Secure Information Sharing among Autonomous Vehicles in NDN

Muktadir Chowdhury
University of Memphis
Memphis, TN 38152
mrchwdhr@memphis.edu

Ashlesh Gawande
University of Memphis
Memphis, TN 38152
agawande@memphis.edu

Lan Wang
University of Memphis
Memphis, TN 38152
lanwang@memphis.edu

ABSTRACT

Autonomous vehicles must communicate with each other effectively and securely to make robust decisions. However, today's Internet falls short in supporting efficient data delivery and strong data security, especially in a mobile ad-hoc environment. Named Data Networking (NDN), a new data-centric Internet architecture, provides a better foundation for secure data sharing among autonomous vehicles. We examine two potential threats, false data dissemination and vehicle tracking, in an NDN-based autonomous vehicular network. To detect false data, we propose a four-level hierarchical trust model and the associated naming scheme for vehicular data authentication. Moreover, we address vehicle tracking concerns using a pseudonym scheme to anonymize vehicle names and certificate issuing proxies to further protect vehicle identity. Finally, we implemented and evaluated our AutoNDN application on Raspberry Pi-based mini cars in a wireless environment.

CCS CONCEPTS

•Security and privacy →Pseudonymity, anonymity and untraceability; Mobile and wireless security; •Networks →*Naming and addressing;*

KEYWORDS

NDN, autonomous vehicles, authentication, privacy

ACM Reference format:
Muktadir Chowdhury, Ashlesh Gawande, and Lan Wang. 2017. Secure Information Sharing among Autonomous Vehicles in NDN. In *Proceedings of The 2nd ACM/IEEE International Conference on Internet-of-Things Design and Implementation, Pittsburgh, PA USA, April 2017 (IoTDI 2017),* 11 pages.
DOI: 10.1145/3054977.3054994

1 INTRODUCTION

Autonomous vehicles are the future of transportation, offering a variety of social benefits such as improving mobility for the elderly, disabled and children, reducing accidents caused by driver errors, decreasing congestion linked to selfish driving behavior, increasing fuel efficiency by reducing unnecessary braking, and enhancing human productivity by freeing us from driving ([5, 16]). To acquire real-time information about their environment, autonomous vehicles use an array of sensing technologies, e.g., sonar devices, cameras, lasers, and radars. However, because sensor data may be inaccurate or incomplete due to limited range, field of view, and obstructions, it is important for vehicles to share information with each other through vehicle-to-vehicle (V2V) communication ([2, 11, 14]). Vehicles can retrieve information that assist autonomous driving (e.g., map publishers) through vehicle-to-infrastructure (V2I) communication [2] so that they can make better decisions.

The implication of data sharing among vehicles is limitless. For example, when a vehicle's camera is malfunctioning or blocked by some object, it may not detect a pedestrian crossing an intersection, thus leading to an accident. However, if a nearby vehicle senses the person and shares the information with that vehicle, the accident may be avoided. As another example, a vehicle miles away from an accident may know the accident from vehicles coming from that area and change its route accordingly. Such data sharing can be useful to not only autonomous vehicles, but also connected vehicles with more limited autonomous driving capability. However, autonomous vehicles have a much higher dependency on the accuracy of the received data as they cannot rely on human drivers to judge the data or correct their behavior.

Despite the potential benefits of a network of autonomous vehicles, there are significant challenges to develop such a network in the current IP architecture. The first challenge is mobility and intermittent connectivity. The second challenge is security vulnerabilities that have been plaguing the Internet. These security problems can also exist in an autonomous-vehicular network and have severe consequences, since false data or unauthorized control commands can cause vehicles to make wrong decisions or maneuvers leading to deadly accidents. Unfortunately, due to the current Internet architecture's focus on a point-to-point communication and channel-based security model (e.g., IPSec [15], TLS [4] and VPN [6, 24]), it cannot support effective data distribution and strong data security in a *mobile* environment. We believe that the new data-centric network architecture *Named Data Networking (NDN)* [37] provides a better foundation to address these issues. NDN's name-based data retrieval removes the dependency on static locations and IP addresses, and its pervasive caching capability helps handle intermittent connectivity [38]. Furthermore, its built-in security features allow authenticating data without relying on a stable channel between two nodes or additional infrastructure support [36]. A number of research efforts have explored vehicular communication in NDN (e.g., [8–10, 32]), but they mainly focus on naming, forwarding strategy and link adaptation layer design, not security issues.

IoTDI 2017, Pittsburgh, PA USA

DOI: 10.1145/3054977.3054994

An NDN-enabled autonomous vehicular network is not immune to potential attacks [26], but applications in this network can take advantage of a number of built-in architectural mechanisms, e.g., *signature* in every data packet [37] and *schematized trust* enabled by hierarchical names [36], to address the attacks. This work focuses on two potential attacks, *false information dissemination* and *vehicle tracking* in such networks. There is an inherent conflict between approaches that address these two types of attacks. On one hand, we need to authenticate data to reduce the chance of accepting false information. This means that the data needs to carry some information identifying the vehicle that produced the data. In NDN, since we derive trust using names, this identifying information is usually in the Interest and Data names. On the other hand, identification information can be used to track a vehicle, so ideally we would like to hide the vehicle's identity. Without this protection, vehicle owners may not be willing to share their data, leading to ineffective decision making by the autonomous vehicles. *The challenge is to prevent both false information injection and vehicle tracking at the same time.*

To this end, we propose a trust model specific to NDN-based autonomous vehicular applications to prevent the injection of false information by either unauthorized vehicles or attackers impersonating other vehicles. We also sketch out a scheme to make it difficult for attackers to track vehicles under this trust model. *Note that the proposed mechanisms are applicable to not only fully autonomous vehicles, but also vehicles with lower degrees of autonomy as long as their behavior is affected by received data.* We have developed a prototype implementation of the proposed trust model and associated validation mechanism on the NDN platform using Raspberry Pi-based mini cars. We have also conducted experiments to investigate delays associated with our implementation.

The organization of the paper is as follows. Section 2 provides background on NDN. We present our threat model in Section 3, and our design including naming scheme, trust model, and privacy protection in 4. Section 5 details the implementation of our design, and Section 6 presents experiment setup and evaluation results. Finally, Section 7 presents related work and Section 8 concludes the paper.

2 BACKGROUND

Today's Internet applications are increasingly data centric. Massive amount of content such as video, audio, news, blogs, tweets, and images is generated and consumed by Internet users. Meanwhile, smart homes, wearables, sensors, vehicles and other Internet of Things (IoT) devices also generate vast amounts of data that needs to be accessed on-demand and at various granularities to enable monitoring and decision making services. To address this fundamental shift, the Named Data Networking (NDN) architecture [37] was proposed to make "data" the primary abstraction of the architecture. Below we describe how NDN works and explain why it serves as a good foundation for building a vehicular network.

