CERIAS

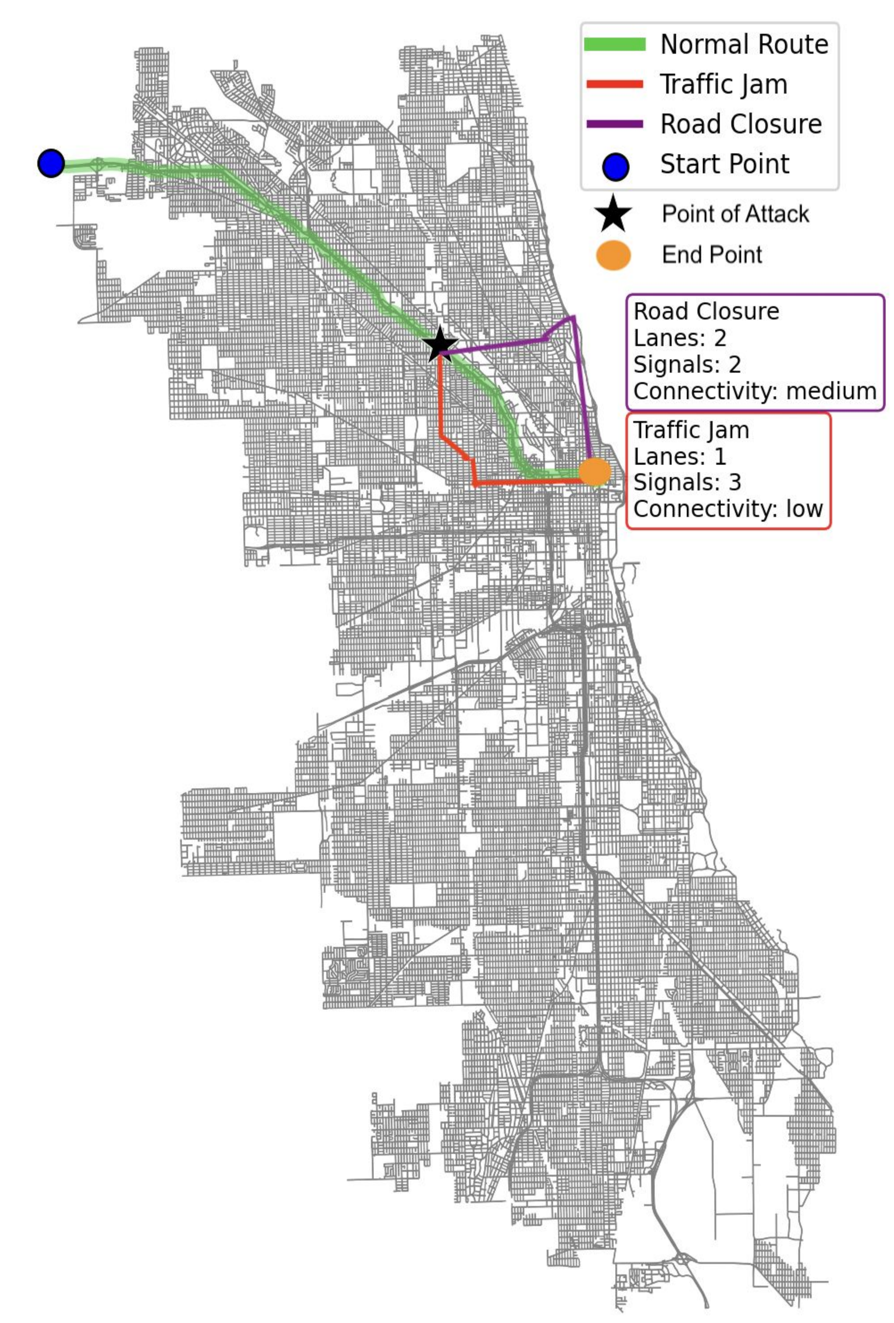
The Center for Education and Research in Information Assurance and Security

Deceptive Directions: Understanding Route Guidance and GPS Spoofing Attacks

Akshit K. Bedi, Abrar Ali, Eunhan Ka, Satish V. Ukkusuri

2025 - NS - JPT-6N8 - Deceptive Directions: Understanding Route Guidance and GPS Spoofing Attacks - bedi2@purdue.edu - Akshit Bedi

