Verifying Physical Endpoints to Secure Digital Systems

Submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Ahren M. Studer

B.S., Computer Science, University of Rochester

B.S., Electrical and Computer Engineering, University of Rochester

M.S., Electrical and Computer Engineering, Carnegie Mellon University

Carnegie Mellon University

Pittsburgh, PA

May, 2011

Thesis Committee:

This research was supported in part by CyLab at Carnegie Mellon grant DAAD19-02-1-0389 from the Army Research Office, the iCAST project, National Science Council, Taiwan under the Grants No. (NSC95-main) and No. (NSC95-org), General Motors through the GM-CMU Collaborative Research Laboratory, and by gifts from Bosch.

Keywords: System Security, Mobile Computing, Trust Establishment, Key Management, Secure Storage, Vehicular Networks

Abstract

The proliferation of electronic devices supporting sensing, actuation, and wireless communication enables the monitoring and/or control of a variety of physical systems with digital communication. Such "cyber physical systems" blur the boundaries of the digital and physical worlds, where correct information about the physical world is needed for the correct operation of the digital system. Often in these systems the physical source or destination of information is as important as the information itself. However, the omni-directional and invisible nature of wireless communication makes it difficult to determine communication endpoints. This allows a malicious party to intercept wireless messages or pose as other entities in the system. As such, these systems require new protocols to associate the endpoints of digital communication with physical entities.

Traditional security approaches that associate cryptographic keys with names can help verify endpoints in static systems where a string accurately describes the role of a device. In other systems, the role of a device depends on its physical properties, such as location, which change over time. This dynamic nature implies that identification of an endpoint based on a static name is insufficient. Instead, we can leverage devices' sensing and actuation capabilities to verify the physical properties and determine the physical endpoints of communication. We investigate three different scenarios where the physical source and/or destination is important and propose endpoint verification techniques: verifying the physical endpoints during an exchange between two smartphones, verifying the receiver of information is in a physical space to enable location-based access control, and verifying the source of information to protect Vehicle-to-Vehicle (V2V) applications. We evaluate our proposals in these systems and show that our solutions fulfill the security requirements while utilizing existing hardware.

Exchanging Information Between Smartphones Shake on it (SHOT) allows users to verify the endpoints during an exchange of information between two smartphones. In our protocol, the phones use their vibrators and accelerometers to establish a human-observable communication chan-

nel. The users hold the phones together while the phones use this channel to bootstrap and verify the authenticity of an exchange that occurs over the higher-bandwidth wireless channel. Users can detect the injection of information from other devices as additional vibrations, and prevent such attacks. Our implementation of SHOT for the DROID smartphone is able to support sender and receiver verification during an exchange between two smartphones in 15 seconds on average.

Location-Based Access Control We propose using location-based access control to protect sensitive files on laptops, without requiring any effort from the user to provide security. With a purely wireless electronic system, verifying that a given device is in a physical space is a challenge; either the definition of the physical space is vague (radio waves can travel beyond walls) or the solution requires expensive hardware to measure a message's time of flight. Instead, we use infrared as a signal that walls can contain. We develop key derivation protocols that ensure only a receiver in the physical room with access to the signal can derive the key. We implement a system that uses the laptop's webcam to record the infrared signal, derive a key, and decrypt sensitive files in less than 5 seconds.

Source Verification for V2V Networks A number of V2V applications use information about nearby vehicles to prevent accidents or reduce fuel consumption. However, false information about the positioning of vehicles can cause erroneous behavior, including accidents that would not occur in the absence of V2V. As such, we need a way to verify which vehicle sent a message and that the message accurately describes the physical state of that vehicle. We propose using LED lights on vehicles to broadcast the certificate a vehicle is currently using. Receivers can use onboard cameras to film the encoding of the certificate and estimate the relative location of the vehicle. This visual channel allows a receiver to associate a physical vehicle at a known location with the cryptographic credentials used to sign a location claim. Our simulations indicate that even with a pessimistic visual channel, visual verification of V2V senders provides sufficient verification capabilities to support the relevant applications.

Acknowledgements

Though this work bears my name, it is the result of guidance and support from many. In a small number of words, I will try to express my gratitude. However, I apologize in advance for I will surely omit someone or fail to put into words how much many people mean to me.

I owe a debt to many faculty and mentors that have influenced me over the past few years. Adrian, I owe you many thanks for the countless hours spent helping me develop the skills associated with conducting research, technical writing, and presenting material. I also appreciate our discussions that took place anywhere between lunches on Craig Street and taxi rides through rural Taiwan, which taught me many lessons beyond the aforementioned skills. Fan Bai, Lujo Bauer, Virgil Gligor, Bo-Yin Yang, and Cynthia McLain: even in our limited interactions, I reaped large rewards from your feedback, advice, and support. I credit Chen Ding and Michael Huang from the University of Rochester for whetting my appetite for research and providing guidance as I finished my undergraduate degree. Finally, I would like to thank Chris Brown from U of R and David Brumley of CMU for their wise words that reminded me to never settle for good enough and that the onus of success falls on one's own shoulders.

The success of much of my work is due to the assistance from many collaborators. Thanks to the insightful feedback and well-balanced guidance of Jon, Max, and Cynthia, we were able to produce two of my favorite papers. I commend Yue-Hsun Lin and Chia-Hsin Chen for their ability to produce implementations under short deadlines with little guidance. When I had too many projects to handle, Hsu-Chun and Hyun-Jin were able to join the efforts and produce results beyond my expectations. I also owe Casey thanks for her contributions to my research and the reminders that the mean is only one of many important statistics.

A number of people have helped me maintain my sanity amongst the pressures of school and provided guidance on a range of topics. Dan, Doug, and Garrett: our homework sessions and everything that we did instead of homework are some of my fondest memories of U of R. Dave, Jonah, and

Paul also deserve my gratitude for the addition of art, rugby, and various hijinks that rounded out my time at Rochester. I thank Mike, Scott and Ginger, Jon, Kathleen, and Addie, Jim and Bonnie, Bryan and Diana, and Cynthia W. for the good times we had in Pittsburgh and wherever else our adventures took us. Thanks to yinz, I have climbed dinosaurs, barely eluded a bar fight, argued over crops, drank a steak, and learned about working on cars and homes.

Finally, the largest tribute belongs to my family. I owe an immense debt to my parents for the amazing environment in which I grew up and their continued love and support. You instilled me with curiosity, tolerated my constant questions, and supported all of my efforts to find the answers. Letting me stay up past my bedtime to learn about negative numbers is my earliest memory, which embodies that environment. Megan, your accomplishments and work ethic have always served as inspirations. To “my boys”, Caleb and Sam: though you two are unable to read this, thank you for dragging me off of the couch and out of the house for walks and for keeping me company all of those late nights I spent working. I am unable to say thank you enough times to Casey for agreeing to become my wife and continuing to tolerate my constantly divided attention. You are my most honest critic, my strongest supporter, my favorite co-pilot, and my best friend. However, you will never be my lookout or getaway driver. Connor, in less than a year and a half, without ever speaking a word that exists in the English language, you have taught me more about life and happiness than anyone else. I hope one day you will be as proud of me, as I am proud of you.

Though many people have contributed to the creation of this document, any and all defects are my own.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The decreasing size, cost, and power consumption of computing and wireless communication has allowed the addition of sensors, actuators, CPUs, and antennae to a variety of devices that traditionally operated without communication or computation. These additions have enabled a number of different cyber physical systems “with integrated computational and physical capabilities” [2]. In addition, to sharing digital information, these systems may sense properties of the physical world and control actuators to change the physical state of a system. Examples of cyber physical systems (CPSs) include:

- Smart Grids to enable more efficient generation and distribution of energy [2].
- Smartphones to enable games, social networking, and financial exchanges with collocated parties [11, 69].
- Location-based access control for digital resources to provide more intuitive access policies [28].
- Vehicle-to-Vehicle (V2V) communication to enable applications to reduce the number of accidents and improve efficiency with more intelligent routing or platooning [3, 6].
- Advanced healthcare systems to enable improved patient monitoring, medication delivery, and prostheses [2].

For these applications, correct data is necessary for correct operation. One extreme example is the warning lamp at Three Mile Island which incorrectly indicated the state of a valve supplying coolant to a nuclear reactor, resulting in a nuclear accident [107]. When describing the physical world, often the physical source or destination of the information is very important, often almost as important as the information itself. For example, the message "valve closed" means very different things depending on the physical configuration of the system and the flow that valve controls. As such, one of the first steps towards performing as intended is allowing participants in the system to verify the physical source or destination of messages.

In a system with a fixed topology and a central authority, public key cryptography allows participants to determine where a message came from and who is able to receive it. Specifically, the authority generates a certificate that binds a public key to a name that describes a device's location and or role in the system. Once the certificates are known, devices in the system can determine the source and destination of messages. Digital signatures allow receivers to determine the physical source of a message. A sender can use a public key to encrypt a message, defining which physical endpoint can decrypt and receive the information.

However, in systems where participants move, a certificate that provides a static description is unable to identify a digital entity as a physical entity at a specific location in the system. Instead, we need a different way to determine the physical embodiment of an endpoint. We cannot rely on the information in a wireless message[1] to determine the physical embodiment of an endpoint since a malicious party can lie about its location. Instead, we need additional information to verify the endpoint of a message. Distance bounding [9] and location verification [18, 19, 83, 97] can help verify the endpoints of communication by verifying the maximum distance between two devices or the location of a device. However, these schemes require

[1]In this work, we use the term wireless to mean communication using radio waves. We will use more specific terms when we discuss any other technology (e.g., infrared, ultrasonic, or visual light) that supports communication without a physical link (i.e., wire) or radio waves.

high accuracy timers (on the order of nanoseconds) to measure the time-of-flight of wireless messages or several monitoring devices to triangulate the location of a device. Instead, we examine how to use devices' actuators and sensing capabilities to enable physical interactions between devices in the system. These interactions allow the establishment of communication channels with properties which allow the verification of the physical source and/or destination of messages on that channel. Once we know the physical endpoints of communication on this secondary channel, we can use cryptographic techniques to verify the endpoints of digital communication on the wireless channel with existing hardware on devices without a network of supporting infrastructure.

Thesis Statement *Physical interactions can significantly enhance security by allowing collocated devices to verify the source and/or destination of data in a number of scenarios where the name of an entity—a cryptographically verifiable item—is less important than its physical embodiment.*

To validate this thesis, we examine three different cyber physical systems to determine how endpoint verification helps improve security, present how to perform endpoint verification in each scenario, and evaluate the proposed systems. In each of the systems, we leverage a communication channel, other than the wireless channel, which exhibits properties that enable endpoint verification. In Chapter 6, we summarize the properties a communication channel must fulfill to enable endpoint verification with our protocols and contrast our approaches with related work that use distance bounding or location verification. General related work is discussed in Chapter 7. We now provide an overview of the technical chapters of this thesis that are related to each scenario.

1.1 Information Exchange Between Smartphones

The exchange of information between two smartphones represents a scenario where identifying the physical source and destination of the exchange is important. Users with smartphones often want to share information with people they meet. This exchange can enable games, social networking, or even support the exchange of money via third parties like PayPal [69]. When performing this exchange, the users want to ensure that an unwanted third party is unable to inject other information as part of a Man-in-the-Middle (MitM) attack. As such, users want to verify that their phones are the sources and destinations in the exchange. We analyze the popular exchange protocol Bump to determine if it accurately verifies the endpoints in an exchange and to deduce what factors contribute to the popularity of an exchange protocol. After finding and demonstrating a MitM attack against Bump, we propose Shake on it (SHOT), a secure replacement for Bump, that exhibits many of the properties that made Bump popular.

Bump uses the physical act of bumping the phones together to indicate the desired endpoints in an exchange. We identify a vulnerability in Bump that allows us to perform a MitM attack against the protocol. We analyze what is necessary to make Bump secure and derive some guidelines for designers to help increase the security and adoption rate of an exchange protocol.

Based on these guidelines, we design SHOT. SHOT is the first exchange protocol to use the phones' vibrators and accelerometers as part of a human-observable exchange that does not require secrecy. Specifically, our protocol verifies that the two phones pressed against each other and vibrating are the phones involved in the exchange.

In Chapter 2, we present our analysis of Bump and the design, implementation, and evaluation of SHOT. We discuss prior exchange protocols for smartphones and other devices in Section 7.1.

1.2 Location-Based Access Control

Location-based access control requires verification that the receiver of a resource is within a specific physical space. We propose location-based access control as a way to protect sensitive information on a lost laptop without requiring user effort (e.g., no password entry, hardware tokens, or biometric entry). Only a receiver in the correct physical space is able to derive the cryptographic key needed to access sensitive files on the laptop.

When a laptop is lost, any information on that laptop may fall into the wrong hands. Encryption of data can protect that information. However, users often feel key management is a burden and use weak passwords or perform other actions which reduce their burden, but subvert the protection provided by encryption. As such, we need a solution that can protect a user's sensitive data, without requiring work from the user.

We propose an approach where the key needed to decrypt sensitive data is only available when the laptop is in a trusted physical space. As such, users can access the majority of their information anywhere, but must be in a specific location to acquire the decryption key needed to access the sensitive data. We call this Mobile User Location-specific Encryption (MULE). If the laptop is lost, whoever finds the laptop must break into the trusted space to access the sensitive data. Given users often keep hard copies of sensitive data (e.g., print outs or flash drives with backups) in the trusted location, access to the trusted space allows access to the data, with or without the laptop.

The challenge here is designing a mechanism that allows the laptop to determine when it is in a trusted physical space, without requiring the user to add more hardware to the laptop or perform some action. Our solution relies on a small computing device that remains in the trusted physical space and performs some calculations as part of the key derivation process. We call this device the Trusted Location Device (TLD). The TLD transmits a secret on a location-limited channel such that only devices in the trusted space can receive the information. When a user wants to access sensitive files, the laptop uses the information from the location-limited channel to generate a key derivation request, which is sent to the TLD. The TLD uses this request to complete the key derivation process. We design the key derivation protocol such that the TLD calculations only produce the correct decryption key if the requesting device has access to the location-limited channel (i.e., the device is in the trusted space). A request from a device outside of the trusted location will produce a random value that will fail to correctly decrypt the sensitive data with high probability. Under this design, reception of the file decryption key allows the laptop to verify it is physically located in the trusted location, enabling location-based access control in order to improve the security of sensitive files on the laptop, without requiring any user effort.

In Chapter 3, we describe the design, implementation, and evaluation of MULE.

1.3 Vehicle-to-Vehicle Applications

In a number of vehicle-to-vehicle (V2V) applications, the location of the physical source of a message is often just as important as the message. For example, receiving the message "braking hard" means different things if the source is a vehicle in front or behind the receiver. If the sender is in front of the receiver, the receiver should brake to avoid a rear-end collision. If the sender is behind the receiver, the receiver may even want to accelerate to avoid being rear-ended. A message can contain the sender's location, but malicious senders can lie about their location. Another challenge is how to detect Sybil [31] vehicles (one physical vehicle using radio messages to claim to be more than one vehicle on the road). If undetected, Sybil vehicles can generate false reports of congestion and negatively impact the accuracy of schemes that try to detect abuses of V2V, which rely on agreement from multiple vehicles [51, 75]. As such, vehicle-to-vehicle applications require a way to verify the relative location of a sender and that it is a physical vehicle. In Chapter 4, we discuss background material on V2V networks and why existing security solutions fail to provide these properties.

We leverage computer vision and blinking LEDs to associate vehicles with wireless messages. In addition to digitally signing messages and broadcasting certificates, each vehicle uses LEDs attached to the front and back of the vehicle to transmit an encoding of its current public key. A receiver will film this blinking sequence to visually verify the location of a vehicle and determine the vehicle's public key. Based on the information from the visual channel, the receiver can determine what cryptographic key a given vehicle is using and if the location in the corresponding wireless message matches the physical location of the sender. Once a receiver can map cryptographic keys to physical vehicles, Sybil attack detection is possible, since the source of the wireless identities can only be in one physical space at a given time.

We provide background information on vehicle-to-vehicle networks in Chapter 4. In Chapter 5, we describe the design and simulation of our vision-based vehicle verification technique.

Chapter 2

Secure Exchange of Information Between Smartphones

As the functionality and computing power of smartphones increase, users are leveraging these devices to perform more of their computing tasks. In addition to traditional tasks such as email, gaming, banking, and maintaining a schedule, the mobility and ubiquity of smartphones allows people to use these devices to establish ad hoc associations with people they meet. For example, people may exchange phone numbers, email addresses, and social network identities or even use their phones to enable the exchange of funds via an online service (e.g., PayPal). During these exchanges, phones typically use the wireless channel to perform the majority of the communication. The wireless channel makes exchanges easier for users since they do not have to carry cables to connect the phones. However, users are unable to observe the endpoints of wireless communication and without a secure protocol a malicious party can insert themselves into the exchange as part of a man-in-the-middle (MitM) attack.

Exchange protocols for smartphones require varying levels of user involvement and provide different security guarantees. The widely deployed Bump [11] and Bluetooth pairing [49] protocols represent distant points in the spectrum of user involvement and security guarantees. Bump requires users to only perform a simple gesture, but, as we demonstrate, is vulnerable to MitM attacks under realistic conditions. Bluetooth pairing [49] is resistant to such attacks, but requires significant user involvement to correctly compare a checksum across phones.

