Robert Deppe, Henrik Graßhoff, Robert Kokorev, and Tino Rohmund

ETSi – An Electronic Tolling Simulation Framework for Attacking Location Privacy

Abstract: Authorities all around the world install Electronic Toll Collection (ETC) systems in order to finance road infrastructure or to reduce congestion and negative health effects like air pollution and accidents. As most of the current systems are mandatory with no easy way to opt-out, preserving the location privacy of its users is a main objective of such systems.
We provide a flexible framework for the simulation of an ETC and attacks on the location privacy of its users. This includes the generation of traffic demand based on persons with different definable behaviours throughout a day and week, the traffic simulation and the execution and evaluation of different attacks.
We conducted a random and a trip reconstruction attack in a variety of scenarios. Our results indicate that an attacker which uses a simulated annealing approach and knowledge of average travels times is able to reconstruct trips and correctly combine them to a wallet with an already good, though not overwhelming quality.

Keywords: Anonymity, ETC, Location, Privacy, Simulation, Toll

DOI Editor to enter DOI
Received ..; revised ..; accepted ...

1 Introduction

Two thirds of the world's population will live in cities by 2050, the UN projects [1]. This poses a big challenge for local authorities to plan and adapt cities to accommodate to the increase of inhabitants, and a major task is coping with the growing mobility demand. In order to finance road infrastructure or to reduce congestion and negative health effects such as air pollution, noise, and accidents, Electronic Toll Collection (ETC) systems are increasingly being deployed around the world.

For example, the European Commission decided in 2009 to improve the interoperability of its member states' ETC systems under the name "European Electronic Toll Service" (EETS) [2]. The harmonisation of the individual systems is intended to allow users to access and pay for toll roads in different countries with the same in-vehicle hardware and a single registration with a Toll Service Provider (TSP). There have been made further legal and technical [3] specifications, and the EETS is currently deployed in the EU members states. This could lead to a toll collection with potentially up to almost 240 Million natural users [4] in the EU only.

Considering such a large number of users, these systems need to be designed carefully in order to meet several requirements which seem contradictory in the first place. Users must be prevented from misbehaving – e.g., not participating or pretending to be a different vehicle – and detected when they do (*integrity*); the system must be installed and maintained at reasonable costs and user-friendlyness (*practicability*), not fail for a longer time period (*availability*) and prices should be able to vary with vehicle characteristics, day and time, road congestion etc. (*flexiblity*) as well as comprehensible to the user (*transparency*). These requirements could be met in the most naïve way by the TSP tracking all vehicles and calculating the user's debt. It is clear that such a privacy-invading system is not desirable (see [5] for a definition of location privacy and the consequences of its loss). Such an invasion cannot be completely avoided – e.g., mandatory in-vehicle satellite positioning hardware or automatic video surveillance –, but the impact on the users' privacy should be kept at a minimum as long as the functioning of the system is provided (*privacy*). However, this is not the case for many systems currently in use (cf. [6, 7]).

1.1 Classification of ETC systems

ETC systems can be classified based on the way debt is collected. Concerning the type of charging, we distinguish three *charging schemes*: distance-based, time-based, and access-based schemes, where a user pays for the distance/time she travelled or to access a specified tolled area [8] like the London Congestion Charge Zone. In terms of involved technology, five fee accumulation systems stand out [3]:

ANPR Tolling roads are equipped with cameras for Automatic Number Plate Recognition (ANPR) and is for example deployed in the urban centres of London [9] and Stockholm. Although this technology is quite flexible and does not require any OBU, it is also a big threat to the users' privacy.

DSRC Debt is accumulated by Dedicated Short-Range Communication (DSRC) between an in-vehicle On-Board Unit (OBU) and a Road-Side Unit (RSU). This can either be a toll plaza with physical barriers or an open-road station which does not slow down traffic. DSRC requires in-vehicle hardware and the maintenance of RSUs by the TSP, but is the most widely used ETC system wordwide.

