of the attacker is to deploy attack resources on edges that have the highest chance of usage. The Betweeness centrality of a node or an edge is the fraction of the shortest paths between all possible pairs of origin-destination pairs that include it. Since the defender wants to make deliveries within the constraints of distance and time, the shortest path between nodes of the tour would be a reasonable choice for the defender although not the most resilient choice. Accordingly, the attacker targets the set of edges with the highest Betweenness centrality in the graph. High centrality edges usually form a cutset separating dense subgraphs.

E. Modularity attacks

An alternate approach to mincut detection is offered by the notion of modularity. Modularity techniques uncover mincuts that partition a graph into two or more modules. The intuition behind modularity mincut detection is to search for graph components which have less edges than expected from an equivalent baseline. The baseline is a random graph [8] where the expected probability of an edge $(i, j)$ is $d_i d_j/|E|$. Modularity of graph $G(V,E)$ is accordingly defined as:

$$Q = \sum_{i,j} A_{ij} - \frac{d_i d_j}{|E|}$$

where $A$ is the corresponding adjacency matrix of graph $G$. To find the modularity mincut, a search algorithm (for modularity maximisation) is used to detect edgesets that can partition the graph into maximally modular subgraphs. A number of optimisation approaches have been proposed.

a) Greedy-modularity: Clauset, Newman, and Moore [5] propose a greedy algorithm to detect modularity mincuts. Starting from a set of disconnected nodes, the edges of the original graph are iteratively added in order to produce the largest possible increase of the modularity at each step. It has a complexity of $O(N \log^2 N)$ on sparse topologies such as road networks.

b) Eigen-modularity: Newman [13] proposed a spectral optimisation approach to modularity maximisation. It combines the intuitions of eigen centrality (node centrality is recursively defined in terms of the centrality of its neighbours) with modularity (expected vs actual edge probability distributions) to isolate mincuts. This method works by calculating the most significant eigenvector of the modularity matrix defined as $B_{ij} = A_{ij} - d_i d_j/|E|$, where...
the first term is the adjacency matrix and the second term is the expected edge probability according to a randomised baseline. The graph is split into two partitions based on the sign of the corresponding element in the eigenvector, with the mincut being the set of edges across the two partitions. When there is no underlying structure to leverage, the eigenvector elements are of the same sign with the method returning a null cutset (as opposed to partitioning the graph into two partitions regardless of underlying structure).

c) Hierarchical-modularity: Blondel et. al [3] propose a hierarchical modularity maximisation technique for mincut detection. It starts with a set of isolated nodes, each within its own partition. Edges are added from the original graph in order to produce the maximum possible increase in modularity. In each iteration, edges and nodes may be reassigned for merging with a different partition with which it achieves the highest contribution to modularity. Each partition is replaced replaced by supernodes, yielding a smaller weighted network. The process is then iterated, until modularity (which is always computed with respect to the original graph) does not increase any further. This method offers a fair compromise between the accuracy of the estimate of the modularity maximum, which is better than that delivered by greedy techniques like the one by Clauset et. al [5], and computational complexity, which is essentially linear in the number of links of the graph.

F. Defense Strategies – Naive defences

Our first defensive strategy is the simplest of all, and is one that has been proposed by several authors in the past in the context of a network interdiction game [20], [15]. The defender navigates via a randomly chosen route to reach destinations on a tour. This is equivalent to the defender undertaking a Markovian random walk to complete deliveries on time. The intuition behind this defense is that by making random choices about the next part of the route the defender can hope to maximise the attacker’s uncertainty about the defender’s current location. Sanjab et. al [15] also provide a proof of the optimality of this defense.

The alternate obvious defense, is to enumerate all the disjoint paths between source-destination pairs and simply choose one of the routes uniformly at random. One might hope that there is enough redundancy within the network structure that multiple (disjoint) path routes exist and that the defender gets through most of the times and absorbs the delays on account of any attacks (being towed out of the attack zone).

Nice as these ideas may seem in theory, we find they do not work at all well in practice. In the Evaluation section, (Section III) we examine the effectiveness of naive defenses against all the attack strategies and show that they are mostly ineffective when examined against real datasets.