In the first part of this chapter, we highlight the danger of relying on popular usable-but-perhaps-not-secure protocols by demonstrating a MitM attack against Bump. Bump is the most popular exchange protocol for smartphones[1] and claims to provide an excellent user experience while ensuring security.[2] Due to the simplicity of the operation, Bump is used in a number of applications. For example, PayPal's mobile app uses Bump to exchange account information and send money to nearby friends[69]. Bump's security is based on the Bump server's ability to determine what phones are physically interacting based on the time, location, and the force with which the two phones were physically bumped together. Bump provides a very nice user experience, but is vulnerable to attack. An attacker may be able to observe when and where users bump their phones together and estimate the force of the bump. Without any secret information to identify the pair of phones to the server, an attacker can submit similar information about a bump to the server. When presented with similar information, the server may transfer data between the wrong phones. We demonstrate how an attacker under realistic conditions can use this approach to launch a MitM attack against Bump. In addition to Bump, other works have also suggested using accelerometers or vibrators and accelerometers to facilitate an information exchange between phones [21, 45, 56, 60, 26, 86]. However, our attack on Bump and prior work [43] demonstrate that a physically present attacker may be able to violate the secrecy of this channel, requiring a new

[1]Popularity is based on data from `http://techcrunch.com/2011/01/19/iphone-ipad-top-app-downloads/` and `http://www.androidapps.com/` which use download count to determine popularity. According to the Bump blog, over 25 million users have installed Bump.

[2]According to Bump's webpage, "With Bump *you* are in control of deciding with whom you share your information. You don't have to worry about anyone being able to get at your information unless you physically bump your phone with theirs." [11]

approach to ensure security.

In the second part of this chapter, we demonstrate how phones can leverage accelerometer readings to assist in the secure exchange of information while maintaining the limited user involvement offered by Bump. We propose Shake on it (SHOT), a protocol designed specifically for smartphones that requires little user interaction. The novel idea is to use phones' vibration function and accelerometers as an authentic, but not secret, human-observable communication channel. The phones leverage data exchanged on the vibration-to-accelerometer channel to verify data exchanged over the wireless channel. Only when the two phones are in physical contact can they communicate via the vibration-to-accelerometer channel, providing demonstrative identification to the users of the devices exchanging information [5]. If a remote party (one not in physical contact) tries to inject information on the channel, users will notice the additional vibrations (a potential attack) and stop the exchange. Given attackers can eavesdrop on this channel [43], the phones use symmetric and asymmetric cryptography to authenticate exchanged information and detect active attacks against the wireless channel, without users having to compare any information. When two people run SHOT, they only have to perform three simple tasks: select what data is to be exchanged, hold the phones together until the phones beep to indicate completion, and cancel the exchange if they feel vibrations from devices other than the two phones. We argue that SHOT occupies a new point on the spectrum of exchange protocols, providing greater security than comparably convenient protocols, and greater convenience than comparably secure protocols.

We present an implementation of SHOT on a commodity smartphone and evaluate its performance. Our analysis and evaluation show that SHOT provides a secure exchange while requiring the same level of user effort and execution time as the popular Bump protocol.

2.1 Problem Definition

The goal of an exchange between smartphones is to provide two people (users A and B) who meet in person a user-friendly mechanism that allows the authentic exchange of information (A's information I_A and B's information I_B) using their phones (P_A and P_B). After the exchange is complete, P_A will have received I_B and P_B will have received I_A, or P_A and P_B will both detect with high probability that an error has occurred and discard the information.

A user-friendly solution is any exchange that only requires limited user involvement and can complete execution in a reasonable amount of time. The exchange should not require the users to type several bytes worth of information into either phone or compare a checksum (e.g., a string of hex digits [49, 55, 101], a series of words [38], or a graphical image [70, 57]). User studies have shown that a redesigned interface [100] or the comparison of words or images [54] reduce the number of errors, but still require non-negligible user involvement. The popularity of Bump indicates that the desired level of user involvement is a simple action, such as a physical gesture, that allows the phone to determine the other phone in the exchange. Solutions exist that require the user to take a photograph of the other phone [61], shake the two phones together [21, 45, 56, 60], or simply point the phones at each other [5], but each solution has drawbacks that negatively impact security or operation (see Chapter 7 for a discussion of related work). Given the wide acceptance of Bump, we consider the execution time of a Bump exchange as a reasonable amount of time. Depending on the platform, Bump takes between 9.4 and 37.8 seconds, from starting the application to receiving the exchanged data. Section 2.7 has more details on the execution time of Bump.

During an exchange, a malicious party M may attempt to inject its own information (I_M) or other data into the exchange such that P_A or P_B accept something other than I_B or I_A, respectively. A secure exchange allows P_A and P_B to exchange I_A and I_B or detect the insertion of any other information into the exchange with a high probability.

After discussing our assumptions, we give a detailed description of our attacker model.

2.1.1 Assumptions

We assume smartphones are equipped with the hardware, software, and connectivity needed to execute a Bump exchange.

Current smartphones are equipped with a vibrator and an accelerometer to provide silent notifications to the owner and to allow correct orientation of an image on a rotatable screen. Smartphones also allow the installation of generic software that can access the vibrator and accelerometer functionality. Smartphones also have Internet connectivity the majority of the time via the cellular network or a local WiFi access point.

2.1.2 Attacker Model

An attacker's goal during the smartphone exchange between phones P_A and P_B is to convince P_A or P_B to accept information other than I_A and I_B. We consider an attacker that may be in the same room as A and B, knows the value of I_A and I_B, and has bounded computational capabilities. We also assume any software on P_A and P_B is outside of the attacker's control.

We assume the attacker has control over the wireless channel between the two phones and is able to intercept, modify, delay, or inject messages. However, the attacker does not control human-observable channels between the two phones (i.e., the visual or vibration channel). The attacker can accurately eavesdrop on these channels. However, users can detect the attacker's attempts to insert information on a human observable channel (i.e., insertion of a third phone into the visual channel or shaking the phones to inject data onto the vibration channel).

An attacker knows a priori what information the two users want to exchange. Given most users are exchanging contact information (e.g., phone numbers or online account IDs), this information can be found online.

It is infeasible for a computationally bounded attacker to break various properties of different cryptographic primitives of the appropriate strength.

Hence, hash functions with sufficiently long outputs are one-way and second pre-image resistant (i.e., given a hash function $h()$ and a hash output $y = h(x)$, an attacker is unable to find x or another value x' such that $h(x') = y$). We also assume digital signatures are secure against selective forgery. This means that without knowledge of the private key, it is infeasible for the attacker to create a signature (σ) for an attacker selected message m such that the corresponding public key verifies the signature message pair (σ, m).

2.2 Bump Exchange Protocol

In this section, we begin by describing the Bump exchange. We then explain why the exchange is vulnerable and provide an analysis of our attack and its impact.

2.2.1 The Bump Protocol

The Bump exchange is meant to allow two phones (P_A and P_B) to exchange information. During an exchange, the users physically bump P_A and P_B together; each phone sends information about the bump and the information it wants to exchange, including a user-selected identifier (ID) and some additional data, to the bump server; the server uses the time, location, and force of the bumps to determine which phones want to exchange information and returns to each phone the other phone's information; and the user decides to save the information based on the ID in the information. Based on information provided on Bump Technology's webpage (`http://www.bu.mp`) and timing of packets sent during a Bump exchange, we were able to determine approximately how the Bump protocol works. We can divide the protocol into two main steps: Initialization and Exchange. Figure 2.1 provides a rough outline of the steps for a single phone involved in a Bump exchange.

Initialization:			
1. P_A	:	$loc \leftarrow findLoc()$	The phone determines its current location.
2. $P_A \overset{Internet}{\longleftrightarrow} S$	:	establish TLS	The phone establishes a TLS connection with the server.
Exchange:			
3. $A \overset{acc.}{\rightsquigarrow} P_A$	:	a_{bump}, t	Bumping the phone induces an accelerometer reading a_{bump} at t.
4. $P_A \xrightarrow{TLS} S$	:	a_{bump}, t, loc, I_A	The phone sends the acceleration, time, and location of the bump and the user's info to the server.
5. S	:	$info = match(a_{bump}, t, loc)$ $resp = info.ID$	The server checks its database of recent bump requests for similar accelerometer readings in a similar location, and returns the ID. $resp = \emptyset$ when no match is found. $resp =$ **"Bump Again"** when > 1 match is found.
6. $S \xrightarrow{TLS} P_A$	:	$resp$	The server returns the ID of the match (or the error code).
7. $P_A \xrightarrow{Screen} A$	:	If (($resp \neq \emptyset$) and ($resp \neq$ **"Bump Again"**)) **" Connect with** $resp$**? Yes/No"**	After a successful match, the phone asks the user if they want to save the information.

8. The phone then sends the user's selection to the server. If both phones return "yes", the server returns the other phone's data. If either phone returns "no", the server tells the other phone the exchange was cancelled.

Figure 2.1: Operations During a Bump Exchange Where User A Uses Phone P_A to Exchange Information I_A Using the Server S.

Initialization involves each phone determining its location and connecting to the server. When Bump begins, the phone uses one of many techniques to determine its current location (Step 1). Popular smartphones can determine their location outdoors using GPS. When indoors, smartphones use nearby cell tower identifiers or WiFi access point information to estimate the current location. Even in the best case, these location estimations are within an accuracy of several meters. The phone then connects to the server via TLS [30] (Step 2). TLS provides secrecy and authenticity of communication with the server.[3] During the TLS handshake, the phone verifies it is communicating with the Bump server and not an attacker impersonating the server. Once established, the communication with the server is encrypted and authenticated to prevent an attacker from intercepting or modifying any information the phone exchanges with the server. Once initialization is complete, the phone knows approximately where it is and that it is connected to the correct Bump server.

After initialization is complete, the users bump phones together to start the actual exchange. During the physical bump, the phone's accelerometer measures the approximate force of the bump (Step 3). Once a bump has occurred, the phone sends the acceleration associated with the bump, when the bump occurred, the phone's current location, and whatever information is to be exchanged to the server, including an ID and the data to be exchanged (Step 4). Using the received information, the server tries to match the bump with other recent bumps based on force, time, and location (Step 5). Because of inaccuracies in the phone's location finding, clock, and accelerometer measurements, the server uses approximate matching and considers any value that is similar a valid match. If no other database entry has similar bump properties, the server responds indicating their was no match found (we use $\emptyset$ to indicate this). If more than one database entry is similar, the server asks the phone to bump again. The assumption is that if,

[3]The Bump FAQ claims the new version of Bump does not use TLS, but is "encrypted end-to-end". However, analysis of intercepted traffic between an iPhone and the Bump server indicates that TLS is still in use. Even if TLS is "phased-out" the server stills needs to perform the matching task, which we attack to subvert Bump.

for example, four phones were bumped at approximately the same time and location with similar forces, a second set of bumps will yield information that makes the pairs distinct. If a single match exists, the server returns the identifier from the matching request (Step 6). The server tags the database entries associated with the match so that the same IDs are not returned to other requests (not shown in Fig. 2.1 for simplicity). The phone asks the user to confirm if she wants to exchange information with the ID the server returned (Step 7). It is important to note that the ID used during this confirmation is a value which an attacker can select to mimic a legitimate party in the exchange (i.e., the server will return the same ID when the exchange is correct and when under attack). After both users in the pair click "Yes", the server returns the data associated with the pair, which the phones then save. If either user clicks "No", the server never releases the full data (only the ID is always sent) and the exchange is cancelled.

2.2.2 Vulnerabilities in Bump and Their Exploitation

Bump is insecure due to three aspects: inaccuracies in the measurement of the physical aspects of the bump, the ability of a sender to control the ID used during confirmation, and server flexibility in accepting delayed bump requests. Given these three items, an attacker is able to successfully launch a MitM attack.

Sensor Inaccuracies. Given limited sensor accuracy, the phone is unable to know its exact location or how hard the phone was bumped. Given inaccuracies in different phones' clocks, the server is unable to know exactly when a bump occurred. Without accurate information, the server must use approximate matching, which can associate bumps that occurred several meters apart, with different forces, and at slightly different times. We were able to confirm that the server will match bumps that differed in each of these three aspects. Unless every user is willing to pay for more accurate sensors for their phones, the server will be forced to use approximate matching and maintain this vulnerability.

Without a way to decrypt TLS traffic, we were unable to determine the range of acceleration, location, or time values the server would consider as similar. However, during testing, we were able to bump 2 phones across the room from each other (∼5 meters apart) with different forces (e.g., slam one against a table and tap the other) at different times, and still have the server match the two phones. This vulnerability allows an attacker to block P_B's bump request and have the server match an attacker's request with P_A. However, a full MitM attack is not achieved because both the confirmation question ("Communicate with [ID]?") and the failure of P_B's request allow the users to detect the attack.

Spoofable ID. Circumventing the confirmation question such that a MitM attack goes undetected in Bump is simple. A participant controls the ID sent to the other phone during an exchange. As such, the attacker can use the same ID as the victim it is impersonating. This is still a valid attack because the underlying data is different (e.g., the email or public key the users accept are the attacker's). Bump could require users to compare all of the exchanged data or a hash of the data to remove this vulnerability. However, that approach would negatively impact usability, resulting in a protocol similar to the Bluetooth exchange. With spoofable IDs, all that remains is ensuring that both P_A and P_B are able to complete exchanges for the MitM to go undetected.

Acceptance of Delayed Requests. Bump accepts stale requests, but will try to find a match as soon as possible to provide a better user experience. Different networks can cause different delays, such that the request from P_A arrives before the request from P_B, which arrives long after when P_B claims to have bumped. Given this potential delay, if no matching database entry exists, the server will wait some period of time before responding to P_A's request, rather than responding you were the only one to bump. The server also considers P_B's request with the older bump time valid because the delay may be network related. However, to improve response times, the server will associate a phone with a similar bump, rather than waiting until the correct request arrives.

This acceptance of delayed requests and preference for fast responses means an attacker can inject her own information into a legitimate exchange and allow both victims requests to complete. All the attacker has to do is delay P_B's request until the server associates P_A's request with an attacker submitted request. After the server associates P_A with the attacker, the attacker can forward P_B's delayed request and a second attacker generated request that allows P_B's exchange to complete successfully.

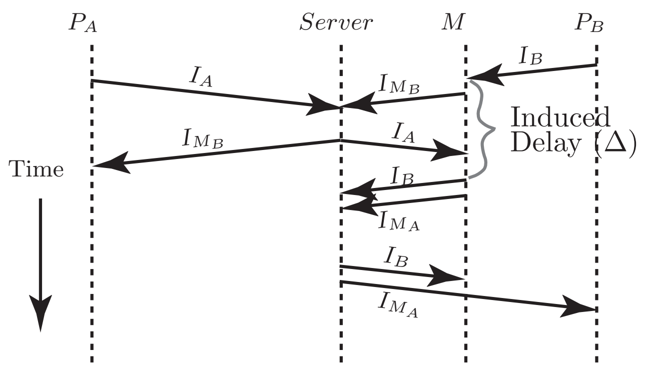


Figure 2.2: Timing of Packet Exchanges During the Attack on Bump.
(Accelerometer and location information omitted for clarity.)

The Attack. An attacker with the ability to submit similar bumps to the server, spoof legitimate users' IDs, and control the wireless channel can successfully complete the following MitM attack against Bump. Figure 2.2 sketches the attack. After A and B bump phones, the attacker delays P_B's request containing I_B and sends its own request impersonating P_B with information I_{M_B} and similar bump values. The server associates the attacker's request with P_A's request since no other similar entries exist in the database. Next, the attacker forwards P_B's delayed request and sends a request impersonating P_A with information I_{M_A} and the same bump values. In response, the server associates those two requests. Since the phones will present the correct ID's to the users, there will be zero indication that an attack has occurred and that the underlying data is incorrect. Delay Δ, the duration of time by which the attacker delays the server's receipt of P_B's request, is an important parameter in the attack. If Δ is too short, the server will correctly associate the legitimate users' bumps or claim all of the bumps were too similar and ask the phones to bump again. If Δ is too large

(> 3 seconds), the server generates an error and returns $\emptyset$ in response to the delayed request.

In the next subsection, we describe how such an attack can happen in practice and analyze the probability of a successful attack based on different values of Δ.

2.2.3 Attack Implementation and Analysis

To successfully perform a man-in-the-middle attack against Bump an attacker must be able to observe the bump and delay packets. A maliciously controlled access point (AP) provides control of packets. In our attack setup, we use a MacBook Pro with OS X 10.5 and Dummynet [20] as an AP. The laptop connects to the Internet via the ethernet port and can forward packets from the WiFi network to the Internet. In a real attack, the attacker could trick users into associating with the attacker's AP by selecting a common SSID (e.g., "linksys"). By default, a phone will use a WiFi network with a known SSID, rather than the cellular network.

To evaluate the attack, we use real phones with three different people to bump the two sets of phones. We use 4 different iPhone[4] 3G smartphones running iOS version 4.2.1 and Bump 2.4.0 (1 each for P_A and P_B and 1 for each of the attacker's roles, P_{M_A} and P_{M_B}). Two humans bump phones P_A and P_B together while another human plays the role of the attacker and tries to bump phones P_{M_A} and P_{M_B} together at roughly the same time and with the same force. This is meant to simulate an attacker observing victims across the room using Bump to exchange phone numbers at a bar or exchange money to reimburse one user for the other user's share of a tab. We configure Dummynet to delay P_B's and P_{M_A}'s requests by Δ as part of the attack (see Figure 2.2).