An NDN network enables users and applications to simply specify a name to fetch desired data using *interest* and *data* packets (Figure 1 shows the packet format). More specifically, a consumer sends an *interest* packet containing the name of the desired data. The routers then use the name to guide each interest towards the

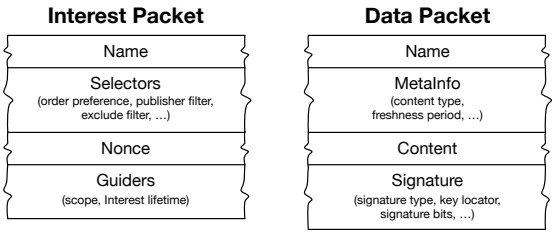

Figure 1: Interest and Data Packet Format [37]

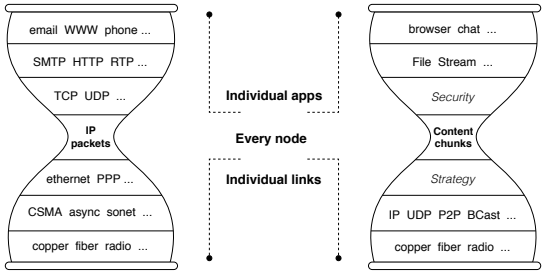

Figure 2: Internet vs. NDN [37]

data producer(s). Routing protocols and forwarding strategies on these routers help direct the interest. When the *data* is found, it is forwarded towards the consumer on the reverse path of the interest and *cached* in the intermediate nodes for future Interests. In a vehicular network where connectivity is constantly changing, name-based data retrieval and caching are two powerful NDN features supporting robust data sharing [39].

Security is a built-in layer in NDN (Figure 2), not an after-thought as in IP, because data authenticity is of paramount importance when the named data can be retrieved from any node (it can be completely forged by an attacker). To address this problem, NDN binds data name and content using a key owned by the data producer to generate a *cryptographic signature* that is part of the data packet. Moreover, the hierarchical name contained in every piece of data provides context for deriving trust. Data authenticity can then be verified using the information contained in the data packet, i.e., the data name, content and signature, as well as a predefined *trust model* specific to the application that generated the data. Furthermore, applications can control access to data via encryption and distribute the data decryption keys as encrypted NDN data. ***This data-centric approach to security moves the focus from securing the data container/channel/perimeter to protecting the data itself, which is much more robust.***

More specifically, inside a data packet, the signature field includes the signature type, key locator, and other information that can be used to verify the data's authenticity and integrity (Figure 1). The *key locator* identifies the data signer's public key, which can be either the name of the key (or more accurately its certificate) or the digest of the key. Note that because a public key is also a piece of NDN data containing a signature, *the data effectively becomes a certificate of the key* if the signature in the data is generated by the

right signer based on the application's trust model and follows a standard certificate format.

To verify the data authenticity, an application-specific ***trust model*** must be established. This is an important step in each application's development process. Different applications may have very different trust models, which depend on the relationships among the entities that participate in the applications and who/what the entities ultimately trust. One example is the five-level trust hierarchy adopted by the Named-data Link State Routing (NLSR) protocol ([12, 31]), which reflects the administrative structure for routing within a network domain.

Based on the trust model for a specific application, a *trust schema* containing a set of *trust rules* can be developed for the application to automatically infer the correct signing key for each received data (or key) [36]. For example, in the NLSR trust model, the trust rule for routing data specifies that its signing key name should contain the originating routing process' name, which should be part of the data name. Every trust rule can be expressed using regular expressions. When a data packet is received, a verification process uses the trust schema to check the following: (1) the key locator field contains a key that is expected by the schema; (2) the key can be retrieved using information contained in the key locator; and (3) the signature matches the data name and content based on the retrieved public key. The verification process repeats these steps for each retrieved key until it reaches the trust anchor whose self-signed public key is pre-configured in the application. If any of these step fails, the data fails the verification and it is discarded.

In summary, NDN provides a solid foundation for addressing *two fundamental issues in a vehicular network*:

First, **the focus on data, rather than the location(s) of the data, along with caching, makes it easy to share data among vehicles with high mobility and dynamic connectivity**. A vehicle can retrieve the desired data from any other vehicle that has a copy of the data, not just the original producer of the data. Essentially, the vehicles become "data mules" which transport data opportunistically. In contrast, the current IP Internet was designed to solve point-to-point communication problems such as accessing a remote server. When nodes change their location, the previous point-to-point communication breaks and they need to rediscover their addresses and reestablish their communication (or use some redirection mechanism). These mechanisms are not an inherent part of the IP architecture and therefore are cumbersome to design, implement, and deploy.

Second, **in a highly mobile environment, data-centric security is vital, as there is no stable session between any two nodes that can be used as a basis for today's channel-based security**. NDN enables every consumer to authenticate received data using the information in the data and an application-specific trust model, regardless of where the data is from and how the data is retrieved.

3 THREATS

Our work focuses on two potential attacks in NDN-based vehicular networks. The first attack is *dissemination of false information* to mislead other vehicles. For example, a vehicle may give false information about the traffic of a road to other vehicles so that they will either avoid that road or flood to it. In this case, the consequence can be extended travel time for the vehicles that are misled by the information. In other cases, the false information may cause another vehicle to make a wrong decision that leads to an accident. Regardless of the consequence, a vehicle should not blindly accept all the data from other vehicles. Note that this attack is not unique to an NDN-based vehicular network, but we believe that it can be addressed more effectively in the NDN architecture.

Let us examine the possible sources of the false information: (1) a malicious outsider not in the same vehicular network; (2) a misbehaving insider; (3) an unintentional insider or outsider (e.g., the data may have been corrupted in that node's cache). The third case can be easily detected by the signature in the data, as it protects the integrity of the data. In the first and second case, the attacker may put the data name and key name associated with another vehicle in the data to avoid being identified. However, as long as the attacker does not have the impersonated vehicle's private key, he/she cannot generate the correct signature and the authentication mechanism should be able to use the signature to detect data that is not from the claimed vehicle and discard it. In addition, the authentication mechanism can reject all data from outsiders as it is very difficult to ascertain the trustworthiness of the data. The remaining data should have been generated by an insider that used its own key to sign the data. Although the data can still be false, there is a higher risk to produce such false data since the node has a higher chance of being captured. In fact, there are proposed schemes to detect such misbehaving nodes and penalize them by revoking certificates or lowering the trust value of the data originated from them [25]. All of the above requires a good naming scheme and trust model to associate each piece of data with the right producer and detect data from outsiders, which is addressed by our work.

The second attack is *vehicle tracking* using data shared by vehicles which often contains location and time information. For example, accident data contains the location and time of an accident and implies that the vehicle is near the accident area at the specified time. Traffic congestion data also typically contains a road segment's name and time, and suggests that the vehicle was traveling on that road at that particular time. If the shared data's name and signing key contains information that identifies the vehicle originating the data, anyone observing the data can track the routes of the vehicle and use this information for a range of purposes. There may be legitimate reasons that warrant the association of location and vehicle identity, e.g., for law enforcement purposes, but there should be sufficient protection in the system to prevent a bystander or an outside attacker from doing so.

Note that data from vehicles may contain other sensitive information such as images, video, and audio of people traveling in the vehicle, but privacy issues in such cases are beyond the scope of this paper.