G. Defense strategies

Better results can be expected by designing defenses that are independent of attack strategies, inspired by a common heuristic for solving zero-sum games. The idea behind this approach is that the defender is indifferent to attack moves, resulting in a lower-bound of defence utility.

a) Inverse centrality defence: Accordingly, our first non-naive inverse centrality defense is a simple route-finding strategy that avoids edges that are commonly used as part of a shortest-path route between origin-destination pairs. And, where the attacker might hope to achieve a high rate of interception. Specifically, the defender chooses routes to explicitly avoid high-centrality edges. The defender scores each edge $i, j$ as a combination of Degree, Betweenness, and Eigen-centrality over node $j$. Degree centrality is easily computed as a local metric, and it is the most approximate form of a node’s significance $C^{D}_{ij} = D_{ij}/|E|$. Betweenness centrality is roughly the proportion of paths a node lies on $C^{B}_{ij} = \sum_{s \neq i \neq t} \rho_{st}(j)/\rho_{st}$, where $\rho_{st}$ is the total number of paths between $s$ and $t$ while $\rho_{st}(j)$ is the number of $st$ paths that include node $j$. And, eigen centrality further incorporates the notion of node significance as a function of being connected to significant nodes $C^{\lambda}_{ij} = C_{ij}^{E} = (\frac{1}{\lambda} \sum_{v \in V} A_{ij} C_{iv}^{E})$, where $\lambda$ is the largest eigenvalue. A defender needs to balance between the various types of centrality hence they compute the harmonic mean over the three measures as follows:

$$C_{ij} = \frac{C^{D}_{ij} C^{B}_{ij} C_{ij}^{\lambda}}{C^{D}_{ij} + C^{B}_{ij} + C_{ij}^{\lambda}}$$

The defender then selects a route $p$ that minimises the cumulative weight of the scores within the path i.e $D^p = \min_{e \in p} C_e$.

b) Mixnet routing defence: Our next defense is more sophisticated and derives from the theory of anonymous communications as developed in traffic analysis literature [7]. The defender assigns a random score (between 0 and 1) to each edge in the graph, and computes a route that minimises the cumulative weight of the route whilst completing the tour. This model was initially presented in Danezis’ work on routing anonymously in sparse networks cited above, so we refer to it as Mixnet-shortest-path routing. In Danezis’ model, each mix-router routes messages by forwarding them to a random neighbour in order to maximise a global-passive adversary’s uncertainty about a message’s location within a mix network. This is strikingly similar to the defender’s interest in resisting an attacker in our scenario. The Mixnet defense promises significant improvement over Markov chain routing proposed by previous work [15]. Mixnet routing not only randomises the defender’s path but incorporates the notion of latency awareness by choosing the route that minimises the cumulative weight of (randomised) edge scores in the route. This increases the likelihood of the defender meeting delivery deadlines as compared to Markov chain routing. Since the random scores are only known to the defender, and each defender uses different edge scores, the attacker is unable to effectively predict the edge a defender may be traveling. As such Mixnet routing achieves a better balance between the maximal adversarial uncertainty of Markov chain approach vs the highly predictable shortest-path routing.

Variants of Mixnet routing are possible that achieve a different tradeoff between adversary uncertainty and latency. For instance, a defender can follow (one of) the shortest path for most part but toss a coin at intermediate nodes and either continue to follow the shortest-path to the next hop, or undertake an $O(\log n)$ random walk and regenerate a shortest-path route from the end of the Markov chain to the next destination on the tour. We will explore these variants in future work.

III. EVALUATION

Previous researchers considered disruptive attacks on networks to be a single-round game. Such a model is suitable for applications
such as a conventional war, in which the attacker has to expend a
certain amount of effort to destroy the defender’s command, control
and communications, and one wishes to estimate how much; or
a single epidemic in which a certain amount of resource must be
spent to bring the disease under control.

However, where attack and defense co-evolve in an adaptive
manner, then we have to consider a multi-round game [11] which
has significant explanatory power in many applications. In our
scenario, we are specifically interested in the various Nash equilibria
that might be possible with pure and mixed strategies.

We consider 12 different road networks corresponding to popular
cities across the world where an automated real-time courier delivery
system might be financially viable. The security game described in
Section II is played in a number of rounds. Each round consists of
an attack described in Sections II-C–II-E, when deployed against
each of the defense strategies in Section II-G.

We simulate the adversarial TSP game using real-world courier
workflows. Our analysis proceeds as follows. We initially investigate
the impact of each attack on the percentage of late deliveries when
shortest-path routing or a defense routing strategy is employed in
a multi-round game whilst also computing Nash equilibria. Nash
equilibrium is the solution to our adversarial TSP game, in which
an attacker and defender choose a strategy while considering the
opponents choice, and neither benefits by changing their strategy.

In the attack phase, the attacker deploys attack assets on a subset
of links of the network and intercepts any automated courier she
can find. She selects edges according to an attack strategy described
in Section II-C and executes it on the basis of information about
the network topology.

In the defence phase, the defender considers the impact of the
attack on their delivery efficiency and chooses a defense strategy
in accordance with available strategy choice and information. A
defense strategy is more efficient if for a given attack strategy, it
compels the attacker to increase the number of attack assets to
achieve the same level of network disruption.

