Thesis Prospectus

Efficient Hyper-Parameter Search for Hardware-Aware Optimization of Deep Learning Systems

Author:

Akshay Chandrashekaran
akshayc@cmu.edu

Thesis Committee:

Prof. Ian Lane (Chair)

Prof. John Shen

Prof. Ole Mengshoel

Dr. Adam Coates

Department of Electrical and Computer Engineering

Carnegie Mellon University

Moffett Field, CA 94035

Abstract

In recent years, machine learning algorithms have been using deep learning (DL) to achieve state-of-the-art results for various tasks. These models are also being used in real-world production systems for tasks like speech recognition, computer vision, machine translation, etc. Existing approaches focus on building really deep models with large features which are tuned to run on a cloud-based server platform. However, with smart-phones becoming more ubiquitous, and gaining more computational power, interest in the usage of machine learning to perform these tasks on the mobile and embedded platforms themselves. In the advent of new home automation technologies, issues of security and privacy are paramount. However, current state-of-the-art techniques involve sending the raw data to data centers over the internet. One solution for this is to move the inference process to the edge device itself. However, edge devices are more computationally constrained.

All machine learning algorithms are governed by a set of tunable hyper-parameters that affect the performance of the algorithms and the structure of models used within. These hyper-parameters are generally set by human experts based on their previous experience and intuition, and only slightly tweaked to improve performance. However, such manual optimization has been shown to be an inefficient search over the hyper-parameter space. Existing automated hyper-parameter optimization techniques suffer from the curse of dimensionality. When multiple hyper-parameters are present, it typically takes more number of iterations to converge to a good solution.

Moving the DL system inference to the edge node necessitates a search for hyper-parameter configurations that can obtain performance close to that obtained by cloud architectures, but still be computationally efficient on the edge device. This can be done by either setting specific constraints on some of the objectives, and optimizing the hyper-parameters in the face of these constraints, or by doing providing a Pareto front, from which an operating point can be decided upon. Both of these approaches require an efficient search of the hyper-parameter space.

In this prospectus, we propose a structured and efficient search hardware-aware approach to produce DL models with performance comparable to cloud, but running on edge devices. First, we propose a novel technique of hierarchical hyper-parameter optimization for multi-objective functions. We split the hyper-parameters based on where they come into effect, and optimize them in a hierarchical manner. We additionally propose an extension to extrapolation of learning curves within the training pipeline, where we also use

the uncertainty in the extrapolation of the learning curve to allow revisiting points which were discarded. We propose to combine the above two techniques to yield the required search approach and demonstrate it on two different tasks: speech recognition and image classification.

Contents

Chapter 1

Introduction

State-of-the-art deep learning systems for commercial products typically run on large servers in data-centers. These systems are governed by a set of hyper-parameters that affect them in terms of performance as well as computational complexity. Deep neural networks have become ubiquitous in the presence of large amounts of training data, inexpensive massively parallelizable hardware like graphics processing units (GPUs) and open source training platforms like TensorFlow, PaddlePaddle, Torch, Theano, etc.

Most of the focus is on training and using systems that are to be run in server platforms. Due to this, focus has almost exclusively been given to performance of these systems in terms of accuracy. Given large enough amounts of data, this has resulted in the construction of large models which can only be run in machines with high compute capacity. Usage of these models for tasks like image classification, machine translation, speech recognition, etc. require that the raw data be transferred over the internet to these server farms. This exposes them to network latency issues as well as privacy issues. With increasing focus on home automation, it is of special concern since there is a potential of sensitive conversations being recorded and sent over potentially unsecured channels and stored in servers over which the user has no control.

One solution for tackling these two issues is to perform these tasks on situated compute nodes like smartphones and other embedded devices. However, this comes with the drawbacks of having low compute and memory capacity. The models trained with the intent of being run on large servers will not be able to run on these embedded platforms.

Although the performance of ML algorithms depends on model capacity, in the case of deep neural networks, this relationship is not strictly true. It is possible to obtain a reasonably good performing model with far fewer parameters. This gives the possibility of finding DNN structures that can potentially run on embedded platforms with performance comparable to that on servers, but still be able to satisfy the memory and computational constraints of the embedded platform.

All machine learning algorithms are governed by a set of hyper-parameters, which influence the size of the models in the algorithm, and the performance of the algorithm. Typically, these hyper-parameters are set by a human expert who then tweaks them till they can get no further improvement. However, manual optimization suffers from human fatigue and human preconceptions. This is exacerbated more in the case of multi-objective optimization where it is difficult for humans to track the effects of optimization on multiple objectives simultaneously.

Automated hyper-parameter optimization has been shown to be effective in surpassing human performance for both single objective and multi-objective scenarios. However, in the presence of increasing number of hyper-parameters, these automated techniques suffer the curse of dimensionality, taking longer to converge to a good solution or set of solutions.

Experts in deep learning typically use their intuition and expertise to predict whether a given system configuration is going to yield a promising result. They do so by looking at the performance of the system in it's early stages. If not, they terminate that build and look for alternate configurations. Recently, work has been done to use models to predict the performance of a system, and induce early termination if the model predicts that the system is not going to outperform the best available model so far.

Given the above individual components, it should be possible to design a system that can train towards multiple objectives, both towards performance and computational efficiency. In this thesis, we propose such a technique that can perform an efficient search of the hyper-parameter surface, incorporating the hierarchy of where hyper-parameters come into effect, and robust extrapolation of the training curves to reduce the wall time for hyper-parameter search.

Chapter 2

Related Work

In this chapter, we investigate three related work that try to solve the identification and authentication problems that span the spectrum of varying constraints on physical boundary.

As the most constrained scenario among the three related work, we present Message-In-a-Bottle (MIB) [19], which leverages a Faraday cage, a special hardware device that ensures authenticity and secrecy of the communication inside the cage. The cage prevents any wireless signals communicating between the sensor nodes inside to be leaked to attackers outside the cage. Also, attackers are infeasible to inject wireless signals into the cage. The main drawback of this approach is that it requires specialized hardware such as the Faraday cage.

Second, as a relatively relaxed constraint scenario, we present Distance Bounding (DB) [12], which leverages estimated distance via wireless signals to be used as a "virtual" boundary. Although there are no physical boundary like MiB, DB requires that a human verifies if there are non-intended devices within the virtual boundary.

Third, as a most relaxed constraint scenario, we present Zero Interaction Authentication (ZIA) [24], which proposes recurring authentication when pairing IoT devices at home by leveraging contextual information (i.e., light and sound). Devices co-located at one household would experience similar context as opposed to devices in a neighbor's house. This scheme increments authenticity score over time to derive a shared key among the devices with certain confidence. The main drawbacks of this approach are that it requires only homogeneous pair of sensors (e.g., light or microphone) and takes a long time to perform contextual verification.