[4]We use iPhones for our attack because it supports a more recent and reliable version of Bump. iPhones are able to run Bump version 2.4, while phones with Android are limited to version 1.3.2. If Bump Technologies secured the application between major revisions, attacking version 1 would be futile. The Android version of Bump was also much less stable during our testing. Over the course of 20 exchanges without attacker interference, the iPhone version was able to exchange data 19 times. However, the Android version was only able to exchange data 7 out of 20 times.

To evaluate the attack on Bump, we vary the induced delay (Δ) from 0 to 3 seconds in increments of 0.5 seconds and run 10 exchanges for each setting. During an exchange, a phone can experience one of three potential outcomes:

SE Successful Exchange: The server returns a potential match and asks if the user wants to communicate with that ID.

BA Bump Again: The server finds multiple similar entries and asks the phone to bump again.

OO Only One: The server is unable to find a similar entry in the database. The server also returns this response if the induced delay is large.

Based on the individual phones' outcomes, we classify the result of the attack as one of the following:

- **Successful Exchange:** P_A receives B's information (**SE**) and P_B receives A's information (**SE**).
- **Successful Attack:** P_A receives the impersonation of B (I_{M_B}) and P_B receives the impersonation of A (I_{M_A}). Both phones have outcome **SE**, but receive attacker's information.
- **Detectable Attack:** P_A receives the impersonation of B (I_{M_B}), but P_B receives bump again or only one. **SE** for A and **BA** or **OO** for B.
- **Other:** P_A and P_B receive Bump Again or Only One. Neither phone experiences outcome **SE**.

We present a summary of our results in Figure 2.3. These results show that a MitM attack against Bump is feasible. The optimal induced delay (Δ) is around 2 seconds. With that delay, 70% of the attacks succeed. With less delay, the server detects multiple similar requests and returns "Bump Again". With more delay, the server detects the long delay for P_B's request and instead of associating the request with I_{M_A}, returns an "Only One" response. For delays longer than 3 seconds, the attack is detectable; the server consistently pairs I_A and I_{M_B}, but returns "Only One" in response to I_B and I_{M_A}.

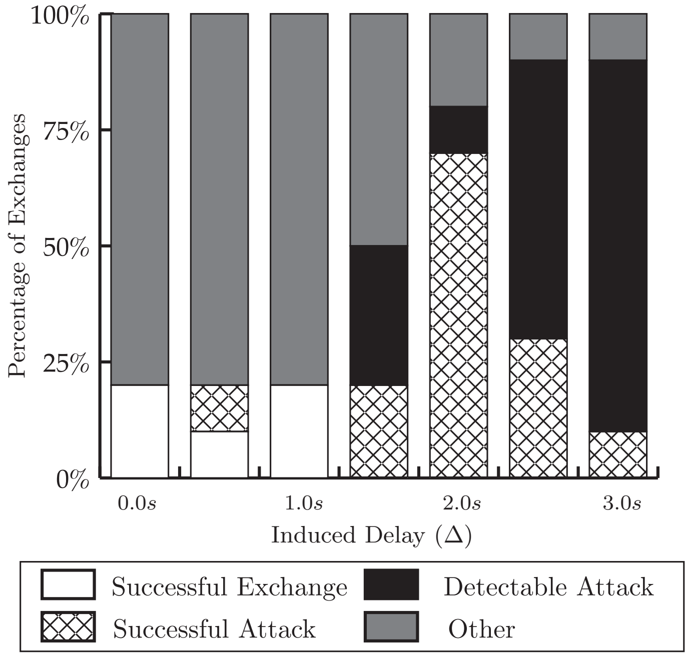


Figure 2.3: Attack Results on Bump with Varying Values of Δ.

2.2.4 Attack Impact

A MitM attack against Bump has varying levels of impact depending on the application using Bump, including user annoyance, interception and/or impersonation during personal communication, and theft of money.

Some attacks merely result in user annoyance. For example, smartphone games such as BumpFour (a version of Connect Four) leverage Bump to establish a connection between opponents. In the event of a MitM attack, a malicious party can insert themselves between the two phones and play against the victims.

When Bump is used to exchange contact information, a MitM attack threatens the security of any communication that relies on the exchanged information. The standalone Bump application, which we attack in Section 2.2.3, exchanges contact information and creates a new address book entry with the received data. If the victims send text messages, phone calls, emails, or other online communication using the information received during a MitM attack, the attacker, rather than the intended user, will receive the information. In addition, the attacker can impersonate one victim if the recipient relies on the information received from Bump (e.g., an email address) to identify the source.

An attacker can exploit Bump to steal money. The PayPal smartphone application allows users to transfer money and uses Bump to identify the transfer endpoints. However, a MitM attack against PayPal does not allow the attacker to empty a user's account. When using Bump to transfer money, the party sending the money selects the amount (or at least confirms the amount) to send. This limits the attacker to stealing only the amount one user meant to send. In addition, PayPal uses an account's email addresses as the ID. Given most users will simply trust the Bump exchange if it does not fail, an attacker could insert their own email address and steal the money. However, diligent users may notice the wrong email address and cancel the exchange. An attacker can create a similar looking email using "typejacking" to effectively evade detection by all but the most diligent of users [29]. It is important to note that the onus of detection in such a scenario is on the users, and Bump itself provides no protection on its own.

Exploitation of Bump provides an attacker with a range of capabilities. While some attacks only allow the attacker to annoy legitimate users, other instances of the attack enable interception of personal communication or money.

In this section, we have analyzed the Bump exchange, demonstrated how an attacker can successfully launch a man-in-the-middle attack against the protocol, and discussed the implications of the attack.

2.3 Exchange Guidelines

This section discusses some of the lessons learned and problems encountered in Bump. Based on these guidelines and challenges, we developed the Shake on it (SHOT) exchange, described in Section 2.4.

Server-based Communication: Smartphones are connected to the Internet the majority of the time via the cellular network or WiFi. By using a server to exchange information, Bump circumvents some challenges associated with trying to establish local communication. For example, iPhones only allow Bluetooth connections to other iPhones or computers.[5] Ad hoc communication over WiFi is another option, but requires users of Android-based phones to subvert the OS to enable phone-to-phone WiFi broadcast.[6] A server allows smartphones from different vendors to communicate quickly and easily, without requiring the owner to modify the operating system.

One drawback to using a server to communicate is that the server needs some information to know which phones are trying to exchange data. With wireless communication, the phones can assume any broadcast data came from the other phone. Given all phones using a protocol share the same server, the server needs a way to differentiate each pair of phones. As such, before the phones communicate via the server, the phones must agree on a value we call the "pair identifier" which allows the server to route traffic from one phone to the other.

Accelerometers as an Authentic Channel: A number of works use a bump, shake, or gesture to intuitively indicate which phones are to exchange data [11, 21, 45, 56, 60]. The accelerometer reading provides a way to convert the physical interaction into a label which identifies potential endpoints to a server and/or to derive a secret used to detect a MitM attack. This human observable communication also provides the users demonstrative identification of the endpoints of the exchange.

Unfortunately, a physically present attacker can observe the movements and leverage real-time motion tracking [58] and high speed cameras to quantify the accelerations during the bump, shake, or gesture. Once the accelerations are known, the information the phones share lacks the secrecy needed to secure communication. Rather than assuming secrecy, we need a protocol that can provide security and usability with only authenticity.

Attack Detection on the Phones: Given attackers can observe any

[5] `http://discussions.apple.com/thread.jspa?threadID=1460770`
[6] `http://code.google.com/p/android/issues/detail?id=82`

information received by the accelerometer, the phones are unable to present any information to the server that allows the server to isolate the correct endpoints in an exchange. Without such a mechanism, the phones may receive the wrong information, but are unable to detect the error. Increasing user involvement by comparing the received data would solve the problem, but reduces usability. Instead, we want a protocol that allows the phones to detect reception of the wrong information from the server while limiting the user involvement to something as simple as putting the phones together.

Based on these observations, we need a protocol that begins by establishing a pair identifier. The phones can use the pair identifier to exchange information using the server and use authentic information exchanged via accelerometers to verify the correct phones and data were involved in the exchange, without requiring secrecy or involving users in the verification.

2.4 The Shake on it (SHOT) Exchange

In this section, we describe SHOT, a secure exchange protocol for smartphones that provides a user experience similar to Bump. We leverage the smartphone's vibration function to transmit an authentic phone-selected message from one phone to the other phone's accelerometer. This allows the phones to bootstrap communicate via an untrusted server and verify received data without involving the user. We begin with a discussion to explain why we made certain design decisions and provide an overview of the protocol to describe the high level goals of SHOT. Section 2.4.2 contains a detailed explanation of SHOT. We conclude this section with a security analysis of SHOT.

The version of SHOT presented here only ensures the integrity of the exchanged information. If secrecy is desired, the two parties can use SHOT to authentically exchange public keys, and use those keys to establish a shared symmetric key for encryption of information.

2.4.1 SHOT Overview

Based on the guidelines in Section 2.3, we want a protocol that takes advantage of smartphones' accelerometers for demonstrative identification and authentic communication. For SHOT, we take this one step further and leverage the smartphone's vibration function to send a phone selected message to the other phone's accelerometer. With this capability, the naive approach would be to have users hold the phones P_A and P_B together. Once together, P_A would vibrate I_A to P_B and P_B would vibrate I_B to P_A. Unfortunately, the vibration to accelerometer channel is too slow for this to be user-friendly (see Section 2.6 for more details on the bit rate).

Several works explain how two parties can exchange data over an insecure medium (e.g., our server) and use an authentic channel (e.g., human comparison or vibration) to verify the exchange, using a checksum derived from the exchanged data [49, 38, 55, 70, 101]. However, these protocols require the phones to exchange data before they can calculate the checksum. If using a server to communicate, the phones need a pair identifier to communicate via the server, before the phones use the vibration channel to exchange the checksum. A protocol could use two long vibrations (one at the beginning to exchange a pair identifier and another to exchange the checksum), but vibrating all of that information would be slow. Users could copy a pair identifier from one phone to the other, but that is cumbersome.

Instead SHOT starts by using the vibration channel to send a message that performs two functions: 1) acts as a pair identifier and 2) acts as a pre-authenticator to verify the authenticity of data exchanged from one phone to the other. Much like Talking to Strangers [5] and Seeing-Is-Believing (SiB) [61], SHOT begins with the exchange of a pre-authenticator over the authentic channel, which allows verification of subsequent exchanges over the insecure channel. Unlike these protocols, SHOT provides demonstrative identification and secure exchange with a single pre-authenticator. After P_A vibrates the pair identifier, the phones exchange data using the server and confirm the exchange in the following fashion:

1. P_B uses the pre-authenticator/pair id to verify a public key from the

server belongs to the phone pressed against P_B.

2. P_A digitally signs I_A and a copy of the other phone's information, which P_A received from the server.
3. P_B verifies the signature from the server is a valid signature over the other phones information and its own information (I_B). If the signature is correct, P_B vibrates back a positive response to indicate that the other phone signed a copy of the information P_B received and sent.
4. P_A uses the vibrated response to determine if it signed its own information and the vibrating phone's information or if an attack occurred.

SHOT does use two vibrated messages. However, one vibration in each direction is needed to allow each phone to verify the information was exchanged with the phone physically pressed against itself. If only one vibrated message was used, either more human interaction is needed, or there is no way for the first phone to verify the other phone received its information. The second vibration is a yes/no response that requires 1 bit of data and can be quickly transmitted in order to maintain a short execution time.

2.4.2 SHOT Exchange

SHOT provides a user friendly way to exchange information between smartphones by leveraging the authentic nature of the vibration to accelerometer channel, communication via a server, and the phones' ample computational capabilities. To minimize computation and communication over the vibration channel each phone has a different role. For the remainder of this chapter, rather than P_A and P_B, we call one phone the Endorser (P_E) and the other phone the Verifier (P_V). After users select what data to exchange and agree on or are assigned each phone's role, SHOT consists of 4 phases to securely exchange data between two smartphones: 1) exchange of the pair identifier, 2) exchange via the server, 3) signing of the data, and 4) confirmation of the data. Figure 2.4 contains the steps associated with the SHOT exchange.

① **Exchange of the Pair Identifier** During the initial phase, the phones exchange a pair identifier which bootstraps communication through the

```
① Exchange of the Pair Identifier:
1.  $P_E$                                   : $h = Hash(K^+_{P_E})$
2.  $P_E \overset{acc.}{\rightsquigarrow} P_V$   : $h$
② Exchange Via the Server
3.  $P_E \rightarrow S$                     : $h, K^+_{P_E}, I_E$
4.  $P_V \rightarrow S$                     : $h, I_V$
5.  $S \rightarrow P_V$                     : $h', K^+_{P_E}', I_E', I_V'$
6.  $S \rightarrow P_E$                     : $h', K^+_{P_E}', I_E', I_V'$
③ Signing of the Data:
7.  $P_E$                                   : $\sigma = Sign(K^-_{P_E}, I_E || I_V')$
8.  $P_E \rightarrow S$                     : $h, \sigma$
9.  $S \rightarrow P_V$                     : $h', \sigma'$
④ Confirmation of the Data:
10. $P_V$                                   : if($h == Hash(K^+_{P_E}')$ and
11.                                             $Verify(\sigma', K^+_{P_E}', I_E' || I_V)$)
                                                  $result$ = "yes"
                                                  save($I_E'$)
                                              else $result$ = "no"
12. $P_V \overset{acc.}{\rightsquigarrow} P_E$   : $result$
13. $P_E$                                   : if($result$ == "yes")
                                                save($I_V'$)
14. $P_s, P_V$                              : Sound Tone
```

Figure 2.4: SHOT Exchange Between P_E and P_V Utilizing the Server S. (X' is used to indicate a potentially modified value of X that has beentransfered over the attacker control wireless medium.)

server. In SHOT, this identifier also is a pre-authenticator that allows P_V to verify P_E's public key ($K^+_{P_E}$) later in the protocol. When the protocol begins, P_E calculates a shortened hash of its public key (Step 1). We truncate a hash output to 80 bits to balance security and time needed to transmit the value over the vibration channel. After the users physically place their phones together, P_E vibrates this truncated hash (Step 2). This vibration demonstrates to P_V that the other phone in the exchange has a public key that hashes to this value. Given that this hash must be unique to P_E for security, we can also use this hash as the pair identifier to help establish communication through the server.

② **Exchange via the Server** Once the phones know how to identify the pair to the server, the two phones use the server to exchange the Endorser's public key and any other information to be exchanged (P_E's info I_E and P_V's info I_V).

During each data transmission (Steps 3 & 4) and data retrieval (Steps 5 & 6), the phone begins the connection by sending the pair identifier (h). The server uses h to know how to record and retrieve the information associated with a pair of phones. If at any point a phone receives the wrong information from the server (e.g., P_V finds that $I_V \neq I_V\prime$ or $h\prime \neq h$), the phone assumes an attack has occurred and aborts the protocol.

③ **Signing of the Data** Once I_E and I_V have been exchanged, the phones start the process of verifying the data. P_E uses its private key ($K^-_{P_E}$) to sign the data it believes was exchanged so P_V can verify if the server or an attacker modified the information. Specifically, P_E signs the concatenation of its own information and the potential copy of the Verifier's information (Step 7). P_E then uses the server to send the signature to P_V (Steps 8 & 9). Once this phase is complete, neither phone has verified any information. P_V has a signature that potentially represents what information the other phone sent and received.

④ **Confirmation of the Data** The final phase has three checks to detect if the exchange was successful:

- P_V verifies the public key it received from the server belongs to the phone P_V is pressed against.
- P_V verifies that the owner of the authenticated public key received I_V and sent the information received in Step 5.
- P_E verifies that the phone it is pressed against received a valid signature. A valid signature confirms that the data P_E signed in Step 6 matches what was sent and received by both phones during the exchange.

P_V begins by verifying the hash of the received public key matches the hash it received over the vibration channel (Step 10). This verifies that the public key P_V received belongs to the phone physically pressed against P_V.

Provided P_V has the other phone's public key, P_V verifies the signature from the server (Step 11). If the public key and signature are correct, P_V knows the data it received came from the phone it is pressed against and that that phone received I_V. We could have the users look at the result on P_V and press "success" or "failure" on P_E [84], but this is work for the user. A more user-friendly approach is for P_V to use the vibration channel to send a short confirmation to P_E (Step 12). This vibration lets P_E know that the phone it is pressed against received a valid signature (verified using P_E's public key) over the exchanged data. As a final step, the phones sound a tone to indicate that the protocol has completed and users can stop holding the phones together.

In the next section we describe why SHOT is secure provided some properties of the underlying cryptographic primitives and authenticity of the vibration channel.

2.5 Security Analysis of SHOT

For SHOT to securely exchange information the following four properties are necessary:

- Only P_E can send P_V a pre-authenticator—a copy of the hash of P_E's public key ($K^+_{P_E}$).
- P_V can detect if it received a copy of $K^+_{P_E}$ or the wrong public key.
- P_E is the only entity that can generate a signature which $K^+_{P_E}$ verifies.
- Only P_V can indicate to P_E if the signature it received verifies the information P_V sent and received.

We discuss the properties needed from different cryptographic primitives to fulfill the second and third properties before discussing the authenticity of the vibration channel which is needed to fulfill the first and last properties.

2.5.1 Cryptographic Primitives

Assuming the inability of an attacker to inject information onto the vibration channel (see next subsection), a hash function that is second preimage resistant allows P_V to detect if it received a copy of P_E's public key. Given an authentic copy of P_E's public key, a signing algorithm that is secure against selective forgery allows P_V to verify P_E received P_V's information and sent the information P_V received.