4 DESIGN

In this section, we first describe the type of vehicular applications considered in our design. We then present a naming scheme and trust model for authenticating information from vehicles in a network. We further develop the bootstrap procedure for the model

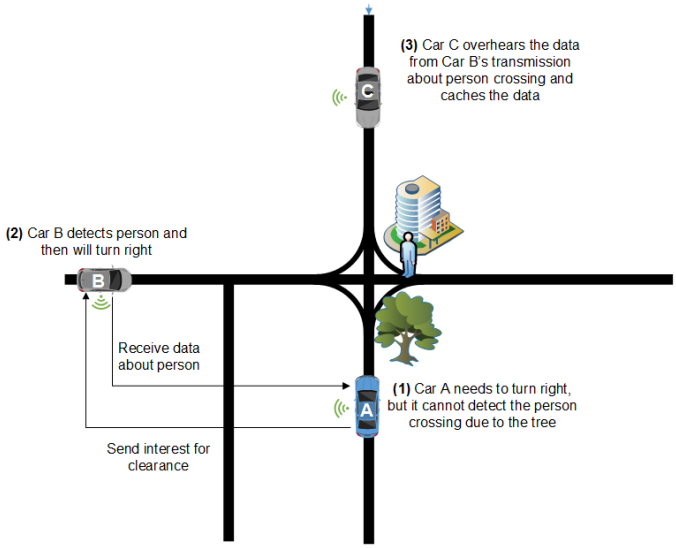

Figure 3: A Scenario of Data Sharing among Vehicles

and propose a preliminary framework for protecting the privacy of the vehicles (and their owners) when the vehicles share data that may disclose their location information.

4.1 Autonomous Vehicular Applications

We envision that each vehicle has a set of sensors that capture information about the vehicle and its environment. Moreover, it has autonomous driving software that makes decisions on its direction, speed and other parameters that affect the vehicle's behavior, using its own sensor data, other vehicles' data, maps and other data sources. The vehicle sends NDN interests via V2V and V2I communication to retrieve data that may potentially affect its decisions, such as detected pedestrians and cyclists in the surrounding area, traffic level of the next road segment it will travel on, and accidents on its path to the destination.[1] Any vehicle that has relevant information to share may respond to the interests. To avoid an implosion of data responses, the vehicles receiving an interest may use a random back-off scheme before sending their data – it will suppress its own reply if it hears another reply during the back-off period. Finally, the data may be cached by any vehicle that overhears it and the cached data can satisfy any interest it matches as the vehicle moves around. A simple example of such a scenario is illustrated in Figure 3.

4.2 Naming Scheme

Naming is one of the most important aspects in designing an NDN application [37]. It is closely related to other parts of the application design, e.g., the trust model for authenticating vehicular data, and it also affects how efficiently the packets from the application can be routed and forwarded. Based on the expected participants and data in the vehicular applications, we designed the following naming scheme for them.

[1]Interested readers can refer to work by Grassi et. al. ([8, 9]) to learn more about interest/data forwarding issues in NDN-based vehicular networks.

Our data name follows this format: */<application-prefix>/<data-type>/<data-location>/<name-marker>/<vehicle-name>/<timestamp>*. Similar to the naming scheme proposed by Wang et. al. in [32], our name format contains geographical scoping and temporal scoping information. However, a major difference from [32] is that our scheme contains the *name (Section 4.3) or pseudo-name (Section 4.4) of the vehicle that produced the data* so that we can associate the data with the origin vehicle and authenticate the data. In addition, because we have vehicle name and timestamp in the data name, we do not need a nonce field to make the names unique as in [32]. Below we explain each name component in more detail.

The first component is the *name prefix of the application*, such as a real-time traffic application or an accident prevention application, which is producing the data.

The *data type* component refers to the kind of data that is produced, e.g., status of the road, sudden speed reduction of other vehicles, or arrival of emergency vehicles. Different types of data require vehicles to process them differently and take different actions, so it is important to differentiate between them.

Since vehicular data is mostly location dependent, we use a name component *data location* to specify the geolocation of the data. Depending on the application, the location name may have different formats – one option is a hierarchical structure that allows vehicles to get information of a road in different granularity, e.g., */usa/tn/memphis/road-name/section1-section2* and another option is a pair of longitudinal and latitudinal coordinates.

The next component *name marker* indicates that the following name component is the vehicle's name. We need this marker because the location may contain multiple name components so it is difficult to know where the vehicle name starts. In our implementation, we use *V1.Vehicle* as the marker.

The *vehicle name* should reflect a vehicle's relationships with other entities in the system and allow derivation of trust based on its relationships. In our design, it has a simple form */<manufacturer-name>/<vehicle-name>*, where the first component identifies the vehicle's manufacturer and second component identifies the vehicle. These components are necessary for data authentication as we will explain in Section 4.3. Note that it is possible to track the vehicle when the vehicle is associated with a single name (Section 3). To address this problem, we propose to use pseudo-names to make vehicle tracking difficult (Section 4.4).

The next component is *timestamp*, which can either be a single value, i.e., a particular point in time, or a range.

Our AutoNDN prototype system follows the above naming scheme, and an example data name is /autondn/road-status/usa/tn/memphis/I-240/20-21/V1.Vehicle/⟨vehicle-name⟩/20170128105534, which indicates that the data is about the status of the interstate I240 between exit 20 and exit 21 at 10:55:34 on Jan. 28, 2017 (see Section 5 for more information about our prototype).

We note that our data naming scheme may not fit every vehicular application. Some applications may need additional name components, while others may prefer different ordering of the name components. Some applications may not need all the name components. If a name component is not expected to be useful in interests (as a query parameter), yet it still provides useful information to consumers, it may be moved into the data content.

Table 1: Trust Hierarchy, Key Names and Data Names in Autonomous Vehicular Applications

Level	Entity	Key/Data Name	Example
1	Auto-Vehicle Organization	/<av-org>/KEY/<key-id>	/auvsi/KEY/<key-id>
2	Manufacturer	/<manufacturer-name>/KEY/<key-id>	/honda/KEY/<key-id>
3	Vehicle	/<manufacturer-name>/<vehicle-name>/KEY/<key-id>	/honda/574G98627Y/KEY/<key-id>
4	Data	/<application-prefix>/<data-type>/<data-location>/<name-marker>/<vehicle-name>/<timestamp>	/autondn/road-status/usa/tn/memphis/I-240/20-21//honda/574G98627Y/20170128105534

4.3 Trust Model

As explained in Section 3, in order to reduce the likelihood of accepting false information, a vehicle should authenticate all received data. Since every data packet in NDN is signed by the producer, it can be verified using the data's signature and a pre-defined trust model for the data (Section 2). Every entity in NDN has its corresponding key(s) that can be used to sign certain data and/or keys. The trust model defines the trust anchor(s) and the chain of trust from the anchor(s) to the data, i.e., the key/data signing privileges of each entity in the chain.

To ensure data authenticity in vehicular networks, we propose a four-level trust model including autonomous-vehicle organizations, manufacturers, vehicles and data (Table 1). First, the trust anchors can be highly reputable organizations that are interested in the deployment of autonomous vehicle technology. They are responsible for certifying the keys belonging to vehicle manufacturers by signing those keys. There can be *multiple* trust anchors and a manufacturer can obtain a certificate from one or more of them (each vehicle can be configured with multiple trust anchors' public keys). Second, every manufacturer certifies the keys of its vehicles. Third, every vehicle signs its own data.