Chapter 3

Proposed Approaches For Applications of Varying Levels of Physical Constraints

In this chapter, we explore how varying constraints determines what relative physical context needs to be proven in order to achieve identification and authentication between devices by examining three different exemplary application scenarios that represent varying levels of constraints on physical boundary. Figure 3.1 depicts the three scenarios on the spectrum of varying constraint levels. First, we present a pairing scenario between a car's infotainment system and a driver's phone, to illustrate the *Most Constrained* solution (of the three examples). Second, we present a smart home or office scenario to illustrate the *Less Constrained* solution. Finally, we present a truck platooning scenario to illustrate the *Least Constrained* solution.

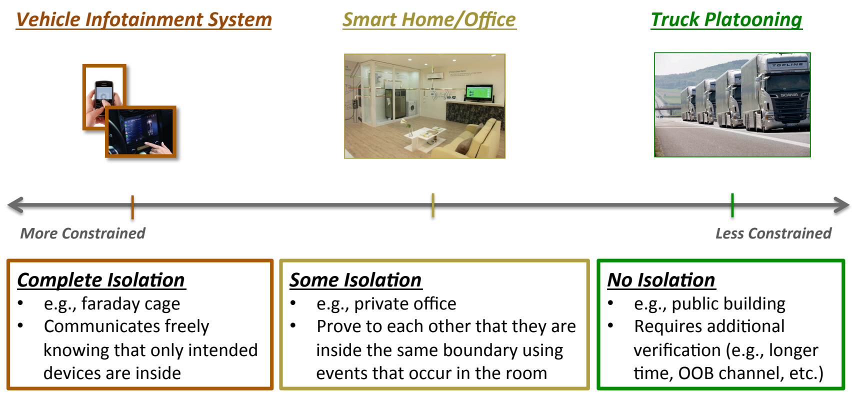

Vehicle Infotainment System
Smart Home/Office
Truck Platooning
More Constrained
Less Constrained
Complete Isolation
• e.g., faraday cage
• Communicates freely knowing that only intended devices are inside
Some Isolation
• e.g., private office
• Prove to each other that they are inside the same boundary using events that occur in the room
No Isolation
• e.g., public building
• Requires additional verification (e.g., longer time, OOB channel, etc.)

Figure 3.1: We propose to explore different illustrative example scenarios that span through a spectrum of constraints on physical boundary.

3.1 Most Constrained: Vehicle Infotainment System

We identify the pairing scenario as the most constrained one because the methodology used in this subsection entails use of a car and its glove compartment as a tightly managed physical boundary.

3.1.1 Problem Definition

With the proliferation of wireless devices using Wi-Fi and Bluetooth technologies, security of their communication is a vital concern as numerous real-world attacks have been reported [28]. Insecure wireless communication may allow attackers to eavesdrop or launch Man-in-the-Middle (MitM) attacks, impersonating legitimate communicating devices.

Efforts to eradicate such attacks have inspired many research proposals as well as industrial solutions, namely to provide secure pairing between devices by "bonding" them to establish a secure channel. However, it is still difficult for human users to easily determine which devices are being paired because of the invisible nature of wireless communication. Hence, researchers propose demonstrative identification, which affirms to the human user which devices are actually communicating leveraging out-of-band channels. [9].

However, many naive solutions attempting to establish such secure pairing for any two devices introduces a trade off. In many cases, increasing security leads to decreased usability, which becomes a significant hindrance for wide adoption of the technology by the general public. On the other hand, decreased usability may cause a security breach in these protocols. This is exemplified by the use case scenario when a user tries to pair her phone with a friend's phone using Bluetooth. The state-of-the-art solutions require the user to either copy a passkey displayed on one device to the other, compare two passkeys displayed on both devices, or to enter a hard-to-guess passkey on both devices. However, the security of such protocols often rely on the passkey not being repeated or easy-to-guess, requiring the users to input hard-to-guess passkeys to guarantee the protocol security [20]. These designs, however, lead to multiple problems in practice. Many devices actually display a repeated and/or easy-to-guess passkeys (e.g., 000000, 123456, etc.) [31]. Also, many users tend to make fatal mistake of inputting easy-to-guess passkeys [29].

In this section, we delve into a specific problem of vehicular environments. The proliferation of smartphones coupled with emerging smarter vehicles allows constant exchange of sensitive information over wireless communication. For example, different automotive manufacturers and smartphone companies established Car Connectivity Consortium (CCC) and have formed *Mirror Link*, a standard for integrating smartphones and the vehicles to enable access to the phones using car's control, display, and speakers [13]. In addition to pairing with personal cars, we expect more frequent pairing use cases for widely deployed rental car services – both traditional and short-term rental cars (e.g., Zipcar).

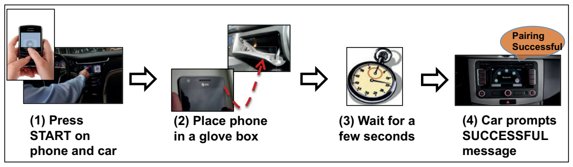
Unfortunately, coupling of smartphones and vehicles introduces a new avenue of potential attacks if the wireless channel is not secured. Although launching such attacks may not seem plausible at a first glance, they are certainly within the realm of possibility especially for high-value targets (e.g., celebrities, politicians, etc.) that provide more incentives for the attackers. Furthermore, such targets are more likely to drive luxury vehicles that embrace next-generation vehicle-to-mobile convergence systems. Are current cars effectively protected from remote attackers attempting to compromise vehicular components? Can we be convinced that the sensitive information in our vehicles is not being maliciously transmitted to attackers in other nearby cars on the road, or in parking lots connected via Bluetooth or Wi-Fi?

To address these problems, we present *MVSec*, the first secure key agreement scheme tailored specifically for vehicular environments, providing strong security guarantees and easy usability. *MVSec* leverages out-of-band channels such as sound or light because commodity hardware such as LED, ambient light sensor, speaker, and microphone are readily available in cars and/or smartphones. *MVSec* allows a user, typically the driver, to simply press a button on each device (the car and the phone) to initiate the protocols. For the protocols that leverage sound, all the user needs to do is to simply verify that both the car and the intended mobile device emit a short beep. Similarly, for protocols that leverage light emission, the user simply needs to place the mobile device in the glove compartment for a short amount of time.

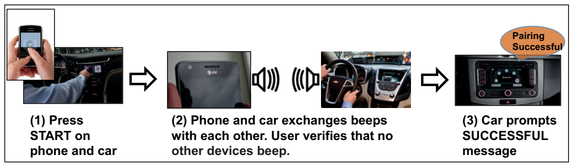
3.1.2 Capturing Contextual Cues

This section presents the overview of the *MVSec* protocols, discusses the OOB channel selection, and then delves into the protocol details. The main goal of *MVSec* is to allow a user to securely pair his/her smartphone with a vehicle such that an attacker will not successfully launch MitM attacks.