Similarly, an attack strategy is more efficient, if it either achieves
an increase in the number of successful interceptions or forces the
defender to expend more resources resulting in deliveries beyond
the delivery window.

To quantify attack efficiency, we measure the percentage increase
in late deliveries induced by the attack as well as the increase in the
tour-completion time – i.e the time required to complete deliveries
per workday. We then examine the how attack efficiency changes
with variance in the delivery window size (the maximum time that
can elapse after delivery time, before a delivery is classed as late);
the impact of number of attackers on the dynamics of attack-defense
strategies.

Initially, we focus on the cities of London and Beijing for which
we have access to real courier traces. Subsequently, we validate our
findings at scale using graph data from ten other cities.

**Assumptions:** We assume that the attacker has perfect infor-
amation about the road network including traffic information as this
information is publicly available. The defender also has access to this
information, however the defender is not aware of the attacker’s suc-
cess rate on a particular route. This simulates the scenario that defend-
ers belong to different administrative domains (i.e no single company
owns them all) and hence their strategy selection is not coordinated.
Our goal is to understand the lower bound for adversary success —
the best-case scenario for operators of fleets of driverless vehicles.
We assume that roads allow movement at the posted speed limits.
Consequently, our analysis is the most optimistic scenario for devel-
opers of driverless vehicle technology, referred to as the defender.

### A. London dataset

Our first dataset contains real courier traces for the city of London
provided by eCourier (www.eCourier.co.uk). eCourier provides this
data through the Open Street Map (OSM) project via a Creative
Commons license. This dataset contains traces of actual courier
movement over an eight week period in 2007 corresponding to half a
million deliveries. Each delivery is associated with a delivery window
which is a binary tuple composed of the earliest delivery time and
the latest acceptable delivery time for the item. We also obtained the
traffic and road maps for London via Open Street Map and generated
a road network graph. Figure 1 shows the distribution of the delivery
windows and we can observe that the average delivery window is
about 2.2 hours. The average tour time in this dataset is ~11 hours.
We assume that each successful interception causes a delay of $M =
10$ minutes. This is roughly the time required for the human driver
to take control after the driverless vehicle has executed an emergency
stop in response to a physical-plane attack. A manual response by
an emergency road-assistance team would increase the delays.

![Delay window(London)](image)

**Fig. 1: Delivery Window (London)**

**Attacks vs. Defenses:** We simulated the multi-round adversarial
TSP game over 83330 tours within this dataset. Figure 2 shows the
impact on delivery time for every combination of nine attack strategies
and three defense strategies, within a multi-round game involving
thirty attackers (we justify this in a future section). The attackers coor-
dinate and intercept as many defenders as possible. Each successful
intercept results in a delay in the defenders journey as the vehicle
executes a graceful stop. The delay value is set to the average amount
of time required by the human assistant to take over the controls and
move the vehicle out of the location where attacks are mounted.
Figure 2 shows that most attacks on driverless vehicles in London are effectively countered by shortest-path routing, with the exception of the Betweenness and Botgrep attacks. Betweenness attacks the edges which lie on the shortest-paths. Consequently, it caused 70% of deliveries to be late when the defender was using shortest-path routing. When the defender switched to Mixnet routing, there were no late deliveries, hence completely mitigating the attack. Mitigation is achieved by the randomness of Mixnet routing, which switches from using high-betweenness edges to leveraging edges that are a part of mid to low conductance cuts, in order to route efficiently.

From Table I we can see that 65% of the late deliveries caused by the Betweenness attack were critically delayed (by >50% of delay window, e.g., 1.1 hours for London) when shortest-path routing is used. The Inverse defense strategy reduced the amount of critical delays by 10%, but Mixnet significantly reduced the critical delays to 1% of the overall late deliveries. When the attacker switches strategies from Betweenness to Botgrep, the Mixnet defense, unlike its effectiveness in defending against the Betweenness attack, was less effective compared to using shortest-path routing. The reason for this is that Botgrep (in common with other modularity-based techniques) attacks low conductance cuts which are crucial to Mixnet’s routing efficiency, as they enable connectivity between sparsely connected localities. Interestingly, the Inverse centrality defense was the most successful at reducing the amount of late deliveries caused by the Botgrep attack. Overall, the amount of critical delays caused by Botgrep is less than Betweenness. Even though Mixnet is less effective against Botgrep compared to Betweenness, it still reduces the amount of critically late delays to 30% of the overall late deliveries, which is lower than if shortest-path routing or Inverse is used.