The certification of a key involves checking the credentials of the entity being certified. For example, a manufacturer needs to present valid documents proving that it is indeed the claimed manufacturer. A vehicle can present a unique secret vehicle identifier (VID) that the manufacturer installed in the vehicle in the factory (see Section 4.4).

Based on the above trust model, a vehicle's data can be verified by the vehicle's key, which can be verified by the manufacturer's key, and the manufacturer's key can be verified by a trust anchor's key, which is self-signed and pre-configured in the application. Figure 4 shows an example of the signing and verification process, with the organization AUVSI (Association for Unmanned Vehicle Systems International) being the trust anchor.

The trust schema based on our trust model should include the following trust rules: (1) the vehicle key that signs a piece of vehicular data should have the <*vehicle-name*> component of the data name as its prefix; (2) the manufacturer's key signing a vehicle's key should match in their first component; (3) the key that signs a manufacturer's key should match one of the trust anchors configured locally. Section 5.5 presents an implementation of this schema for our AutoNDN application.

We can extend our trust model to accommodate other entities in the system. For example, due to privacy considerations, we will introduce Certificate Issuing Proxies (CIP) to help vehicles obtain

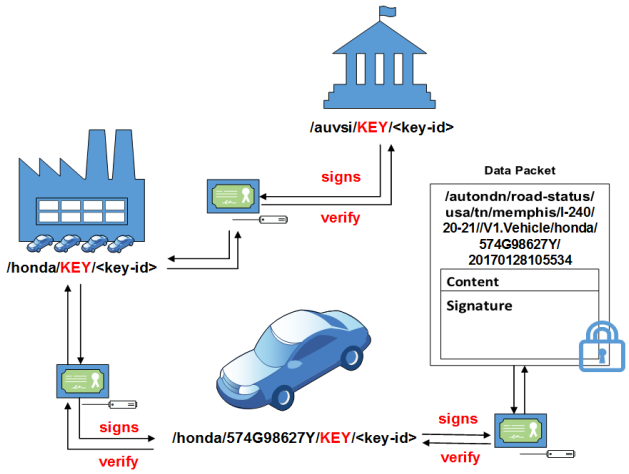

Figure 4: Example Signing and Verification Hierarchy (Real names such as Honda are used here for illustration purposes only. Due to privacy considerations, we will use pseudonyms to replace the real names as described in Section 4.4.)

manufacturers' pseudonyms and the certificate of their own keys from the manufacturers. In order for a vehicle to trust a proxy, the proxy's key must be certified by the vehicle's manufacturer. However, the proxy does not have the authority to certify vehicles' keys, because it cannot verify the identifies of vehicles.

4.4 Vehicle Tracking Prevention

Our trust model requires a vehicle to put its name into its data's name and key name. This gives rise to privacy concerns as the vehicle can be easily tracked if it keeps using the same name to publish its data. To mitigate the threat of vehicle tracking, rather than changing the trust model, we propose to *let each vehicle use a set of pseudonyms so that it can publish consecutive data items using different pseudonyms and the associated keys.* Each vehicle's complete pseudonym *n* has two components /<MP>/<VIP>, where <MP> is the manufacturer's pseudonym and <VIP> is the vehicle's individual pseudonym. MPs are generated by the manufacturers and provided to the vehicles. VIPs are generated by vehicles whenever they need new pseudonyms. We use pseudonyms for manufacturers to prevent leaking information about vehicles. Otherwise, if a

vehicle travels along a road segment with few other vehicles, it will produce a series of data items containing the same manufacturer name, the same road segment and similar timestamps. Such information can be used to infer that the data items belong to the same vehicle.

Since a vehicle cannot have an unlimited number of pseudonyms, it will have to reuse a pseudonym after time T, which is determined by the number of pseudonyms N and the average data-publishing rate R, i.e., $T = N/R$. The longer the T, the more difficult it is to track the vehicle. This means that, for a given date-publishing rate, a larger N is more desirable. We envision that every vehicle will have a large number of pseudonyms and certificates initially installed by its manufacturer. However, as the keys expire over time, every vehicle needs to generate new pseudonyms and the associated keys in order to maintain a sufficient number of pseudonyms. To bootstrap this process, each vehicle is given a secret vehicle identifier (VID) by its manufacturer. The VID is used as an input to generate pseudonyms so as to prevent pseudonym collision among vehicles. It is also used by vehicles to prove their identity when they request for certificates.

The certificate issuing process can potentially expose part of a vehicle's identity information. More specifically, if a vehicle requests for a new certificate by sending an interest containing the manufacturer's name prefix, an attacker who captures packet traces in the vehicular network can infer that a vehicle from that manufacturer was near a certain location at a particular time. Hence we need to provide a *certificate issuing mechanism that is resistant to vehicle tracking*. To this end, we introduce a new entity to the system called ***Certificate Issuing Proxy (CIP)*** which requests certificates on a vehicle's behalf. A CIP may be located at a gas station, car dealer, road-side unit, or even home (Figure 6). The idea is similar to periodically filling up gas in a vehicle, except that we fill up certificates in the vehicle in this case. To request for a new certificate via a CIP, the vehicle sends an interest using the CIP's name prefix, and the CIP relays the interest to the manufacturer using a network that is only accessible by authorized personnel at the facility (not an open-access wireless link) so that it is difficult for an attacker to capture a packet trace directly from the CIP.

Each CIP needs to obtain its own certificates from manufacturers, so that vehicles and manufacturers can trust it. Before a vehicle uses a CIP's service, it requests for all the CIP's certificates and verifies that at least one of the certificates is from its manufacturer. Some CIPs may have very limited capabilities. For example, a home CIP may be only authorized to relay requests for specific vehicles owned by the homeowners. These rights may be specified using certain name components in the CIP's key.

Note that when a vehicle sends an interest for a new certificate to a CIP, it needs to include the actual manufacturer's name, encrypted using the CIP's key, as one of the name components. This is for the CIP to route the interest to the correct manufacturer. Below we explain the scheme for certificate issuance as shown in Figure 6:

(1) The vehicle first retrieves the CIP's public key K_1 and associated certificates. It then validates that one of the certificates is from its manufacturer.