To achieve this goal, we first need to overcome the challenge of providing *demonstrative identification*, to ensure that the vehicle and the intended smartphone are in fact communicating with each other. We leverage out-of-band (OOB) communication channels as a solution. Different from the in-band channels used by the devices, e.g., Wi-Fi or Bluetooth, an OOB channel is a separate communication medium between the communicating devices (e.g, humans, light, sound, vibrations, etc.).


(1) Press START on phone and car
(2) Place phone in a glove box
(3) Wait for a few seconds
(4) Car prompts SUCCESSFUL message
Pairing Successful

(a) Strong OOB Channel (light)


(1) Press START on phone and car
(2) Phone and car exchanges beeps with each other. User verifies that no other devices beep.
(3) Car prompts SUCCESSFUL message
Pairing Successful

(b) Weak OOB Channel (sound)

Figure 3.2: Overview of *MVSec* using light and sound as OOB channels

Out-of-band Channel Selection

MVSec leverages two types of OOB channels for different protocols. According their characteristics, they are categorized into *strong* and *weak* OOB channels.

Strong OOB Channel. A strong OOB channel guarantees both *secrecy* and *authenticity.* We select **light** in a vehicle's closed glove compartment as the strong OOB channel because it helps to provide both of these security properties with minimal human involvement. We assume that the glove compartment does not leak any light signal, thus provides secrecy. This channel also provides authenticity because only the vehicle will emit light signals. This is because no other device is inside the compartment as the driver first verifies that other devices are not placed inside the compartment during protocol execution.

In addition to considering the security properties, we choose light because of the following requirements of the OOB channel. The OOB channel needs to (1) be readily available in vehicles and smartphones today in order to be easily deployable, and (2) provide high usability, i.e., the OOB channel needs to have a relatively fast data rate and be easy and pleasant to use (i.e., should not require user diligence nor annoy the users). We define relatively fast data rate to be faster than the OOB channel used as baseline case, which is manual human input. This OOB channel allows such usability because the only task that the user performs is to press a button on both the vehicle and the smartphone, and place the smartphone inside the glove compartment. After waiting for a few seconds, during which the vehicle transmits signals via blinking lights to the smartphone, the pairing process successfully completes. The steps are depicted in Figure 3.2(a).

Weak OOB Channel. A weak OOB channel provides only *authenticity.* We select **sound** signals as the weak OOB channel. This channel helps to provide authenticity, again, with minimal human involvement, because we assume that it is easy for the user to identify that the sound beeps are originating only from the intended devices (e.g., vehicle and driver's smartphone). If an unintended device beeps, the user simply aborts the protocol. We assume that the beeps are sufficiently long and loud enough for the user to easily identify the origin of the beeps. We assert that this is a realistic assumption, because smartphone users generally distinguish who's phone is ringing when (s)he hears a phone ring. We also use sound signals because of the ubiquitous deployments of microphones and speakers in vehicles and smartphones. Figure 3.2(b) depicts the steps the user performs. We provide different variation of protocols using different cryptographic primitives that leverage the two aforementioned OOB channels.

3.1.3 Tasks

We decompose this project into tasks, which have been completed. First task is to design the security protocol, by applying modifications on Diffie-Hellman protocols leveraging the OOB channels. Second task is to analyze this protocol to successfully defend against MitM attacks. Using the protocol, the third task is to implement *MVSec*'s prototype, which is used in the fourth task of conducting a user study, by recruiting over 20 participants. Lastly, the results are evaluated. We present a summary of the tasks in Chapter 5.

3.2 Less Constrained: Smart Home and Office

We now present an application scenario that is less constrained than that of the previous scenario, namely autonomous IoT device pairing in a smart home or office setting. This is less constrained in terms of physical boundary, because the home or office inherently provide a physical separation between devices inside and outside. However, unlike the previous scenario, wireless signals as well as some attenuated sound or vibration signals still propagate to the devices outside of the boundary.

3.2.1 Problem Definition

Securing the IoT network becomes a necessity as introducing more IoT devices comes at a cost of potential privacy leakage as these devices are equipped with sensors that monitor activities within a home or an office space [25, 26, 15]. While securing the connectivity of the IoT devices is crucial, it is non-trivial for an end user to perform security configurations on the devices. We refer to performing security configurations (e.g., configuring Wi-Fi WPA2 for secure pairing) in this chapter as establishing a secure channel between devices (i.e., cryptographic key agreement). First, many IoT devices come without any I/O mechanisms. Trying to configure a motion detector without a proper display and keypad will be extremely cumbersome for home owners. Second, there are too many IoT devices to configure. One can already easily find tens of IoT devices in a home or office. Furthermore, the number of IoT devices in a smart home are even projected to increase to 500 within the next decade [14]. To exacerbate the problem, the devices may need multiple security reconfigurations during the life span of these devices. Such reconfigurations may be due to multiple reasons – new cryptographic algorithm may be introduced requiring longer keys, or the IoT devices may themselves be compromised and the keys may have been breached – all of which, require reestablishment of cryptographic keys. In addition to the aforementioned I/O problems, many of the devices, such as smoke detectors, HVAC systems, etc., are permanently installed and are extremely cumbersome to physically reinstall after reconfigurations. Considering a long life span of many of these devices, which usually range in the order of years, security reconfiguration introduces a serious usability drawback. While newer IoT devices are starting to be equipped with NFC or other out-of-band channels such as light to help in solving the I/O problem, such solutions still do not scale with many devices requiring multiple reconfigurations and additional specialized hardware [4, 3].

The aforementioned drawbacks necessitate an autonomous pairing mechanism for the IoT devices that require no human involvement in any of the pairing processes. To address this concern, we propose *Perceptio* (which means perception in Latin), an autonomous pairing scheme based on the similarity of contextual information collected by sensors of the IoT devices. *Perceptio* makes use of the findings that IoT devices co-located within a physical boundary such as a room will experience more of the similar events over time as opposed to potential devices owned by an attacker, who may be outside of the physical boundary. Specifically, physical boundary such as homes, offices, or factories are naturally enforced with a notion of physical security. For example, access rights to an apartment complex are only granted to the residents, while office or factory buildings also enforce access rights to their employees. Visitors are also granted access after an implicit or explicit delegation of trust. In this section, we are inspired by such notion of trust within a physical boundary because it provides a natural separation of trust. Hence, we envision devices within a physical boundary would also trust each other compared to devices that are outside of an apartment or office building.

However, many challenges exist to fully address the aforementioned problem. First, we identify new challenges related to devices that have disparate sensing capabilities. We find from our main observations of recent trends in commercially available IoT devices, which reveals the emergence of special-purpose sensing devices with only a small number (often one) of embedded sensors, typically optimized for a specific application for cost, power consumption, and form factor. For example, Passive infrared motion detectors are only equipped with a single infrared sensor to monitor movements [11]. Hence, it is infeasible to directly compare context collected from different devices with different sensor modalities because differing sensor types produce completely different signals. However, we make use of the findings that many sensors of different modalities actually respond to and perceive the same context. Specifically, *Perceptio* leverages a common feature across different modalities, namely the time intervals of starting points of the common events. Because *Perceptio* leverages the time intervals, it does not require the devices to be time synchronized.