Next, we allowed attackers and defenders to adapt to each other’s strategies. For the city of London, we found a pure Nash equilibrium between the Botgrep attack strategy and Mixnet defense strategy. When adaptation is allowed, and the attacker employs the Betweenness attack, the defender can deploy Mixnet to suitably counter it.

The attacker could counter the defender’s move with the Botgrep attack, maximising the attacker’s payoff against Mixnet. However, no other strategy increases the defender’s payoff, and no other attack improves the attacker’s payoff, hence constituting a Nash equilibrium.

Impact of attacks on tour time: We also measured the increased costs imposed by defenses by measuring the numbers of hours a courier would need to work for. With no attacks, the average working day is ~11 hours, shown by the length of an average tour when shortest-path routing is used (Figure 3). Even where defenses are successful in minimising the impact of attacks by ensuring the deliveries reach on time, the length of the workday increases significantly which constitutes the extra cost of resilience.

The Betweenness attack induces the largest increase in tour time compared to all other attacks on this dataset. Although the Inverse defense reduces the amount of late deliveries compared to shortest-path routing shown in Figure 2, it induces a higher tour length compared to shortest-path routing, with a worst-case tour time of around 24 hours. Interestingly, the average tour length when Mixnet is deployed against Betweenness is only slightly more than when Mixnet routing is deployed under no attack; we note however the the worst-case tour length is significantly higher under attack.

We also observed that Botgrep does not induce as many late deliveries as the Betweenness attack, due to the overall tour time being relatively similar. Although Mixnet reduces the amount of late deliveries compared to Inverse and shortest-path routing for Botgrep, as shown in Figure 2, it incurs a longer tour time. Interestingly, the tour time incurred by Mixnet defense against Botgrep is similar to the tour time for the Betweenness attack with shortest-path routing, which is the highest number of late deliveries in this dataset.
Impact of delivery-window size: Next, we investigated whether increasing the delivery window — the buffer times available to a courier before a delivery is classed as late — would reduce the number of late deliveries. The average delivery window size within the dataset is 2.2 hours (Figure 1). The results of increasing the delivery-window size are show in Figure 4. Reductions in late deliveries start at around a 75% increase in the delivery window, which is 3.85 hours. Reducing the percentage of late deliveries to a serviceable level of 5% of total deliveries, requires significant increase in the delay-window size, which has implications for the numbers of hours a courier needs to work for to complete the day’s work (or an increase in the number of couriers). For example, in order to reduce late deliveries from 75% to 10% for the Betweenness attack, this would require a delivery window increase of around 250% or around 5.5 hours per delivery.

Impact of attacker strength: To investigate the impact of the number of attacker units on late deliveries, we controlled for the number of attack resources available to the attacker. We assume these attack units are coordinated by a single attacker who coordinates the placement and strategies of all the attack units using a command-and-control network. As shown in Figure 5, as the number of attackers increases, increasing numbers of edges get attacked which result in increasing delivery times. We identified that on average, significant increases in late deliveries occur between 10 and 30 attackers. As well as this, attacks other than Betweenness and Botgrep show minimal or no increases in late deliveries, regardless of how many attackers are deployed. From these observations, we decided to run our previous experiments with a baseline of 30 attackers.

B. Beijing dataset

Our second dataset for the city of Beijing is two orders of magnitude larger than the London courier dataset. This dataset contains traces generated by 30000 couriers over a period of three months between making a combined total of 75 million deliveries. Figure 6 shows the distribution of the delivery windows and we can observe that the average delivery window is about 30 minutes. The average tour time in this dataset is ~4 hours, owing to a large number of couriers being engaged for a fraction of a working day.