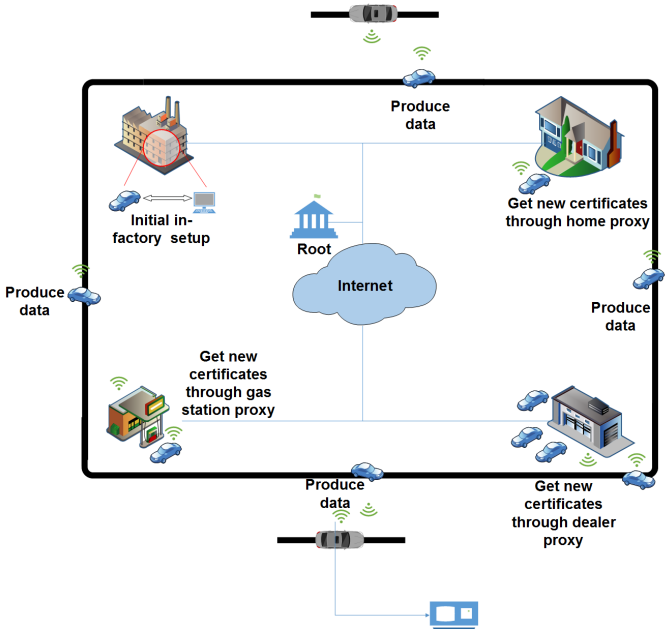

Figure 5: Obtaining Certificates from Manufacturer and Proxies

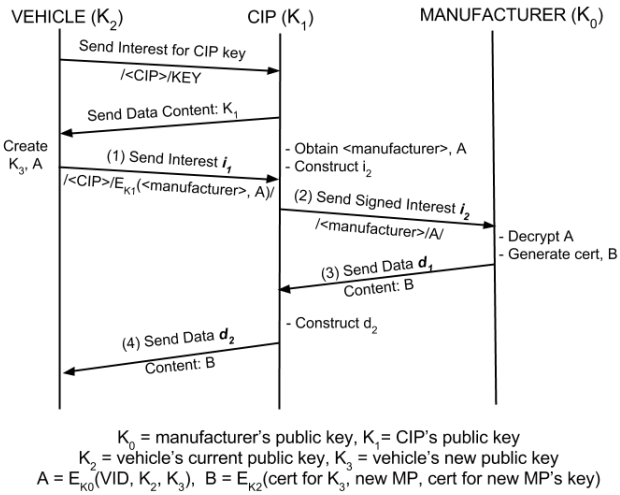

Figure 6: Pseudonym Certificate Issuance through Proxy

(2) The vehicle generates a random pseudonym based on its VID, chooses a manufacturer's pseudonym from those received from the manufacturer, and creates a new key K_3. It encrypts its VID, one of the vehicle's current public keys K_2, and the new public key K_3 using the manufacturer's public key K_0. It then encrypts this information along with the manufacturer's name using the CIP's key K_1. Next, it sends an interest (i_1) to the CIP with the name $/<CIP>/E_{K_1}(<manufacturer>, E_{K_0}(VID, K_2, K_3))$.

(3) CIP receives i_1, decrypts the component that follows <*CIP*> and determines the manufacturer's name. CIP cannot decrypt the components after manufacturer's name as they are encrypted using K_0. This ensures that CIP cannot obtain the VID and other sensitive information. Next, CIP constructs and sends a new interest (i_2), signed with K_1, with the name /<*manufacturer*>/E_{K_0}(*VID*, K_2, K_3).
(4) After receiving i_2, the manufacturer checks that the interest is signed by one of the authorized proxies and then extracts the VID, K_2, and K_3. The manufacturer verifies the VID and creates a certificate for the new key K_3. It then encrypts this certificate, a future manufacturer's pseudonym, and the manufacturer's pseudonym certificate, using K_2, to prevent the CIP from obtaining the information.[2] This information is included as content in the data packet d_1.
(5) CIP receives the data packet d_1, constructs d_2 by changing its name to match i_1, and forwards it to the vehicle, which will be able to decrypt the data and store the certificate and other information. The manufacturer's future pseudonym and certificate will be used by the vehicle the next time it creates a pseudonym.

If we use a unique manufacturer's pseudonym for every vehicle's pseudonym, then there will be too many such pseudonyms and keys for the manufacturer to generate and get certified. This is not a scalable solution. Instead, the manufacturer can give the same manufacturer pseudonym and key to different vehicles in different geographic areas, e.g., one pseudonym may be used by 50 vehicles each in a different country. This kind of reuse will lower the risk of vehicle tracking through the manufacturer's pseudonym.

5 SYSTEM PROTOTYPE

Since there are no autonomous vehicles available for our experimentation, we decided to used mini cars containing a Raspberry Pi that provides basic communication and processing capabilities to prototype our system. Our AutoNDN application is composed of four modules: *communication*, *sensor*, *control*, and *calibration*. The interaction among the modules is shown in Figure 7. The Communication module is responsible for sending and receiving all the interests and data packets by interacting with the NDN forwarder (NFD). It also validates all the received data. The Sensor module obtains data using on-board sensors. The Control module interfaces with the Sensor module and the Communication module to know the status of the current road and the next road to make decision accordingly. It sends decisions to the Calibration module that handles the low-level movement and direction of the car. The application is written in C++ using the ndn-cxx [30] library. It runs on the NDN platform [29], which was cross-compiled on a server and transferred to the RPi for deployment. We implemented the basic trust model and validation rules, but not the scheme for preventing vehicle tracking.

[2]The vehicle can publish the manufacturer's pseudonym certificate when it publishes new data with a name containing that manufacturer's pseudonym to speed up validation of the data by other vehicles.

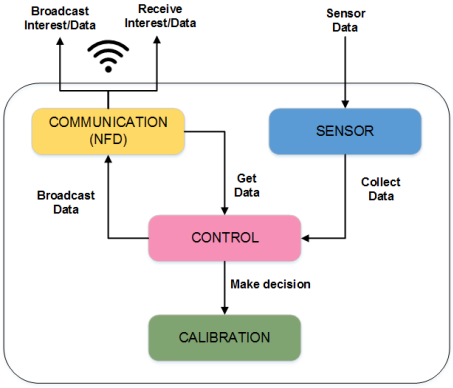

Figure 7: System Overview

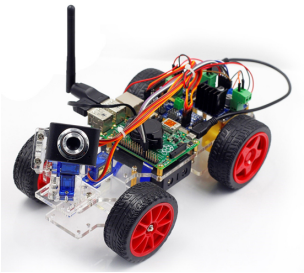

Figure 8: Raspberry Pi Car [28]

5.1 Hardware

We used Sunfounder Raspberry Pi car kits (Figure 8) to develop our prototype system. The car has a Raspberry Pi that controls its motion, communication and sensing. The motion of the car is powered by the two rear wheels using a 5V DC motor each. The front wheels are used to turn the vehicle in a desired direction. For wireless communication, it uses a Wi-Fi dongle with an antenna that is capable of ad-hoc communication. The car comes with a camera, but it is currently not activated for the prototype.

5.2 Control Module

When the AutoNDN program is run, the control module loads the map of roads from a file that also contains the starting and ending coordinates for the current trip. Dijkstra's algorithm is used to decide the shortest path that the vehicle would travel on. The vehicle publishes data about the road it is currently on and retrieves data for the next road. If it receives data indicating that the next path is not good for travel then Dijkstra's algorithm is run again between the end of this road and the end point, and the path is altered. Figure 10 shows the algorithm for the prototype.