Yet, challenges still remain, as different sensors located at different positions within a room would produce similar but not equal signals. Hence *Perceptio* tolerates these error by leveraging a fuzzy commitment scheme, which bases its error tolerance on Reed-Solomon error correction mechanism.

We evaluate *Perceptio* by placing different devices equipped with disparate sensor types – a geophone, a microphone, an accelerometer, a motion detector, and a power meter – in an office. We also place attacker's devices just outside the office. We conduct experiments to allow these devices to capture events caused by human subjects that occur in an office setting.

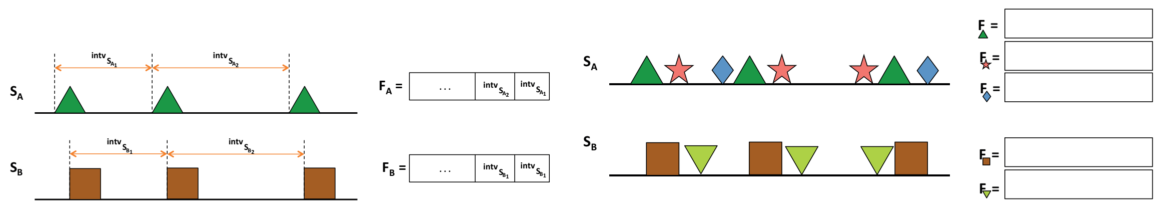
3.2.2 Capturing Contextual Cues

As with any cryptographic key agreement protocols, *Perceptio* needs to bootstrap its trust from a source of entropy. We leverage the inherent randomness of events occurring in a room (e.g., knocking, walking, talking, etc.) as its source of entropy in its cryptographic protocol. Specifically, *Perceptio* leverages the fact that it is infeasible for an attacker to guess a series of event occurrence in sub-second granularity. Hence, fingerprint extraction from contextual information is a vital part of *Perceptio*. We now discuss the design choice of the fingerprint extraction algorithm, and how multiple event types affect *Perceptio* fingerprinting.

We now address the seemingly impossible challenge of trying to "fingerprint" disparate sensor modalities across IoT devices. This is possible because sensors of different modalities "perceive" the same context even though their numerical representation may be different. Hence, *Perceptio* abstracts out from differing numerical sensor data, and leverages temporal domain as common feature across the devices. Specifically, *Perceptio* makes use of the fact that starting points of commonly observed events are spaced out at equal time intervals, and captures the collection of these time intervals as fingerprints. Figure 3.3(a) depicts an example of how two sensors of different modalities fingerprint a commonly observed context. Assuming that both $Sensor_A$ and $Sensor_B$ observed an event, their numerical representations of the event are shown as *triangles* and *circles*, respectively. Note that intervals between the starting times of adjacent *triangles*, denoted as $intv_{S_{A_1}}$ and $intv_{S_{A_2}}$, while those of *circles* are denoted as $intv_{S_{B_1}}$ and $intv_{S_{B_2}}$. *Perceptio* makes use of the fact that $intv_{S_{A_i}}$ and $intv_{S_{B_i}}$ are very similar to each other. Subsequently, the intervals are converted into bits and appended to the fingerprints as:

$$\begin{aligned} F_A &= \{intv_{S_{A_n}} || intv_{S_{A_{n-1}}} || ... || intv_{S_{A_1}}\} \\ F_B &= \{intv_{S_{B_n}} || intv_{S_{B_{n-1}}} || ... || intv_{S_{B_1}}\} \end{aligned} \tag{3.1}$$

However, we need to address an additional challenge of how disparate sensors have varying events (e.g., walking, door opening, talking, etc.) that they are responsive to. For example, consider $Sensor_A$ and



(a) Single event commonly observed (b) Events observed by each sensor

Figure 3.3: Creating F with starting point intervals.

$Sensor_B$ of different modalities, and the corresponding events that the two sensors observe. Events that only $Sensor_A$ and $Sensor_B$ observes are denoted as $E_A\overline{E_B}$ and $\overline{E_A}E_B$, respectively. Events that both modalities observe in common are denoted as E_AE_B. For example, a microphone will be more sensitive to people speaking, geophones will be more sensitive to footsteps of a walking event, and both of these sensors are responsive to a running coffee machine (i.e., $E_{mic}\overline{E_{geo}} = \{talking\}$, $\overline{E_{mic}}E_{geo} = \{walking\}$, and $E_{mic}E_{geo}$ =$\{$*running coffee machine*$\}$). However, the core idea of *Perceptio* lies in the fact that most pairs of disparate sensor types have sets of common events that they respond to with varying sensitivity.

In Figure 3.3(b), $\{\blacklozenge, \blacktriangle, \bigstar\}$ and $\{\blacktriangledown, \blacksquare\}$ depict set of signals that are observed by $Sensor_A$ and $Sensor_B$, respectively. Hence, each sensor will first locally determine similar events (i.e., clustering events into different clusters), and extract corresponding fingerprints per event type. Hence $Sensor_A$ will have three distinct fingerprints (i.e., $F_{\blacklozenge}$, $F_{\blacktriangle}$, and $F_{\bigstar}$), while, $Sensor_B$ will have two distinct fingerprints (i.e., $F_{\blacktriangledown}$ and $F_{\blacksquare}$).

Perceptio bootstraps its trust from the entropy of the occurrence of captured events. These occurrences are converted to the intervals of the starting points, which in turn are translated into the bits of the fingerprint. Hence, the entropy of the fingerprint depends on the length of the fingerprint. This is depicted in Equation 3.2. F depicts the concatenation of bit values of each intervals, $intv_{S_{A_i}}$, for $i = [1, n]$ intervals, where $[1, n]$ represent integers from 1 to n, inclusive. If the length of F are greater than l_{min}, a minimum length of a fingerprint, F is truncated to l_{min} bits. If less than the minimum length, the fingerprint is discarded due to lack of enough entropy.

$$F_{E_A} = \begin{cases} \Big[F\Big]_{l_{min}}, & \text{if } |F| \geq l_{min} \\ \emptyset, & \text{otherwise} \end{cases} \tag{3.2}$$

Figure 3.4 depicts the flow chart diagram of fingerprint generation steps. First a sensor captures contextual information for fingerprint time period, t_F, producing a raw sensor data. This is first input to *Signal Detection* module, which distinguishes signals of events (e.g., walking, talking , etc.) against ambient noise and outputs the corresponding indices of the event signals. Subsequently, these indices, along with the raw sensor data, are input to *Event Clustering* module, which performs unsupervised learning to cluster signals of similar events via K-Means clustering. Hence, module outputs different cluster IDs and the corresponding indices of the signals belonging to the clusters. The output is then input to *Fingerprint Extraction* module, which finally converts the cluster indices into fingerprints to be used in *Perceptio* protocol. We now present the implementation details of *Signal Detection* and *Event Clustering* modules.