Attacks vs. Defenses: Figure 7 shows that most attacks had some degree of impact on the amount of late deliveries when shortest-path routing was used. Specifically, the Betweenness, Eigen-modularity and Botgrep attacks were successful in inducing high rates of late deliveries. The Betweenness attack is effectively mitigated by the Mixnet defense strategy, similar to London. Unlike London, the Inverse defense strategy incurs a higher percentage of late deliveries when employed against the Betweenness attack.
<table>
<thead>
<tr>
<th>Attack Strategy</th>
<th>Lon</th>
<th>Bei</th>
<th>Bris</th>
<th>Bham</th>
<th>Cam</th>
<th>Gla</th>
<th>Edinburgh</th>
<th>Del</th>
<th>Cha</th>
<th>Bos</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Betweenness</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shortest</td>
<td>65</td>
<td>94</td>
<td>85</td>
<td>96</td>
<td>100</td>
<td>94</td>
<td>81</td>
<td>98</td>
<td>98</td>
<td>34</td>
</tr>
<tr>
<td>Inverse</td>
<td>55</td>
<td>99</td>
<td>88</td>
<td>98</td>
<td>100</td>
<td>88</td>
<td>82</td>
<td>97</td>
<td>40</td>
<td>72</td>
</tr>
<tr>
<td>Mixnet</td>
<td>1</td>
<td>10</td>
<td>25</td>
<td>2</td>
<td>75</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td><strong>Random</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shortest</td>
<td>3</td>
<td>22</td>
<td>18</td>
<td>9</td>
<td>54</td>
<td>15</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>Inverse</td>
<td>7</td>
<td>27</td>
<td>18</td>
<td>3</td>
<td>45</td>
<td>9</td>
<td>13</td>
<td>13</td>
<td>8</td>
<td>13</td>
</tr>
<tr>
<td>Mixnet</td>
<td>7</td>
<td>24</td>
<td>16</td>
<td>6</td>
<td>51</td>
<td>15</td>
<td>13</td>
<td>12</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td><strong>Degree</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shortest</td>
<td>0</td>
<td>4</td>
<td>25</td>
<td>0</td>
<td>15</td>
<td>31</td>
<td>11</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Inverse</td>
<td>1</td>
<td>2</td>
<td>21</td>
<td>3</td>
<td>8</td>
<td>35</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mixnet</td>
<td>0</td>
<td>1</td>
<td>54</td>
<td>0</td>
<td>3</td>
<td>13</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Eigen-C</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shortest</td>
<td>0</td>
<td>3</td>
<td>32</td>
<td>0</td>
<td>17</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Inverse</td>
<td>0</td>
<td>2</td>
<td>42</td>
<td>0</td>
<td>12</td>
<td>5</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mixnet</td>
<td>0</td>
<td>1</td>
<td>77</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>66</td>
<td>14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Infomap</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shortest</td>
<td>0</td>
<td>58</td>
<td>31</td>
<td>1</td>
<td>58</td>
<td>10</td>
<td>2</td>
<td>12</td>
<td>4</td>
<td>45</td>
</tr>
<tr>
<td>Inverse</td>
<td>2</td>
<td>59</td>
<td>34</td>
<td>0</td>
<td>46</td>
<td>12</td>
<td>9</td>
<td>9</td>
<td>2</td>
<td>42</td>
</tr>
<tr>
<td>Mixnet</td>
<td>14</td>
<td>4</td>
<td>30</td>
<td>7</td>
<td>50</td>
<td>52</td>
<td>11</td>
<td>40</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td><strong>Hierarchy</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shortest</td>
<td>5</td>
<td>86</td>
<td>76</td>
<td>11</td>
<td>95</td>
<td>35</td>
<td>26</td>
<td>85</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>Inverse</td>
<td>7</td>
<td>77</td>
<td>69</td>
<td>23</td>
<td>94</td>
<td>31</td>
<td>23</td>
<td>83</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>Mixnet</td>
<td>3</td>
<td>54</td>
<td>62</td>
<td>4</td>
<td>83</td>
<td>50</td>
<td>27</td>
<td>68</td>
<td>38</td>
<td>3</td>
</tr>
<tr>
<td><strong>Greedy</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shortest</td>
<td>5</td>
<td>86</td>
<td>76</td>
<td>11</td>
<td>95</td>
<td>35</td>
<td>26</td>
<td>85</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>Inverse</td>
<td>7</td>
<td>77</td>
<td>69</td>
<td>23</td>
<td>94</td>
<td>31</td>
<td>23</td>
<td>83</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>Mixnet</td>
<td>3</td>
<td>54</td>
<td>62</td>
<td>4</td>
<td>83</td>
<td>50</td>
<td>27</td>
<td>68</td>
<td>38</td>
<td>4</td>
</tr>
<tr>
<td><strong>Eigen-M</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shortest</td>
<td>0</td>
<td>72</td>
<td>2</td>
<td>3</td>
<td>82</td>
<td>0</td>
<td>32</td>
<td>39</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Inverse</td>
<td>1</td>
<td>74</td>
<td>3</td>
<td>5</td>
<td>81</td>
<td>2</td>
<td>27</td>
<td>30</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Mixnet</td>
<td>0</td>
<td>19</td>
<td>2</td>
<td>2</td>
<td>80</td>
<td>1</td>
<td>15</td>
<td>60</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td><strong>Botgrep</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shortest</td>
<td>46</td>
<td>68</td>
<td>76</td>
<td>14</td>
<td>83</td>
<td>19</td>
<td>59</td>
<td>64</td>
<td>26</td>
<td>3</td>
</tr>
<tr>
<td>Inverse</td>
<td>36</td>
<td>74</td>
<td>77</td>
<td>15</td>
<td>82</td>
<td>27</td>
<td>48</td>
<td>57</td>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td>Mixnet</td>
<td>30</td>
<td>21</td>
<td>73</td>
<td>13</td>
<td>89</td>
<td>47</td>
<td>44</td>
<td>9</td>
<td>62</td>
<td>1</td>
</tr>
</tbody>
</table>