RFID Vehicles passing a toll station are identified via Radio Frequency Identification (RFID) of radio waves emitted from an in-car antenna. This technology is very comparable to DSRC and mostly deployed in the US.

GNSS Using a Global Navigation satellite System (GNSS) like the US GPS or the EU's Galileo system, every car accumulates position data and transmits, typically using a cellular network, those to the TSP, who automatically calculates the fee based on this data. Similar to those schemes requiring an OBU, in-car hardware is neededd for satellite positioning and the transmission of location data. ETC by GNSS raises questions of reliability – e.g., no satellite signal or manipulated location data – and location privacy as it potentially enables live-tracking of vehicles.

Smartphone-based The use of smartphones for toll collection and payment is similar to the GNSS scheme and currently limited to a few pilot projects.

For enforcement purposes, ETC systems of the DSRC, RFID and GNSS schemes are often supplemented with ANPR cameras or mobile controls, which enhance the system's integrity but threat the users' privacy on the other hand.

There exist a number of different ETC models, both in the literature and real-world implementations, for all of these technical schemes. Despite their differences, one thing many of these models have in common is the fact that the TSP gradually gathers location-time data of its users. Though amount and content of these data differ between ETC models, one can naturally ask the question: "To what extend can an adversary track ETC users by exploiting location-time data?"

1.2 Our contribution

In order to estimate the privacy impacts of ETC, we developed a Python framework ETSi (**E**lectronic **T**olling **Si**mulation Framework). The framework uses the SUMO traffic simulation package [10] as a foundation. In particular, the framework can realise ANPR, DRSC and RFID systems on any SUMO map with RSUs that can be placed either by hand or automatically. It then generates traffic based on settable parametres – e.g., home and work districts on the map, number of simulation days –, runs the simulation and collects location data which are thought to be stored by the TSP. The amount and content of this data depend on the ETC system which is to be simulated. Based on the accumulated data, the framework can apply an attack to harm the users' location privacy and evaluate this attack. A main advantage of our framework is the modularity: different maps can be used as inputs, the stored location data can be altered by editing the simulation script, and attacks as well as their evaluations can be exchanged without much effort. To the best of our knowledge, this is the first sophisticated modular framework for the simulation of privacy attacks on ETC. In addition, we provide the results of two simulations: a random attacker and an attack by trip reconstruction.

Organisation

The paper is organised as follows: Section 2 presents a variety of ETC schemes proposed by past research. Section 3 gives an overview of the ETC scheme applied in our implementation and the main components of the framework. We then evaluate both our framework and the results of two applied attacks in section 4.

2 Related Work

Research has focused on the development of privacy-preserving ETC models in different schemes.

Popa et al. [11] propose VPriv, a flexible GNSS-based ETC system in which a driving vehicle's OBU uploads anonymised, random-tagged location-time tuples to the TSP. At the end of a billing period, the TSP broadcasts subfees for every recieved tuple to every OBU. The OBU then calculates the total fee based on the subfees for the corresponding tuples, sends this fee to the server and zero-knowledge proves in a two-party compucation the fee's correctness without revealing which tuples were used. For fraud prevention, authorities use random “spot checks” using, e.g., ANPR. An observed client is challenged to provide a tuple in a time-spatial neighbourhood of the spot check (*reconciliation*). As these spot checks reveal non-anonymised time-location data to the TSP, its number is a trade-off between law enforcement and privacy.

PrETP [12] is a similar model, but in contrast, the OBU calculates sectional fees on its own, sends them to the server and commits to these fees using a homomorphic commitment. Using this homomorphic property, the TSP assures that all sectional fees were used by the OBU to calculate the total fee at the end of a billing period. Fraud prevention is again realised by random spot checks, and the main difference is that the TSP does not collect any location-time data apart from those spot check data. Pricing is theoretically flexible, but the number of possible prices must be kept small for practicability purposes.