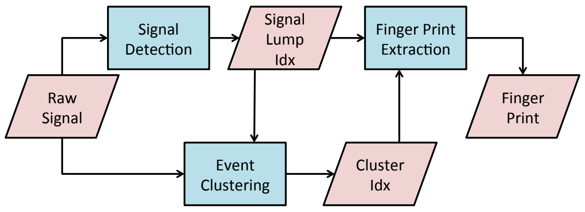
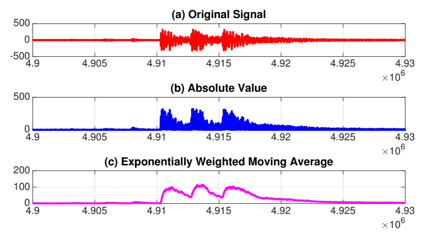


Figure 3.4: Overview of *Perceptio* fingerprint generation flow chart.

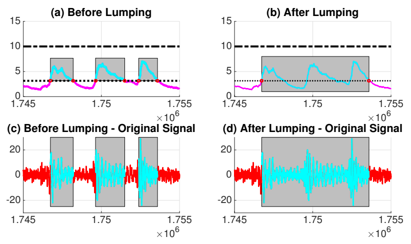
Signal Detection

The goal of signal detection module is to identify the signals that represent events of interest as opposed to ambient noise. We break down the tasks into two steps – (1) performing a moving average; and (2) thresholding and signal detection. We first compute a moving average to smooth out the signal for noise removal using *Exponentially Weighted Moving Average (EWMA)*. Figure 3.5(a) illustrates this effect, as the first plot depicts the original geophone signal of the event of a person walking. The second plot depicts the absolute value of the original plot, and the third plot depicts the *EWMA* of the absolute value.

We then perform *thresholding* for signal detection. We note that we have two types of thresholding, namely a lower-bound (Thr_{lower} and an upper-bound (Thr_{upper}) threshold. We leverage Thr_{lower} to distinguish event signals to ambient noise. On the other hand, we leverage Thr_{upper} to remove any signals of high amplitude, in order to thwart an eavesdropping attack. We note that Thr_{upper} can be a function of Thr_{lower} after certain calibration phase. This is depicted in Figure 3.5(b) (a), where we apply a lower-bound thresholding to the *EWMA* signal using the lower dotted line (i.e., $Thr_{lower} = 3$). The signal above the threshold are highlighted with a gray box. Also, we apply an upper-bound thresholding as well using the upper dotted line (i.e., $Thr_{upper} = 10$). For more accurate event clustering, however, we implement a signal *lumping* technique to group segmented parts of the event signal into a single event signal, as shown in Figure 3.5(b) (b). Specifically, we disregard short discontinuities between adjacent segmented signals above threshold to "lump" the signals into one continuous group of signal event. From the indices returned by these steps, we determine the signal of interest in the original signals as presented in Figures 3.5(b) (c) and (d), depicting before and after lumping technique, respectively. Finally, this module outputs the corresponding indices of detected signal to the *Event Clustering* module.



(a)



(b)

Figure 3.5: Illustration of (a) raw geophone signal, followed by corresponding absolute value, and subsequent exponentially weighted moving average; (b) thresholding and signal isolation.

Event Clustering

We implement event clustering to reliably categorize observed events into appropriate cluster groups. Though some additional work may increase the accuracy and efficiency of the clustering results, we present a preliminary proof-of-concept implementation details.

We select a set of features per sensor to reliably separate perceived events via clustering such as maximum amplitude, duration, and area under the curve and its variants. We leverage K-Means Clustering to cluster data points of similar groups, without prior training, by taking in hypothesis cluster number, K. Hence, we make use of Elbow method to infer the optimum value of K [18], which tests several number of K cluster hypothesis to find the optimal K value. Consequently, K-means clustering and elbow method enables each sensor device to roughly group perceived events to categories without the burden on the manufacturers to train specific events for all the devices.

3.2.3 Tasks

Different tasks are broken down into the following categories. First, we collect a real-world data by deploying sensors in an office space. We then design and implement a fingerprint extraction algorithm and apply it to the collected data. This algorithm translates the data into stream of bits that can be used in the protocol we design. The protocol enables devices to tolerate for subtle errors due to differences in sensing modalities while experiencing same events. Furthermore, we analyze the protocol to demonstrate its robustness against active attacks. We then implement the *Perceptio* prototype and analyze experimental results. Detailed tasks and their corresponding timeline are presented in Chapter 5.

3.3 Least Constrained: Truck Platooning

We now present an application scenario with the least constraint among the three examples we present in this chapter. We present a truck platooning scenario, where there is no physical boundary that exists to keep the unintended devices out. Rather, the trucks may be traveling openly, with the possibility of attackers driving along side the trucks.

3.3.1 Problem Definition

Vehicle platooning is a method of having a group of vehicles drive in a single file to follow the preceding vehicle and ultimately the leading vehicle, which is gaining large traction today. Platooning is getting significant attention in the commercial trucking industry [5, 2, 7, 21, 32] as it provides benefits of increase in fuel efficiency (and reduction in CO_2 emissions), driving safety, road efficiency , and driver convenience [6]. The consumer vehicle industry is also preparing to incorporate personal vehicles into platooning to take advantage of the aforementioned benefits [27, 8].

Because platooning leverages wireless communications to transmit control messages such as accelerating, braking, and steering information from the participating cars, securing the communication is extremely crucial as attacks may result in life-threatening collisions, damage to high-value vehicles and cargo, and loss of business. Specifically, vehicle platooning uses Dedicated Short-Range Communications (DSRC) and Wireless Access in Vehicular Environments (WAVE) as de facto standards for vehicle-to-vehicle (V2V) communications [23, 17]. The current DSRC/WAVE model assumes Public Key Infrastructure (PKI) that authenticates each vehicle's public key by leveraging certificates signed by a trusted third party, such as a Certificate Authority (CA). Unfortunately, this model is susceptible to impersonation attacks such as masquerading or sybil attacks (impersonating as non-existing or "ghost vehicles") [22, 10].

The root cause of the aforementioned problems is due to the fact that the vehicles have no way of binding their digital certificates with their physical identities. This is on the contrary to the analogy of web server authentication in TLS/SSL. The certificates of the server contains the identifier (i.e., URL), which can be compared to the URL that the user has visited via the web browser, hence naturally *binding* the "physical identity" with the identity included in the certificate. Even though the certificate of the vehicles contain their identifiers, other cars in the platoon have no way of ***verifying*** the physical identity of the certificate. This is exemplified in Figure 3.6, where *Cars A* and *B* are vehicles in am existing platoon. *Car C* is a vehicle that wishes to join the platoon, and *Car M* is an attacker's car driving in an adjacent lane. In this example, *Cars A* and *B* receives *Car C*'s certificate, but is unsure if the certificate is actually from *Car C* or *Car M* in the adjacent lane.