Table I: % of Critically Delayed Late Deliveries

Fig. 7: Late Deliveries vs. Attacks (Beijing)

than if shortest-path routing is used. This is because London has higher redundancy in terms of the number of disjoint shortest paths within the road network. With an increased number of attackers running the betweenness attack, London can be expected to show a similar trend i.e Inverse performs worse than shortest-path routing. This does not mean there is not enough redundancy between origin-destination pairs in Beijing, however routing techniques based on shortest-paths cannot locate such routes. Mixnet routing however can do so. Similar to the London dataset, the Mixnet strategy also significantly reduces the amount of critical delays. This indicates the importance of leveraging low-conductance paths rather than shortest-path routing to construct better defences.

In both London and Beijing, attacks leveraging betweenness centrality and low-conductance cuts are fairly successful, while Mixnet is the only serviceable defense. Table I, shows that the Betweenness attack causes 94% of late deliveries to be critically delayed and 99% of these delays to be critical when the Inverse defense is employed. Modularity-based attacks also induce high percentages of critically delayed late deliveries. High percentages of critical delays are induced by the majority of attacks with the exception of Degree and Eigen-Centrality, which is unlike the London dataset where high percentages of critically delayed late deliveries are only induced by the Betweenness and Botgrep attacks. Modularity attacks target low conductance cuts which Mixnet uses to improve routing efficiency. We expect to observe a higher amount of late deliveries when Mixnet is deployed against modularity-based attacks. This is
demonstrated by the Hierarchy and Greedy attacks in Figure 7, with Mixnet incurring a lower percentage of late deliveries compared to the Inverse defense. In regards to conductance-based attacks, we find that Botgrep is a successful attack in London whereas Hierarchical-modularity is the most successful attack in Beijing.

As with London, we also found a pure Nash equilibrium in Beijing between the Hierarchy-modularity attack and Mixnet defense.

Impact of attacks on tour time: Next, we investigated the impact of the different attack strategies on the tour time of couriers. From Figure 8 we can observe that the average tour time for a working day is \( \sim 4 \) hours, shown by the length of an average tour when shortest-path routing is used with no attack. Only a minimal increase in tour time is observed when defense strategies are employed with no attack strategy used. Similar to the London dataset, the Betweenness attack significantly increases the tour time of a courier to around 11 hours when no defensive-routing strategy is used. The Inverse defense further increases this, incurring a tour time of around 18 hours. Mixnet performs well to keep the tour time low, showing only a small increase compared to when it is employed against no attack strategy. The Inverse defense further increases this, incurring a tour time of around 18 hours. Mixnet performs well to keep the tour time low, showing only a small increase compared to when it is employed against no attack strategy. From Figure 8, we can also identify that on average modularity-based attacks significantly increase the tour time compared to the average tour time by at least 50%. Mixnet however manages to reduce the tour time incurred by the modularity-based attacks by around the same amount.

Impact of variable delivery windows: The next experiment we ran on this dataset was to investigate the impact of variable delivery windows on the amount of late deliveries. Figure 9 shows us that significant decreases in the amount of late deliveries only occur after at least a 100% increase in the existing delivery window. From Figure 6, we identified that the average delivery window was 30 minutes for this dataset. From our previous observation, we can deduce that a substantial reduction in late deliveries will be seen with a delivery window of about one hour. For an ideal amount of late deliveries, such as around 10% like the London dataset, an increase of at least 250% to the delay window (2.5 hours) is required. However, we can observe that attacks such as Betweenness still incur a very high percentage of late deliveries even with a 250% increase in the delivery window. This suggests that increasing the delivery window alone does not resolve the attack.

Impact of the number of attackers: Our final experiment on this dataset was to investigate the impact of the number of attackers on late deliveries. From Figure 10 we can see that on average, there are significant increases between 1 and 30 attackers. The baseline is derived by observing how many attackers are required to induce significant failure rates using any attack with shortest-path routing. We observe that Botgrep, Eigen-Modularity, Greedy modularity, and Betweenness are very successful even at a fairly low attacker count of 5–10 attackers. However, to keep our experiments consistent with the London dataset for comparison, we decided to use the same baseline of 30 attackers, as it covers the significant increases in late deliveries for attacks on averages.