As [13] points out, large-scale driver collusion is a potential thread to the reliability of VPriv und PrETP. When the TSP observes a driver at a spot check, he reveals the check location. On a large scale, if spot check locations change only insufficiently, drivers could cooperate to learn the spot locations and cheat. As a solution, the authors present Milo, a variation of PrETP, in which the TSP and the OBU process the reconciliation in a blind identity-based encryption protocol without the OBU learning the actual spot location.

P4TC [14] is a post-pay, access- or distance-based DSRC scheme ETC model with an elaborate documentation of its protocols. Debt accumulation is done by the RSU without learning the driver’s identity. The TSP records that a (but not which) vehicle passed this RSU and additionaly stores the RSU previously visited by the same vehicle. The authors prove that the TSP theoretically does not learn enough information to link transactions with certainty. As they note, tracking users might be achievable by solving a specific instance of the KNAPSACK problem, which is NP-complete but could be *“practically solvable for ‘real world’ instances”*.

3 The framework

Our framework is able to simulate a DSRC/RFDI toll collection system and provides a pipeline to generate traffic demand and attack the resulting ETC data. The framework is solely written in Python, uses SUMO [10] for simulations of urban traffic (cf. technical documentation for further details), and is available as a Docker container.

3.1 ETC scheme and attacker model

A user accumulates debt whenever she passes an RSU, which can be positioned at intersections at will and priced flexibly. When passing, she initiates a *transaction* (tag, time, RSU id, price, user id). The TSP then adds the price to her *wallet*, i.e., updates (user id, old debt) to (user id, new debt), and stores the anonymised transaction in his database, thus, without the user id (see figure 1). At the end of a billing period, each user is charged to pay her wallet debt, which is then set to zero for the next period.

We assume that the adversary is the *honest-but-curious* TSP or has his knowledge. This assumption is not unrealistic: A TSP with far-reaching user tracking capabilities could lose public acceptance – and thus, the toll collection as a whole –, and the collected data could be abused by authorities, e.g., for law enforcement purposes.

In detail, the adversary owns a complete list of anonymised transactions, consisting of tuples (tag, time, RSU id, price) for every transaction initiated by a vehicle passing an RSU, as well as each user’s total debt at the end of a billing period, thus a list of tupels (user id, debt). It is possible to add justifiable knowledge for more sophisticated attacks. For example, an adversary might use public data, own traffic observations or physically track individual vehicles to gain information on average travel times during different hours of a day or average lengths of trips. This knowledge was used in our trip reconstruction attack (see section 4.1).

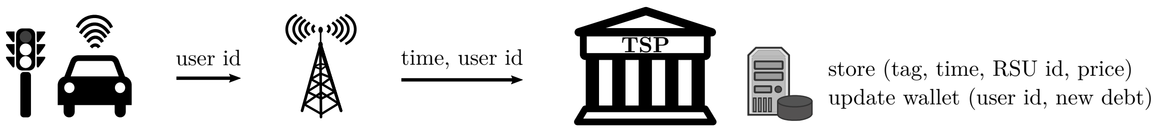

user id
time, user id
TSP
store (tag, time, RSU id, price)
update wallet (user id, new debt)

Fig. 1. Data transmission and storage in the debt accumulation

3.2 Maps and modules

We first explain the preparation of maps. Basically, any SUMO map is suitable to be used in ETSi. Our simulations took place on three different maps: a 15×30 Manhattan grid with a street length of 270 and 80 metres, which roughly and partly reflects the real Manhattan street layout to the south of Central Park, a spider grid with ten sectors and ten rings in a distance of 100 metres, and an adoption of the freely available SUMO map of the German city Ingolstadt from the InTAS project [15] of the Technische Hochschule Ingolstadt.