Route Guidance Attack Visualization



Scenario	Travel Time	Increase (mins)	Increase (%)
Without Attack	32 mins	–	–
Traffic Jam	45 mins	+13 mins	+40.6%
Road Closure	53 mins	+21 mins	+65.6%

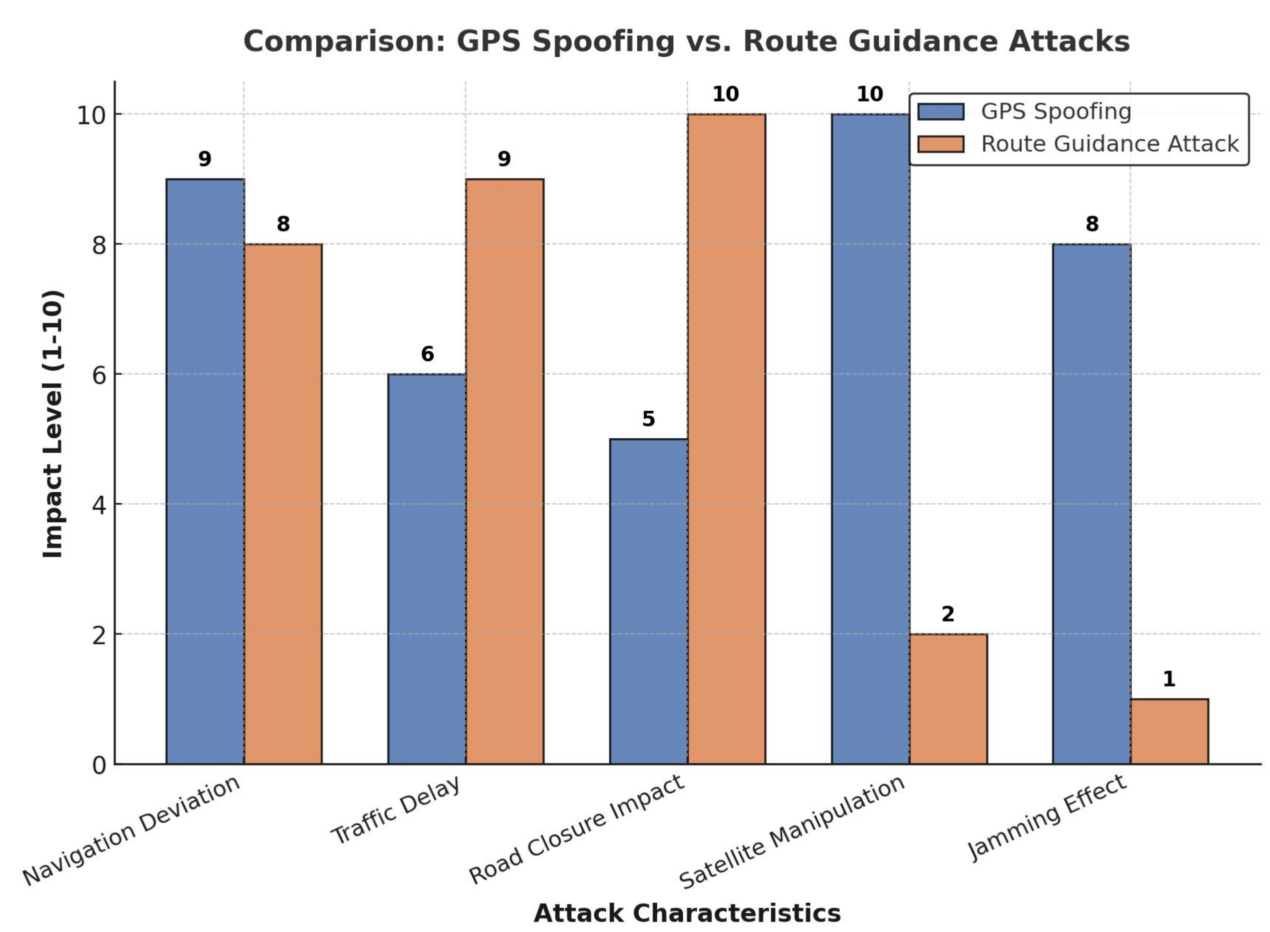
Overview of Route Guidance Attacks

A **Route Guidance Attack** occurs when GPS navigation systems are manipulated to redirect vehicles away from their optimal or intended routes. These attacks exploit vulnerabilities within GPS signals or routing algorithms, forcing vehicles onto inefficient or unsafe alternative paths. In this study, we analyzed the impact of such attacks specifically within the Chicago area, using realistic scenarios to quantify their effect.

