Master Defense

Detecting Sybil Attacks using Proofs of Work and Location for Vehicular Ad-Hoc Networks (VANETS)

Presented by:
Niclas Bewermeier

Electrical and Computer Engineering

December 14, 2018

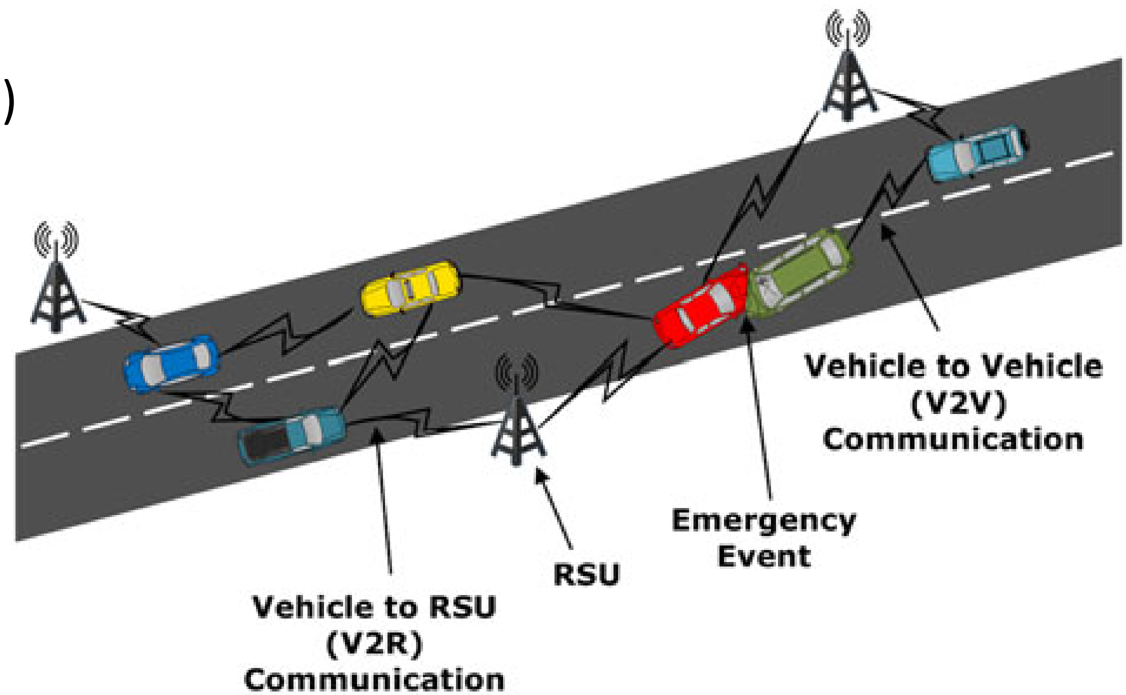
Outline

- **Introduction**
- Sybil Attack Detection using Proofs of Work and Location Solution
- Evaluations
- Conclusion and Future Work

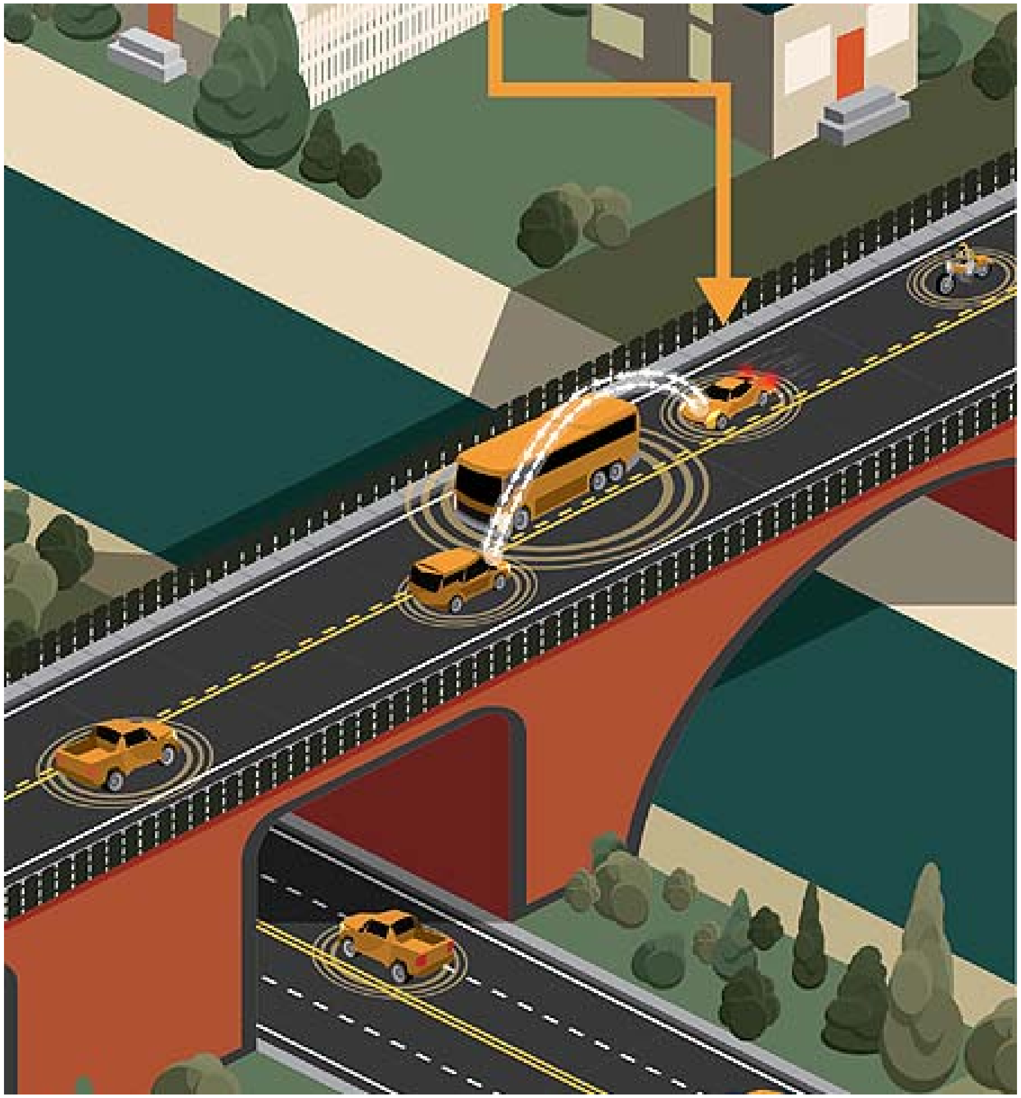
Vehicular Ad-Hoc Networks (VANETs)

- Vehicles communicate
  - With each other (V2V)
  - With infrastructure (V2I)

- Objectives:
  - Improve
    - Road safety
    - Traffic efficiency
    - Infotainment



Safety-related Applications for VANETs

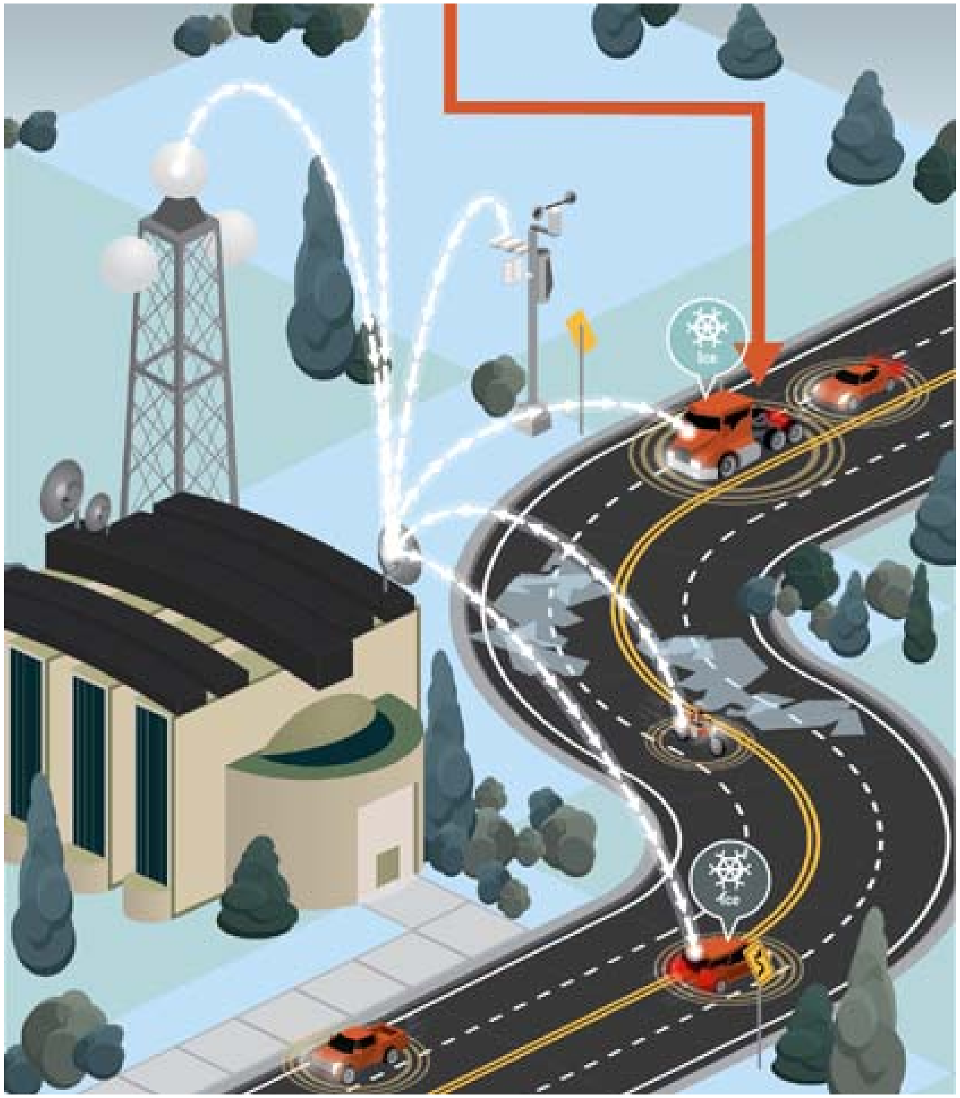


Do-not-pass Warning

Safety-related Applications for VANETs

Emergency Electronic Brakelight Warning

Safety-related Applications for VANETs



Road Weather Connected Vehicle Applications

Authentication in VANETs

- Vehicles need to exchange various messages
  - Warning against congestion, accident
  - Emergency on the road
  - Many other cases

- Authentication of messages is very important
  - Ensure that messages are sent from intended nodes and also from legitimate members, i.e., protect against
    - Impersonation attacks
    - Data modification attacks
    - Sending false information by external attackers.