If an attacker wants P_V to accept a different public key, the attacker has to find a different public key ($K^+_{P_M}$) such that the truncated hashes are the same (i.e., $Hash(K^+_{P_E}) == Hash(K^+_{P_M})$).[7] However, if the hash function is second-preimage resistant, it is infeasible for an attacker to find such a public key, even if the hash is truncated to 80-bits [84].

Without a way to convince P_V to accept a different public key, an attacker needs to produce a signature over incorrect exchange information which $K^+_{P_E}$ verifies. If the authentic public key were to verify an attacker generated signature for the message $X||I_V$, P_V would believe the other phone signed that message, indicating P_E sent X instead of I_E. However, if the signature scheme used is secure against selective forgery, it is infeasible for an attacker to produce such a signature. Without a signature which verifies these values, P_V will reject the signature and thwart the attack.

Provided the hash function is second pre-image resistant and the signature scheme is secure against selective forgery, the phones will detect any type of active attack against data exchanged over the wireless channel during the SHOT exchange. Once the attack is detected, the phones will discard the information. This fail safe operation does mean an attacker can launch a Denial-of-Service attack against SHOT. However, DoS attackers are outside of the scope of this work since the attacker could also jam the wireless channel to prevent any wireless communication.

2.5.2 Vibration as an Authentic Channel

For SHOT to be secure, only P_E and P_V are able to send information on the vibration to accelerometer channel. If an attacker were to send information on the channel, a user can detect the vibrations and abort the exchange. Without a way for attackers to send messages on the channel, only P_E is

[7]We assume the attacker does not know P_E's keys so $K^+_{P_E} \neq K^+_{P_M}$.

able to send a pre-authenticator of its public key and only P_V is able to send confirmation of a valid signature.

Users can detect when other parties are trying to send information on the vibration to accelerometer channel. The two users have in mind what phones are supposed to exchange information, and will detect if some other device is vibrating against the phones during the exchange. Holding two phones together is an intuitive way to indicate which phones should exchange information and should experience a low rate of operator error. Instead, we have to worry about situations where the attacker tries to remotely induce vibrations. However, without a way to focus those vibrations precisely on the phones (some kind of audio version of a laser), the user(s) holding the phones will notice the additional vibrations and stop the exchange. For example, consider an attacker which produces a loud tone at a low frequency in an attempt to vibrate the two phones remotely. The tone needs to be quite loud to induce vibrations which are comparable to the vibrations from another phone in direct physical contact.[8] Even if the frequency of the tone is below the human audible range, a tone with this much energy is perceptible (e.g., the users' clothes and skin will vibrate). As such, users can detect the additional vibrations and abort the exchange.

Without a way to break the security properties provided by the underlying cryptographic primitives, an attacker must find a way to inject a message on the vibration to accelerometer channel to successfully subvert a SHOT exchange. However, the users can detect attempts to remotely induce vibrations and thwart such attacks.

2.6 SHOT Implementation

In this section, we describe our implementation of SHOT for the Motorola DROID smartphone and how we use a java program on a desktop as the server to simplify non-vibrator to accelerometer communication.

[8]Audio engineers suggest using accelerometers as "contact microphones" to reduce pickup from sources that are not in direct physical contact [67].

2.6.1 SHOT for Android

Our implementation of SHOT is written for and tested on Motorola DROID smartphones with Android version 2.2. However, the system can be ported to any mobile phone with a vibration function and accelerometer. If installed on other phones effective bit rates for the vibrator to accelerometer may change with different hardware based on access to vibration functions[9] or accelerometers. In this section, we describe how we achieve communication between phones and between phones and the server, what library and parameters we use to perform cryptographic operations, and how users run the protocol.

Communication between phones using the vibrator and the accelerometer uses the `android.os.Vibrator` class to vibrate a phone, the `android.hardware.SensorManager` to access the phone's accelerometer, and a `android.hardware.SensorEventListener` to know when the accelerometer has new data. With the DROID, we use a simple on/off keying and test varying bit lengths from 60ms per bit to 100ms per bit to test the reliability of the channel (see Section 2.7.2 for comparison of reliability versus bit rates). For example with 100ms/bit, the phone vibrates for 100 ms to transmit a 1. Given the possibility of errors, we use Reed Solomon to allow the Verifier to recover from errors during reception of the pair identifier. We use Reed-Solomon encoding with 8 bit symbols and include 4 error correction symbols. This allows the Verifier to recover from errors in 2 independent bytes. When sending the confirmation, the Verifier transmits 2 bytes of 1s to indicate "yes" or sends 0xC003 (2 ones, 12 zeros, 2 ones) to indicate "no". Given the redundancy in the message, the Verifier simply transmits the two bytes without any error-correction. The Endorser only considers the confirmation a "yes" if the confirmation contains a series of several consecutive 1s.

Communication with the server uses TCP sockets. At the beginning of the protocol the phone connects to the server and maintains a single TCP

[9]For example, Apple only allows reduced access to the vibrate function if distributing an application through the App Store.

connection over the course of the entire protocol.

All of the cryptographic operations use the Bouncy Castle[10] Java package. We use SHA-1 to create the pre-authenticator and to verify the public key. The implementation uses 1024-bit RSA to sign and verify the exchanged information. One could generate a new RSA key pair for each exchange. Instead, we generate a key pair during the first execution and save the key pair for future executions to keep subsequent execution times shorter and more consistent.

To run the protocol, all the users have to do is hold the phones together back-to-back with one phone facing up and press a button to start the exchange. This orientation provides good transmission between the phones' vibrators and accelerometers and allows the phones to automatically assign Endorser or Verifier roles. Currently, the phone with the screen facing down becomes the Verifier. Once the exchange is started, the users simply hold the phones together until the phones beep to indicate completion. Once the confirmation is complete, the phones play a lighter tone to indicate success and a harsher sound to indicate a failed exchange. These tones are selected such that they do not induce a false "yes" or "no" on the vibration channel during the final confirmation.

2.6.2 Java-Based Server

Instead of using local communication (Bluetooth or ad hoc WiFi), SHOT uses a server on the Internet to transfer the majority of data between phones. Our server is a Java program running on a desktop machine that listens for incoming connections and uses a database to store and lookup information based on the received hash value.

Currently, the server does not verify if any of the information is correct (e.g., hash a potential Endorser_Key to verify it matches Pair_ID). Instead the server acts as a means to forward information between the two phones. In the end, the phones perform the final verification.

[10]`http://www.bouncycastle.org/`

Platform	Connect & Send 1kB	1024 RSA Sign	1024 RSA Verify
iPhone	90.0 ms	201.5 ms	8.4 ms
DROID	91.2 ms	41.9 ms	7.0 ms

Table 2.1: Network and Cryptographic Performance for the iPhone and DROID.

Millisecond per bit	60	80	100
Rate of successful transmissions	0.53	0.67	0.94

Table 2.2: Impact of Vibration Time on Transmission Reliability.

2.7 Evaluation

In this section, we evaluate the reliability of the vibration-to-accelerometer channel used in SHOT, and compare the performance of Bump on both iPhone and DROID to SHOT on DROID. We also present some microbenchmarks to help explain differences in the execution times.

2.7.1 Microbenchmarks

Table 2.1 shows the average time required to perform network and cryptographic operations on either an iPhone 3G with iOS 4.2.1 or a DROID with Android 2.2. Each value represents the average from 20 executions of the operation. We measure the time needed to connect to the Bump server over WiFi and send a kilobyte of information using TCP without TLS, and the time needed to sign a random value and verify the signature with 1024-bit RSA keys. The phones exhibit similar network capabilities and require roughly 90 ms to establish a connection and send the data. However, the DROID appears to have greater processing power. Signing is almost five times faster on the DROID than on the iPhone (41.9 ms versus 201.5 ms). Verifying is also faster on the DROID (7 ms versus 8.4 ms).

2.7.2 Reliability of the Vibration Channel

To measure the reliability of the vibration channel, we exchange the 112-bit message consisting of a pre-authenticator and error-correcting code payload while varying the time needed to communicate one bit from the vibrating phone to the other phone's accelerometer from 60 to 100 ms and measured how often the receiving phone successfully decodes the message. The results of this test are summarized in Table 2.2. With 60 ms/bit encoding, the receiver is able to successfully decode the message 53% of the time. However, decoding is successful all but one time with 100 ms/bit. The limiting factor for using vibration and accelerometers for communication appears to be the scheduling on the DROID. There is a function call to turn on the vibrator for a set period of time, but multitasking prevents the system from promptly turning on and off the vibrator for values smaller than 75ms.

2.7.3 Complete Protocol Execution Times

We measure the execution time of each protocol over the course of 10 successful exchanges. During our evaluation, all of the wireless communication is sent over the WiFi network, as opposed to the cellular network. For Bump, we measure execution time from when the application is started until the phone receives the other phone's information. For SHOT, we measure the execution time from when the application is started to the tone at the end of SHOT. Figure 2.5 summarizes the results of our evaluation.

The performance of Bump is highly platform dependent, requiring an average of 21 seconds on the DROID and almost half of that (10.4 seconds) on the iPhone. The DROID appears to have faster processing and similar network performance (see Section 2.7.1), but appears to take longer to determine its location. Our experiments were performed indoors and the iPhone appears to quickly switch from trying to use GPS to using WiFi to estimate its location before connection to the Bump server (total execution time between 9.4 to 13.5 seconds). However, the DROID version of Bump appears to spend a variable amount of time trying to use GPS before switching to using WiFi (total execution between 14.2 to 37.8 seconds).

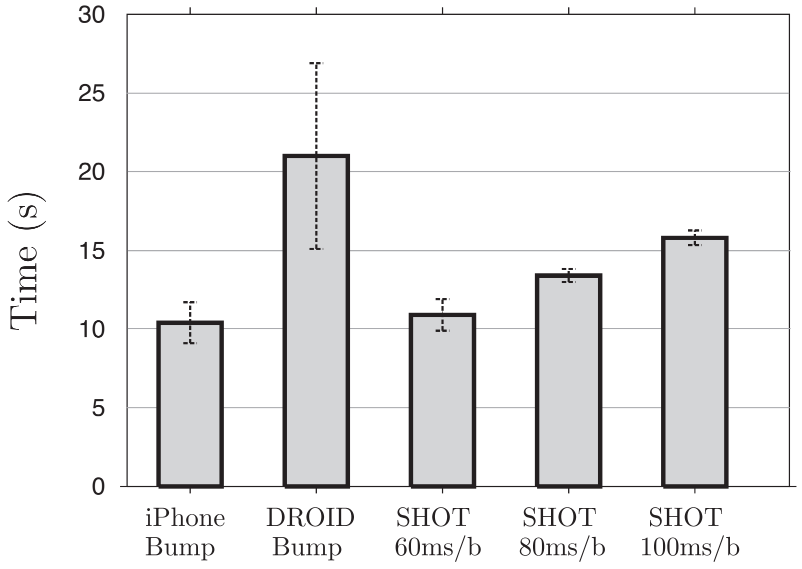


Figure 2.5: Average Execution Time for Bump and SHOT.
(1 error bar = one standard deviation.)

With fixed size pre-authenticators and confirmation vibration, SHOT provides different execution times depending on the encoding used. When compared to Bump on the iPhone, SHOT provides similar execution times when using the the less reliable 60ms/b encoding (9.9 to 12.8 seconds). However, the more reliable version of SHOT with 100ms/b encoding (15.1 to 16.9 seconds) is slower than the version of Bump for the iPhone. Compared to the DROID version of Bump, SHOT provides faster execution times. The performance of Bump on DROID may improve with the next release for DROID (DROID uses v1.3.2 while iPhone uses v2.4). Note that our implementation of SHOT is not optimized for efficiency, and the purpose of this evaluation was simply to confirm that SHOT can execute as quickly as Bump.

2.8 Summary

The exchange of information between smartphones allows users to play games, share contact information, and even transfer money. If a malicious party is able to perform a Man-in-the-Middle (MitM) attack during an exchange, she can interfere with games to annoy users, intercept communication, and even steal money. We present a MitM attack against Bump, the most popular smartphone exchange protocol, to demonstrate the relevance of this threat. Our analysis of this attack under realistic conditions shows that a malicious party can launch a MitM attack against Bump and succeed 70% of the time.

We also present SHOT, a new secure and simple-to-use smartphone exchange. SHOT uses the phones' accelerometers and vibrators to establish a human-observable channel between two phones that are held together. Since users can feel the vibrations, this channel provides demonstrative identification of the sources and destinations in the exchange, and allows users to detect if an attack is occurring (e.g., a remote party is trying to vibrate the phones). SHOT leverages asymmetric cryptography and the authentic vibration channel to bootstrap authentic communication over the higher-bandwidth wireless channel.

Our implementation and evaluation of SHOT indicates the protocol can complete in roughly the same amount of time as Bump, while providing greater security and placing similar demands on the user. As such, SHOT represents a new point on the spectrum of exchange protocols, providing improved verification of the physical source and destination of an exchange or reduced user involvement when compared to existing solutions.

Chapter 3

Mobile User Location-specific Encryption (MULE): Location-Based Access Control to Protect Sensitive Data on Lost Laptops

Lost/stolen laptops are a major cause of leaked data [78, 90]. In previous years, lost or stolen laptops resulted in the exposure of over 30 million unencrypted records [72]. These leaks expose financial account information, health records, social security numbers, and other crucial data. This issue is not limited to corporate laptops that contain hundreds or thousands of records. As financial institutions and the IRS move towards electronic systems (i.e., e-statements and e-filing), more home users are storing sensitive data on their personal laptops. The loss of a personal laptop may leak the owner's account numbers, social security number, or even health records.

A simple solution to preventing such leaks is to authenticate users and encrypt sensitive data [63, 87]. However, a recent survey found that only half of security and privacy professionals indicate their companies encrypt data[62]. Without access to policies, we cannot know why so few companies are using encryption. However, without a standard that enforces encryption, most users view the technology as an unnecessary burden. Company IT personnel will also see encryption technologies as a burden, similar to the headache associated with account management. When given the choice, users often prefer convenience over security and only worry about leaking data after losing their laptop [77]. What makes securing files on laptops hard is that users and administrators want a system that "just works" with little or no actions on their part. However, there is often a tradeoff between security & user effort.

In this chapter, we examine the extreme case of no user effort and little IT administration to determine what level of security is possible. Our goal is to remove user effort associated with encryption technology while achieving the same or better security compared to traditional password-based approaches. Prior work by Corner & Noble [25] reduces user overhead associated with securing files by leveraging a cryptographic token which shares secrets with the laptop. Provided the token is within radio range of the laptop, the user can access files. Once the token is out of range, files are encrypted. Such an approach has the advantage that all of the user's files are protected, but requires the user to carry a token to allow access to any of the files. We lower the threshold and examine how much security one can achieve with no per laptop secrets on company managed devices and zero user effort (i.e., no password entry, biometric entry, or possession of cryptographic tokens) in the common case.

We observe that only a fraction of the files on users' laptops are sensitive and most users only access those sensitive files when they are in a location they feel is difficult for malicious parties to access (e.g., a home office or a desk at work). Our approach is to encrypt user-specified sensitive files and leave all other files unencrypted and always accessible. Given the majority of accesses to sensitive files occur in a trusted location, Mobile User Location-specific Encryption (MULE) uses location-specific information from the trusted location to automatically derive a decryption key and allow access to the sensitive files. This is one scenario where verification of the physical destination of messages provides useful security properties: if the laptop is able to receive the location-specific information it is in the trusted location and should have access to the sensitive files. Once the user is inactive, logs off, or puts the computer to sleep, the system automatically re-encrypts files and deletes the key. In the rare case that a user wants to access sensitive files outside the trusted locations, the user can enter a secondary password to gain access. This password-based access also provides a fail-safe mechanism in case location-specific information or services are no longer available in a trusted location.

Convenience is the major advantage of MULE. The user can access files without performing any additional actions in the common case (accessing non-sensitive files or accessing sensitive files in a trusted location). Ideally, only in rare cases would MULE require user effort. We could use SHOT (see Chapter 2) as one potential way to verify the laptop is in the trusted location. The system would require a special device in the trusted location that acts as a key escrow and has an accelerometer and the ability to vibrate. However, that approach can negatively impact usability and provides more functionality than necessary. The laptop must have physical contact with the other device; automatic access to sensitive files is not possible if the laptop is on the user's lap in the trusted location. Another usability drawback is that SHOT only ensures authenticity of the information devices exchange. As such, a laptop would first use SHOT to prove presence in the physical space and exchange ephemeral keys. The devices will use the ephemeral keys to secure a subsequent exchange of the file decryption key. Instead, we want a protocol that allows only devices in a physical space to derive the key.

A malicious entity in possession of the laptop with unconstrained access to a trusted location could access sensitive data. We propose a new attacker model, the *Outsider Thief (OT)*, to more accurately reflect the threat of a laptop thief. We describe this adversary model in more detail in Section 3.1.

Our implementation of MULE is able to acquire the location-dependent key is less than 5 seconds depending on the security level, type of location-specific information, and technique used. Once the key is known, access to encrypted sensitive files is transparent to applications. Our implementation also includes the necessary tools to automatically re-encrypt sensitive files when the system is idle for a set period of time or put to sleep (e.g., the owner closes the laptop).

We investigate the level of security one can achieve with encrypted sensitive files when no user effort is required. Rather than relying on something users know, have, or are, we explore using **where the user** is to perform access control. We propose two new mechanisms that derive keys based on location-specific information to allow the laptop to infer that it is physically located in a trusted location, without proving to any other device that it is in the trusted location. We evaluate an implementation of our system on readily available hardware.