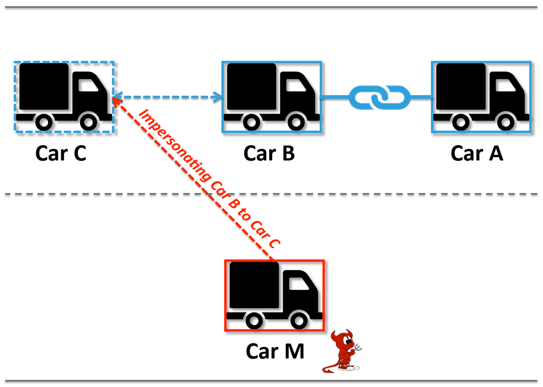


Figure 3.6: Overview diagram depicting vulnerabilities of platooning systems to impersonation attacks.

To address the above problem, we present *Convoy*, an autonomous authentication and verification scheme of platooning vehicles. *Convoy* performs the verification by binding the certificate with the physical context. *Convoy* is based on the findings that vehicles wishing to form a platoon can prove to each other that they are indeed traveling together using context information captured from their sensor data. This is possible because *Convoy* requires a vehicle wishing to join the platoon (*Car C*) to follow a rear vehicle of the platoon (*Car B*) for a period of time resorting to automated technologies such as Adaptive Cruise Control (ACC), which enables a vehicle to autonomously follow the front car with constant headway and keeps the vehicle within the traveling lane (ACC is prevalent in many vehicles today) [1, 30]. Consequently, unique *road conditions* per lane (e.g., bumps, cracks, potholes, etc.) cause similar vibrations between potential vehicles of a platoon as they travel on the same lane in a single file, as opposed to a car traveling on an adjacent lane (Cars A, B and C as opposed to Car M). Furthermore, *traffic conditions* cause vehicles traveling in a single file to experience similar acceleration, deceleration, braking, and steering. Hence *Convoy* leverages these conditions as sources of entropy to establish a symmetric cryptographic key between a pair of vehicles, naturally binding the physical context to the symmetric key. Subsequently, the symmetric key is used to authenticate the certificates to delegate the bindings of physical context to the certificates.

Yet, challenges remain to achieve the aforementioned goals. First, different cars wishing to join as a platoon (e.g., Cars A, B, and C) would experience similar but different context, leading to numerically unequal signals. In order to compensate for subtle differences in the signals between the cars, *Convoy* makes use of an emerging cryptographic primitive called *Fuzzy Commitment* [24, 16] that relies on error-correcting codes to establish a shared symmetric key from similar-but-unequal signals capturing a common context.

Second, comparison of contextual information increases room for error because similar context may exist for attackers at times (e.g., two lanes may have similarities at times). Hence, *Convoy* requires the vehicles to repeat the protocol for multiple iterations over time (and increment a confidence score upon a successful termination of each iteration), thereby increasing the probability of vehicles traveling together to experience more similar context, while vehicles traveling in another lane would not. Hence, *Convoy* thwarts attackers driving on an adjacent lane to the platoon, attempting to claim a membership in the platoon or impersonating an existing member in the platoon.

3.3.2 Capturing Contextual Cues

The goal of *Convoy* is to enable vehicles trying to establish a secure platoon to verify that their public keys indeed belongs to the vehicles driving in the platoon as opposed to an attacker's car driving in an adjacent lane. Hence, the protocol binds the physical context, namely innate context experienced by the vehicles driving together in a platoon, to their public keys. Specifically, *Convoy* enables this binding by making use of the shared context as sources of entropy in establishing a shared symmetric key between a pair of cars. This symmetric key is used to verify the owner of the public keys by having the pair of vehicles each compute a MAC over each car's public key.

Convoy leverages the following innate context for a platoon to leverage as sources of entropy to the key agreement protocol – unique *road* and *traffic conditions*. First, each lane has *varying road conditions* that are hard-to-guess by attackers. The road conditions not only differ at different segments of the road, but also across different lanes. This is often due to many factors such as patches, bumps, cracks, pot-holes, etc. Second, traffic conditions are inherently random as they vary when different vehicles on the road travel together, causing the platoon to accelerate, brake, and steer differently. Hence, the vehicles traveling on the same lane within a platoon will experience similar road and traffic conditions, while an attacker traveling in an adjacent lane will not. We note that the control messages (e.g., acceleration, brake, and steer messages) are encrypted to be shared only within the members of a platoon to thwart attackers from mimicking the traffic conditions. However, a new vehicle wishing to join a platoon (*Car C*) follows the read vehicle of the platoon (*Car B*) resorting to ACC so that C's traffic information would be similar to that of B. This is depicted in Figure 3.6, where Vehicles A and B already travel as an existing platoon and Vehicle C is trying to join the platoon, while an attacker's vehicle, M, is traveling in an adjacent lane.

Convoy leverages the aforementioned findings to extract context fingerprints to have a newly joining vehicle (C) to prove to the vehicles in a platoon (A and B) that they are traveling together. Each vehicle running *Convoy* protocol makes use of a multi-axis accelerometer to capture the context. Specifically, the road condition is captured by an axis perpendicular to the road (Acc_{Road}), while the traffic condition is captured by the axis parallel to the lane (Acc_{Traf}). The captured context is then extracted to fingerprints, which will be used in the protocol shown later in this chapter to prove to each other that they are in fact driving together in a platoon.

Fingerprint Extraction Algorithm and Implementation

The *extractF()* function takes in a raw signal (i.e., Acc_{Road} or Acc_{Traf}) and the fingerprint capture time, t_F, to encode the signal to a bit stream, F, of length l_F. The main idea behind this extraction algorithm is to capture abrupt changes in the raw signals and encode them into *high bits* (i.e., bit value '1's), while encoding the rest of the signal as *low bits* (i.e., bit value '0's). The extraction is divided into the following phases: (1) Pre-processing, (2) Derivative, (3) Bit Translation. The phases are illustrated in Figure 3.7.