- Message authentication can be achieved using digital signature

Authentication vs. Privacy

- There is a conflict between privacy and authentication

Authentication

- A proof that you are a legitimate user.
- Achieved by giving some information about yourself, i.e. a signature

X

Privacy

- You do not want to reveal information about yourself
  - Your location
  - Your identity
  - Your activity

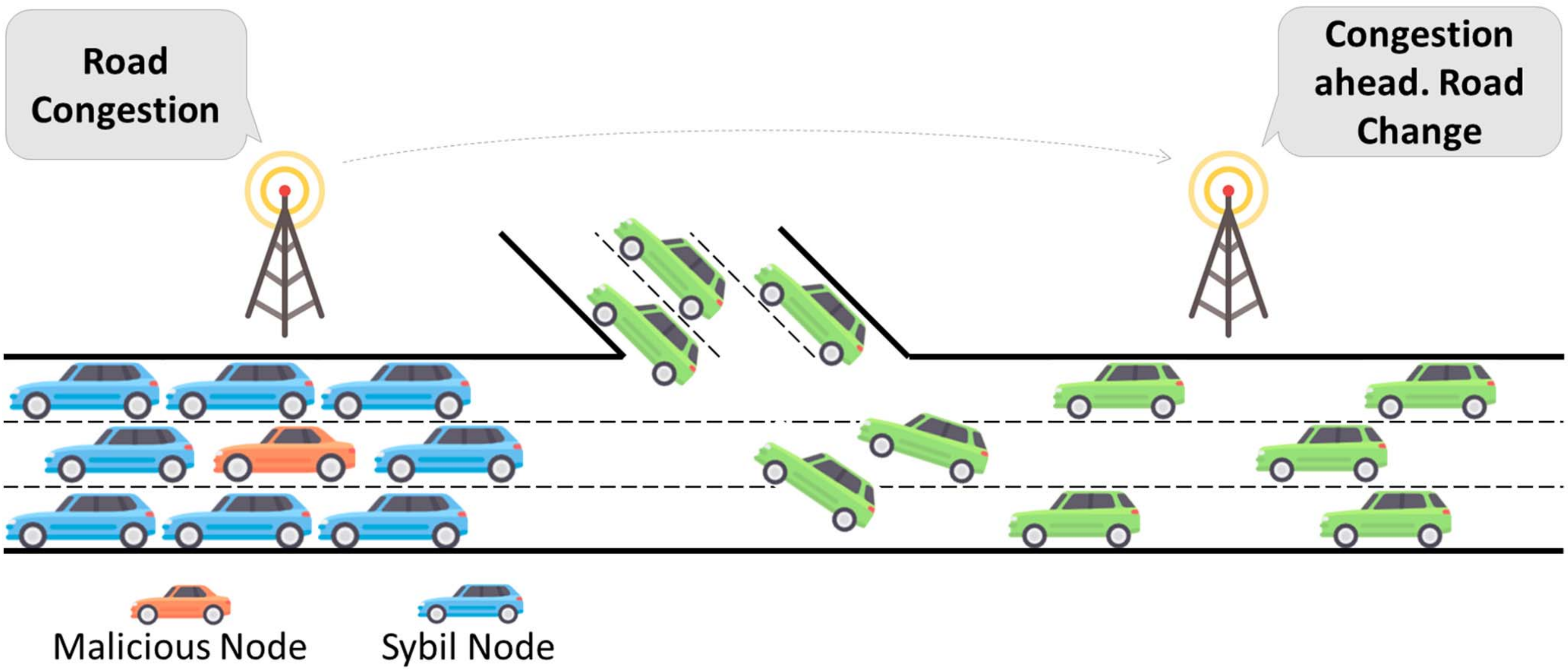
↓

Anonymous Authentication

Anonymity is ”the state of being not identifiable within a set of subjects called the anonymity set”.

What is Sybil attack?

- An attacker pretends to be multiple simultaneous vehicles at different locations.
- The credibility of received events increases when large number of vehicles report the same event.
- Traffic management needs accurate number of cars.



Contributions

- We propose a Sybil attack detection scheme based on time-stamped and anonymously signed messages issued by RSUs.

- We employ the concept of Proof-of-Work (PoW) to limit an attacker's ability to create multiple Sybil nodes. We also provide a method to determine appropriate PoW-target values with respect to time.

- We apply a Threshold Signature scheme to be secure against RSU compromise attacks.

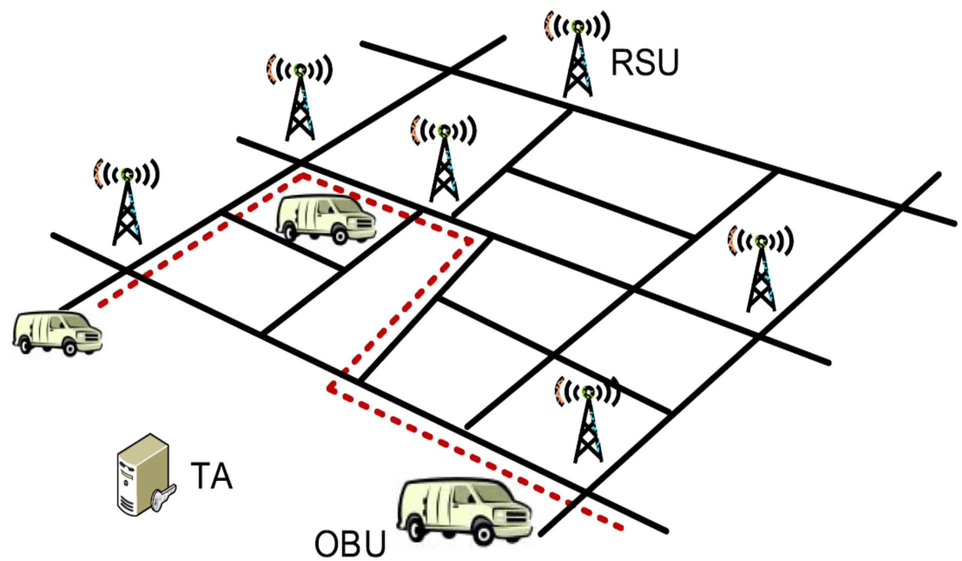
- We conduct extensive simulations to evaluate the performance of the proposed scheme.

Outline

- Introduction
- **Sybil Attack Detection using Proofs of Work and Location Solution**
- Evaluations
- Conclusion and Future Work

RSU:

- Provides wireless access to users within its coverage
- RSUs are interconnected (RSU backbone network).



OBU:

- Can communicate with RSUs and other vehicles via wireless connections.

Off-line Trust authority:

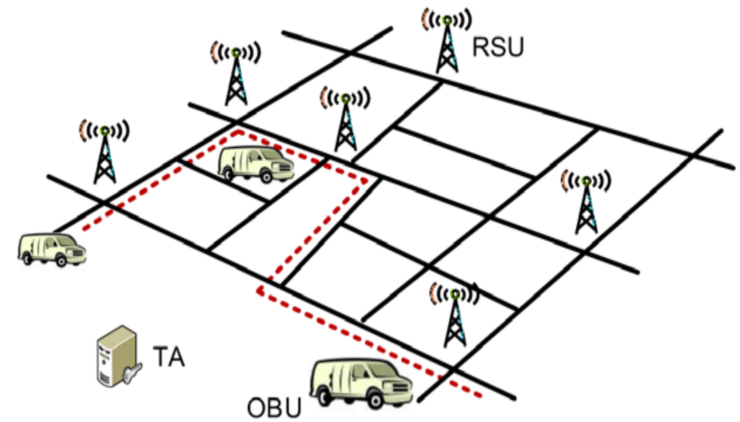
- Responsible for system initialization
- Connected to RSU backbone network
- Does NOT serve vehicles for any certification purpose

WU1

Slide 12

WU1 we said it is better if each vehicle has certifictaes and psudonyms

Windows User, 12/13/2018



Two steps

1. Trajectory Generation
2. Sybil attack detection

Definition of Trajectory

A vehicle anonymously authenticates itself using its **trajectory**.

- When passing by an RSU, a vehicle obtains an authorized message as proof of presence at particular RSU at a given time
- A set of consecutive authorized messages form a **trajectory**

Trajectory Generation

- In future conversations, a vehicle uses its individual trajectory to authenticate itself

Sybil Attack Detection

Assumption:
The mobility of vehicles is independent.

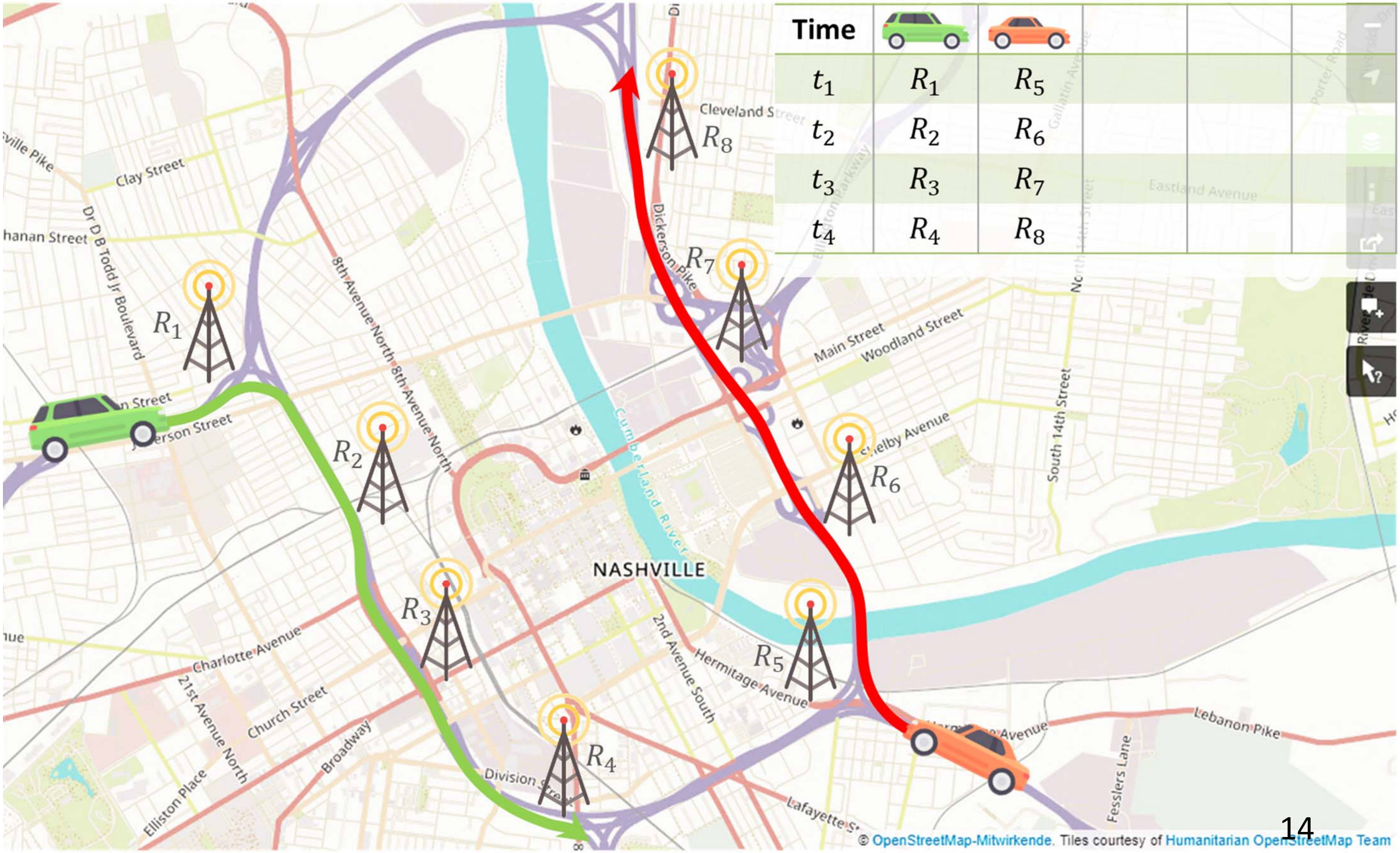
This means individual vehicles move independently, and therefore would not travel along the same route for all the time.