3.1 Problem Definition

Users habits indicate the need a system that allows the encryption of sensitive data while requiring minimal administrative effort and zero user effort during common accesses and moderate effort otherwise. Such encryption should, with high probability, prevent an adversary who steals the laptop from accessing files. Minimal administrative effort applies to a corporate setting and means IT personnel at most have to keep a list of not-yet-stolen laptops (i.e., there are no per-laptop secrets on a corporate server). A scheme requires zero user effort, if a user can access sensitive files without entering a password or biometric, carrying around a hardware token, or adding additional hardware to their laptop. The main challenge is how to protect the key needed to decrypt sensitive data without requiring user or administrative effort.

To avoid user effort in the common case, we need a system that allows the laptop to automatically derive the key based on location-specific information when in a trusted location (the common accesses). The laptop can use this location specific key to decrypt sensitive files and provide the user access.

Location-based access control [27] addresses the different problem of proving to an outside system that a device/user is in a location and thus should have access to a resource. Our problem involves a laptop that wants to prove to itself that it is in a specific location (i.e., retrieve location-specific information and ultimately a decryption key). The laptop can leverage information and computation from other devices already in the location to perform this proof, but the other devices perform no authentication of the laptop and require no per-laptop secrets.

3.1.1 Assumptions

Sensitive Data Access Patterns. We assume users rarely access sensitive data outside of trusted locations. For example, users will work on taxes in their home or access customer accounts in the office. This assumption remains true for individuals that travel for work. Companies often disallow individuals from taking sensitive files out of the office [24]. For example, human resource employees often access employee records. However, while traveling to recruiting events, company policy may dictate that laptops must not contain unencrypted copies of employee records. Some users rarely access sensitive files in the same location. For example, a consultant that frequently travels may have no real office. For users without a trusted location, MULE provides no real advantage.

Available Hardware and Software. We assume laptops are equipped with a video camera and trusted computing technology, MULE users will accept the cost of installing a small device in the trusted location, and corporations that use MULE will have a whitelist of company laptops or a blacklist of stolen company laptops and a Public Key Infrastructure (PKI) or at least the means to distribute authentic public keys to their employees.

The majority of commodity laptops have webcams mounted into the frame around the display and already come equipped with trusted platform modules (TPMs) [98], an inexpensive coprocessor that enables a number of security related operations. Extracting keys from the TPM is infeasible without expensive hardware and extensive time.

Given the lack of laptop accessible location-identifying information in home and work offices, we assume users or their companies will install a Trusted Location Device (TLD) as part of MULE. The TLD provides location-specific information and responds to any machine that wants to run the key derivation protocol. Instead of performing location-based access control, the TLD performs no authentication of the requesting device. TLD secrets, location-specific information, and inputs to the key derivation process form the foundation of the secrecy of location-specific keys in MULE. The TLD is a spare machine connected to an inexpensive ($20) microcontroller to transmit location-specific information. The TLD also requires zero user effort after plugging it into the trusted location and minor maintenance in the corporate setting to ensure protection of keys (see Section 3.2 for more details).

We assume corporations keep track of company owned laptops using some type of unique identifier that the laptop knows. This could be a value the company assigns to the laptop (e.g., a network assigned name or IP address) or a value found in hardware (e.g., the MAC address of the laptop). When the laptop is lost/stolen, IT or property management personnel will remove the laptop from the company whitelist (or add it to a blacklist) of laptops that are allowed various services (e.g., access to the corporate wireless network). We leverage this company assigned unique identifier during calculations such that, once the laptop is stolen, the calculations fail, but no authentication of the identifier is performed—as part of MULE.

Any corporation with a secure web-site already uses a PKI to protect communication with the site. We assume that the company can use this same PKI to identify TLDs. Otherwise, IT personnel can easily create an in-house PKI for free using open source software like OpenSSL [66] and manually distribute the necessary credentials (i.e., CA's public key) to identify TLDs.

3.1.2 Outsider Thief (OT) Attacker Model

We propose the Outsider Thief (OT)—a realistic attacker to model a laptop thief. The thief has complete control over a stolen laptop, may visit a company office, and can launch attacks on the wireless network, but is unlikely to break into a user's home. After the OT steals the laptop, she can install any software and try to guess the user's login password. We assume the laptop has no malware before it is stolen, otherwise any sensitive data the user accesses could be leaked. Malware defenses are an important problem that is outside of the scope of this work. We assume fear of legal retribution prevents an OT from breaking into a home to access a home user's trusted location. Corporate trusted locations are publicly accessible, but are protected by guards and IT personnel which prevent an OT from successfully compromising devices in the trusted location. An OT can overhear, intercept, and inject messages on the wireless network. However, when outside of a trusted location, an attacker is unable to access the location's constrained channel [52] (e.g., infrared signals that are unable to pass through walls or sounds in an insulated room). We recognize that stronger attacker models exist, but we expect that defending against those adversaries requires additional user effort.

3.1.3 Requirements for Location-Specific Information Used to Derive Keys

When location-specific information is used for key derivation, the information must fulfill the following requirements to ensure successful and secure operation of MULE.

Easily Accessible. Once the laptop is powered on, placed in a trusted location, and the user is logged in, the laptop should have access to the information required for key derivation.

Unique to a Location. If the information is not unique, the laptop may automatically decrypt a user's sensitive files while outside of the originally defined location, an obvious security vulnerability.

Bounded Range. Location information should only be accessible within the location. Information accessible from outside of a building will apply to more than the location the user trusts.

Significant Entropy. Information used to derive the key within a location needs to have significant entropy so that it is hard to guess. Limited entropy would enable an attacker to guess the necessary values, spoof the location, and recover a key.

3.2 MULE Overview

MULE's goal is to protect sensitive files on mobile devices with zero user effort in the common case. Standard user login works independent of MULE and provides a form of weak user authentication. All non-sensitive files on the laptop are left unencrypted and are always accessible. Only user-specified sensitive files are encrypted. Figure 3.1 depicts an overview of the operation of MULE.

When a user tries to access a sensitive file, MULE contacts a Trusted Location Device (TLD) which helps the laptop derive the key needed to decrypt sensitive files with zero user effort. The TLD generates a nonce and transmits it over the constrained channel [52]. We call this TLD generated nonce a location-specific message (m) because the properties of the constrained channel ensure that only devices within the trusted location can access the m associated with the current run of the protocol. A TLD is unable to authenticate requesters without per-laptop secrets and will respond to any key derivation request. However, the key derivation calculations are such that a TLD produces the wrong output if the requester uses the wrong m in calculations (e.g., the client is in a different location). After the TLD has helped derive the key, the user can access sensitive files without having performed any extra actions. During the rare occasion when a user accesses sensitive files outside of a trusted location, MULE will lack the correct location-specific information and key derivation will fail. In that case, we sacrifice some usability to preserve security and ask the user to enter a password as part of a location-independent key derivation scheme. The

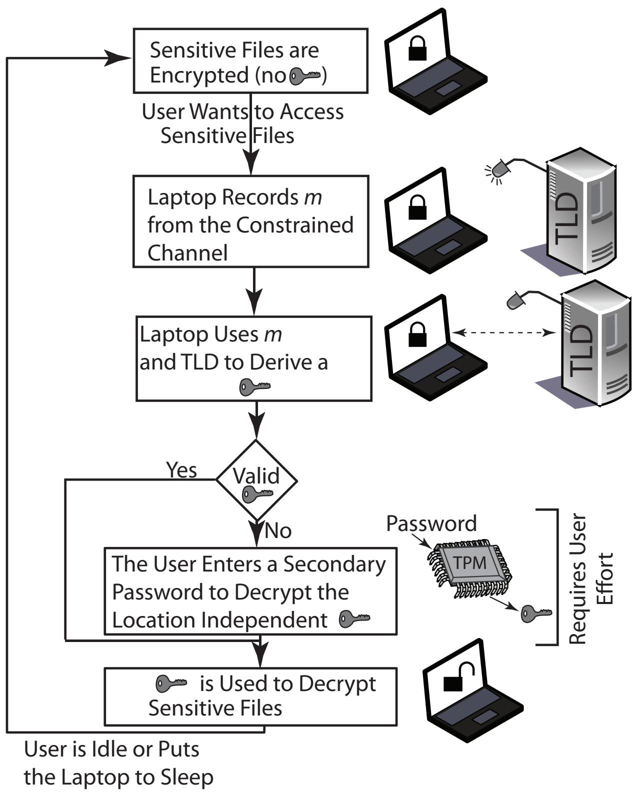

Sensitive Files are Encrypted (no)
User Wants to Access Sensitive Files
Laptop Records *m* from the Constrained Channel
TLD
Laptop Uses *m* and TLD to Derive a
TLD
Valid
Yes
No
The User Enters a Secondary Password to Decrypt the Location Independent
Password
TPM
Requires User Effort
is Used to Decrypt Sensitive Files
User is Idle or Puts the Laptop to Sleep

Figure 3.1: Operation of MULE.
User effort is only needed when location-specific key derivation fails.

password allows the TPM on the laptop to decrypt a location-independent key which can decrypt the files. Once a valid key is available, the sensitive files are decrypted. When a user is idle for some set period of time, logs off, or puts the laptop to sleep, the laptop will re-encrypt the files and delete the key.

In this chapter, we present two location-specific key derivation protocols that leverage the same implementation of a constrained channel. The reason we have two protocols is that home and corporate users are willing to accept different levels of management overhead and use cases. In both protocols, we use an infrared (IR) LED and the laptop's webcam to implement a constrained channel. IR cannot pass through objects (e.g., walls, window blinds, or people). If one is viewing sensitive files on the screen, one should close the blinds to prevent an OT across the street from seeing the display—and also prevent access to the constrained channel. One downside to this constrained channel is the limited number of bits the TLD can reliably transmit to the laptop in a fixed amount of time. Based on the Nyquist frequency, a camera that captures X frames per second is limited to $X/2$ bits per second when using an on/off encoding scheme.[1] We could add multiple LEDs to encode more bits per frame. However, the user would have to pay careful attention to how the laptop is positioned within a trusted location to ensure the LEDs are easily differentiable so the laptop can successfully decode m. Instead, we design key derivation protocols which are secure despite the use of a 20 to 30 bit long m which changes with each run of the protocol.

In Section 3.3, we describe the Home Key Derivation (HKD) protocol. In the home scenario, we assume the user wants to turn an existing computing device (e.g., old desktop or wireless router) into a TLD, hang the IR LED over the desk in the office (or other trusted location), and leave the system alone. Without access to the constrained channel, an OT can sit outside of the user's home, intercept all wireless communication, and find it infeasible to recover the key. If the laptop is ever stolen, the user can feel secure knowing a thief needs to break into her home to steal the TLD or attack the location-independent mechanism to recover a key which will decrypt sensitive files.

In Section 3.4, we describe the Corporate Key Derivation (CKD) protocol. In a business setting, malicious parties may surreptitiously gain access to the office for any number of reasons (e.g., a corporate open house or interview). As such, an OT may access the trusted location with the laptop in her possession. In that case, we leverage the company's guards and IT personnel to maintain the secrecy of the sensitive files. Guards will prevent an OT from breaking into locked rooms and subverting TLDs.[2] The com-

[1]The LED on is a 1. The LED off is a 0.

[2]The TLD can be secured behind doors and use a wired connection to send data to microcontrollers which transmit data over the constrained channel.

pany's IT personnel can maintain a simple white-list of valid MULE clients based on some static unique ID for the laptops (e.g., a network name, IP address, or MAC address). During key derivation, the TLD will verify a laptop supplied name is in the whitelist and use it as input to the key derivation function. Even though the TLD performs zero authentication that the supplied name belongs to the laptop, this ensures that after a laptop is stolen and administrators remove the laptop name from the white-list, the thief is unable to recover the key needed to decrypt the files using the TLD.

Outside of the trusted locations, during designation of a trusted location, or when automatic key derivation fails, MULE uses a location-independent key storage and retrieval technique that uses TPMs and a secondary password (different from the user's login password) to securely manage a key that can access the sensitive files (see Section 3.5).

In Section 3.6, we discuss implementation details of the protocols, management of encryption/decryption of files for application transparency, key management such that multiple keys can access the files, and management of the automatic re-encryption of files so that a user does not unknowingly leak data.

3.3 Home User Scenario

Within the home setting, MULE needs a key derivation technique that "just works" once the TLD is powered on and in the trusted location. Pairing the laptop with the TLD could establish a strong shared key which could be used to secure later key derivations, but requires user effort and is vulnerable to human error [100]. As such, MULE should perform all of the tasks necessary once the laptop is in a trusted location. The simplest key derivation protocol would have the laptop derive a decryption key from a fixed m. Without access to the constrained channel, an OT will be unable to derive the key. However, an attacker in possession of the laptop could quickly brute force the key, given the limited entropy of m. Instead, the TLD can possess a strong secret to help derive the laptop's decryption key and use m to ensure the requesting laptop is in the room and protect communication over the wireless network. With Encrypted Key Exchange (EKE) [7], the TLD and the laptop can leverage m as a weak secret to establish a session key. The session key can protect the laptop's subsequent file decryption key request. Instead of using m to establish a key which is used to protect the derivation of a different key, we would like a protocol that allows the laptop with knowledge of m to successfully acquire a file decryption key with less communication and computation overhead. Since a new m is randomly generated for each run of the protocol, we can use a more efficient protocol. With blind-signatures [22], a laptop can store a secret k and use a TLD signature ($k^d \mod N$) as a decryption key, without revealing either value to the TLD or an eavesdropper. If the signature request is concealed using symmetric encryption with m as the key, the laptop with the correct m can derive the key in a single round of communication over the wireless channel. Provided symmetric encryption functions as a Pseudo-Random Permutation (PRP), a device with the wrong m (a device outside of the room) will receive seemingly random output from the TLD.

In the remainder of this section, we describe the key derivation protocol, how the laptop first associates a key with a trusted location, and an analysis of our protocol.

3.3.1 Home Key Derivation (HKD)

The HKD protocol consists of 4 main steps: initialization, input hiding, TLD calculations, and key recovery. During initialization, the laptop verifies it is interacting with a known TLD.[3] At the same time, the TLD generates random location-specific information (m) for use during this instance of the protocol and transmits it using the constrained channel. During input hiding, the laptop multiplies a value stored on the laptop (k) with a random number to generate a blinded request and uses m to encrypt the blinded value. The TLD's calculations include using m to decrypt the ciphertext and recover the laptop's blinded message, signing the value, and returning the result to the laptop. To recover the key needed to decrypt sensitive files

[3]Section 3.3.2 discusses how the laptop first learns the appropriate TLD and public key for a trusted location.

(a signature on k), the laptop unblinds the signature. Figure 3.2 contains a summary of the HKD protocol. The goal of the protocol is to ensure that only a device with knowledge of m can successfully retrieve the correct signed value from the TLD, without revealing the laptop's long-term value k or a signature on k. If a client uses an incorrect m (the key for the cipher), the decryption at the TLD will produce the wrong value and result in a different signature.

Initialization:			
1.	$TLD \rightarrow L$:	K^{+}_{TLD}	TLD responds with its public key (N, e).
2.	L :	if(!($k = get\_k(K^{+}_{TLD})$) quit;	L retrieves the long-term secret for this K^{+}_{TLD} from its harddrive and quits if no record exists.
3.	TLD :	$m \xleftarrow{R} \{0,1\}^{\ell}$	TLD generates a random value of length ℓ to act as the current location-specific information.
4.	$TLD \xrightarrow{CC} L$:	m	TLD transmits the location-specific information over the constrained channel.
Input Hiding:			
5.	L :	$R \xleftarrow{R} \mathbb{Z}^{*}_{N}$	L generates a random number relatively prime to N,
6.		$b \leftarrow R^{e}\, k \mod N$	uses R raised to the public exponent e to blind the long term secret,
7.		$c \leftarrow Encrypt_{m}\{b\}$	and uses m to symmetrically encrypt the blinded value.
8.	$L \rightarrow TLD$:	c	L sends the concealed value to TLD.
TLD Calculations:			
9.	TLD :	$v \leftarrow Decrypt_{m}\{c\}$	TLD uses m to decrypt the message
10.		$\sigma \leftarrow v^{d} \mod N$	and signs the value.
11.	$TLD \rightarrow L$:	σ	TLD returns the signed value.
Key Recovery:			
12.		$K \leftarrow \sigma R^{-1} \mod N$	L unblinds the result to retrieve $k^{d} \mod N$. Note: $(R^{e}k)^{d}R^{-1} \mod N = R^{ed}k^{d}R^{-1} \mod N = Rk^{d}R^{-1} \mod N = k^{d} \mod N$

Figure 3.2: The HKD Protocol Between a Laptop (L) and a TLD.

Initialization When the protocol starts, the TLD sends its RSA public key (see Step 1 in Figure 3.2 where N is the RSA modulus and e is the public exponent). At this time, the laptop verifies it is talking to a known TLD by checking metadata stored with the encrypted sensitive files. If this TLD is unknown, the laptop considers this an untrusted location and stops key derivation (see Step 2). At the same time, the TLD randomly generates the location-specific information m of length ℓ and transmits it over the constrained channel (see Steps 3 & 4). Since the constrained channel provides a slow rate of transfer, ℓ is 20 to 30 bits to reduce the transmission time to a few seconds. Note that here we use infrared as a constrained channel, but any medium that allows the TLD to confine the transmission of m to the physical room will work.