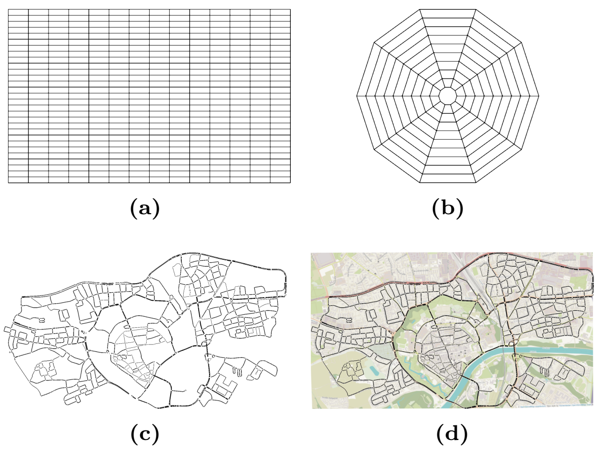


Fig. 2. The two maps used in our simulations: (a) Manhattan grid, (b) spider net, and (c) a map of the German city Ingolstadt. The latter is shown with an OpenStreetMap overlay in (d).

Induction Loops Detectors, also called *E1 Detectors*, can be used to gather location-time information [16]. To use these detectors, we placed Traffic Light Systems (TLS) at intersections and used the Python script `generateTLSE1Detectors` to equip every lane of a TLS intersection with a detector [17]. Note that this script places detectors only at intersections with TLS, however, E1 detectors can also be placed by hand. This enables an easy automated placement of RSUs at intersections as well as the manual placement anywhere on the map. When using the script `netgenerate` (which we did to create the grid and spider network), one can set TLS on every junction by default [18]. The map of Ingolstadt was manually edited: we chose a proper section of the downloadable map, removed dead ends for the sake of simplicity, equipped all intersections with TLS using the graphical map editor `Netedit` and adjusted some narrow road situations to avoid dead locks and vehicle teleporting caused by congestion.

Once a map has been prepared, the framework's four modules can be applied:

Generation This module generates the traffic demand by creating so called *agents* and trips herefore. An agent persistently represents a vehicle throughout the entire simulation. In our implementation, the module creates a definable number of agents for a definable number of days. Different agent types are implemented – e.g., there are classes for full-time and part-time workers with weekly work hours based on European statistics [19] –, and every agent visits some random appointments throughout a week.
When executing this module, one can set home and working districts on the map each as a union of convex polygons. Each agent then randomly choses a home and work address among the addresses in the corresponding districts. The module outputs an XML file consisting of the routes of the individual agents.

Simulation Based on a map and suitable trip file – presumably, but not neccessarily, generated by the agent generation –, this module executes the simulation. It runs a SUMO simulation either with or without the SUMO graphical user interface and outputs two XML files: one file containing the full knowledge – i.e., all non-anonymous transactions, all final wallets and which transaction belongs to which trip –, called *challenger knowledge* and the adversary's *attacker knowledge* (see subsection 3.1).

Attacker Given the attacker knowledge file, this module executes the interchangeable attack. Our simulations included a random attack as well as a trip reconstruction, but more elaborate attacks are imaginable and could be subject of further research. The attacker module attempts to match each transaction with its corresponding agent and outputs an XML file of the proposed matches.

Challenger The challenger module is in charge of both the actual true challenger knowledge as well as the proposed output given by the attacker. It then compares the attack's quality and outputs a measure of quality – e.g., "60% of transactions assigned to correct agent" – which can be set flexbily. We implemented the above mesaure of correctly assigned transactions as well as a measure for correctly reconstructed trips.