Slide 13

WU2 Do not use converstation this is for humams instead use communication
Windows User, 12/13/2018

- Proposed Scheme -

Authentication using Trajectories

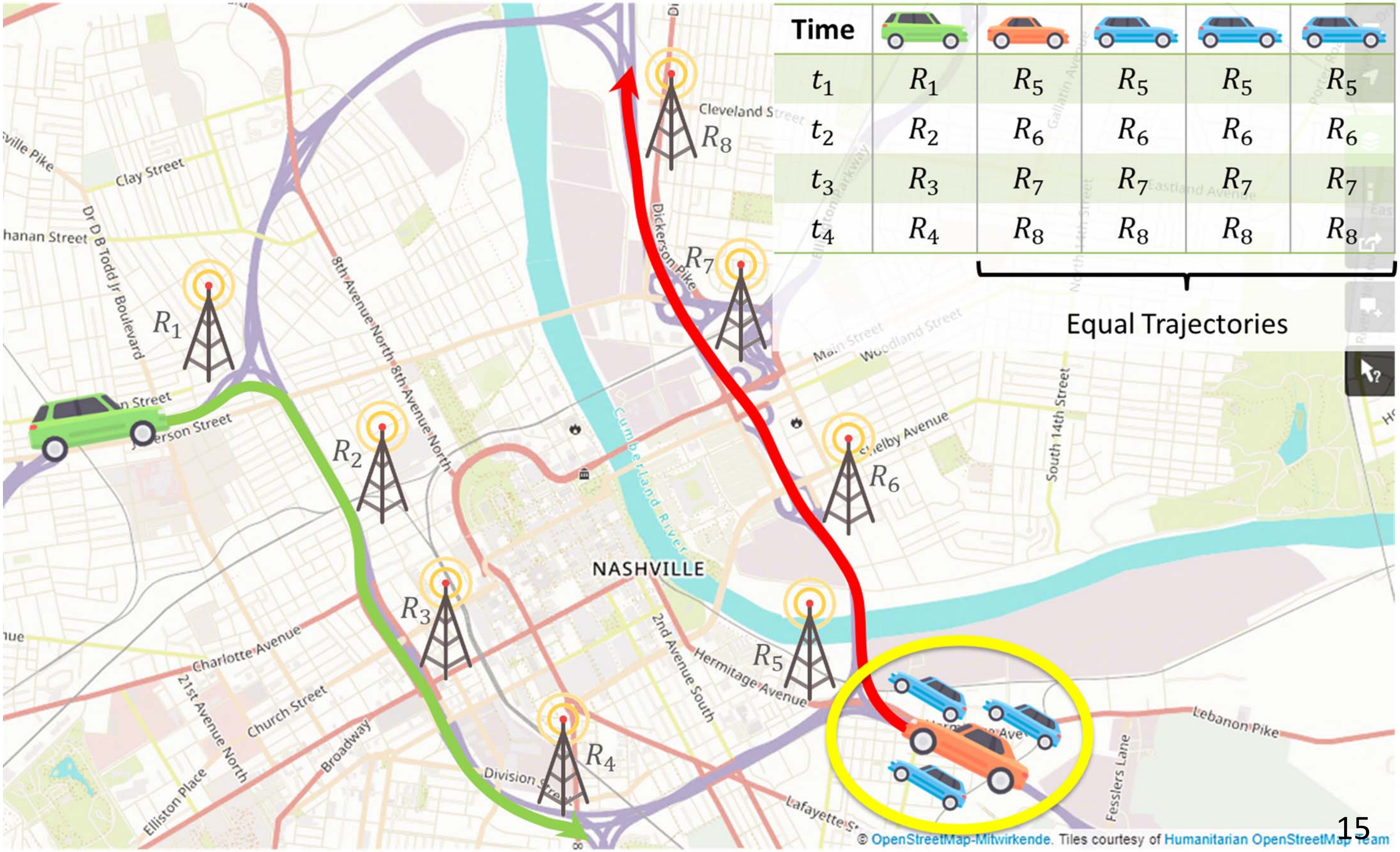


Time				
t_1	R_1	R_5		
t_2	R_2	R_6		
t_3	R_3	R_7		
t_4	R_4	R_8		

Authentication using Trajectories

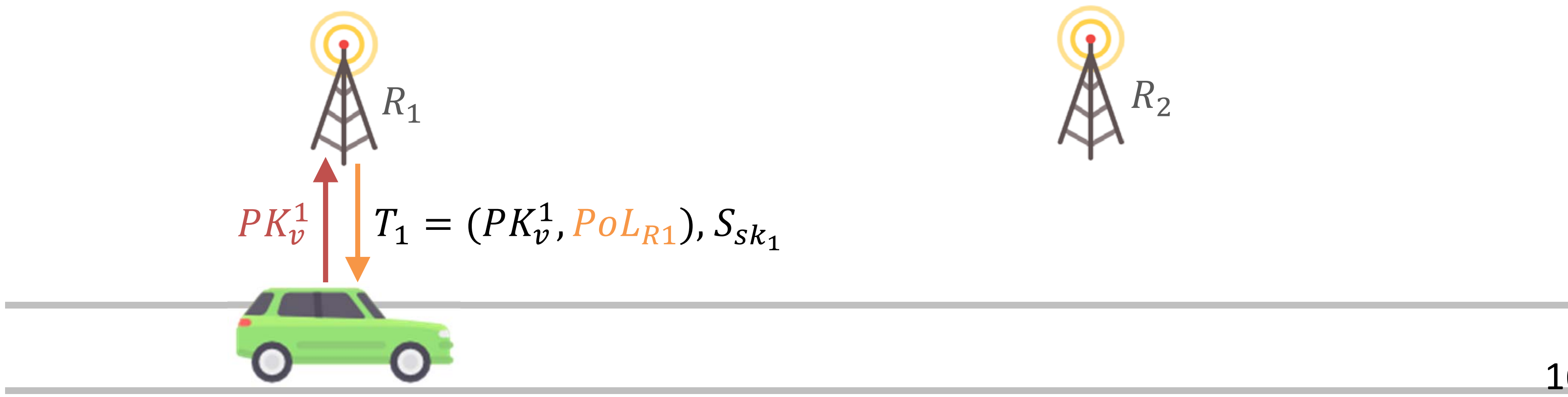
Time					
t_1	R_1	R_5	R_5	R_5	R_5
t_2	R_2	R_6	R_6	R_6	R_6
t_3	R_3	R_7	R_7	R_7	R_7
t_4	R_4	R_8	R_8	R_8	R_8

Equal Trajectories



Obtaining Authorized Timestamped Messages

1. Vehicle generates Private/Public Key pair SK_v^1/PK_v^1
2. Vehicle requests an authorized message by submitting PK_v^1
3. RSU generates Proof-of-Location: $PoL_{R1} = t_s, tag_{R1}^p$
4. RSU signs on (PK_v^1, PoL_{R1}): $S_{sk_1}(PK_v^1, PoL_{R1})$
5. RSU issues authorized message $\mathrm{T}_1 = (PK_v^1, PoL_{R1}, S_{sk_1})$ to vehicle



Proof-of-Work

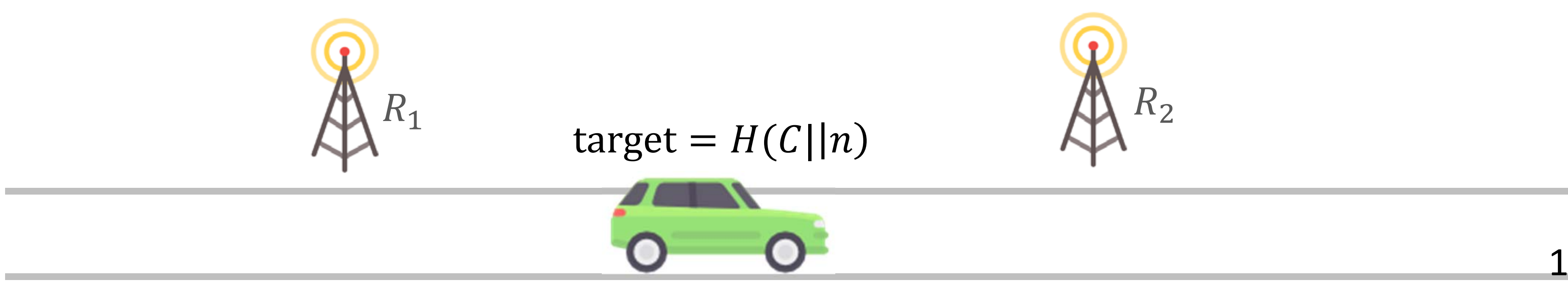
Goal: Prevent vehicles from creating multiple trajectories at a time.

Vehicle:

Upon receiving of T_1

- generate challenge $C = H(T_1)$
- start running the PoW algorithm:
  - Calculate target $= H(C||n)$ WU4e incrementing n.
  - Keep the lowest target.

The longer it takes a vehicle to traverse from R_1 to R_2, the lower the value of target should become due to the probabilistic behavior.