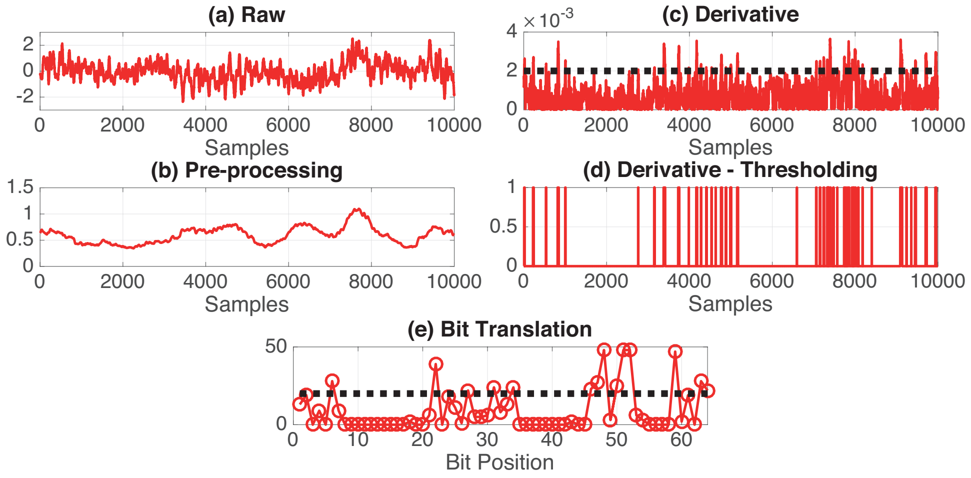


Figure 3.7: Illustration of fingerprint extraction phases – (a) depicts a raw data; (b) depicts the moving average of the absolute value of the raw signal; (c) depicts the resulting signal after taking the derivative; (d) depicts the binary signal after thresholding; and (e) depicts the bit window (e.g., 64 bits of bit length).

Pre-processing. We pre-process raw sensor data to minimize noise and maximize detection of information (i.e., context from road and traffic conditions). Hence, first process raw sensor data to remove high frequency noise components in the raw data which would otherwise appear as noise in the following steps. We compute the absolute value of the entire signal to capture the magnitude of the samples independent of the sign. Next, with goals to remove high frequency noise in the data, we compute the average of samples as a sliding window, computing a moving average of the raw signal.

Derivative. Taking the pre-processed signal, we take the derivative of the signal to obtain, $S'[x]$.

The derivative helps to detect abrupt changes in the smoothed signal. The derivative of the signal with respect to time rewards abrupt changes in the signal while penalizing slow stagnant changes. We note that this is computationally synonymous to applying a bandpass filter to capture the signal of interest, while implementation may be realized in different ways. This is depicted in Figure 3.7 (c). The dotted lines indicate the thresholding value, Thr_{Deriv}, to capture sudden changes. The resulting binary signal, $S'_{binary}[t]$ is computed as Equation 3.3 and illustrated in Figure 3.7 (d).

$$S'_{binary}[t] = \begin{cases} 1, & \text{if } S'[t] > Thr_{Deriv} \\ 0, & \text{otherwise} \end{cases} \tag{3.3}$$

Bit Translation. $S'_{binary}[t]$ is then divided into l_F windows (e.g., 64 windows for 64 bits) of size $bitWnd$. For each window, we compute the summation of the values of $S'_{binary}[t]$. The resulting summation is depicted in Figure 3.7 (e). If this sum is greater than a threshold (depicted with the dotted line), Thr_{trans}, the encoded bit is 1, and 0 otherwise, for the fingerprint, F, as depicted by Equation 3.4.

$$F[n] = \begin{cases} 1, & \text{if } \sum_{t=0}^{bitWnd} S'_{binary}[t] > Thr_{trans} \\ 0, & \text{otherwise} \end{cases} \tag{3.4}$$

Subsequently, each truck verifies if the fingerprint contains enough entropy by computing the ratio of high bits. If the ratio is below certain threshold, the fingerprint is discarded and new fingerprint is generated in the next iteration.

3.3.3 Tasks

We now decompose this project into different tasks at hand. Similar to what is mentioned in Chapter 3.1.3, we first design the fingerprint extraction algorithm to explore how the driving data collected by an accelerometer can be converted into a stream of fingerprint bits. We then design its security protocol to leverage the *road* and *traffic* conditions as sources of entropy, which is error-tolerant to take into account for different but similar signals captured by different trucks in the platoon. We also perform security analysis to verify that the protocol successfully defends against described attacks. Finally, we implement the prototype and analyze the experimental results. Furthermore, Chapter 5 presents a list of the tasks and their corresponding schedule.

Chapter 4

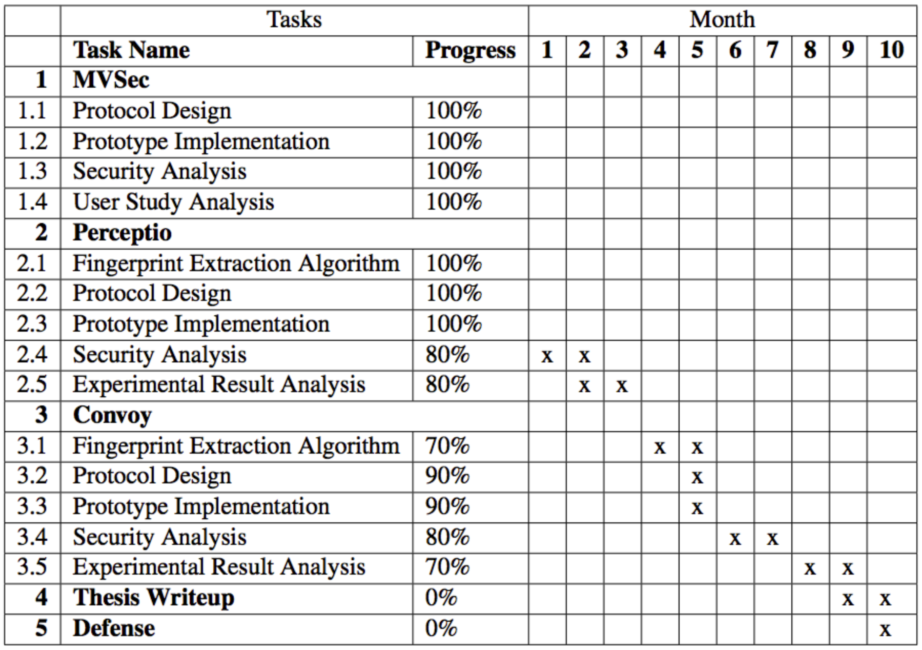
Summary

In this thesis prospectus, I present analysis on solutions for IoT devices to bind digital identities to physical identities, ultimately enabling them to correctly identify and authenticate each other. More specifically, I present the solution of allowing IoT devices to prove their unique set of relative physical context, which is governed by how the devices are constrained in terms of physical boundary. Hence, I provide analysis on exploring varying levels of constraints and their influences on the relative physical context in different application scenarios such as (1) pairing a smartphone with vehicular infotainment system, (2) smart home or office, and (3) truck platooning. Preliminary results demonstrate the feasibility of the aforementioned solution. Although I demonstrate three specific application scenarios through this thesis, I expect to contribute the generic methodology of complementing security protocols leveraging physical context for the IoT devices.

Chapter 5

Research Plan

In this chapter, I provide the summary of the remaining and completed tasks, and their corresponding timeline, as depicted in Table 5.1. Currently **MVSec** project is completed. I will first concentrate on completing the remaining tasks of **Perceptio** project, namely, working on security and experimental analysis. Subsequently, I will then work on completing **Convoy** project, by looking into different tasks at hand, namely extracting contextual fingerprint from the sensor data, designing and implementing the protocol, and analyzing security and experimental results. Finally, I will spend time to write my dissertation and prepare for defense.