Input Hiding After receiving m, the laptop uses a random number R to blind the laptop's long term secret k, and uses m to encrypt the result in Steps 5 to 7. Here, R is a random number relatively prime to N. Blinding the value hides k and temporarily conceals m. Blinding with the random value R^e prevents the TLD (or any device with knowledge of m) from recovering k from $R^e k \mod N$. Blinding also ensures that the value encrypted is different for each protocol run. This prevents an entity outside of the room from intercepting c and recovering m from a brute-force attack. When testing a potential key for the cipher, the output of decryption appears as a random value and an attacker cannot verify if the revealed value matches the laptop's original message, thus verifying the correctness of a guess of m. In Step 7, an implementer must select a cipher such that a failed decryption does not leak information about m. If the size of the blinded value is a multiple of the cipher's block size, Electronic Code Book (ECB) mode is sufficient. However, the addition of deterministic padding to the plaintext would allow an attacker to recover m based on c (e.g., if padding is a series of 0s the attacker will try different values of m until decryption results in a plaintext ending in 0s). If the length of the blinded value is not a multiple of the cipher block size, a stream cipher or ciphertext stealing can encrypt the blinded value such that no predictable plaintext is used.

TLD Calculations After receiving c (Step 8), the TLD uses the correct m to decrypt the message (Step 9). If c was generated with m, the TLD has a copy of $R^e k \mod N$ (or whatever the laptop sent). If c was generated with a different key, the TLD will have a pseudo-random value that differs from the laptop's original input with high probability. After using the private exponent d to complete the blind signature, the TLD returns the result to the laptop as σ (Steps 10–11). At this point, m **is no longer location-specific information**. An attacker can take c and σ and recover m by finding the x such that $Decrypt_x\{c\} == \sigma^e \mod N$. However, learning this m is useless since the TLD uses a new random m for each run of the protocol and blinding conceals k and k^d.

Key Recovery Once the laptop has the TLD's response (σ from Step 11), the laptop will use $R^{-1} \mod N$ to recover the file decryption key/signature ($k^d \mod N$) from the TLD's response. Provided the laptop and TLD were using the same value for m, the laptop will now possess a deterministic signature on k (static across time) which it can use as a decryption key. If the laptop used incorrect location-specific information ($m_{Laptop} \neq m_{TLD}$), the end result will contain a signature on a pseudo-random value—the TLD signed a different value than the laptop sent—that will fail to decrypt the sensitive files.

In the remainder of this section, we discuss how our protocol allows TLD identification with zero user effort and why our protocol is secure against an OT.

3.3.2 Trusted Location Designation

Before using HKD, a laptop must acquire the correct public key (N, e) associated with a trusted location. Given the attacker is unable to send or receive information on the constrained channel, learning the correct public key for a trusted location requires zero user effort. When designating a location as a trusted location, the laptop generates a random 128 bit or larger k. After performing HKD with the new k and a potential public key, the laptop can verify if the device which controls the constrained channel used the public key the laptop received over the wireless channel. Given an OT is unable to access the constrained channel, this verification ensures the TLD—which controls the constrained channel—has the public key the laptop received. Once the laptop knows it is interacting with the TLD with the correct public key, the laptop saves k and the public key (N, e) as metadata with the encrypted sensitive files so future runs of HKD can access all of the necessary data. All a user has to do during designation of a new location is to recover the location-independent key (see Section 3.5) so the sensitive files can be decrypted using HKD or the location-independent mechanism. Section 3.6.3 has more details on how we manage sensitive files encrypted under multiple keys.

After a sample run, the laptop can check if it has the correct public key by verifying the signature it received from the potential TLD. If the signature is valid, there is a $1 - 2^{-\ell}$ chance that the public key used is the correct public key for this trusted location. There is a small probability $(2^{-\ell})$ an attacker impersonated the TLD and was able to guess m. With a correct guess of m, the attacker could correctly decrypt c and produce a valid signature for the claimed public key. For a stronger guarantee, the laptop could run HKD multiple times before accepting N, e. Given m is randomly generated independently for each run of the protocol, the laptop will detect an attack after n runs of the protocol with probability $1 - 2^{-n\ell}$.

3.3.3 Security Analysis

In this section, we discuss how HKD prevents an attacker from recovering $k^d \mod N$ (the value used as the file decryption key). Provided RSA is secure, blind signatures conceal the original message [22], m and R are random, and the block cipher used is a good pseudo-random permutation, an OT attempting to discover $k^d \mod N$ will be unsuccessful. Section 3.3.2 already discussed why an attacker is unable to pose as a TLD during setup and generate $k^d \mod N$ once in possession of the laptop (if the impersonation was successful, the attacker would know the private key needed to generate the signature). The remainder of this section discusses how the key remains secure when the laptop is still in the user's possession and after the OT has stolen the laptop.

While the user still has possession of the laptop, an attacker can eavesdrop on HKD to try and recover the key from messages from multiple protocol runs, or try to directly run the protocol with the TLD. After eavesdropping on one run of the protocol, an eavesdropper will have a blinded message ($R_1^e k$) and a blind signature ($R_1 k^d$). An attacker that could recover k or k^d from these messages would be able to defeat blind signatures (something infeasible assuming blind signatures conceal the original message and the attacker lacks the randomly generated R values). After eavesdropping on n protocol runs, the attacker will have n messages ($R_1^e k, R_2^e k, ..., R_n^e k$) and n signatures ($R_1 k^d, R_2 k^d, ..., R_n k^d$). Even with n pairs, the attacker is still unable to recover k or k^d given each R_i is random and the number of unknowns matches the number of equations. Without knowledge of k, direct interaction with the TLD provides little information to the attacker. As we discuss in the next paragraph, it is also infeasible to retrieve the desired signature without knowledge of the current m.

With possession of the laptop, an attacker will know k and attempt to recover $k^d \mod N$ by calculating the signature, deriving the value based on previously recorded messages, or via interacting with the TLD. If an attacker were able to generate $k^d \mod N$ without knowledge of d (or the factors of N) the attacker can compromise RSA (something infeasible assuming RSA is secure). An attacker can try to recover k^d using k and data from previously recorded runs of the protocol (e.g., the n pairs of messages ($R_1^e k, R_2^e k, ..., R_n^e k$) and signatures ($R_1 k^d, R_2 k^d, ..., R_n k^d$)). Now, the attacker can recover the various R_i^e by multiplying a message with $k^{-1} \mod N$. However, the attacker is unable to isolate k^d from the signatures. The RSA assumption dictates that given N and R_i^e (the values an attacker has) it is infeasible to recover R_i (the value the attacker needs). As such, the attacker needs a different approach to learn k^d.

If an attacker can submit k to the TLD concealed with the correct m, the attacker can recover k^d and decrypt the user's sensitive files. However, we assume an OT is unable to access the constrained channel and thus lacks knowledge of m. Instead, the attacker can try to guess m and interact with the TLD by itself or intercept communication between a legitimate user and the TLD to recover the current m. On her own, the attacker has a $2^{-\ell}$ chance of guessing the correct m for a given run of the protocol. A geometric distribution describes the probability of success after x attempts ($x - 1$ failures followed by one success). As such, an attacker will need 2^{ℓ} attempts on average to successfully guess m. To make the attack less feasible, the TLD can rate limit requests, forcing an attacker to invest more time to perform the large number of attempts. If an attacker wants to leverage information from a legitimate run of HKD, the attacker must intercept the first message and try to determine the current m which produces the other user's blinded message $R_O^e k_O$. However, the block cipher and the random selection of R_O prevent the attacker from verifying if the guess for m was correct. Given the randomness of R_O^e and the pseudo-randomness of the cipher, the majority of decryptions look like potential legitimate messages. As such, interception of c_O provides little help to an attacker trying to recover the current m. After the TLD responds, the attacker can recover the last m from c_O and σ_O, but once the TLD responds it will use a different m for the next run of the protocol.

Without access to the constrained channel, an attacker is unable to recover the m needed to interact with the TLD and will be unable to recover the signature needed to decrypt the user's sensitive files. The best an attacker can do is recover m after the TLD responds. However, at that time the TLD will expect a different m for the next run of the protocol.

3.4 Corporate User Scenario

In the corporate setting, an OT is able to access the constrained channel. If HKD was used, an OT could recover file decryption keys for stolen laptops. The company PKI and an administrator maintained list of company laptops allow us to design a protocol such that legitimate users can access sensitive files on a laptop with zero user effort, but an OT in possession of the laptop is unable to access the same data. In the remainder of this section, we discuss why the HKD fails to work in the corporate settings, how the Corporate Key Derivation (CKD) works, and why CKD is secure.

With access to the office, a malicious party can receive or send data on the constrained channel and circumvent any security provided by HKD. With the ability to receive data from the constrained channel, an attacker could return to the office after stealing a laptop and use the TLD to automatically derive the file decryption key. With the ability to send on the constrained channel, an OT could pose as a TLD during initial MULE setup, trick a laptop into deriving a file decryption key based on the attacker's RSA key pair, and generate the file decryption key once in possession of the laptop.

The Corporate Key Derivation (CKD) must authenticate the TLD, cease to work once the laptop is reported stolen, and only succeed when the laptop has access to the constrained channel. With a company PKI/trusted company public key, TLDs possess authority signed certificates to identify themselves as legitimate. A TLD certificate and Transport Layer Security (TLS) [30] can prevent an attacker from spoofing the TLD or eavesdropping on other devices' communication with the TLD. However, execution of HKD over TLS is insufficient, because an OT can steal a laptop, return to the office, and acquire the file decryption key. The TLD could authenticate laptops and only perform HKD for laptops not yet reported stolen. However, this means significant administrative overhead with per-laptop secrets on the TLD, or the addition of laptops to the company PKI and the maintenance of a Certificate Revocation List (CRL) to identify stolen laptops. Instead, the TLD uses the laptop's long-term secret and a company assigned laptop identifier (without any authentication the laptop is the one it claims to be) as input during key derivation, and refuses to derive keys for laptops that report an identifier not in the company whitelist (or quits if the laptop is present in a blacklist). The TLD uses a keyed hash/Message Authentication Code (MAC) as an efficient way to generate an output given a laptop secret and ID pair, without leaking any information about the key used in the MAC. To ensure derivation only succeeds for devices in the trusted location, the laptop xors its long-term secret with the location specific m, and the TLD xors m with the laptop's input. If a laptop uses an m that differs by one or more bits, the TLD will use the wrong input to the MAC and fail to derive the correct key. Encryption/decryption of the input using m as the key would provide the same results as xor, but requires additional computation. The Corporate Key Derivation (CKD) can protect sensitive data from a laptop thief, provided the laptop is reported stolen before an OT can enter the corporate trusted location.

In the remainder of this section, we describe the CKD protocol and discuss why it is secure. Trusted location designation within CKD is similar to designation within HKD, except the laptop can verify the public key for the TLD via the corporate PKI or a public key distributed manually (e.g., IT personnel installs it with other software).

3.4.1 Corporate Key Derivation (CKD)

In the CKD protocol, the TLD uses TLS to prevent attackers from posing as the TLD or interfering during legitimate key derivation, xors an input with m to implicitly check if the other device is in the trusted location, and uses a MAC to derive keys. Figure 3.3 contains the various steps involved in CKD. The protocol is divided into three main steps: initialization, application of m, and TLD calculations.

Initialization:		
1. $L \overset{TLS\_init}{\leftrightarrow} TLD$	: Cert'$_{TLD'}, \sigma_{TLD'}$	TLD sends a certificate and signature as part of the TLS handshake.
2. L	: if(!verify($\sigma_{TLD'}, Cert_{TLD}$)) quit	L uses the locally stored certificate to verify this is a trusted TLD.
3. TLD	: $m \xleftarrow{R} \{0,1\}^{\ell}$	TLD generates a random value of length ℓ as the current location-specific information
4. $TLD \xrightarrow{CC} L$	: m	and transmits the location-specific information over the constrained channel.
Apply m**:**		
5. L	: $x \leftarrow k \oplus m$	L xors m with the long term secret value (k) stored on the laptop
6. $L \xrightarrow{TLS} TLD$	: ID_L, x	and sends its unique ID and the xored value to the TLD.
TLD Calculations:		
7. TLD	: $k \leftarrow m \oplus x$	TLD recovers the laptop's long term secret,
8. TLD	: if ($ID \notin$ whitelist) quit	verifies the ID is in the whitelist,
9. TLD	: $K_{ID_L} \leftarrow MAC_{K_{TLD}}(ID_L \| k)$	uses a keyed hash to derive the key for this ID_L, k combination,
10. $TLD \xrightarrow{TLS} L$	: K_{ID_L}	and returns the calculated value to the laptop.

Figure 3.3: The CKD Protocol Between a Laptop (L) and a TLD.

Initialization The laptop initiates a TLS connection with the TLD and uses locally stored certificates or public keys to verify the TLD. At the same time, the TLD generates a random value to use as location-specific information and transmits it over the constrained channel.

Application of m The laptop xors the location-specific information with its long-term secret (k) as part of the implicit check that the laptop is in the trusted location. The laptop sends the result of the xor operation and the laptop's ID to the TLD.

TLD Calculations The TLD performs two operations to ensure only laptops with access to the constrained channel that are on the company whitelist acquire decryption keys. The TLD xors the received value with m to recover k and verifies the provided ID is in the whitelist. Provided the ID is in the whitelist, the TLD uses the TLD key (K_{TLD}) to generate the MAC of the laptop ID concatenated with k. If the laptop is outside of the room and uses the wrong m, the TLD will use the wrong k as input to the MAC. With the wrong input, the output will fail to decrypt sensitive files. If the whitelist indicates the laptop is stolen (i.e., the ID is absent from the list), the TLD quits the current run of CKD. With a MAC, the TLD can use the same secret (K_{TLD}) to securely generate different outputs for different IDs. For example, if laptop ID_1 is stolen, it is infeasible for an attacker to recover $MAC_{K_{TLD}}(ID_1||x)$ when posing as ID_2 (i.e., a secure MAC ensures the attacker is unable to find y such that $MAC_{K_{TLD}}(ID_2||y) == MAC_{K_{TLD}}(ID_1||x)$).

In Section 3.4.2, we discuss how secrecy of k and the use of a whitelist of valid IDs ensures attackers are unable to use the CKD to recover the keys needed to access sensitive files on a stolen laptop.

3.4.2 Security Analysis

The security of the laptop's decryption key K_{ID_L} relies on the security of TLS, the secrecy of the laptop's long-term secret before the laptop is stolen, and the timely update of the whitelist after the laptop is stolen. We assume

TLS is secure and laptops are able to correctly identify the TLD based on secure distribution of public keys (a company CA key or a copy of the certificate for the TLD). As such we only consider how an attacker can attack the CKD to recover K_{ID_L} before or after stealing the laptop and how tunneling of m can cause the decryption of sensitive files while the laptop is still in the legitimate user's possession.

Before the laptop is stolen, an attacker is unable to recover K_{ID_L} because it lacks knowledge of the long-term secret k. Given TLS authenticates the TLD, an attacker is unable to impersonate the TLD and trick the laptop into sending k to the attacker. At this time, an attacker can pose as the laptop and interact with the TLD while using the laptop's ID. However, without knowledge of k, the attacker is unable to know what inputs will generate the correct output from the TLD. Without the encrypted files, the attacker will be unable to test a potential key and verify the guess was correct. The attacker can collect a large set of potential keys by sending a large number of requests to the TLD. However, k is a long sequence of bits (i.e., 128 bits or more) randomly generated during assignment of the office as a trusted location. The chance of an attacker correctly guessing k and thus retrieving the key from the TLD while the user still has possession of the laptop is negligible.

Once the laptop is stolen, the user will report the theft to the company's IT department which will consequently remove the laptop's ID (ID_L) from the whitelist (or add ID_L to a blacklist). Even though the attacker now knows k and can interact with the TLD, the MAC function and the secrecy of K_{TLD} ensure the attacker will be unable to acquire the key needed to access the sensitive files. Once the ID_L is removed from the whitelist, an attacker will be unable to retrieve values from the TLD that are derived using ID_L. To learn K_{ID_L}, the attacker can recover K_{TLD} and derive the key itself or find different inputs ID_2 and k_2 which, when sent to the TLD, produce the same output. Provided the TLD is securely locked up in a physically guarded room and lacks any software vulnerabilities, the only way to recover K_{TLD} is to send inputs to the TLD and analyze the responses to recover K_{TLD}. Provided the MAC function used is secure, this type of attack is infeasible.

Next an attacker could try to find a still legitimate identifier (ID_2) and an input (k_2), such that $MAC_{K_{TLD}}(ID_L||k) == MAC_{K_{TLD}}(ID_2||k_2)$. An attacker with the ability to discover such an ID_2, k_2 pair would be able to perform selective forgery of the MAC function. Assuming the MAC function is secure, selective forgery is infeasible. The only way an attacker can successfully acquire K_{ID_L} is to steal the laptop and return to the trusted location before the theft is reported.

An attacker can cause a laptop still in the legitimate user's possession, but outside of a trusted location, to derive the correct key with CKD. Xoring k with the shorter location-specific information ensures that only devices with access to the constrained channel can successfully derive keys. If the laptop xors k with the wrong value, the TLD will xor with m and produce a different value as input to the MAC. If the laptop is unable to access the constrained channel the chance of accidentally guessing the correct m is $2^{-\ell}$. However, an attacker can relay information from the constrained channel in the trusted location to a location outside of the trusted location. Assuming users quickly report stolen laptops to the company, this tunneling action only accidentally reveals sensitive files to legitimate users still in possession of the laptop; the TLD will refuse to derive the correct key for a reported stolen laptop.