Days	Agents	Mean (%)	StDev	Agents	Mean (%)	StDev	Agents	Mean (%)	StDev
	Manhattan			Spider net			Ingolstadt		
3	300	1.03	0.02	200	1.64	0.03	400	0.98	0.02
7	300	0.73	0.01	200	1.12	0.02	400	0.65	0.01
10	300	0.6	0.01	200	0.93	0.01	400	0.53	0.01
7	600	0.57	0.01	500	0.86	0.01	800	0.53	0.01
7	1000	0.49	0.01	700	0.8	0.01	1200	0.47	0.01

Table 1. Evaluation of random attacker

4 Evaluation (WIP)

4.1 Attack evaluation

Based on job market statistics in the European Union of the past years [19], we assumed a distribution of 80% full-time and 20% part-time workers. We executed a random and a trip reconstruction attack on the three maps in figure 5. Home and work districts were set as follows: Manhattan was split vertically in two halves with the left half being the home district and the right half being the work district, the spider net was split horizontally in halves, and Ingolstadt has both districts covering the whole map. Prices were set to 1 for all tolling stations and each result is based on ten simulations.

Once an attack has been executed, we measure the similarity of a wallet (i.e., a set of transactions) A proposed by the attacker and the real wallet C using the *Jaccard Coefficient*

$$J(A,C) = \frac{|A \cap C|}{|A \cup C|} \in [0,1]$$

which is equal to 1 iff A and C contain the exact same transactions and 0 in the case of two disjoint wallets. Our attackers' *success* is then defined as

$$\frac{1}{|\mathcal{A}|} \sum_{A \in \mathcal{A}} \max_{C \in \mathcal{C}} J(A,C) \tag{1}$$

where $\mathcal{A}$ and $\mathcal{C}$ are the sets of attacker/challenger wallets, and can be interpreted as the average maximum similarity of an attackers wallet compared to a real wallet. Note that the success is being calculated by a script of the challenger module and could be rather easily replaced with a different measure of success.

Random attack

The random attacks assigns each transaction a random vehicle. We conducted these attacks in the cases of table 1 and evaluated the attacks as shown in formula 1. Our results are plotted in the following two figures:

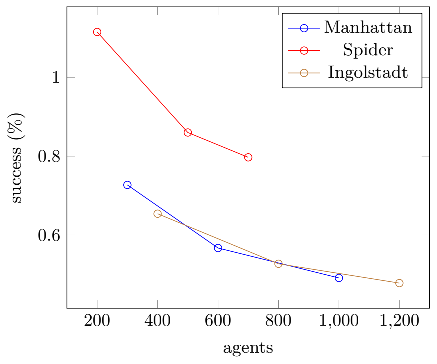


Fig. 3. Success rate of random attacks with a fixed number of seven simulated days (see table 1).

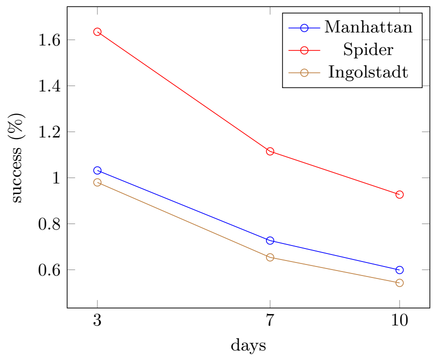


Fig. 4. Success rate of random attacks with a fixed number of agents (see table 1).

Our results suggest that the random attack worsens with an increase in the number of agents as well as in the number of simulated days. This is no surprise, since an increasing number of agents reduces (at least in the most simple case of equal prices) the probability of assigning a transaction to the correct wallet.