C. Synthetic dataset

Our third dataset, is composed of synthetic courier traces combined with real road network data generated via OSM data for the following cities: Birmingham (UK), Boston (USA), Bristol (UK), Cambridge (UK), Chicago (USA), Delhi (India), Edinburgh (UK) and Glasgow (UK). The purpose of this dataset is to expand our analysis beyond London and Beijing. To generate synthetic traces, we use the London database as a basis. The number of couriers are maintained but the locations are randomised in a distance-preserving manner i.e the distance between consecutive locations is identical on both the synthetic and real job cards for any courier. In the London dataset, each courier has a job card created on a per-day basis, that lists the delivery locations, times, and a delivery window which
We noticed that for some cities in our synthetic dataset, such as Edinburgh in Figure 11(e) show that Eigen-Centrality with low-conductance edges across dense clusters are identified and targeted by modularity-based attacks.

Figure 11(b) show that the majority of attacks are not that effective. For example, the degree centrality attack has no impact on late deliveries even when no defense is employed. Interestingly, for most cities in our synthetic dataset, Eigen-Centrality attacks have little or no impact on late deliveries. More specifically, our results for Edinburgh in Figure 11(e) show that Eigen-Centrality with shortest-path routing incurs a very small number of late deliveries, but Mixnet causes ~70% of late deliveries. To investigate this further, we looked at the number of critically delayed late deliveries for these cities shown in Table I. We identified that Mixnet causes 66% of the late deliveries for the Eigen-Centrality attack on Edinburgh to be critically delayed. Overall, we observed that for all cities in our synthetic dataset, except for Chicago and Boston, the modularity-based attacks incur the highest amounts of critical delays. The results from the table do show that for all cities, Mixnet reduces the amount of critical delays — however, not substantially. This means that while Mixnet is able to mitigate the attack to some extent, these cities have relatively lower numbers of low-conductance cuts (ρ < 0.076) across localities which restricts the number of redundant paths available to Mixnet whilst under attack.

Effect of Attack Strategies on Tour Time: Our next experiment on the synthetic dataset was to investigate the impact on attack strategies on tour time. Figure 12 shows the effect of attacks on tour time. Overall, we observed that for most cities, the average tour time for a working day is between 8 and 10 hours. Boston has a slightly higher average tour time of ~12 hours. The average tour time is shown by the length of an average tour when shortest-path routing is used with no attack. Interestingly, we identified that increases in tour time correlates with the percentage of late deliveries shown in Figure 11. For example, in Figure 11(e) our results show that Mixnet incurs a high amount of late deliveries when deployed against Eigen-Centrality. In Figure 12(e) we can see the same increase in tour time when Mixnet is used, with the tour time increasing from ~10 hours to nearly 20 hours. The Betweenness attack also incurs the highest tour time for all cities, with Mixnet effectively reducing the tour time as well as the amount of late deliveries.

Impact of Delivery Window Size: Our final experiment on this dataset was to investigate the impact of the size of delivery windows on the amount of late deliveries. As previously described, the synthetic dataset is based on the London dataset such that the delivery window size is maintained the same. Therefore we can state that for all cities in our synthetic dataset, the average delivery window is 2.2 hours. The results of this experiment are shown in Figure 13. For all cities we observed that the betweenness attack, regardless of the defense strategy, incurred the highest percentage of late deliveries even with an increase in the delivery window. For all cities, we noticed that substantial decreases in late deliveries only occur after an increase in delivery window of around 100% (4.4 hours). As well as this, Mixnet also substantially reduces the percentage of late deliveries in all cities and in some cases almost reducing the percentage of late deliveries to nearly 0% such as in Boston (Figure 13(j)). For most cities however, we would consider an ideal amount of late deliveries to be around 10% like with London and Beijing. From the results we can deduce that to achieve the ideal amount of late deliveries, we would require at least a 200% increase in the delivery window (6.6 hours). However for most cities, the Betweenness attack still incurs over 60% of late deliveries even with a 200% increase in the delivery window, suggesting that increasing the delivery window alone will not resolve this attack.

D. Discussion

Driverless vehicles are expected to be foundational components of future transport systems. The challenge of securing such computing devices straddles the design of transport networks as well as vehicular security. We have demonstrated, that the topology

![Beijing](image-url)
Fig. 11: Effect on % of Late Deliveries for Attack Strategies
of road networks plays an important role in the security of driverless vehicles. In particular, host-level attacks on driverless vehicles can be scaled by exploiting weaknesses in the structure of road-networks. The general lesson is that securing driverless vehicles also requires manufacturers to work with city planners to ensure redundancy to withstand targeted attacks.