In this section, we have presented the CKD protocol and discussed why it allows the secure automatic key derivation without per laptop secrets on the TLD, provided the laptop is reported stolen before an attacker can access the trusted location.

3.5 Location-Independent Key

Users outside of a trusted location may need access to sensitive data. Given these accesses are infrequent, we assume users will accept the solution to require some interaction and time to maintain a high level of security. When outside of a trusted location, we use a secondary password – one different than the user's login password – to retrieve a location-independent key. The simplest solution is to use the password itself as the key. However, an attacker can brute force a password in a matter of hours. Similar to Bitlocker [63], we use the laptop's TPM to bind [98] a key based on the user's password. Here we discuss how MULE uses the user's password and the TPM to protect the location-independent key.

During installation, MULE generates and encrypts the location-independent key (K_{Ind}) that is later un-bound whenever the user accesses sensitive files outside of any trusted location. When the user first installs MULE, the system generates a random K_{Ind} and takes as input from the user a secondary password. Next, MULE asks the TPM to generate a non-migratable asymmetric key pair ($K^{+}_{MULE}, K^{-1}_{MULE}$) (i.e., the key is only accessible on this TPM) and require the user's secondary password to permit operations which utilize the secret key. The TPM can encrypt K_{Ind} without requiring the password. MULE stores the encrypted K_{Ind} in the user's home directory until the user tries to access protected files and HKD or CKD fail to derive a key. At that time, MULE asks the user for the secondary password. MULE passes the entered password and the encrypted copy of K_{Ind} to the TPM. In response, the TPM will only permit calculations which use K^{-1}_{MULE} and decrypt and return K_{Ind} when given the correct secondary password.

Without the secondary password, an OT would need to invest a significant amount of time (e.g., decades) to access user's sensitive files. To access files using the location-independent key, an OT needs to guess the password or one of the keys. We assume K_{Ind} is at least a 128-bit randomly generated value and the TPM uses a 2048 bit RSA private key so it is computationally infeasible for attackers to guess either of the keys. If an OT tries to guess the user's secondary password, the TPM's built-in guessing attack defenses and the fact that only that TPM can use K^{-1}_{MULE} and decrypt the data (i.e., the attack is non-parallelizable) will prevent the attacker from accessing the files for several decades on average (see Section 3.7.2 for more details about guessing attacks against the TPM we use in our implementation).

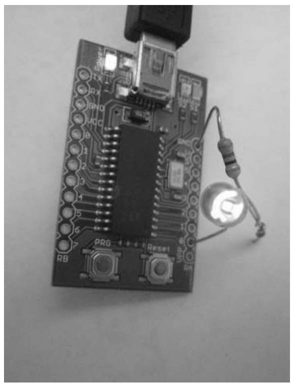
3.6 Implementation

Our implementation of MULE runs on a HP 6730b with a 2.4 GHz Intel Core 2 Duo processor and 2GB of RAM running Ubuntu 2.6.28 that uses EncFS[4] to store sensitive data as encrypted filesystems. For a TLD, we use a Dell Optiplex 755 with a 3.2 GHz Intel Core 2 Duo processor with 4GB of RAM connected to a Universal Bit Whacker[5] with an IR LED (see Figure 3.4(a)). We use OpenSSL for all of the cryptography in the protocol. We use AES with a 128 bit key (m prepended to 0s) as a cipher for HKD. HKD signature generation uses 2048-bit RSA with TLD side blinding to prevent timing attacks [10]. We use HMAC with SHA1 as the MAC in CKD. TLS in CKD uses ephemeral Diffie-Hellman with 2048-bit RSA authentication during setup with AES256 and SHA1 to protect communication. We use video for Linux two (V4L2) to directly capture frames from the laptop's webcamera. As a trusted location, we attach the IR transmitter to the underside of a bookshelf on a desk, pointing towards the back of the desk (see Figure 3.4 (b)). In this setup, the laptop can only see the LED (i.e., be in the trusted location) while the laptop is open and on the desk. In the remainder of this section, we discuss how our implementation of MULE transmits data over the IR channel, manages encrypted copies of sensitive files under multiple keys, provides automatic access to encrypted files, and protects users who forget to close sensitive files.

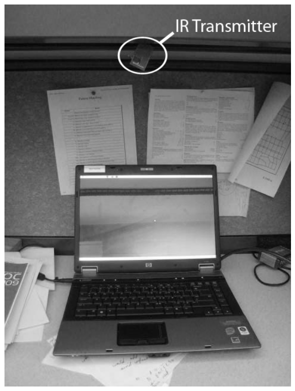
3.6.1 Transmission of Data Over the IR Channel

Our constrained channel uses a simple on/off encoding with two frames per bit (i.e., LED brightness above a threshold for 2 consecutive frames means 1). The laptop performs 3 steps to capture a message over the constrained channel: locate the position of the LED, tell the TLD to begin transmission, and decode the message. When the TLD is not transmitting a message, the LED repeatedly transmits the sequence 1010 to help the laptop determine the position of the LED. Once the camera is on, the laptop records six frames

[4]`http://www.arg0.net/encfs`
[5]`http://www.schmalzhaus.com/UBW/index.html`



(a) IR transmitter



(b) An Example Trusted Location

Figure 3.4: The IR Transmission Device and the Desk Setup as a Trusted Location.

and looks for pixels that follow an alternating on-on-off-off pattern. Any pixels that match the pattern are considered the LED and are used to build a mask such that any other pixels are ignored. With the mask determined, the laptop tells the TLD to begin transmission. The TLD generates a 20 bit random sequence and prepends 0101 as the header. The laptop records the output from the masked images (i.e., ignoring any non-LED pixels), looking for the 0101 start sequence followed by 20 bits for m and quitting once the LED returns to transmitting 1010.

3.6.2 Location Independent Key Implementation

Our laptop includes an Infineon SLB 9635 TT v1.2 TPM which allows our implementation to protect the RSA decryption key with the user's password such that a party needs that TPM and the password to decrypt the location-independent key. Our implementation uses the TCG Software Stack (TrouSerS)[6] to interact with the TPM and our own code to manage the encrypted location-independent key (K_{Ind}).

[6]`http://trousers.sourceforge.net`

3.6.3 Multiple Keys & Encrypted Filesystems

In MULE, a single encrypted filesystem is protected under the location-independent key (K_{Ind}) and at least one location-dependent key ($K_{Loc_1}, K_{Loc_2}, ...$). Knowing one of the keys should grant access, but the different keys produce different cipher text (thus different values). Storing multiple copies of the filesystem, each encrypted under a different key, wastes space and can lead to outdated information when one copy is changed and the keys needed to access the other copies are unknown. Our approach is to generate a random filesystem key (K_{FS}) to encrypt the filesystem. MULE uses OpenSSL to encrypt copies of K_{FS} under the location-independent key and any relevant location-dependent keys (i.e., $\{K_{FS}\}_{K_{Ind}}$, $\{K_{FS}\}_{K_{Loc_1}}$, $\{K_{FS}\}_{K_{Loc_2}}, ...$). Once MULE acquires any of the keys, it can decrypt K_{FS} and mount the sensitive files using EncFS. MULE also stores any location relevant information (the public key for HKD, the certificate for CKD, and various long-term secrets) with each encrypted key to simplify key derivation (e.g., identify the TLD needed to derive the key to decrypt K_{FS}).

3.6.4 Encrypted Filesystem Management

One of the major goals in this work is to reduce user overhead associated with accessing encrypted files. This includes both mounting the encrypted filesystem when the user tries to access sensitive files and unmounting the filesystem to ensure no sensitive information is leaked when the laptop is lost.

3.6.4.1 Mounting the Filesystem

To make accessing the encrypted filesystem unobtrusive, we replace the folder that contains the decrypted files (i.e., the mount point of the encrypted filesystem) with a MULE script. When a user double clicks on the "folder", our script runs, generates a key for EncFS, overwrites the script with the mounted filesystem, and opens the filesystem in a new window. When using a terminal, a user must execute the "folder" rather than simply change directories (i.e., "`cd ~/; ./sensitiveFolder`" rather than"`cd`

∼/`sensitiveFolder`").

3.6.4.2 Unmounting the Filesystem

To protect data, we need to ensure that the encrypted filesystem is unmounted before an OT can steal the laptop. If the encrypted filesystem is left mounted when the laptop is put to sleep, an OT could steal the laptop, wake up the system, and access the files without having to discover a key. We could leave unmounting the filesystem up to the user. However, it is dangerous to assume a user will remember to unmount a filesystem that is automatically mounted. To prevent leaks, MULE installs a number of scripts to automatically unmount the filesystem, re-insert the scripts which perform automatic mounting, and closes applications accessing sensitive files when the user logs off, has been idle for too long, or the laptop goes to sleep (e.g., the user closes the laptop and leaves the trusted area). We recognize that automatically closing applications negatively impacts usability. Fortunately, auto-save functionality will help reduce the loss of data due to unexpected application termination. In addition to a script launched at log-off, our implementation replaces the Gnome screen-saver with a copy of our script and places a copy of the script in `/usr/lib/pm-utils/sleep.d/`. Respectively, these scripts ensure that, once the screen-saver starts (i.e., the user has been idle) or the machine sleeps or hibernates, the sensitive files will be unmounted.

3.7 Evaluation

In this section, we evaluate the performance of our implementation. We first discuss the amount of time from when a user clicks on an encrypted filesystem and when the filesystem is mounted and displayed in a new window. We also discuss how our implementation performs when an attacker attempts to guess the secondary password associated with the location-independent mechanism.

Operation Average (Standard Deviation)	Home Delay in ms (σ)	Corporate Delay in ms (σ)	Loc-Ind Delay in ms (σ)
Connect to TLD	10.40 (3.69)	140.72 (8.6)	11.71 (4.671)
Film Sequence	4329.85 (17.12)	4378.99 (42.781)	1671.19 (39.518)
Decode Sequence	30.24 (5.33)	41.01 (8.58)	—
Calculations & Send Data	2.13 (0.05)	1.96 (0.09)	—
TLD Operations	47.62 (15.54)	19.54 (14.45)	—
Unblind Result	0.06 (0.002)	—	—
Close TLS	—	5.36 (3.48)	—
TPM-based Decryption	—	—	981.5 (5.01)
Decrypt K_{FS}	9.57 (0.13)	9.66 (0.18)	9.63 (0.15)
Mount FS	10.92 (0.26)	10.90 (0.30)	11.1 (0.25)
Open Window	78.3 (2.5)	79.2 (2.7)	76.8 (2.3)
Total	4627.6	4803.6	2806.6

Table 3.1: Average Time and Standard Deviation for Various Operations for HKD, CKD, and the Location Independent Mechanism (After HKD Fails)

3.7.1 Time to Mount an Encrypted FS

Table 3.1 contains the time in milliseconds for various operations associated with each protocol. The values here represent the average and standard deviation across 20 runs of each protocol. In the remainder of this section, we discuss a number of results. The home protocol is faster than the corporate protocol due to less computation. The location-independent mechanism provides the fastest key recovery mechanism. The performance is slower than expected for some operations because we are using a bash script to call a number of different programs. Finally, our key derivation protocols require almost 5 seconds, but our constrained channel limits the potential speedup.

When comparing the performance of the HKD and CKD protocols we find that the home protocol is faster due to less computation. The major differences are the result of the connection setup and the computation performed by the TLD. HKD has a faster setup since TLS is not used. However, HKD requires more computation for the TLD because key derivation involves signature generation. With our current setup, both the blind signature and TLS use 2048-bit RSA keys. However, with TLS the keys are used to sign ephemeral Diffie-Hellman values which are then used to establish a shared key. This connection setup takes roughly 141ms versus the 10ms in HKD. During actual key derivation, HKD is slower since the TLD must generate a signature as opposed to the symmetric operations associated with TLS and the MAC-based key derivation.

There are two reasons why the location-independent mechanism is fastest: the constrained channel has long setup times and a slow transmission rate and we ignore the time associated with entering the password. For the home and corporate protocols, the majority of the time associated with key derivation is "Film Sequence" which includes video initialization, detecting the LED, and capturing the sequence. Video initialization takes on average 1212.8 ms. Once the program is able to capture frames, the system records 11 frames—5 to allow for automatic brightness adjustment and 6 for LED detection—in 466 ms. The system then takes on average 2706 ms to record the next 60 frames or 30 bits at 2 bits/frame. After filming these frames the next 30 ms are used to decode the 20 bit sequence m. In total, the use of IR and a webcam as a constrained channel consumes almost 4.5 seconds. In comparison, the location-independent mechanism spends 1208 ms initializing video and 518 ms capturing frames to determine no LED exists (1.7 of the total 2.8 seconds). During evaluation, our script read the secondary password from a file to remove human variability from the results. However, spawning a password entry window and manually entering the secondary password would add significant time to the location-independent protocol that is highly dependent on the user's ability to quickly remember and type in the secondary password.

By using a bash script to call a number of separate programs some operations take longer than expected. Specifically, the script calls our programs which perform the home or corporate protocol, our TPM-based decryption program if automatic key derivation fails, the OpenSSL command line tool to decrypt K_{FS}, EncFS to mount the filesystem, and the file browser Nautilius to open a window with the mounted filesystem. As a result, a number of generally fast operations contribute considerable overhead. For example, decrypting K_{FS} requires a single AES decryption, but takes almost 10 ms in our current implementation.

Overall, the automatic key derivation schemes can provide access to files in less than 5 seconds. This does seem like an exceptionally long time, but is only incurred during the first access to a set of sensitive files. Once the files are mounted, the only additional overhead is a result of using EncFS. The majority of that time is associated with receiving the location-specific information. If we were to use a different constrained channel or use an IR receiver rather than a webcam, we could increase the transmission rate and speed up the protocol while maintaining the same security level. However, these other options require additional hardware for the laptop, a requirement that contradicts some of our initial design goals.

3.7.2 Attacks on the Location-Independent Password

For the location-independent mechanism to remain secure, the TPM must implement some type of defense against guessing the password. Prior work indicates that only some TPMs have such a defense [80] so we want to evaluate what defenses are present on the Infineon TPM. To test the defenses, we send a series of decryption requests to the TPM with the wrong password and measured how long each guess takes. We also periodically send a correct password to determine how that impacts the defense.

Our test finds that on average testing a wrong password requires 626ms. After a single wrong password, the TPM enters a lockout period where it refuses to respond to even the correct password for more than 2 minutes. Even after removing the battery and power from the laptop, the defense continues to ignore requests. This represents positives and negatives with respect to MULE. With such strong TPM defenses, attacking a limited entropy secondary password will take a long time. For example, consider an attacker trying to guess the password. An 8 character user-selected password has on average 24 bits of entropy [13]. Without the TPM, a modern computer that can perform half a million guesses a second (2^{19}) would take 2^{23} guesses on average or 2^4 seconds (less than one minute) to recover the password. With the Infineon TPM, an attacker can only perform roughly one guess every 2^7 seconds and requires roughly 2^{30} seconds or $\approx$ 34 years to discover the secondary password. However, these defenses will also impede a user who has trouble recalling a password or accidentally mistypes a password.

3.8 Discussion

If we use a non-migratable binding key on the TPM to encrypt a file decryption key, the current TPM is the only TPM that can access the key. If this TPM were to fail, the user would be unable to access files outside of trusted locations. Users should back up files in case of laptop loss in a way that does not rely on the laptop or the laptop's TPM. With a backup, the user can copy sensitive files to a new laptop and use the new TPM and a new location-independent key.

For both HKD and CKD, TLD secrets are needed to derive keys. To ensure continued operation of the location-based protocols, users or company IT personnel could copy the secrets onto some other medium. If a TLD ceases to function, users could copy the secrets to a new TLD. Home users could simply store the TLD secrets on a USB drive in the trusted location.[7] IT personnel could store the data in a company safe behind locked doors so physical security could prevent unwanted access.

[7]Without access to the location, an OT is unable to access the contents of the USB drive.

3.9 Summary

Users and corporate IT personnel want security solutions that simply work and want to avoid any schemes that require additional effort or administrative overhead. In this work, we designed Mobile User Location-specific Encryption (MULE), a system that requires zero user effort and limited IT administration in the common case to protect sensitive data on a laptop. MULE remains secure when facing an Outsider Thief (OT), our model of a laptop thief. Based on the observation that the majority of accesses to sensitive documents occur while located in a trusted location, we designed the Home Key Derivation and Corporate Key Derivation protocols which allow a laptop to automatically derive the key needed to access sensitive les based on the physical location of the laptop.

Our implementation of MULE on commodity provides automatic protection of sensitive les with limited delay during the initial access (i.e., less than 5 seconds to automatically derive the key and decrypt the les). As such, MULE represents scenario where verification of the physical receiver (the laptop receiving location-specific information) helps improve security.

Chapter 4

Background on Vehicle-to-Vehicle Networks

This thesis presents an approach to verify the physical source of information in wireless vehicle-to-vehicle (V2V) networks. The purpose of this chapter is to provide background information on vehicle-to-vehicle communication and its application (Section 4.1) and the associated standards (Section 4.2). These standards provide a good foundation for security, but still leave a number of unfulfilled requirements. After discussing the requirements (Section 4.3), we highlight proposed key management schemes meant to fulfill these requirements in Section 4.4. We discuss additional related work in Chapter 7.

4.1 Applications of Vehicle-to-Vehicle Communication

Vehicle-to-Vehicle communication can support safety applications, platooning, convenience applications, and commercial applications.