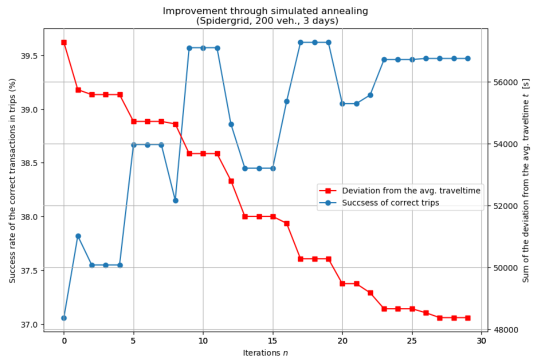
Days	Agents	Mean (%)	StDev	Agents	Mean (%)	StDev	Agents	Mean (%)	StDev
				Evaluation of wallets					
	Manhattan			Spider net			Ingolstadt		
3	300	7.28	0.24	200	9.99	1.08	400	10.19	0.09
7	300	4.99	0.13	200	6.99	1.59	400	7.23	0.20
10	300	4.47	0.14	200	6.3	0.14	400	6.56	0.27
7	600	3.24	0.05	500	3.73	0.1	800	4.50	0.09
7	1000	2.49	0.28	700	2.88	0.09	1200	3.37	0.03
				Evaluation of trips					
	Manhattan			Spider net			Ingolstadt		
3	300	37.45	0.47	200	37.21	1.08	400	38.99	0.46
7	300	38.79	0.44	200	39.24	1.59	400	40.23	0.79
10	300	38.95	0.46	200	36.87	0.82	400	39.82	0.55
7	600	33.16	0.64	500	28.72	0.65	800	33.66	0.33
7	1000	28.09	1.36	700	24.01	0.34	1200	29.53	0.29

Table 2. Evaluation of trip reconstruction attack wallets

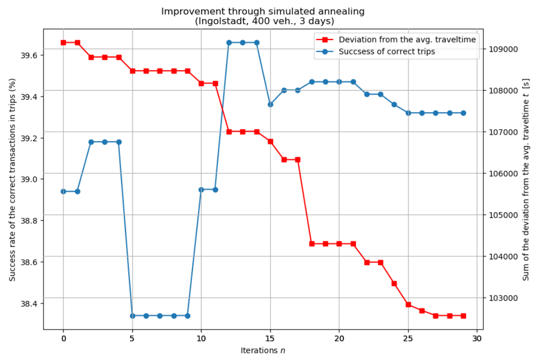
Trip reconstruction attack

The improved attacker tries to track trips and then combine them. The trips are reconstructed from a transaction by looking for the detector that can be reached by the first or last detector. The deviation from the average travel time is then compared with the real travel time. The times with the lowest deviation are saved in each case and then one of them is randomly selected, which is then selected as the new detector. If there is no nearby detector to reach in a realistic time, the trip is considered as over and will be saved. In addition, the deviation from the average values is saved. Transactions that could not be assigned are given a high weight (deviation time). The trips with the highest deviations (weights) are released again and the attacker tries to reconstruct a better trip from them. At the end, work continues with the trips that have the smallest deviation. Trips that are as similar as possible (run over as many identical edges as possible) are put together. These are then put together and an attempt is made to bring the sum of the costs up to those of the wallets.

This attack is being evaluted by formula 1 both for wallets and the trips, which is present in table 1 and figures 6 and 7. It is suggested that the attacker's success of reconstructing trips decreases with an increasing number of agents, but rises the more days are being simulated. Compared to the random attacker, this approach has a significantly higher rate of correctly assigned wallets.



(a)



(b)

Fig. 5. The trip reconstruction attacker's simulated annealing approach improves with the number of iterations.