5.3 Calibration Module

The Calibration module controls the car's direction and hind-motors for movement. It uses the Wiring Pi library [33] to access the RPi's GPIO pins and I2C interface. It communicates with the L298 driver to control the direction of the motor wheels (forward or backward). To control the speed of the car, this module generates Pulse Width

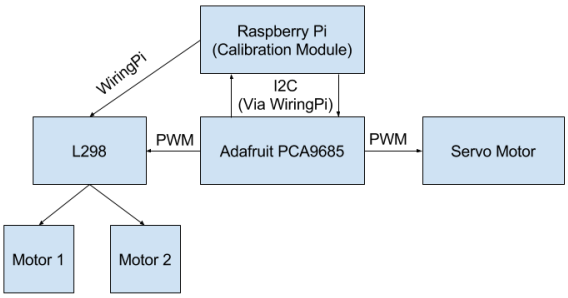

Figure 9: Interactions between Calibration Module and Drivers

Modulation (PWM) using Adafruit's PCA9685 PWM/Servo motor driver [1] for the L298 driver. PWM is also provided to the servo motor to control the direction of front wheels and the camera. The overview of this interaction can be seen in Figure 9. This module communicates with the Adafruit driver via I2C for which Adafruit provides a Python library. The library was ported to C++ for easier integration with the rest of the code. We could have used PyNDN2 [20] library, which is a Python library for NDN. Since we would like to conduct large scale simulations in the future, we decided against it as the NDN simulation environment, ndnSIM [3], is written in C++. Another consideration for using C++ was that the camera module processing might need the speed and efficiency provided by the compiled C++ code considering the lower processing capabilities of the RPi.

5.4 Sensor Module

Since the focus in this paper is on security, we are treating the Sensor module as a black box. To simulate the data from the sensor module, the control module generates the status of the current road arbitrarily. We plan to use the camera on the car to acquire images and video of the road, process the data in sensor module, and then send it to the communication module for sharing with other cars as well as to the control module for decision making.

5.5 Communication Module

The communication module handles the transmission of interests and reception of data. It is important to note that no IP setup needs to be done as the data is being broadcast over the WiFi ad-hoc network (each car needs to join the same ad-hoc network to communicate). Before the communication module passes data to the control module, it verifies the data using the trust schema loaded in its validator, which is extended from the ndn-cxx validator.

Listing 1 is an example of the trust schema configured in our AutoNDN application. The order in which rules appear are important, as the validator applies rules to a data packet one by one. A rule can be applied to a data packet, if its name matches the "filter" property specified in the rule. The security section of the configuration file contains three rules: Road Status Rule, Hierarchical Rule, and Hierarchical Exception Rule. The Road Status Rule will be applied to the data packet whose name starts with /autondn/road-status, since the program is publishing the traffic related data under this name prefix. The "key-locator" sub-field specifies the relationship

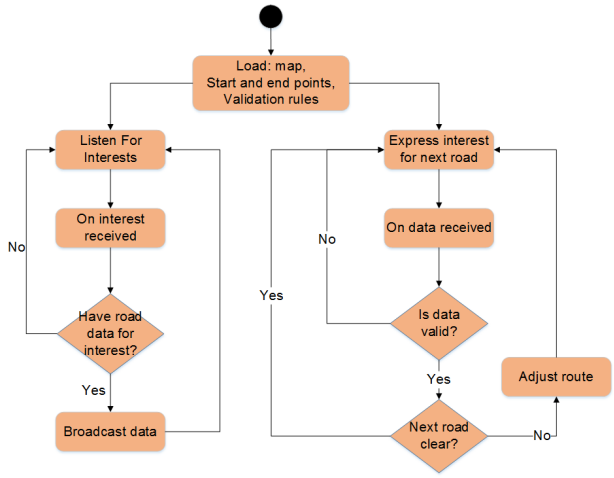

Figure 10: Publishing/Retrieving Data and Adjusting Route

between a data name and the expected signing key name. If the actual signing key does not satisfy this rule, then the data will be dropped. Otherwise, the key will be retrieved to validate the data. The other two rules, Hierarchical and Hierarchical Exception, are for validating the vehicle's and the manufacturer's key, respectively. The name of the manufacturer is a prefix of a vehicle's name, so the relationship between the names of their keys is of type hierarchical (as defined in the ndn-cxx library). On the other hand, the name of the trust anchor's key does not have any relationship with the name of the manufacturer's key, so a Hierarchical Exception rule is used. Every vehicle has the certificate of the trust anchor pre-installed. It publishes not only its own certificate but also the manufacturer's certificate so that other vehicles can retrieve all the necessary certificates for data validation directly from this vehicle without having to retrieve any certificate all the way from the manufacturer.

6 EVALUATION

Our experiment setup consists of seven nodes: one root organization, three manufacturers, and three mini cars. The root organization issues certificate for the manufacturers, which in turn issue certificates for their respective manufactured cars. Besides serving its own certificate, each car also publishes the certificate of its manufacturer to speed up data validation by other cars. The certificate of the trust anchor is configured in each car. The cars are equipped with Raspberry Pi of three different models, namely Raspberry Pi Model B (RPi-B), Raspberry Pi-2 Model B (RPi-2B) and Raspberry Pi-3 Model B(RPi-3B). Table 2 shows the specification of the RPi's. The CPU power increases significantly from RPi-B to RPi-2B, but the difference between RPi-2B and RPi-3B is relatively small. In addition, the two later models have twice the RAM as the first model. Note that we installed the same OS and kernel version on all three RPi's.

We are interested in the total delay between the time when a car sends an interest and the time when it validates the received data. This delay depends on whether the certificates needed to validate the data are locally available or not. If the certificates are

Listing 1: Schema for Data Validation

```
security
{
  validator
  {
    rule
    {
      id "Road Status Rule"
      for data
      filter
      {
        type name
        regex ^<autondn><road-status><>+
      }
      checker
      {
        type customized
        sig-type rsa-sha256
        key-locator
        {
          type name
          hyper-relation
          { ; <make-id><vehicle-id><KEY><ksk-123><ID-CERT>
            k-regex ^([^<KEY>]*)<KEY><ksk-.*><ID-CERT>$
            k-expand \\1 ; extract vehicle's name
            h-relation equal
            ; /autondn/road-status/<data-location>/<time-stamp>/V1
                 .Vehicle/<make-id><vehicle-id>
            p-regex ^<autondn><road-status><>+<V1.Vehicle>(<><>)$
            p-expand \\1 ; extract vehicle name
          }
        }
      }
    }
    rule
    {
      id "Hierarchical Rule"
      for data
      filter
      {
        type name
        regex <><><KEY><ksk-.*><ID-CERT><>$
      }
      checker
      {
        type hierarchical
        sig-type rsa-sha256
      }
    }

  rule
  {
     id "Hierarchical Exception Rule"
     for data
     filter
     {
      type name
      regex <><KEY><ksk-.*><ID-CERT><>$
     }
     checker
     {
      type fixed-signer
      sig-type rsa-sha256
      signer
      {
        type file
        file-name "autondn-root.cert"
      }
     }
    }
  trust-anchor
  {
   type file
   file-name "autondn-root.cert"
  }
}
; validator

  cert-to-publish "honda.cert"
  cert-to-publish "honda-vehicle.cert"
}
```

Table 2: Specification of On-board RPi's

SoC	RPi-B	RPi-2B	RPi-3B
Core Type	ARM1176JZF-S	Cortex-A7	Cortex-A53 64-bit
No. Of Cores	1	4	4
CPU Clock	700 MHz	900 MHz	1.2 GHz
RAM	512 MB	1 GB	1 GB

not available then the cars need to fetch two certificates, the data producer's and the manufacturer's. Thus, we partitioned the total delay into various delay components to examine their contribution. Consumer-side delay includes data fetching delay, certificate fetching delay, and validation delay. Producer-side delay consists of signing delay and certificate sending delay.