Previous work has shown that host-level attacks can be mounted via sensor saturation [21], [18], [16] or by exploiting the impact of adversarial inputs on machine-learning techniques [9], [17], [10], [19], [4]. Up until now, no scalable methods of crippling an entire fleet of transport vehicles has been proposed. We have shown how any host-level attack requiring physical proximity can be transformed into a practical attack that can target a fleet at the scale of an entire city.

Generally, we found that the mainstay of routing techniques used by driverless vehicles – shortest path routing – is highly vulnerable to Betweenness attacks in all cities we examined. This highlights the need to develop better defences. We propose Mixnet routing, inspired by the theory of anonymous communications, which performs significantly better. Mixnet combines the notions of routing efficiency with randomness and seeks alternates to the shortest paths. This reduces the failure rates by half, but it also introduces a cost penalty. Fleet operators should keep these costs in mind when planning to switch their fleet from human-driven to driverless vehicles. While the cost savings might seem attractive, it is worth noting that driverless vehicles are also easier to disrupt. In many cities, none of the defences we analyse produced serviceable
results – even after deploying randomised defences – A coordinated
attack by approximately 10–30 attackers, can cause between 20% to 50% of deliveries to be delayed, at a minimum, considering the application of Mixnet routing strategy. In cities like Beijing as few as 8 attackers are able to cause significant levels of disruption. An increase in the number of attackers reduces delivery rates roughly linearly as the number of attackers with some threshold points.

Relaxing the buffer times courier results in some improvement of the delivery rate (a reduction of 5–20%) against the most powerful attacks, but are accompanied by increased costs of a longer workday.

Switching the routing strategy from the default (shortest path) to a more resilient Mixnet helps, however further switching does not help in most cases. We found pure Nash equilibriums between attacks and defences in London (Mixnet vs Botgrep), Beijing (Mixnet vs Hierarchical modularity), and Boston (Mixnet vs Random). An equilibrium predicts the strategic behaviour of attackers and defenders, and specifically that switching from these strategy combinations is unlikely under the assumption of rationality.

For Bristol, Birmingham, Edinburgh, Delhi, Glasgow, Chicago, and Cambridge (UK), we predict mixed-strategy equilibriums

While some defences can be effective against some attacks many challenges remain. Our work demonstrates that the well known shortest-path routing strategy will fail miserably, with delayed deliveries approaching 80–100% in most cities we analysed, with the exception of Chicago, which has a lattice road-structure that offers slightly better resilience (60% late deliveries). This motivates the need for secure routing algorithms that can withstand attacks targeting a rather small fraction of the network. These findings have important implications for the reliability of autonomous vehicles and highlights the constant effort operators of driverless vehicles may have to put into making their schedules work, even as most of the deliveries are rated late by their customers.

IV. RELATED WORK

The game-theoretic background to the problem at hand lies in the search game within predator-prey games [1665:17-22], also
known as hider–seeker games. This is a zero sum game between a single predator and a single mobile prey. The predator and prey move about in a search region. The game ends with positive payoff to the predator when it meets the prey. As a bio-inspired example, the *blancanella* wasp finds larvae by searching for visible evidence of leaf-mining. Wasps are attracted by the appearance of holes or other leaf deformation created by the leaf-mining larvae. The game begins when the wasp lands on the leaf to search for the larvae, who in turn is alerted by the vibrations caused by the landing wasp triggering evasive behaviour by larvae. When the wasp encounters a feeding hole, it repeatedly inserts its ovipositor violently in the area to ambush the prey. The game ends either with the wasp paralysing the larvae or abandoning the leaf. The formalisation of this problem is well studied within pursuit-evasion games [1].

A particular form of hider–seeker game called an interdiction game [20] which was originally developed to understand and intercept drug smuggling in the 90s. In an interdiction game, one or more smugglers (hiders) attempt to traverse a path between two nodes on a network while the police (seeker) patrol certain routes intensively to interdict smugglers. Both the players are intelligent and adapt to each other to avoid being predictable. Our work uses Wood’s game formulation as the starting point.

Work on security games and robotic patrolling has focused on concrete applications of path-disruption games [2]. Here the hider attempts to reach a well known target whereas the seeker wishes to prevent that. The dynamics of attack and defence strategies is well understood in the static target problem — a static target is perfectly in that we haven’t considered the effects of congestion, we offer a lower bound of adversarial success as congestion will further reduce the fraction of on-time deliveries. There are several avenues for future work. First, our analysis would be improved by considering the effects of congestion. Second, our analysis may be improved by considering temporal aspects (observing how variance on the traffic graphs impacts our results). Finally, we do not attempt to address the challenging problem of providing countermeasures i.e how to build redundancy into the road network and designing defensive-routing schemes that can leverage that redundancy when needed.

### REFERENCES