Slide 17

WU4 Why you hash T1 first why you do not put it here directly with n
Windows User, 12/13/2018

Proof-of-Work Verification

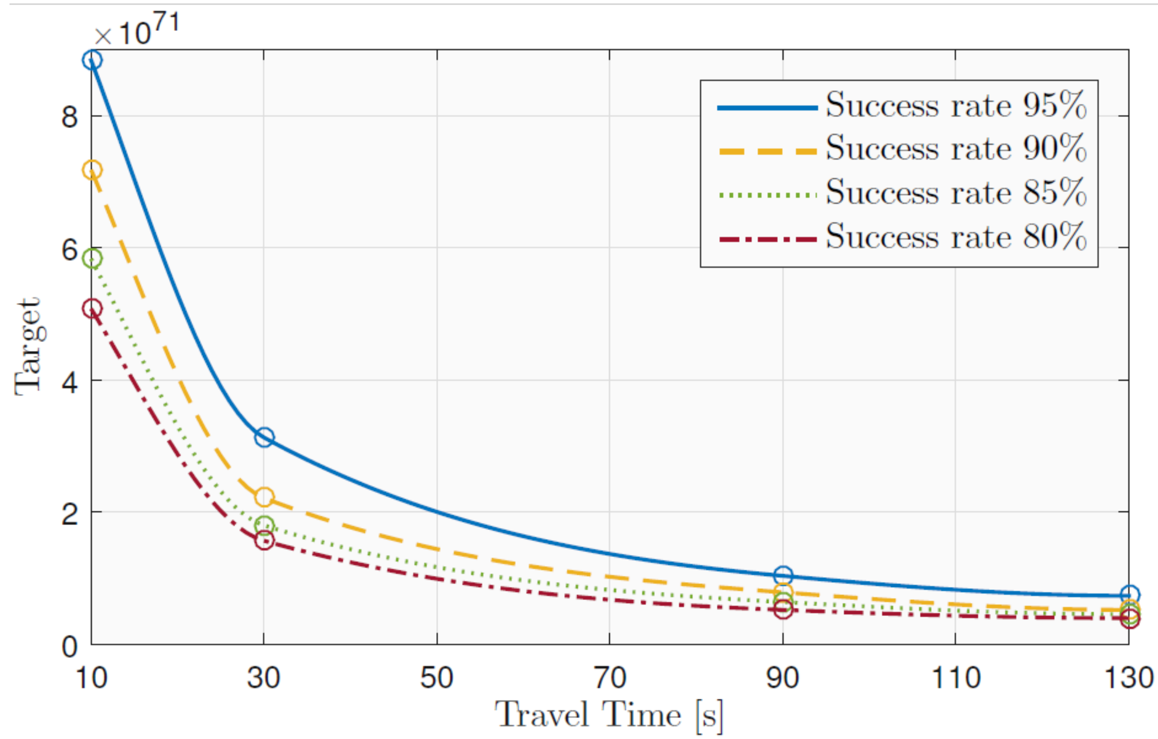
Vehicle:

1. generates Private/Public Key pair SK_v^2/PK_v^2
2. signs on previously obtained authorized message T_1, PK_v^2, target, and n: $S_{SK_v^1}(T_1, PK_v^2, \text{target}, n)$
3. requests a new authorized message by submitting $L_1 = (T_1, PK_v^2, \text{target}, n), S_{SK_v^1}$

Proof-of-Work Verification

RSU:

1. Verify if
   $H(n||c) \stackrel{?}{=} \text{target}$
2. Determine travel time using t_s of PoL_1:
   $\Delta T = t_c - t_s$
3. Look up expected target and check if
   $\text{target} \leq \text{target}_{\text{lookup}}$

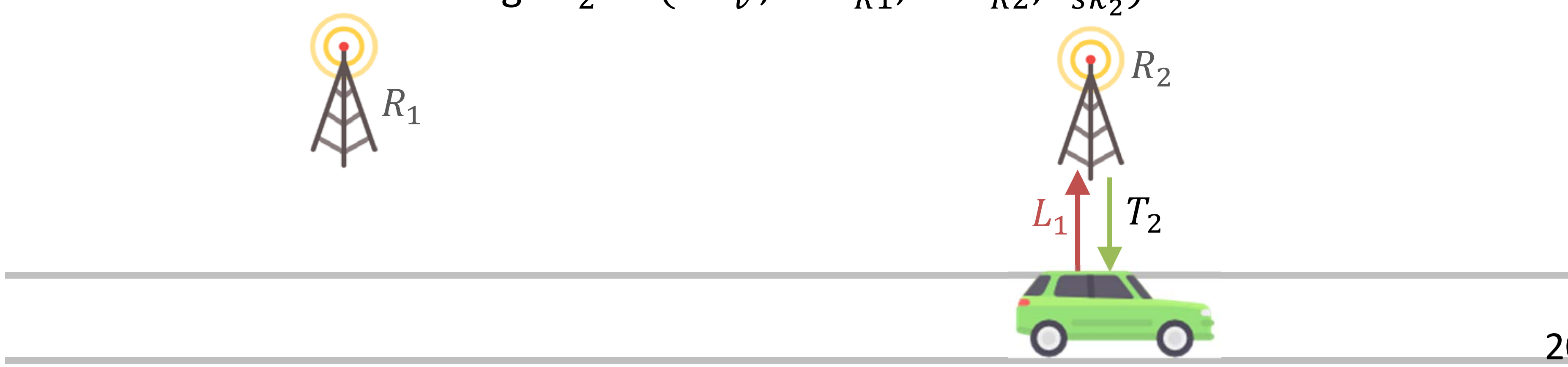


Vehicle: **Message Verification**

1. generates Private/Public Key pair SK_v^2 / PK_v^2 ✓
2. signs on previously obtained authorized message T_1, PK_v^2, target, and n: $S_{SK_v^1}(T_1, PK_v^2, \text{target}, n)$ ✓
3. requests a new authorized message by submitting $L_1 = (T_1, PK_v^2, \text{target}, n), S_{SK_v^1}$ ✓

RSU:

1. verifies Proof-of-work ✓
2. verifies $S_{SK_v^1}$ and S_{sk_1}
3. generates Proof-of-Location: $PoL_{R2} = t_s, tag_{R2}^p$
4. signs on $(PK_v^2, PoL_{R1}, PoL_{R2})$: $S_{sk_2}(PK_v^2, PoL_{R1}, PoL_{R2})$
5. issues authorized message $T_2 = (PK_v^2, PoL_{R1}, PoL_{R2}, S_{sk_2})$ to vehicle

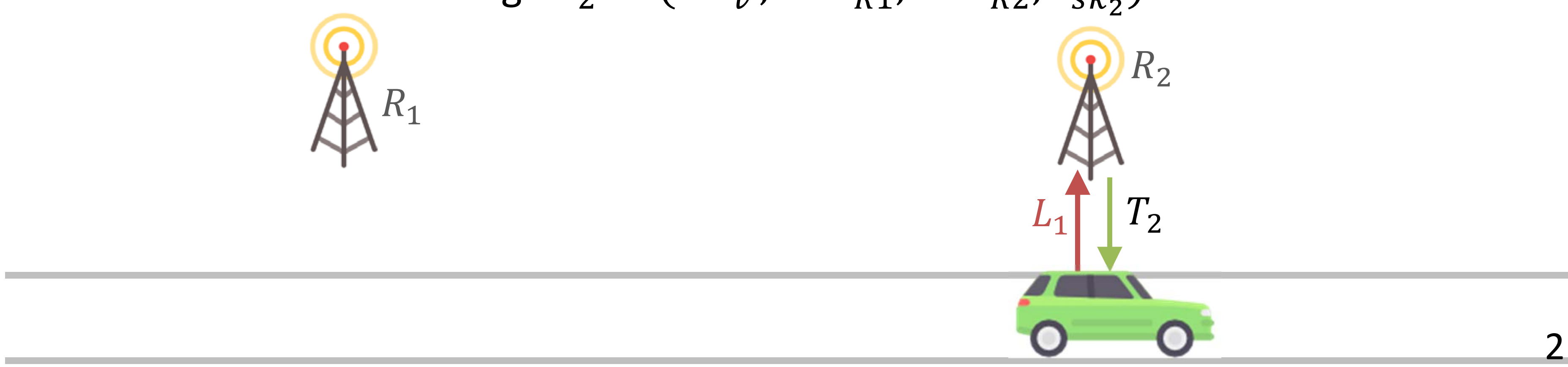


Vehicle: **Message Verification**

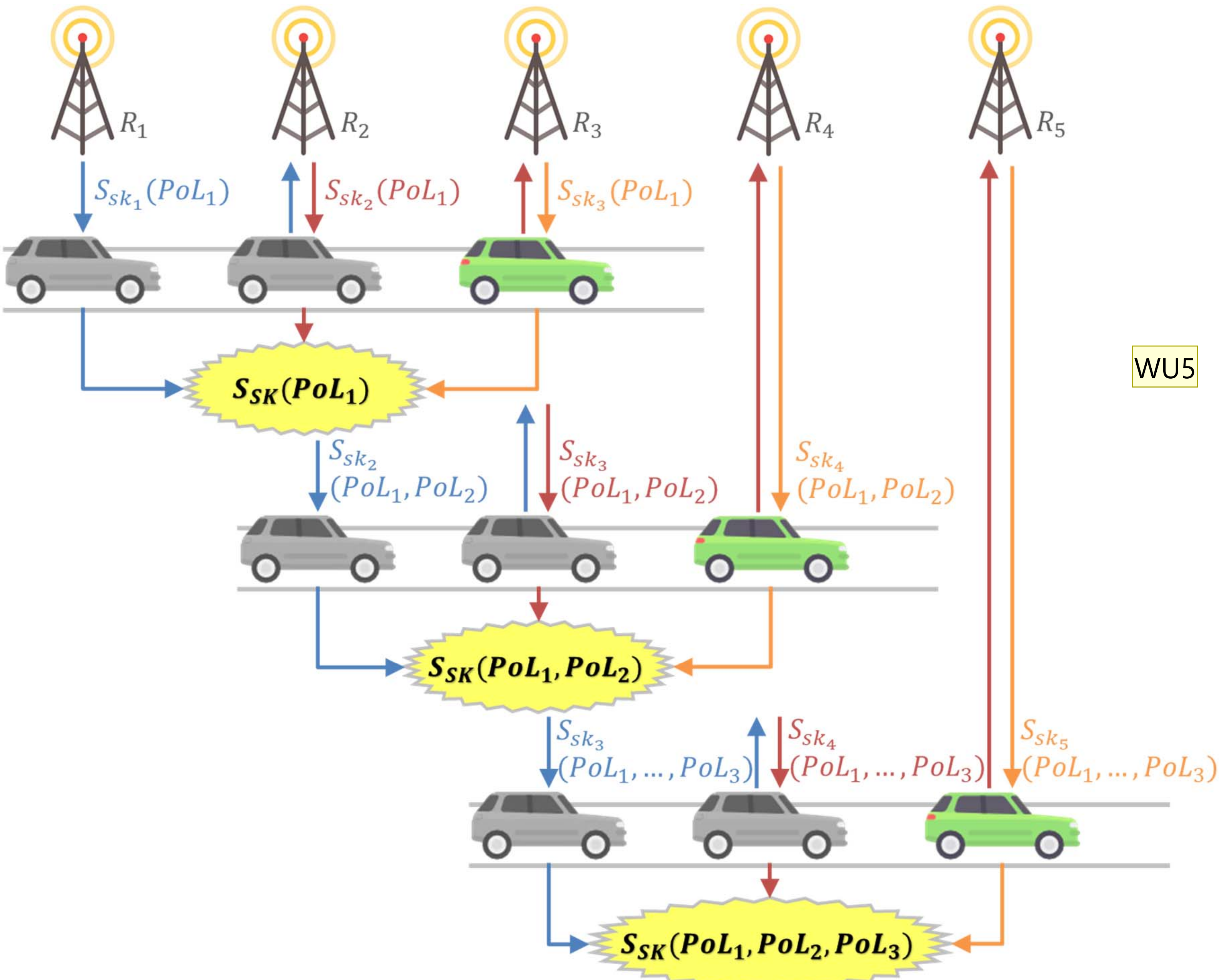
1. generates Private/Public Key pair SK_v^2/PK_v^2 ✓
2. signs on previously obtained authorized message T_1, PK_v^2, target, and n: $S_{SK_v^1}(T_1, PK_v^2, \text{target}, n)$ ✓
3. requests a new authorized message by submitting $L_1 = (T_1, PK_v^2, \text{target}, n), S_{SK_v^1}$ ✓

RSU:

1. verifies Proof-of-work ✓
2. verifies $S_{SK_v^1}$ and S_{sk_1}
3. generates Proof-of-Location: $PoL_{R2} = t_s, tag_{R2}^p$
4. signs on $(PK_v^2, \boxed{PoL_{R1}, PoL_{R2}})$: $S_{sk_2}(PK_v^2, PoL_{R1}, PoL_{R2})$
5. issues authorized message $T_2 = (PK_v^2, PoL_{R1}, PoL_{R2}, S_{sk_2})$ to vehicle



Collaborative Trajectory Generation



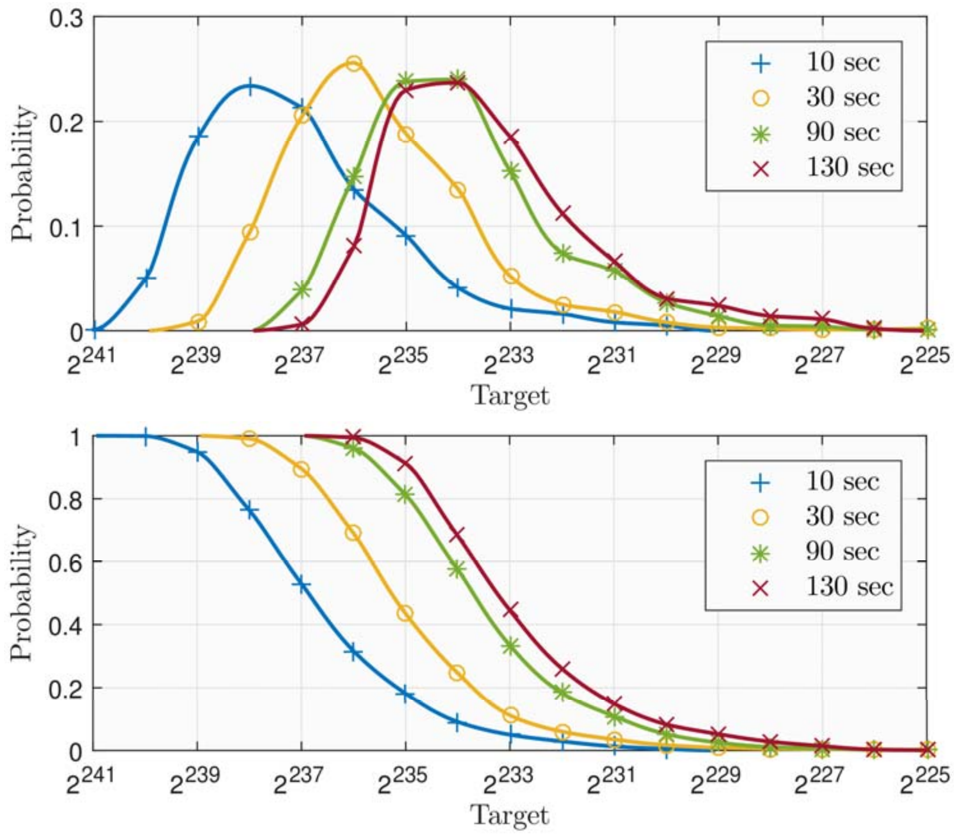
WU5

Slide 22

WU5 using threshold signature was not focused on here
Windows User, 12/13/2018

Selection of PoW Targets

1. Run PoW algorithm for constant times to obtain probability distributions.



<u>Experiment Setup:</u>

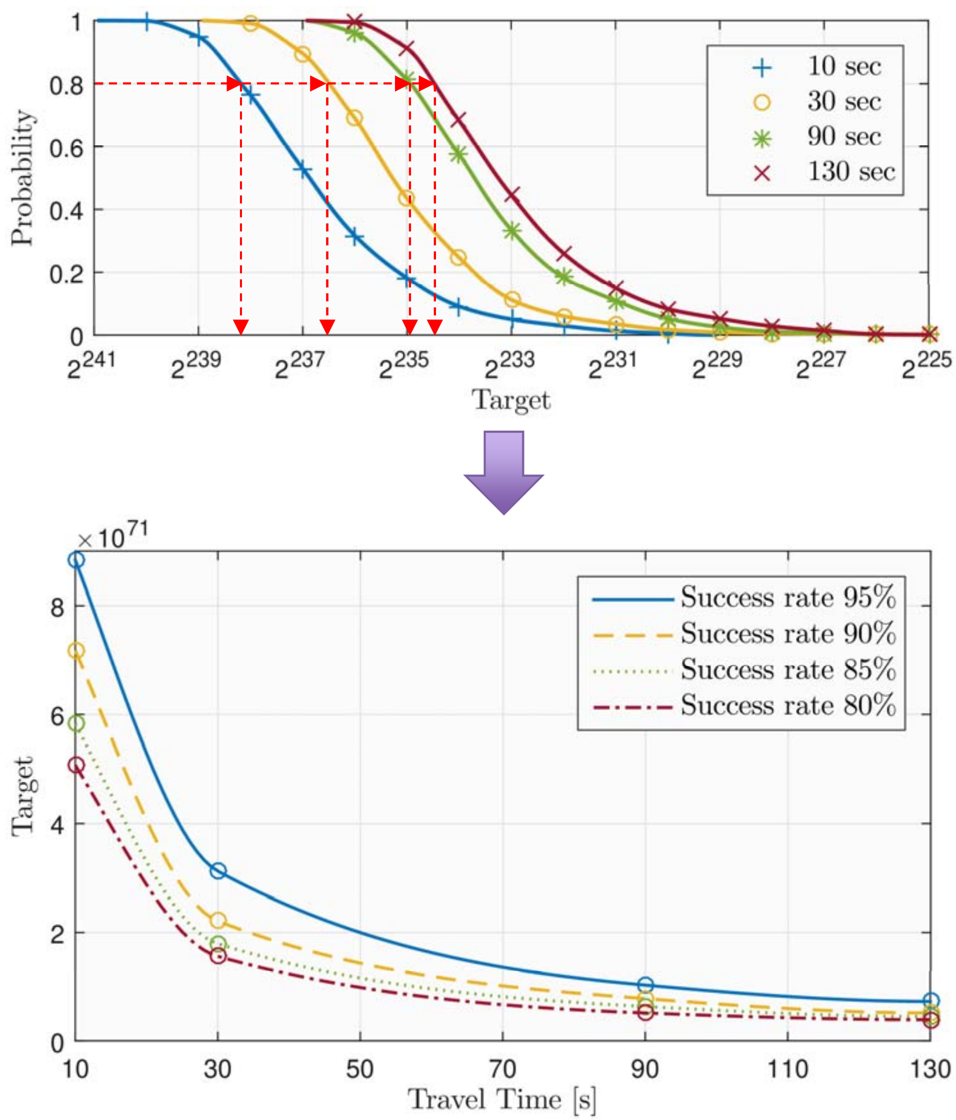
- Raspberry Pi 3
  (1.2 GHz processor, 1 GB RAM)

- Travel times:
  10 sec, 30 sec, 90 sec, 130 sec

- Number of samples:
  1000 per travel time

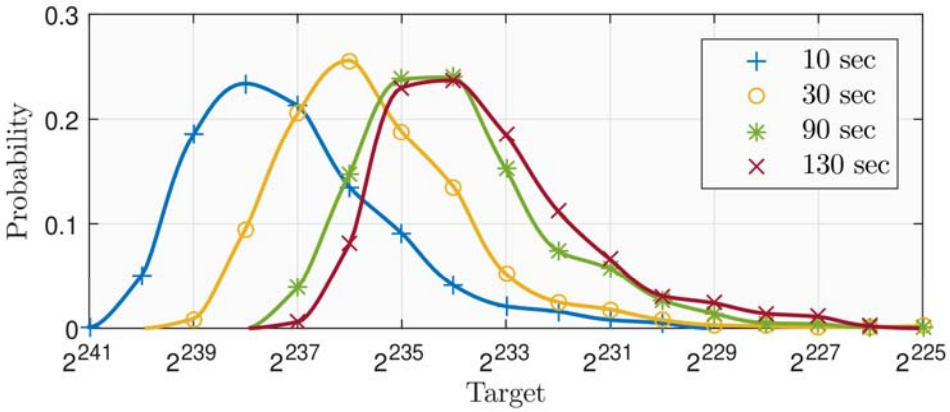
Selection of PoW Targets

2. Map data into Target Lookup Table



- Proposed Scheme -

Selection of PoW Targets



<u>Experiment Setup:</u>

- Raspberry Pi 3
  (1.2 GHz processor, 1 GB RAM)

- Travel times:
  10 sec, 30 sec, 90 sec, 130 sec

- Number of samples:
  1000 per travel time

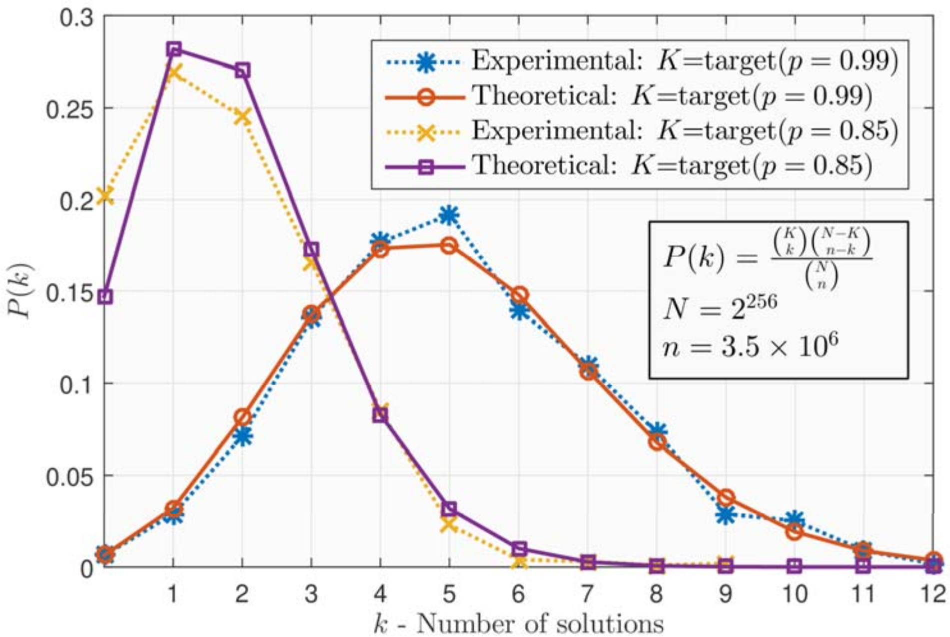
Mathematical Model:

Hypergeometric Distribution: $P(k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$

$N = 2^{256}$ *;* Output range of SHA-256
$K =$ target ; Target at given probability
$n =$ Number of hashes per travel time on RPi 3
$k = 1$; Number of solutions

Selection of PoW Targets

3. Run PoW algorithm for constant time and obtain the number of solutions found for *defined target values.*



Experiment Setup:

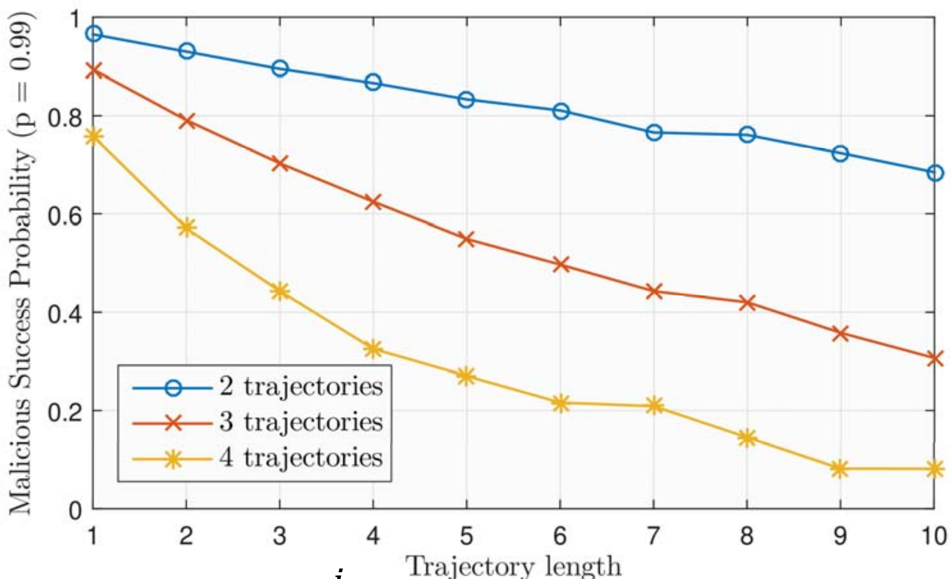
- Raspberry Pi 3
  (1.2 GHz processor, 1 GB RAM)
- Travel time: 90 sec.
- Targets:
  at $p = 0.99$: 16.74×10^{70},
  at $p = 0.85$: 6.34×10^{70}
- Number of samples:
  1000 per target

Hypergeometric Distribution: $P(k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$

$N = 2^{256}$; Output range of SHA-256
$K =$ target ; Target at given probability
$n = 3.5 \times 10^6$; Number of hashes per 90 sec. on RPi 3
$k =$ Number of solutions

Selection of PoW Targets

4. Run PoW algorithm for constant time and obtain the number of solutions found for *defined target values.*
   – *Probability of generating k trajectories of length j*



$$p = \left(\sum_k^0 \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}} \right)^j$$

$N = 2^{256}$; Output range of SHA-256
$K = 16.74 \times 10^{70}$; Target at given probability
$n = 3.5 \times 10^6$; Number of hashes per 90 sec. on RPi 3
$k =$ Number of solutions
$j =$ Trajectory Length

<u>Experiment Setup:</u>

- Raspberry Pi 3
  (1.2 GHz processor, 1 GB RAM)
- Travel time between two RSUs:
  90 sec.
- Trajectory length: 1…10
- Number of simultaneous trajectories:
  2, 3, 4
- Number of samples:
  1000 per target

Two Steps

1. Trajectory Generation ✓
2. Sybil attack detection

Sybil attack detection

During a conversation (initialized by a **vehicle or an RSU**):

1. Participating vehicles should provide their trajectories for verification
2. The conversation holder verifies each trajectory
3. The conversation holder conducts **online Sybil attack detection**
4. Proceeding with the conversation

Trajectory T_v of vehicle v:

$$S_{SK_v^n}(PK_v^n, PoL_1, PoL_2, PoL_3, \dots PoL_n)$$
$$T_v = PK_n^n, PoL_1, PoL_2, PoL_3, \dots PoL_n, S_{SK_v^n}$$

Sybil attack detection

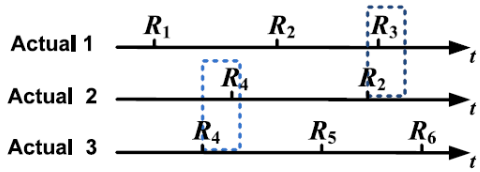
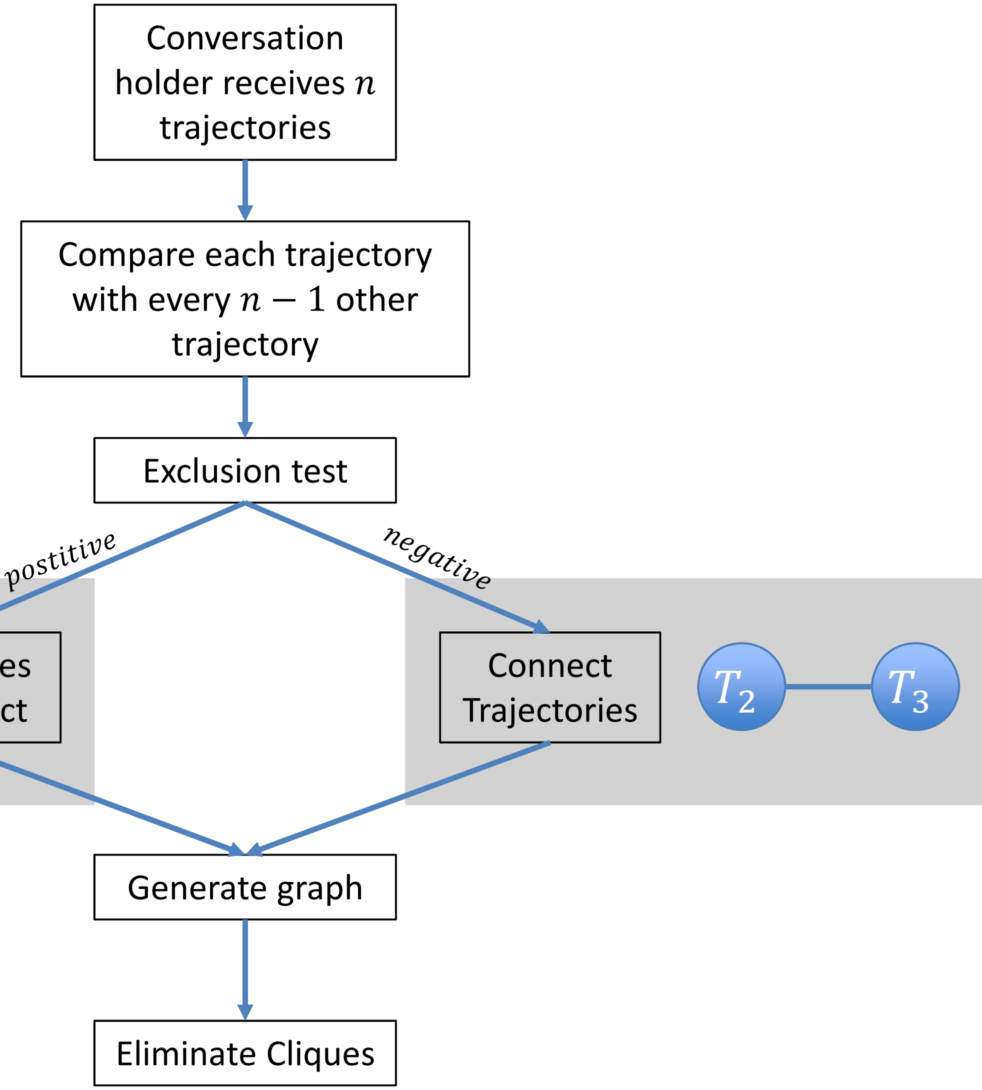
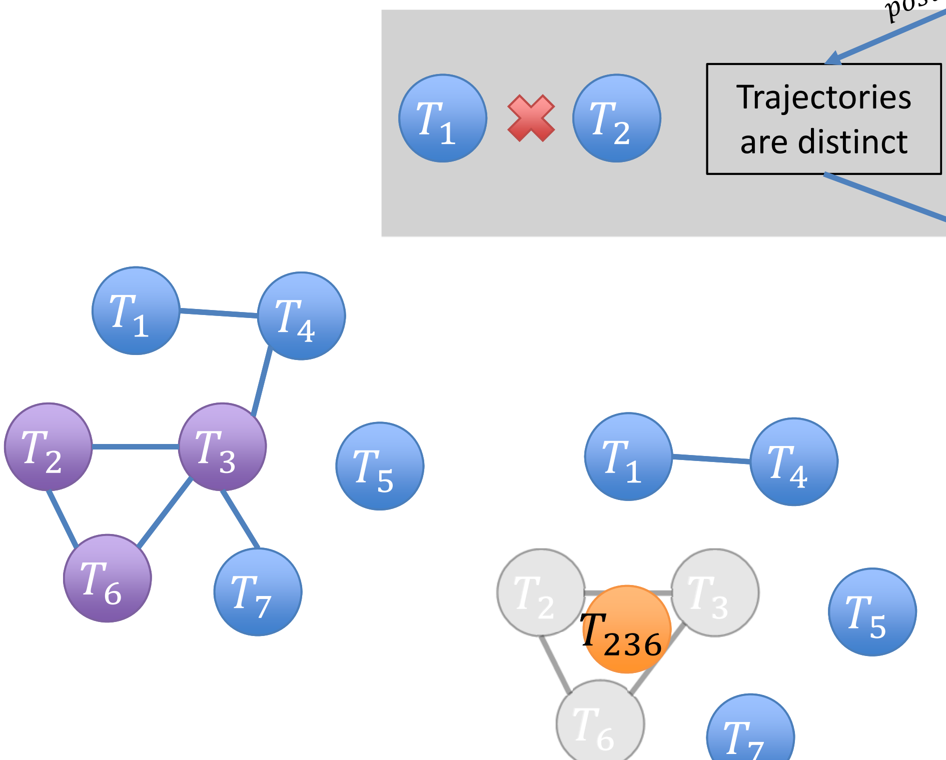
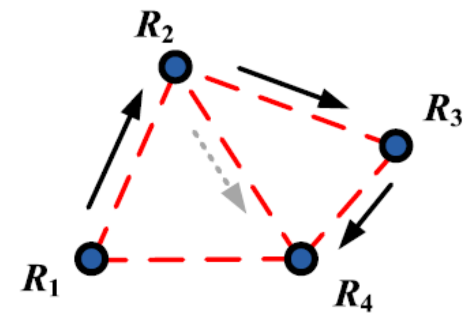


Fig. 3. Checking for distinct trajectories by using a check window (denoted as the box of dotted line) and counting the total number of different RSUs contained in a pair of trajectories.





Sybil attack detection



Features of forged trajectories:

1. A forged trajectory is a proper subset of the actual trajectory

2. Any two forged trajectories cannot have two distinct RSUs at the same time (otherwise the malicious vehicle would appear at two locations at the same time)

Features of actual trajectories:

1. It is very hard (if not impossible) for a single vehicle to traverse between a pair of RSU's shorter than a time limit
=> *traverse time limit: the shortest time for a vehicle to travel between any pair of RSUs in the system*

2. Within a limited time period, the total number of RSUs traversed by a single vehicle is less than a limit
=> *trajectory length limit: the maximum number of RSUs involved in a trajectory within an event*

Both limits can be measured based on the distance and speed limitations of each road segment and the layout of RSU deployment

Sybil attack detection

Exclusion test: examine whether two trajectories are distinct

Two trajectories pass the test (positive test) if:

- Two distinct RSUs within time window (*traverse time limit)* (T1, T2)

or

- Number of RSUs in merged RSU sequence larger than *trajectory limit* (T1, T3, if limit is 5)

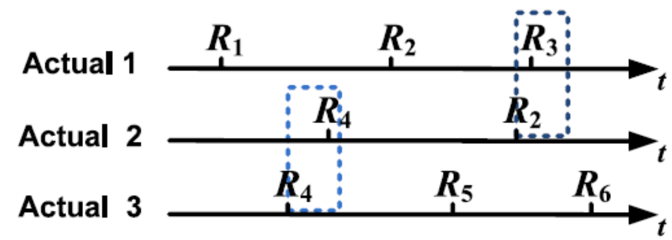


Fig. 3. Checking for distinct trajectories by using a check window (denoted as the box of dotted line) and counting the total number of different RSUs contained in a pair of trajectories.