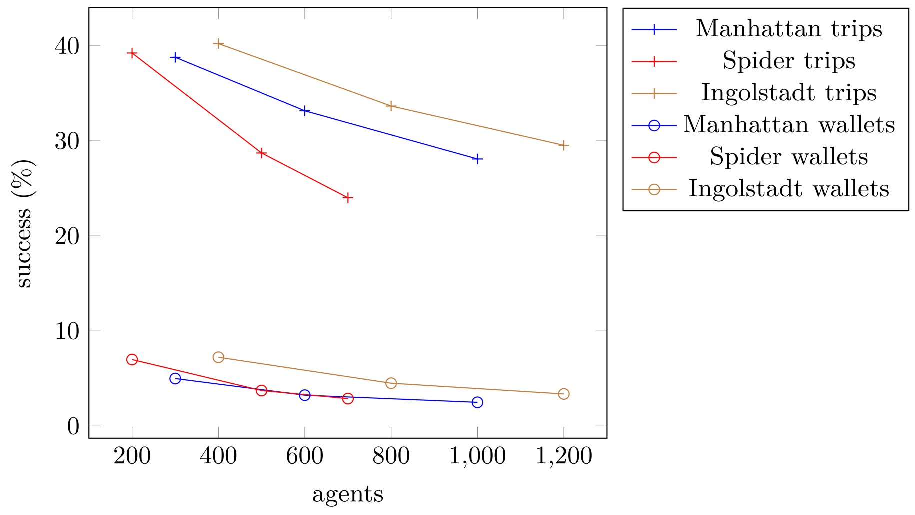


Fig. 6. Success of trip reconstruction attacks with a fixed number of seven simulated days (see table 1).

4.2 Performance

SUMO simulations are generally time-consuming. To evade unsuitably long simulations, the agent generation uses a speed-up factor for accelearation: departure times were contracted by this factor to avoid long periods with sparse traffic. We checked that this does not lead to collisions in the sense that a vehicle departs from a detector before it arrives there (due to short times of staying), but can not guarantee this, and thus, the speed-up factor remains an unpleasant needle in the simulations.

It generally stood out that the major factor of influence on the runtime is the number of simulated days, with the map (i.e., the number and placement of detectors) and the number of agents being secondary. The number of agents influences the process of cleaning detector data (i.e., removing detector data, which are only one time step apart and result from a waiting vehicle), which gets slower when the number of agents are higher.

4.3 Framework evaluation and further work

Though we think that our framework provides a good basis, it still has some rough edges which make some improvements and further work obvious.

The demand generation is currently limited to full-time and part-time workers and could be diversified to match reality more precisely. We think of implementing night-time workers and adding weekly appointments like groceries, sport courses or taking the children to school.

In SUMO, vehicles trapped in dead locks of turning cars or heavily congested streets are teleported. This is problematic for the trip reconstruction attacker: a car is moved to another place on the map, which obviously makes a plausible trip reconstruction impossible. The two main parametres that affect the appearance of teleportation are the number of cars and the map's topography. We ran numerous simulations beforehand to see whether teleporting happens and to adjust the maximum number of cars suitable for the map as well as tight street constellations. However, there is no automatic or systematic procedure of dealing with teleporting vehicles. On the other hand, one could argue in favour of allowing teleportation, since there would exist artefacts and errors in a real-world TSP's database. Implementing an attacker capable of dealing with such artefacts is of interest on its own. More generally, the addition of an even more sophisticated attack is imaginable and could be subject of a thesis or a further paper.

During simulations, we observed that some detectors (with no costs) on the Ingolstadt map lead to erroneous data. These data have been cleaned manually (using a small Python script explicitly written for this purpose), but time could be spent to investigate the origin and fixing of this errors.

One issue that was due to time limitations is the fact that all our simulations had every detector equally priced. We are majorly interested in the behaviour of our attacker on maps with different price levels such as higher priced low emission zones.