We simulated two primary attack scenarios along the route from **O'Hare Airport to Downtown Chicago**:

- **Traffic Jam Scenario**:
  - **Original Travel Time**: 32 minutes
  - **After Attack**: 45 minutes *(+13 minutes, +40.6%)*
- **Road Closure Scenario**:
  - **Original Travel Time**: 32 minutes
  - **After Attack**: 53 minutes *(+21 minutes, +65.6%)*

The simulations accounted for key route hyperparameters, including the number of lanes, traffic signal density, and road connectivity, integrating these directly into shortest-path routing algorithms.



METHODOLOGY

1. Data & Tools

- Chicago road network from **OpenStreetMap** (via OSMnx) and Python
- Routing analysis with **NetworkX**

2. Attack Scenarios

- Route: **O'Hare → Downtown**
- Scenarios:
  - **Traffic Jam** (delay simulation)
  - **Road Closure** (segment removal)

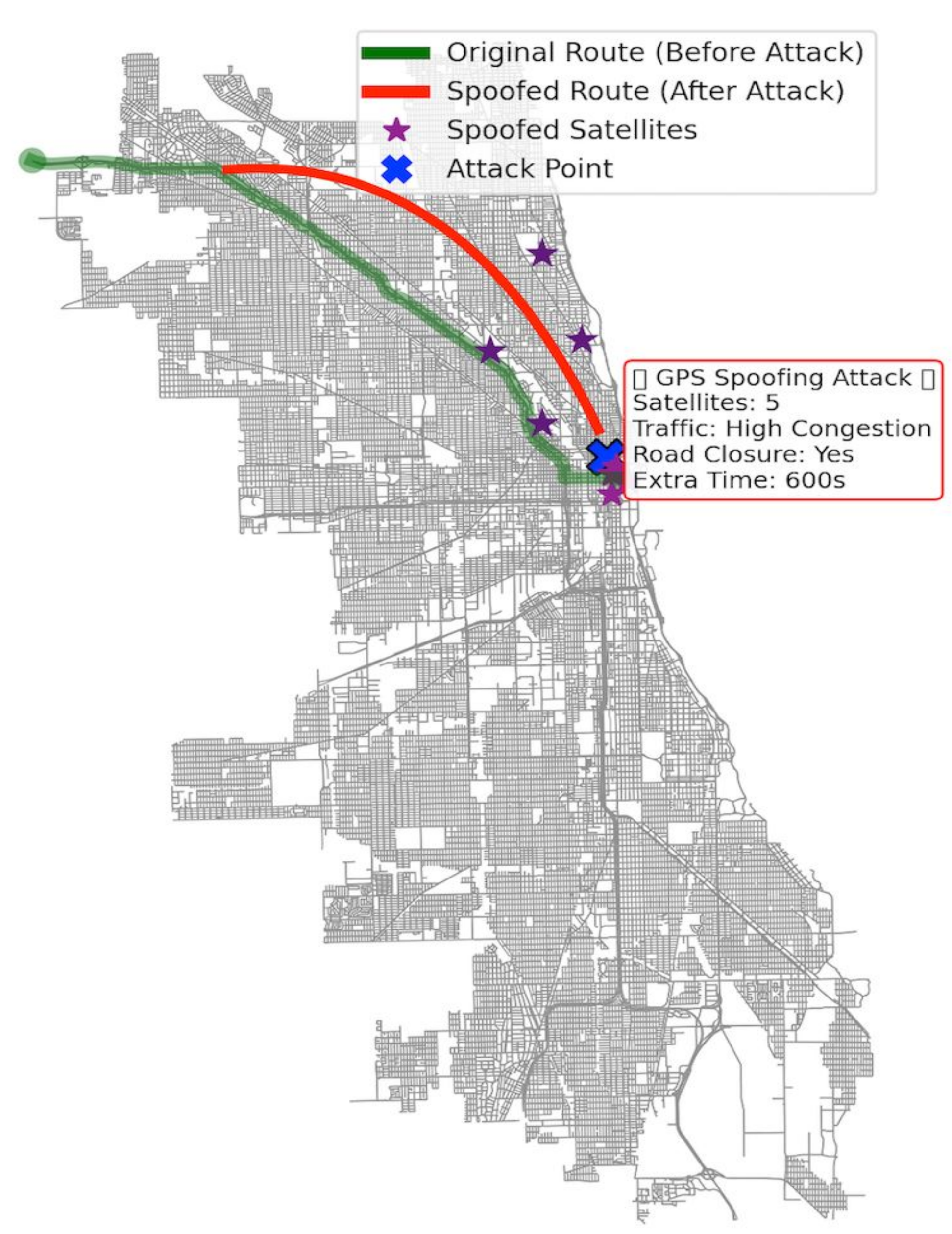
3. Simulation

- Shortest route calculations (**original vs. attacked**)
- Integrated hyperparameters (lanes, signals, connectivity)

4. Evaluation

- Compared **original vs. attack travel times**
- Calculated **absolute and percentage increases**

GPS Spoofing Attack Visualization



Overview of GPS Spoofing Attack

GPS spoofing attacks manipulate a receiver's position and timing data by transmitting counterfeit signals that mimic legitimate satellites. These attacks pose significant threats to navigation, transportation, and security systems by leading vehicles or devices off their intended routes.

The simulation models GPS spoofing impacts through two primary attack scenarios along the **O'Hare to Downtown Chicago** route

- **Spoofed Satellites:** Alters location accuracy by affecting multiple satellites.
- **Traffic Congestion:** Increases delays as spoofed routes lead to congested areas.
- **Road Closure:** Misguides navigation systems into high-delay or closed roads.
- **Satellite Jamming** – Degrades navigation by blocking legitimate GPS signals.

Future Research Directions