In all other cases, the pair of trajectories fails the test (negative test, T2, T3)

Outline

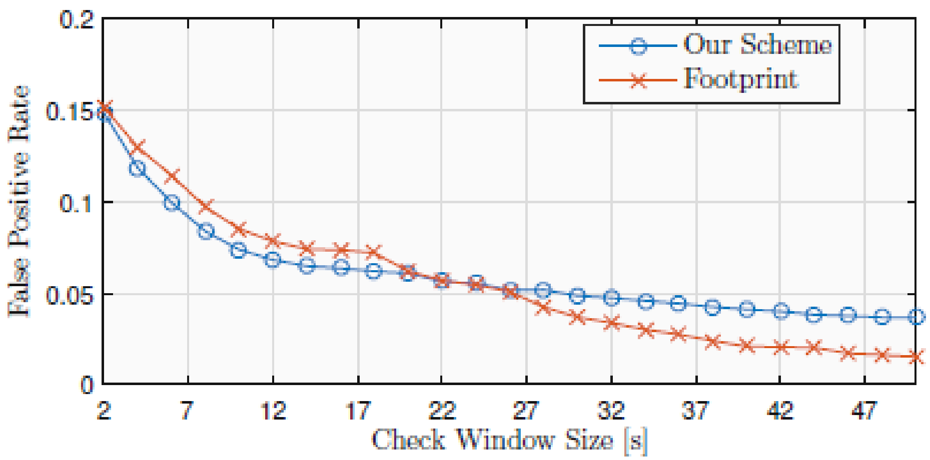
- Introduction
- Sybil Attack Detection using Proofs of Work and Location Solution
- **Evaluations**
- Conclusion and Future Work

Simulation - Setup

- Map of Nashville, TN (75.5 km x 33 km)
- Generation of 160 random routes
- Truncate routes into 460 trajectories according to trajectory length limit
- 0.1 x 460 malicious vehicles
- Every malicious vehicle generates up to 15 forged Sybil trajectories

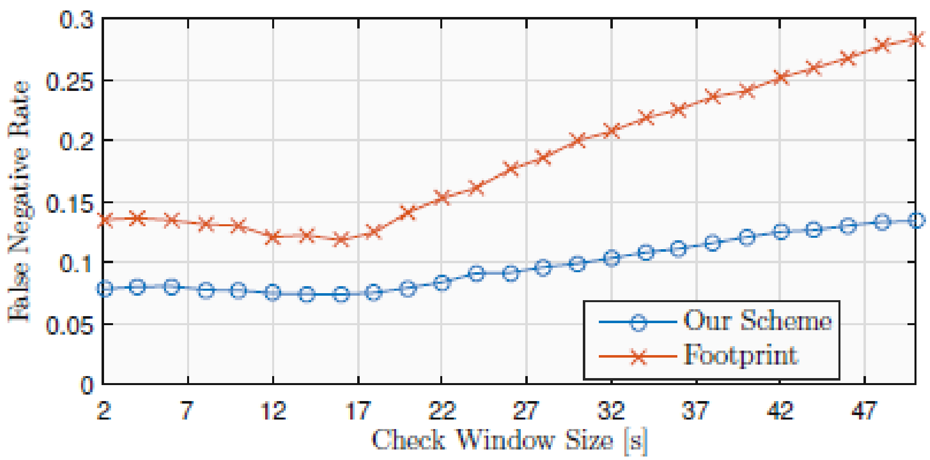
Simulation - Results

1. Impact of the Check Window Size

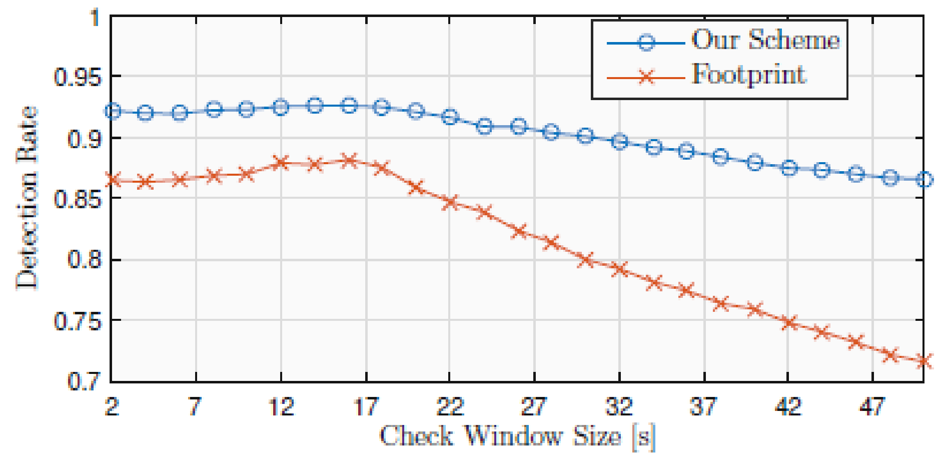


(a) Check window size versus false positive rate.

- Variable Check Window Size: 2, …, 50
- Constant Trajectory Length Limit: 15 sec
- Number of runs per setting: 30



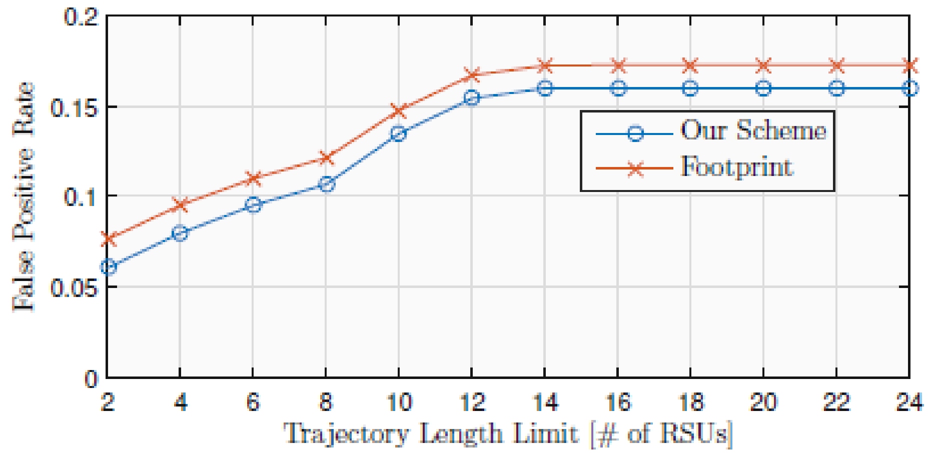
(b) Check window size versus false negative rate.



(c) Check window size versus detection rate.

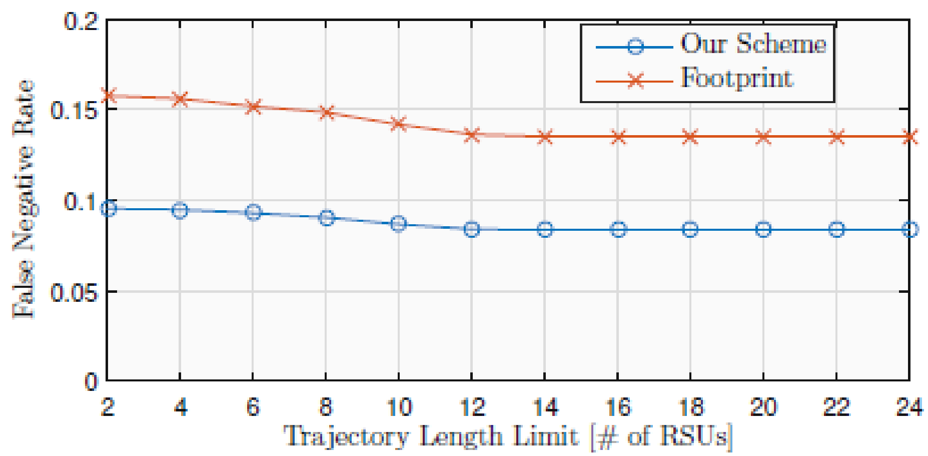
Simulation - Results

2. Impact of the Trajectory Length Limit

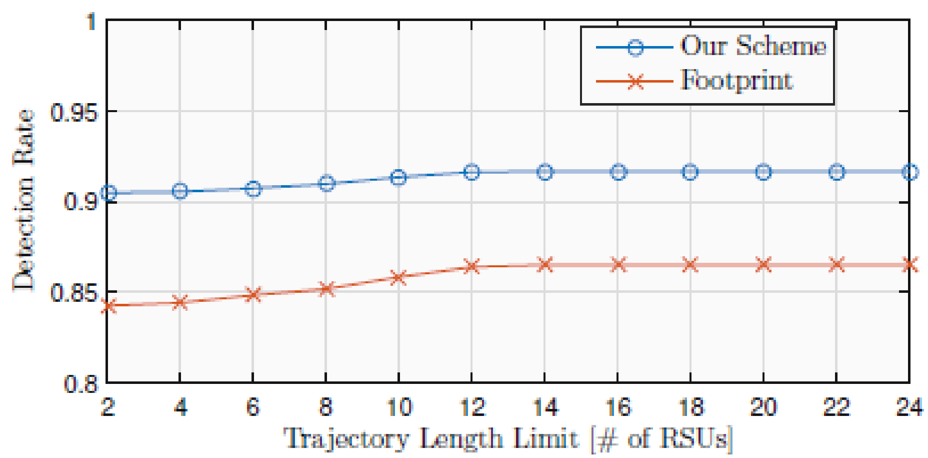


(a) Trajectory length limit versus false positive rate.