In each of our experiments, two cars run, one as a consumer and another as a producer, ten times, and the averages of the corresponding delays are taken. The producer car travels on the experiment track first and generates data for the roads, and the consumer car follows the producer and gets data from it. The number of interest-data exchange in each run is the same as the number of roads in the track (ten). After the first time a car (car_1) consumes data from another car (car_2), the former has all the certificates needed to authenticate data from the latter. Consequently in the subsequent data consumption from car_2, car_1 does not have to request for the necessary certificates, i.e., there is no certificate fetching delay. Therefore, we only considered the first interest-data exchange in each run when calculating the certificate fetching delay. To erase the memory of a car about interactions with another car, NFD is restarted in both cars before each run.

Figure 11 shows our experiment results. The data fetching delay of the consumer includes the delay for the consumer's interest to reach the producer, the data signing delay of the producer (RSA is used for data signing), and the delay for the data to reach the consumer. The lower the processing power of the producer, the higher the data signing delay, and the higher the fetching delay from the producer. Due to RPi1's much lower CPU power. its data signing delay is more than twice that of the other two RPi's, so retrieving data from RPi1 is much slower. The certificate sending delay of the producer mainly includes the time to calculate a SHA256 digest of a certificate. Because the digest calculation is much faster than RSA signing, the certificate sending delay is much smaller than the data signing delay. Data validation delay is how much time it takes for a car to validate data after retrieving certificates. As such, it depends on the processing power of the car so the RPi1 takes twice

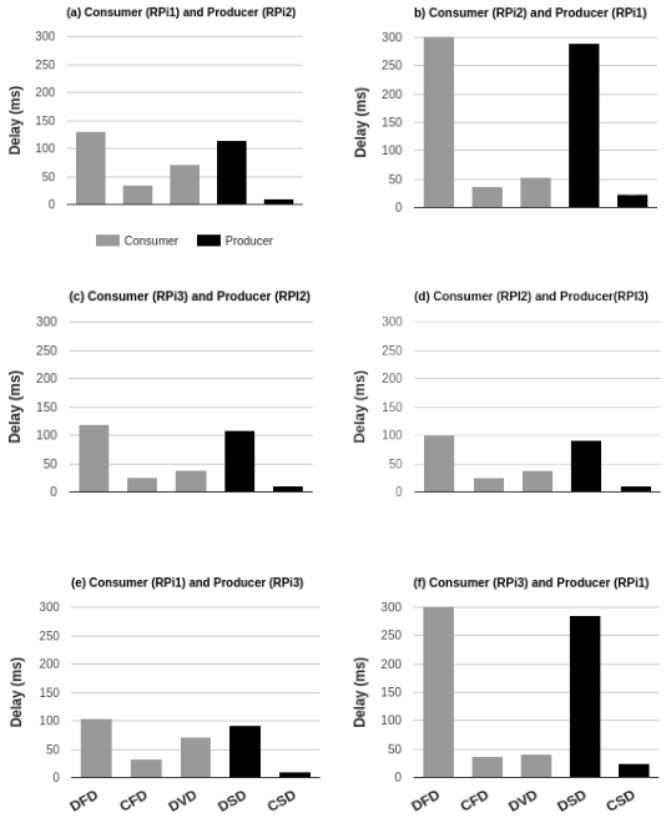

Figure 11: Consumer and Producer Delays in Different Runs (Consumer Delay: Data Fetching Delay (DFD), Certificate Fetching Delay (CFD), Data Validation Delay (DVD); Producer Delay: Data Signing Delay (DSD), Certificate Sending Delay (CSD))

as much time to validate the data compared to the other two RPi models. However, because the RSA algorithm has faster validation than signing, the data validation delay is significantly smaller than the data signing delay.

The good performance demonstrated by RPi2 and RPi3, as shown in Figure 11(c) and (d), is encouraging. Since real vehicles can have even more powerful CPUs and higher storage capacity, the delays caused by these factors will be even smaller for real vehicles.

7 RELATED WORK

NDN's location-independent data retrieval approach makes it a perfect fit for ad-hoc network environments. Meisel et. al. [17] analyzed the benefits of NDN-based forwarding in an ad-hoc network over IP-based solutions. Nevertheless, the ever-changing topology of Vehicular Ad-Hoc network (VANET) still makes the routing or forwarding of interest packets in such an environment challenging. Consequently most of the NDN-based VANET efforts are focused on improving the forwarding decisions of interest/data packets [35]. VANET via NDN (VNDN) [8] used a simple greedy forwarding strategy to spread NDN Interest packets and introduced a new layer called Link Adaptation Layer (LAL) to take advantage of the underlying wireless media efficiently. The later work [9] by the same authors replaces interest broadcasting with geolocation based forwarding, where interests are forwarded toward the locations where data is more likely to be found.

VANET security in the NDN context is a less researched area [26], but there are schemes ([13, 21, 27]) proposed in the context of the current IP network for data authentication and privacy protection in VANETs. Raya et. al. [23] identified various vulnerabilities in VANET and the inherent challenges to protect them from being exploited. In addition to presenting an authentication scheme, the paper put forward the hardware prerequisites for securing vehicular communication. Petit et. al. [22] loosely categorized pseudonymous authentication schemes based on different cryptographic schemes: asymmetric cryptography, identity-based cryptography, group signature and symmetric cryptography. The pseudonymous authentication scheme proposed in this paper is of the asymmetric cryptography type. [22] also analyzed different phases of a pseudonym's lifecycle: pseudonym issuance, pseudonym use, pseudonym change, pseudonym resolution, and pseudonym revocation. Our scheme does not need to resolve and revoke pseudonyms thus decreasing the communication overhead. Moreover, since our scheme is based on NDN, we do not need to handle changing network identifiers, such as IP or MAC addresses.

A major security threat in VANET is the dissemination of false data [26]. Information shared by a vehicle can be incorrect, which can mislead other vehicles and potentially disrupt the whole network. Unlike previous work ([7, 25, 34]), we use name-based trust schemas to facilitate the detection of false information.

8 CONCLUSION

NDN's data-centric approach to security moves the focus from securing data container/channel/perimeter to protecting the data itself. Taking advantage of the built-in security features in NDN, we have developed a trust model and naming scheme to authenticate data produced by autonomous vehicles. Since names can be used to track vehicles, we also sketched out a pseudonym and proxy-based scheme to make it difficult to associate different data items with the same producer vehicle. We successfully implemented the trust model and validation scheme in our AutoNDN application. Our prototype system based on mini cars demonstrates that the design behind the authentication scheme is transferable to real cars with much more powerful processing and storage capabilities. Our next step is to flesh out the details in the vehicle-tracking prevention scheme and implement it in our prototype. Furthermore, we plan to evaluate our schemes at a larger scale in ndnSIM [19] and Mini-NDN [18].