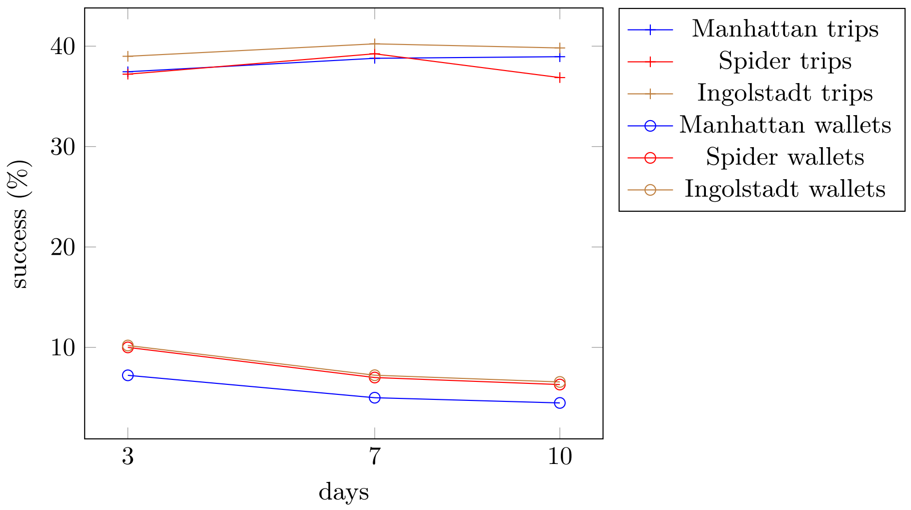


Fig. 7. Success rate of random attacks with a fixed number of agents (see table 1).

5 Conclusion

Based on the program SUMO, we implemented the Python framework ETSi for simulating electronic toll collection and attacks on the users' location privacy. Our framework is capable of generating traffic demand with flexibly settable home and work districts on any properly prepared SUMO map. We executed a random and trip reconstruction attack which uses a simulated annealing approach and evaluated our attacks with The results show that while the random attacker lacks success and rapidly decreases with an increasing number of agents and days, the trip reconstruction attacker could obtain notable success in reconstructing user trips and is significantly better in wallet assignment than the random attacker. Apparently, this attacker's trip success is stable or even slightly improves with more simulated days, but worsens with an increase in agents; the reconstruction of wallets diminishes with both more agents and days.

References

[1] United Nations, Department of Economic and Social Affairs, Population Division, "World urbanization prospects: The 2018 revision," 2019.

[2] European Commission, "Commission decision on the definition of the european electronic toll service and its technical elements," 2009.

[3] European Parliament, Directorate-General for Internal Policies , "Technology options for the european toll service," 2014.

[4] European Statistical Office, "Passenger cars, by type of motor energy and size of engine," 2021.

[5] A. J. Blumberg and P. Eckersley, "On locational privacy, and how to avoid losing it forever," *Electronic frontier foundation*, vol. 10, no. 11, pp. 1–7, 2009.

[6] American Civil Liberties Union, "Newly obtained records reveal extensive monitoring of e-zpass tags throughout new york," 2015.

[7] Family Lawer Magazine, "I-pass tollway data, divorce, and privacy," 2009.

[8] European Commission, Directorate-General for Mobility and Transport, "Study on "state of the art of electronic road tolling"," 2015.

[9] Transport for London, "Road user charging in london."

[10] "Simulation of urban mobility, https://eclipse.org/sumo/," 2022.

[11] R. A. Popa, H. Balakrishnan, and A. J. Blumberg, "Vpriv: Protecting privacy in location-based vehicular services," 2009.

[12] J. Balasch, A. Rial, C. Troncoso, B. Preneel, I. Verbauwhede, and C. Geuens, "Pretp: Privacy-preserving electronic toll pricing.," in *USENIX Security Symposium*, vol. 10, pp. 63–78, 2010.

[13] S. Meiklejohn, K. Mowery, S. Checkoway, and H. Shacham, "The phantom tollbooth: Privacy-preserving electronic toll collection in the presence of driver collusion.," in *USENIX security symposium*, vol. 201, 2011.

[14] V. Fetzer, M. Hoffmann, M. Nagel, A. Rupp, and R. Schwerdt, "P4tc-provably-secure yet practical privacy-preserving toll collection," *Proceedings on privacy enhancing technologies*, vol. 2020, no. 3, pp. 62–152, 2020.

[15] InTAS Project, "Intas (ingolstadt traffic scenario for sumo) github repository."

[16] SUMO Documentation, "Induction loops detectors (e1)."

[17] SUMO Documentation, "generatetlse1detectors.py."

[18] SUMO Documentation, "Abstract network generation."

[19] European Statistical Office, "Average number of usual weekly hours of work in main job, by sex, professional status, full-time/part-time and economic activity," 2021.