- Constant Check Window Size: 7
- Variable Trajectory Length Limit: 2,…,24
- Number of runs per setting: 30



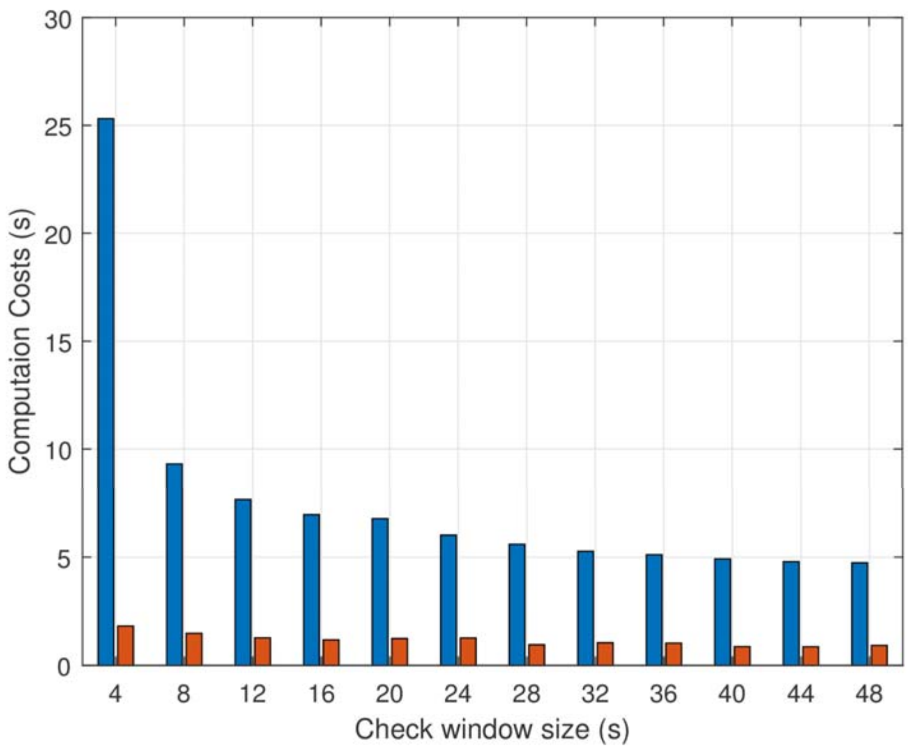
(b) Trajectory length limit versus false negative rate.

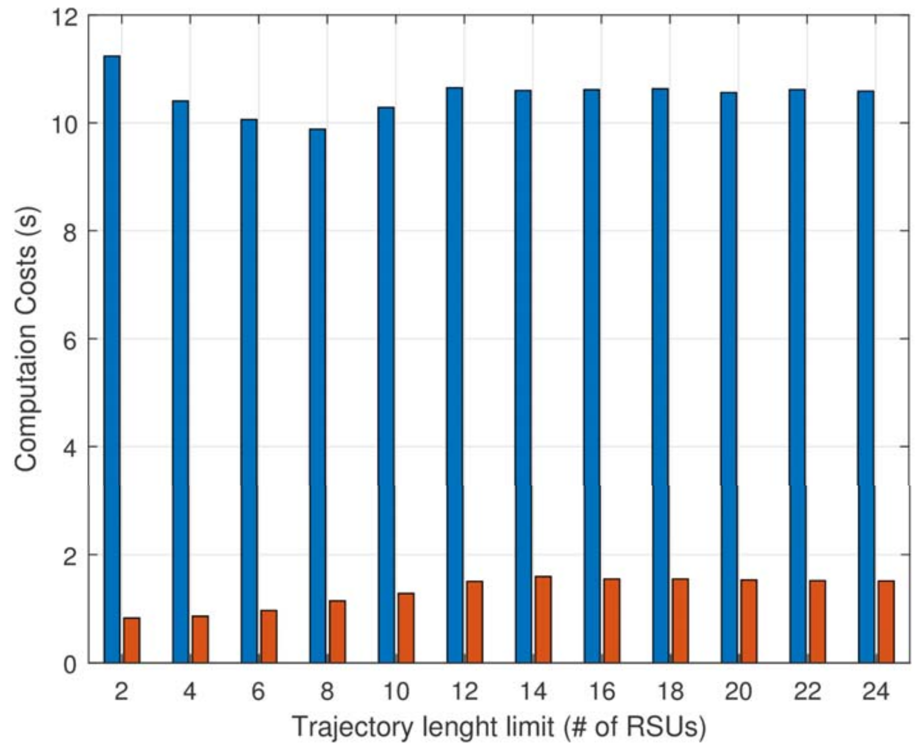


(c) Trajectory length limit versus detection rate.

Simulation - Results

3. Computation Cost of Eliminating Sybil Nodes



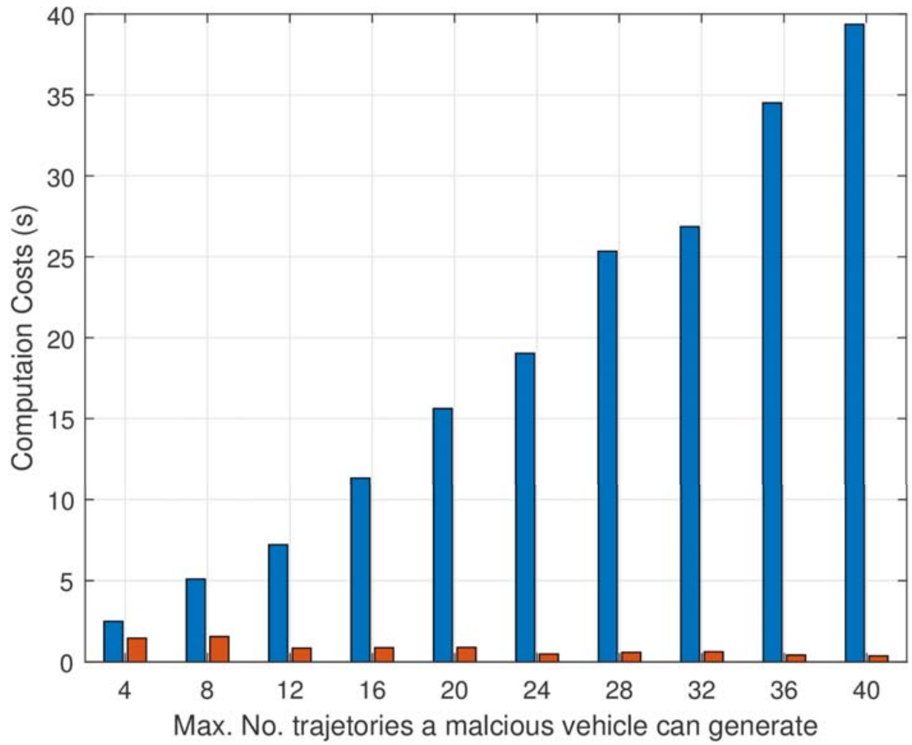


- Variable Check Window Size: 4,…,48
- Constant Trajectory Length Limit: 15
- Number of runs per setting: 15

- Constant Check Window Size: 18
- Variable Trajectory Length Limit: 2,…,24
- Number of runs per setting: 15

Simulation - Results

3. Computation Cost of Eliminating Sybil Nodes (cont.)



- Constant Check Window Size: 18
- Constant Trajectory Length Limit: 15
- Variable number of trajectories per malicious vehicle: 4,…,40
- Number of runs per setting: 15

Outline

- Introduction
- Sybil Attack Detection using Proofs of Work and Location Solution
- Evaluations
- **Conclusion and Future Work**

Conclusion

- In this thesis, we have proposed a Sybil attack detection scheme using anonymous trajectories.
- A threshold signature scheme was used which requires RSUs to collaborate in issuing authorized Proof-of-Locations, mitigating security threats caused by compromised RSUs.
- In order to prevent malicious vehicles from launching Sybil attack, the concept of Proof-of-Work was used to limit the number of trajectories a vehicle can create simultaneously.
- A method on determining appropriate target values with respect to vehicles' travel times has been introduced.
- Our simulation results show that our scheme can achieve high detection rates while maintaining low false positive rates.
- By limiting the number of Sybil trajectories using Proof-of-Work, we can drastically reduce the time for detecting Sybil attack.
- Our scheme is secure against t compromised RSUs, and if t is large enough, compromise attack is infeasible .

Future Work

- We are planning to investigate additional heuristics WU6he exclusion test, that will allow to better identify two honest vehicles traveling common routes during their trips. This would further decrease the false positive rate.

- We are going to review more sophisticated Proof-of-Work algorithms, such as solutions that involve operations on memory, where the solving time is less dependent on the available computational resources.

- In order to become more suitable to the short contact times of V2V and V2I communication in VANETs, approaches to further reduce the time it takes to eliminate Sybil trajectories will be studied.

Slide 41

WU6 Do not put much text here see my comments on the thesis about using blockchain and making a scheme in case there are no RSUs
Windows User, 12/13/2018

Publication

Journal Paper

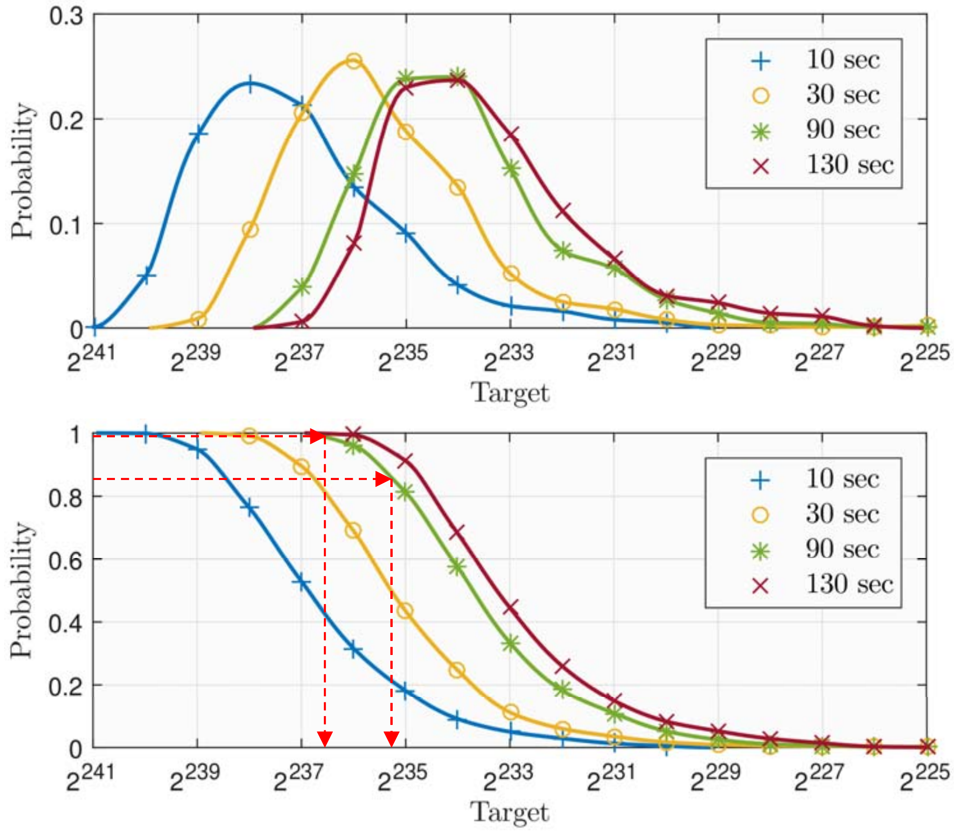
- Mohamed Baza, Niclas Bewermeier, Mahmoud Nabil, Kemal Fidan, Mohamed Mahmoud, and Mohamed Abdallah. "*Proofprint: Detecting Sybil Attacks Leveraging Proofs of Work and Location in VANETs*", to be submitted to IEEE Access.

Thank you!

Questions?

Selection of PoW Targets

1. Run PoW algorithm for constant times to obtain probability distributions.



<u>Experiment Setup:</u>

- Raspberry Pi 3
  (1.2 GHz processor, 1 GB RAM)

- Travel times:
  10 sec, 30 sec, 90 sec, 130 sec

- Number of samples:
  1000 per travel time