REFERENCES

[1] Adafruit 16-Channel 12-bit PWM/Servo Driver - I2C interface - PCA9685. (????). https://www.adafruit.com/product/815
[2] 2012. IEEE Draft Guide for Wireless Access in Vehicular Environments (WAVE) - Architecture. *IEEE P1609.0/D5, September 2012* (Oct 2012), 1–74.
[3] Alexander Afanasyev, Ilya Moiseenko, and Lixia Zhang. 2012 (revised October 2012). *ndnSIM: NDN simulator for NS-3.* Technical Report NDN-0005. NDN Project.
[4] T. Dierks and E. Rescorla. 2008. *The Transport Layer Security (TLS) Protocol Version 1.2.* RFC 5246. RFC Editor. http://www.rfc-editor.org/rfc/rfc5246.txt http://www.rfc-editor.org/rfc/rfc5246.txt.

[5] Daniel J Fagnant and Kara Kockelman. 2014. Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations for capitalizing on self-driven vehicles. *Transportation Research Board* (2014).

[6] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis. *A Framework for IP Based Virtual Private Networks.* Technical Report.

[7] Philippe Golle, Dan Greene, and Jessica Staddon. 2004. Detecting and Correcting Malicious Data in VANETs. In *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*. ACM, 29–37.

[8] Giulio Grassi, Davide Pesavento, Giovanni Pau, Rama Vuyyuru, Ryuji Wakikawa, and Lixia Zhang. 2014. VANET via Named Data Networking. In *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*. IEEE, 410–415.

[9] Giulio Grassi, Davide Pesavento, Giovanni Pau, Lixia Zhang, and Serge Fdida. 2015. Navigo: Interest forwarding by geolocations in vehicular Named Data Networking. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*. IEEE, 1–10.

[10] Giulio Grassi, Davide Pesavento, Lucas Wang, Giovanni Pau, Rama Vuyyuru, Ryuji Wakikawa, and Lixia Zhang. 2013. Vehicular Inter-networking via Named Data (poster). In *ACM HotMobile*.

[11] IEEE 802.11 Working Group and others. 2010. IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std* 802, 11 (2010).

[12] AKM M. Hoque, S. O. Amin, A. Alyyan, B. Zhang, L. Zhang, and L. Wang. 2013. NLSR: Named-data Link State Routing Protocol. In *ACM SIGCOMM ICN Workshop*.

[13] Yih-Chun Hu and Kenneth P Laberteaux. 2006. Strong VANET security on a budget. In *Proceedings of Workshop on Embedded Security in Cars (ESCAR)*, Vol. 6. 1–9.

[14] John B Kenney. 2011. Dedicated Short-Range Communications (DSRC) standards in the United States. *Proc. IEEE* 99, 7 (2011), 1162–1182.

[15] S. Kent and K. Seo. 2005. *Security Architecture for the Internet Protocol.* RFC 4301. RFC Editor. http://www.rfc-editor.org/rfc/rfc4301.txt http://www.rfc-editor.org/rfc/rfc4301.txt.

[16] Hod Lipson and Melba Kurman. 2016. *Driverless: Intelligent Cars and the Road Ahead.* Mit Press.

[17] Michael Meisel, Vasileios Pappas, and Lixia Zhang. 2010. Ad Hoc Networking via Named Data. In *Proceedings of the fifth ACM international workshop on Mobility in the evolving internet architecture*. ACM, 3–8.

[18] NDN Project Team. 2017. Mini-NDN GitHub. (2017). https://github.com/named-data/mini-ndn.

[19] NDN Project Team. 2017. ndnSIM: NDN simulator. (2017). http://ndnsim.net.

[20] NDN Project Team. 2017. PyNDN2. (2017). https://github.com/named-data/pyndn2

[21] Adrian Perrig, Ran Canetti, J Doug Tygar, and Dawn Song. 2000. Efficient Authentication and Signing of Multicast Streams over Lossy Channels. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*. IEEE, 56–73.

[22] Jonathan Petit, Florian Schaub, Michael Feiri, and Frank Kargl. 2015. Pseudonym schemes in vehicular networks: A survey. *IEEE communications surveys & tutorials* 17, 1 (2015), 228–255.

[23] Maxim Raya, Panos Papadimitratos, and Jean-Pierre Hubaux. 2006. Securing Vehicular Communications. *IEEE Wireless Communications* 13, 5 (2006).

[24] E. Rosen and Y. Rekhter. 2006. *BGP/MPLS IP Virtual Private Networks (VPNs).* RFC 4364. RFC Editor.

[25] Sushmita Ruj, Marcos A Cavenaghi, Zhen Huang, Amiya Nayak, and Ivan Stojmenovic. 2011. On Data-Centric Misbehavior Detection in VANETs. In *Vehicular technology conference (VTC Fall), 2011 IEEE*. IEEE, 1–5.

[26] Salvatore Signorello, Maria R. Palattella, and Luigi A. Grieco. 2016. Security Challenges in Future NDN-Enabled VANETs. In *Proceedings of The 3rd International Workshop on the Emerging Future Internet and Network Security (EFINS 2016).*

[27] Ahren Studer, Fan Bai, Bhargav Bellur, and Adrian Perrig. 2009. Flexible, extensible, and efficient VANET authentication. *Journal of Communications and Networks* 11, 6 (2009), 574–588.

[28] SunFounder 2017. Smart Video Car Kit. http://www.amazon.com/gp/product/B014KK89BW?psc=1&redirect=true&ref_=ox_sc_act_title_1&smid=ADHH624DX2Q66/. (2017).

[29] NDN Project Team. 2017. Library/NDN Platform. http://named-data.net/codebase/platform/. (2017).

[30] NDN Project Team. 2017. ndn-cxx Library. http://named-data.net/doc/ndn-cxx/current/. (2017).

[31] Vince Lehman, A K M Mahmudul Hoque, Yingdi Yu, Lan Wang, Beichuan Zhang, and Lixia Zhang. 2016. *A Secure Link State Routing Protocol for NDN.* Technical Report. University of Memphis, UCLA.

[32] L. Wang, R. Wakikawa, R. Kuntz, R. Vuyyuru, and Lixia Zhang. 2012. Data naming in Vehicle-to-Vehicle communications. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*. 328–333.

[33] WiringPi 2017. GPIO Interface library for the Raspberry Pi. http://wiringpi.com/. (2017).

[34] Gongjun Yan. 2010. *Providing location security in vehicular Ad Hoc networks.* Ph.D. Dissertation. Old Dominion University.

[35] Muhammad Azfar Yaqub, Syed Hassan Ahmed, Safdar H Bouk, and Dongkyun Kim. 2016. Interest Forwarding in Vehicular Information Centric Networks: A Survey. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. ACM, 724–729.

[36] Yingdi Yu, Alexander Afanasyev, David Clark, Van Jacobson, Lixia Zhang, and others. 2015. Schematizing Trust in Named Data Networking. In *Proceedings of the 2nd International Conference on Information-Centric Networking*. ACM, 177–186.

[37] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang. 2014. Named Data Networking. *ACM SIGCOMM Computer Communication Review (CCR)* 44, 3 (Jul 2014), 66–73.

[38] Yu Zhang, Alexander Afanasyev, Jeff Burke, and Lixia Zhang. 2016. A Survey of Mobility Support in Named Data Networking. In *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on*.

[39] Zhenkai Zhu, Alexander Afanasyev, and Lixia Zhang. 2013. *A New Perspective on Mobility Support.* Technical Report NDN-0013. NDN. http://named-data.net/techreports.html