This study highlights the real-world implications of GPS spoofing and jamming on urban transportation. Future research can focus on **developing countermeasures**, such as AI-driven anomaly detection in GPS signals, **resilient navigation systems**, and **multi-sensor fusion techniques** to mitigate spoofing threats in smart city infrastructure.

Code_Hyperparameters_Overview

Scenario	Travel Time	Increase (mins)	Increase (%)	Satellites Affected	Jamming Type	Traffic Condition	Road Closure	Extra Time (s)	Signal Interference	Navigation Accuracy
Without Attack	32 mins	–	–	0	None	Normal	No	0	None	High
GPS Spoofing Attack	42 mins	+10 mins	31.3%	5	Spoofed	High Congestion	Yes	600	High	Low
Traffic Jam	45 mins	+13 mins	40.6%	0	None	High Congestion	No	780	None	Medium
Road Closure	53 mins	+21 mins	65.6%	0	None	Closed Road	Yes	1260	None	Low
Satellite Jamming	48 mins	+16 mins	50.0%	7	Jammed	Severe Congestion	No	960	Extreme	Very Low

Pseudo-code

```
Input: OpenStreetMap Road Network (Graph), Origin-Destination pairs, Attack Hyperparameters

For each Origin-Destination pair:
    origin_node ← find_nearest_node(origin)
    destination_node ← find_nearest_node(destination)

    original_route ← shortest_path(graph, origin_node, destination_node)
    original_time ← calculate_travel_time(original_route)

    attack_graph ← copy(graph)

    If Attack_Type is "Road Closure":
        closed_node ← find_nearest_node(closed_road_location)
        remove_node(attack_graph, closed_node)

    Else if Attack_Type is "Traffic Jam":
        jam_node ← find_nearest_node(traffic_jam_location)
        increase_edge_weights(attack_graph, jam_node, delay_factor)

    attacked_route ← shortest_path(attack_graph, origin_node, destination_node)
    attacked_time ← calculate_travel_time(attacked_route)

    intersection_node ← find_intersection(original_route, attacked_route)

    Record results: original_time, attacked_time, intersection_node

Output: Routes, intersection points, travel times, attack impacts
```