	Tasks		Month									
	Task Name	**Progress**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	**MVSec**											
1.1	Protocol Design	100%										
1.2	Prototype Implementation	100%										
1.3	Security Analysis	100%										
1.4	User Study Analysis	100%										
2	**Perceptio**											
2.1	Fingerprint Extraction Algorithm	100%										
2.2	Protocol Design	100%										
2.3	Prototype Implementation	100%										
2.4	Security Analysis	80%	x	x								
2.5	Experimental Result Analysis	80%		x	x							
3	**Convoy**											
3.1	Fingerprint Extraction Algorithm	70%				x	x					
3.2	Protocol Design	90%					x					
3.3	Prototype Implementation	90%					x					
3.4	Security Analysis	80%						x	x			
3.5	Experimental Result Analysis	70%								x	x	
4	**Thesis Writeup**	0%									x	x
5	**Defense**	0%										x

Table 5.1: Timeline of research plan

Bibliography

[1] Adaptive Cruise Control . `http://path.berkeley.edu/fhwa-ear-program-project-cacc-cooperative-adaptive-\\cruise-control-simulation-using-cooperative-acc`, 2016.

[2] Daimler Trucks is Connecting Its Trucks With The Internet. `http://media.daimler.com/deeplink?cci=2742821`, 2016.

[3] ElectricImp. `https://electricimp.com/`, 2016.

[4] ElectricImp BlinkUp$^{\text{TM}}$. `https://electricimp.com/platform/blinkup/`, 2016.

[5] European Truck Platooning Challenge – Creating Next Generation Mobility. `https://www.eutruckplatooning.com/home/default.aspx`, 2016.

[6] Impact of Platooning on Traffic Efficiency. `https://trid.trb.org/view.aspx?id=1262546http://papers.sae.org/2014-01-2438/http://papers.sae.org/2013-26-0142/`, 2016.

[7] Peloton. `http://peloton-tech.com/`, 2016.

[8] Platooning. `http://www.volvogroup.com/GROUP/GLOBAL/EN-GB/RESEARCHANDTECHNOLOGY/TRANSPORTSOLUTIONS/PLATOONINGTRIALS/PAGES/DEFAULT.ASPX`, 2016.

[9] D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *NDSS*, 2002.

[10] N. Bissmeyer, J. Njeukam, J. Petit, and K. M. Bayarou. Central misbehavior evaluation for vanets based on mobility data plausibility. In *Proceedings of the Ninth ACM VANET*, 2012.

[11] Bosch. ISC-BPR2 - Blue Line Gen2 PIR Motion Detectors. `http://resource.boschsecurity.com/documents/BlueLine_Gen_2_Data_sheet_enUS_2603228171.pdf`.

[12] M. Cagalj, S. Capkun, and J. P. Hubaux. Key agreement in peer-to-peer wireless networks. *Proceedings of the IEEE*, 94(2):467–478, Feb 2006.

[13] C. C. Consortium. Mirror Link. `http://www.mirrorlink.com/`.

[14] Deloitte. The Digital Predictions 2015. The Deloitte Consumer Review, 2015.

[15] J. Han, A. Shah, M. Luk, and A. Perrig. Don't sweat your privacy using humidity to detect human presence. In *Workshop On UbiComp Privacy - Technologies, Users, Policy (UbiPriv)*, 2007.

[16] A. Juels and M. Sudan. A fuzzy vault scheme. In *Information Theory, 2002. Proceedings. 2002 IEEE International Symposium on*, pages 408–, 2002.

[17] J. Kenney. Dedicated short-range communications (dsrc) standards in the united states. *Proceedings of the IEEE*, 99(7):1162–1182, July 2011.

[18] D. J. Ketchen and C. L. Shook. The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal*, 17(6):441–458, 1996.

[19] C. Kuo, M. Luk, R. Negi, and A. Perrig. Message-in-a-bottle: User-friendly and secure key deployment for sensor nodes. In *ACM Conference on Embedded Networked Sensor System (SenSys)*, Nov 2007.

[20] C. Kuo, J. Walker, and A. Perrig. Low-cost manufacturing, usability, and security: An analysis of bluetooth simple pairing and wi-fi protected setup. In *USEC*, 2007.

[21] M. P. Lammert, A. Duran, J. Diez, K. Burton, and A. Nicholson. Effect of Platooning on Fuel Consumption of Class 8 Vehicles Over a Range of Speeds, Following Distances, and Mass. SAE Technical Report, 2014.

[22] C. Laurendeau and M. Barbeau. *Ad-Hoc, Mobile, and Wireless Networks: 5th International Conference, ADHOC-NOW 2006, Ottawa, Canada, August 17-19, 2006. Proceedings*, chapter Threats to Security in DSRC/WAVE, pages 266–279. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[23] Y. J. Li. *Quality, Reliability, Security and Robustness in Heterogeneous Networks: 7th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, QShine 2010, and Dedicated Short Range Communications Workshop, DSRC 2010, Houston, TX, USA, November 17-19, 2010, Revised Selected Papers*, chapter An Overview of the DSRC/WAVE Technology, pages 544–558. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[24] M. Miettinen, N. Asokan, T. D. Nguyen, A.-R. Sadeghi, and M. Sobhani. Context-based zero-interaction pairing and key evolution for advanced personal devices. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 880–891, New York, NY, USA, 2014. ACM.

[25] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin. Private memoirs of a smart meter. In *Proceedings of the 2Nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, pages 61–66, New York, NY, USA, 2010. ACM.

[26] S. Pan, N. Wang, Y. Qian, I. Velibeyoglu, H. Y. Noh, and P. Zhang. Indoor person identification through footstep induced structural vibration. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, HotMobile '15, pages 81–86, New York, NY, USA, 2015. ACM.

[27] T. Robinson and E. Chan. Operating Platoons On Public Motorways: An Introduction To The SARTRE Platooning Programme. Proceedings of the 17th ITS World Congress, Oct 2010.

[28] K. Scarfone and J. Padgette. Guide to bluetooth security. *NIST Special Publication*, 800:121, 2008.

[29] E. Uzun, K. Karvonen, and N. Asokan. Usability analysis of secure pairing methods. In *USEC*, 2007.

[30] B. van Arem, C. J. G. van Driel, and R. Visser. The impact of cooperative adaptive cruise control on traffic-flow characteristics. *IEEE Transactions on Intelligent Transportation Systems*, 7(4):429–436, Dec 2006.

[31] S. Viehbock. Brute forcing Wi-Fi Protected Setup. When poor design meets poor implementation. `http://sviehb.files.wordpress.com/2011/12/viehboeck_wps.pdf`.

[32] Will Knight. 10-4, Good Computer: Automated System Lets Trucks Convoy as One. MIT Technology Review, May 2014.