In 2009 over 33,000 people in the US were killed in vehicular accidents [65] and transportation consumed 13.27 million barrels of fuel per day on average [12]. Advanced technology packages with blind spot detection, pre-collision braking, autonomous cruise control, and other capabilities represent one way to reduce fatalities and fuel consumption.[1] However, these optional packages require radar, lasers, or other expensive sensors, with costs close to a thousand dollars, and are thus unavailable in economy vehicles. Wireless vehicle-to-vehicle (V2V) communication represents an inexpensive alternative to reduce the number of accidents via safety applications and reduce fuel consumption via platooning. Vehicular-ad-hoc-network (VANET) safety applications leverage periodic wireless beacons from nearby vehicles to alert the driver of dangerous situations. For example, a receiver can use the position, speed, and acceleration of nearby vehicles from these beacons to warn the driver of a vehicle in a blind spot or a vehicle further up the road performing emergency braking [3]. Platooning leverages V2V communication to allow a series of cars to semi-autonomously drive in a tight formation on a highway, reducing accidents due to driver error and reducing fuel consumption by as much as 15 to 20% [6, 103].

In addition to the aforementioned applications, Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication also support a number of convenience and commercial applications [3]. However, for the remainder of this work we will focus on verification of vehicles as senders for the aforementioned platooning the safety applications. Convenience applications allow toll payments at high speeds, alert drivers of congestion, or help find parking lots with available spaces. Commercial applications allow advertisements for nearby services (e.g., nearby restaurants or prices at nearby stations) or sharing of multimedia. These applications will initially drive the adoption of V2V, given that safety and platooning applications will not function properly when legacy vehicles (those without V2V capabilities) are abundant. For example, if drivers rely on electronic warnings to detect vehicles in blind spots, a drive may fail to notice the presence of a legacy vehicle and cause an accident. The same is true for platooning since when the lead vehicle changes lanes, the computers in following vehicles need to know if the adjacent lane is clear before following the leader and chang-

[1]`http://blog.ford.com/article_display.cfm?article_id=29188` or `http://www.jdpower.com/autos/articles/Adaptive-Cruise-Control/`

ing lanes. However, the convenience and commercial applications can work under sparse deployment (e.g., high speed toll payment only requires the driver's vehicle and the toll collection authority to have communication capabilities). In this work, we focus on the safety applications and platooning since malicious activity against those applications can cause physical harm to the drivers. The convenience and commercial applications also require security (e.g., prevent an impostor from stealing money from a driver's toll account money or lying about parking status or gas prices to lure a vehicle). However, authenticating the source of those messages (i.e., the toll authority, a gas station, or parking lot) is much easier since that is a fixed identity at a fixed location (Section 4.3 discusses the challenges associated with key management for vehicles). As such, for the remainder of this work, we limit our discussion of V2V applications to VANET safety applications and platooning and their associated challenges.

4.2 Vehicle-to-Vehicle Standards

In this section, we describe the current state of standards as they pertain to V2V safety and platooning applications.

In 2003, the FCC allocated the 5.850-5.925 GHz band for Dedicated Short Range Communications (DSRC).[2] DSRC allows wireless communication between vehicles' On-Board Units (OBUs). The physical and lower layer standards for this communication are defined in 802.11p [47]. Compared to other 802.11 standards, 802.11p uses a set of parameters selected to improve performance at vehicular speeds.

V2V safety applications leverage periodic broadcasts to collect information about nearby vehicles. Every 100-300ms each OBU broadcasts a beacon message which contains the OBU's location, heading, speed, and acceleration [91]. Receiving vehicles monitor these messages and alert the driver if the information represents a hazardous situation [16] (e.g., the vehicle is changing lanes but a message reports a vehicle in the blind spot). Given erroneous messages can cause accidents or convince drivers to ig-

[2] `http://wireless.fcc.gov/services/index.htm?job=about&id=dedicated_src`

nore the system, an IEEE standard exists to secure those messages. IEEE 1609.2 [48] defines the format for an Elliptic Curve Digital Signature Algorithm (ECDSA) signature appended to each message and the format for periodically broadcasting the certificate that allows receivers to verify the message. The addition of a Public Key Infrastructure (PKI) to identify valid sources and digital signatures to detect modified messages is a good start towards securing V2V. However, these steps only ensure that something with knowledge of a certified key pair is the source of a message. There is no guarantee the data in the message is correct or that a vehicle was the source of the message. The goal of our work on verification for V2V is to ensure that a vehicle created a message and to verify the content of the message. In addition to challenges associated with verifying a vehicle generated a message, broadcasting 10 signed messages a second that contain a vehicle's position enables tracking of drivers. Consumers are unlikely to adopt V2V technology if it violates their privacy. In the next section, we discuss a number of requirements that remain to ensure proper operation and successful adoption of VANET.

At this time, vehicle platooning lacks a standard, but could operate with minor modifications to the VANET standards. Platooning requires the same information as the safety applications (location, speed, heading, and acceleration). Different platooning systems have proven successful with information from only neighboring vehicles in the platoon [89] or the lead vehicle and the neighboring vehicles [73, 88, 104]. As such, the majority of platoon operation can use the VANET messages to operate. However, a new standard is needed to handle platoon formation, a vehicle joining, and a vehicle leaving. Given the limited set of additional commands needed, only the addition of a few bits are needed to provide platooning support. For the remainder of this work, we assume platooning can operate with the same set of beaconing messages as VANET safety applications and focus on how to enable sender verification within that standard of operation.

4.3 Vehicle-to-Vehicle Security Requirements

A PKI and digitally signing every message provide a foundation for securing V2V applications. However, this fails to fulfill all of the properties needed for V2V. In this section, we present a summary of the required properties that were originally proposed in our prior work [96]. The properties are: sender validity and message integrity, short-term linkability, long-term unlinkability, and traceability. Prior work on securing VANETs relies on key management and detection of misbehavior to determine when an authority should revoke a key. In the next section, we discuss the proposed key management solutions and if each fulfills the properties. We discuss misbehavior detection for V2V in Chapter 7.

Sender validity and message integrity. A recipient should be able to verify that a message came from a valid OBU (i.e., a vehicle). In addition, the recipient should be able to verify that the message has not been tampered with in transit.

Sender validity and message integrity often are also referred to as *authenticity*. Authenticity prevents malicious outsiders (e.g., an attacker with just a laptop and a radio) from injecting bogus messages that might disrupt the normal operation of the VANET. It is important to note that authenticity does not imply correctness of the contents of the message; only that a vehicle sent the message and that its contents were not modified.

Short-term linkability. When the same sender sends two or more messages within a small time frame, a recipient should be able to verify that these messages came from the same sender. We would like to enforce short-term linkability in a way such that a malicious OBU cannot launch a Sybil attack [31] where a single OBU poses as multiple vehicles. Short-term linkability is a desirable property in several VANET applications [37] and necessary for existing misbehavior detection proposals [36, 51, 75].

Short-term linkability does not hurt drivers' privacy. Vehicles' mobility patterns are constrained by roads, other vehicles, and physical limits (e.g., limits on braking, accelerating, or turning). If a vehicle is detected at some location X at time t, then at $t + \delta t$ (where δt represents a small time increment) the vehicle must be in the vicinity of location X. Therefore, being able to track a vehicle in the short-term does not impact drivers' privacy.

Long-term unlinkability. A basic privacy requirement is that a wireless message does not reveal the driver's name, the vehicle's license plate number, or other personally identifying information. This is easy to achieve if the certificate authority (CA) does include such information in each certificate. However, an anonymized certificate fails to protect a driver's privacy. If each vehicle is assigned a single unique certificate, observation of the same certificate at different locations at different times allows a receiver to link those messages and infer that the same vehicle has been at both places at the times of the observations.

As such, a security solution for V2V should provide long-term unlinkability. With long-term unlinkability, if the same OBU sends two messages m and m' more than δt time apart, then an adversary should not be able to determine if the same sender generated both m and m' based on message contents and where the messages were received. An attacker may be able to use RF fingerprinting [35] or additional information besides for the content of the wireless messages to track a vehicle. However, preventing privacy violations based on that information is outside the scope of this work.

Traceability. If an OBU misbehaves, an authority should be able to trace the identity of the misbehaving OBU from a transcript of the messages sent. The exact definition of misbehavior depends on the application and can be detected using various techniques. We discuss misbehavior detection for V2V in Chapter 7.

The exact response to misbehavior is a policy decision once an authority has identified the misbehaving entity. Fines or other punishments can help act as a deterrent, punishing the offender and convincing others not to misbehave. Revocation (removal of the offending identity from the set of valid participants) prevents future misbehavior from the same identity from impacting the V2V network. However, revoking a vehicle means it will be a physical vehicle on the road that is unable to participate in V2V applications. This represents the same dangerous situation as a legacy vehicle.

4.4 Proposed Key Management Solutions

Prior work on fulfilling the aforementioned security properties has focused on proper key management. We present each key management scheme based on the technique used and discuss if the approach fulfills the properties listed in Section 4.3. Specifically, we discuss prior techniques that use trusted hardware, linked certificates, sharing of multiple keys, group signatures for every message, group signatures to generate self-signed certificates, and group signatures to request certificates. We present a summary of this discussion in Table 4.1.

For this work, we assume that periodically changing keys and certificates provides long-term unlinkability. Existing work has shown that randomly changing keys fails to protect a driver's privacy when facing an adversary that can eavesdrop on the majority of a vehicle's beacons [14, 34, 82]. Mix-zones (areas where multiple vehicles simultaneously change keys) or silent periods (short intervals without messages) are needed to achieve privacy when facing attackers with large networks of monitoring hardware. However, these approaches negatively impact the safety applications [41]. Currently, there is no solution that provides privacy in all cases with minimal impact on the applications. As such, we assume access to several keys and the ability to periodically change keys is sufficient for long-term unlinkability.

This summary indicates that existing schemes are prohibitively expensive, fail to fulfill all of the requirements, or sacrifice availability to fulfill the requirements, assuming detection of key theft is fast and accurate. The major issue is that none of the schemes verify that a cryptographic key corresponds to a physical vehicle. Instead, the solutions begin by assuming only vehicles have access to keys and hardware protections or timely detection and revocation ensure non-vehicle entities are unable to use the keys. As such, further work is needed to provide an ideal security solution for V2V safety applications and platooning.

Scheme	Sender Validity	Message Integrity	Short-term Linkability	Long-term Unlinkability	Traceability
Trusted HW [76]	●	●	●	●	●
Linked Certificates [40]	◐	●	●	●	●
Shared Keys [108]	◐			●	
Group Signatures [8]	◐	●		●	●
Group Signature Signed Certificates [15]	◐	●		●	●
Group Signatures to Request Certificates [96]	●	●	●	●	●

Legend: ●Fulfills Property ◐Partially Fulfills Property

Table 4.1: A Comparison of Prior V2V Key Management Schemes

4.4.1 Secure Hardware

Raya et al. [76] propose the use of tamper-proof hardware to secure VANETs. An OBU's secure hardware stores a large number of ECDSA key pairs and certificates and computes the digital signatures for the OBU. This ensures only the intended vehicle has access to the key. Besides for the tamper-proof hardware, this approach functions like the IEEE standard. Provided key extraction is infeasible, this approach fulfills the aforementioned properties. The major drawback is that secure hardware is prohibitively expensive.

Sender validity is guaranteed since extraction of a vehicle's private keys is infeasible, limiting V2V participation to vehicles, assuming the tamper proof hardware can verify it is still in a vehicle. Since only a single vehicle knows the private key, the signature appended to each message provides message integrity and traceability. The secure hardware controls when keys are used, forcing the use of the same key over a short period of time for short-term linkability. After some interval, the hardware uses a new key to sign messages to prevent tracking and provide long-term unlinkability. The authority keeps a record of the keys given to each vehicle and can use the signature attached to a message to determine the original source of the message (traceability). The downside to this approach is that secure hardware is prohibitively expensive. For example, the current state of the art tamper-proof hardware, the IBM 4764 [46], costs on the order of $9,000.

4.4.2 Linked Certificates

Haas et al. [40] suggests using linked certificates to allow efficient revocation of certificates. This way quick detection and revocation ensures vehicles are using keys correctly. Otherwise, the protocol works the same as the IEEE standard. However, if a vehicle lacks the connectivity to download new revocation information, a non-vehicle can use a stolen key—one that should be revoked—to sign messages and violate sender validity.

To make revocation more efficient, all of a given vehicle's certificates contain an identifier that conditionally links the certificates. This way an authority only has to disclose a small piece of information to revoke all of a vehicle's keys. The authority uses a Pseudo-Random Number Generator (PRNG) with a secret seed to produce a given vehicle's identifiers. Before a vehicle is revoked, the randomness of the PRNG ensures that receivers are unable to tell if two certificates belong to the same vehicle. To revoke a vehicle, the authority discloses the seed used to generate the identifiers. With knowledge of this seed, receivers can identify all of the revoked vehicle's certificates.

Given that normal computing hardware is used, a vehicle can now use any key at any time. To ensure short-term linkability and prevent Sybil attacks, the authority provides a vehicle with certificates with non-overlapping periods of validity; only one certificate is valid at a given time.

This approach provides the same properties as the trusted hardware scheme at a much lower cost. ECDSA signatures in messages ensure integrity. Numerous non-overlapping certificates provide short-term and long-term linkability. Changing to a new certificate (that the authority has not revoked) prevents tracking. The authority keeps a mapping between vehicles and certificates to trace misbehavior. The drawback with this scheme is the need for connectivity to receive timely revocation information. If a receiver has outdated revocation information, a malicious party may extract keys from a vehicle and use those keys to pose as a vehicle (violating sender validity).

4.4.3 Shared Keys

With shared keys, multiple vehicles are assigned the same certificates and ECDSA key pairs [108]. This provides strong privacy properties and is the only technique that prevents a malicious authority from violating drivers' privacy [41]. However, it fails to provide any of the other properties.

When using shared keys, the authority assigns the same ECDSA key pairs and certificates to multiple vehicles. The vehicles use those keys and certificates to verify messages according to the IEEE standard. The authority begins by generating a large pool of keys and certificates. Whenever, a vehicle needs keys, the authority randomly selects a subset of the pool, gives a copy of those keys and certificates to the vehicle, and records what keys were assigned to the vehicle. The authors do not suggest the use of non-overlapping keys to limit when a given certificate is valid.

This approach provides the strongest privacy properties. Vehicles can periodically change certificates to provide long-term unlinkability. Since multiple vehicles own the same certificates, even an authority is unable to track a vehicle's movements. Both modification of messages and Sybil attacks are possible. If a nearby vehicle knows the same key as a sender, the two vehicles can impersonate each other to violate short-term linkability and message integrity. With over-lapping certificates a single vehicle can use multiple certificates at once to launch a Sybil attack. Finally, if a malicious party ever extracts a key and tries to use it as a non-vehicle, the authority must either revoke the key, possibly revoking keys for valid vehicles, or allow the key to remain, violating sender validity.

4.4.4 Group Signatures

Group signatures are a cryptographic tool that allows members of a group to anonymously sign a message. Boneh and Shacham proposed using group signatures with verifier local revocation to secure V2V applications. Under their approach, a vehicle signs every V2V message with a group signature. To revoke a vehicle, the authority publishes a revocation token that allows a receiver to de-anonymize the group signatures from the revoked vehicle. This way quick detection and revocation ensures only vehicles are generating messages. The approach provides strong privacy properties, but fails to ensure sender validity or short term linkability.

A group signature is anonymous in that a receiver verifying a signature knows a member of the group generated the signature, but is unable to determine which member of the group generated the signature. A receiver is only able to isolate a specific group member after the authority has revoked that sender.

The anonymity of group signatures ensures long-term unlinkability. At the same time, an authority is able to de-anonymize the group signature to trace misbehavior. In addition to requiring connectivity to distribute revocation information (making sender validity a challenge), group signatures fail to provide short-term linkability. Since a group signature only indicates a message is from a group member, receivers are unable to link subsequent messages back to the same vehicle over a short period of time.

4.4.5 Group Signature Signed Certificates

Calandriello et al. [15] proposed using group signatures to generate self-signed certificates in VANETs. With this approach, a vehicle generates its own ECDSA key pair and uses a group signature to identify that pair as valid. This was designed to improve efficiency compared to a purely group signature approach (group signature operations are over an order of magnitude slower than ECDSA signatures). However, the scheme is unable to provide more properties than the basic group signature based scheme.

Since only one vehicle knows the ECDSA key pair, receivers can now link any messages signed by the private key back to that vehicle. However, the anonymity of the group signature allows a sender to craft many certificates at the same time as part of a Sybil attack, violating short-term unlinkability. Given the need for connectivity to revoke vehicles, periods without connectivity permit attackers with stolen and revoked certificates to produce beacons that appear valid.

4.4.6 Group Signatures to Request Certificates

We previously proposed using group signature as a way to request short-lived certificates from an authority [96]. Once a vehicle's key is revoked, the authority will no longer provide a certificate in response to a request. This fail-safe approach fulfills all of the security requirements, but suffers from a lack of availability if a vehicle is unable to connect to the authority.

In this approach, a vehicle generates its own ECDSA key pair and uses a group signature to anonymously request a certificate for that pair from an authority. If the requester has not been revoked, the authority returns a certificate that is valid for a short period of time. The vehicle uses the certificate and the corresponding ECDSA key pair to broadcast verifiable messages. If the requester has been revoked, the authority refuses to return a certificate. We also modified the group signature such that the authority can detect multiple requests from the same group member within a short window of time.

This approach provides all of the properties needed, but is too restrictive and can negatively impact V2V performance if a connection to the authority is not available. Since a single vehicle is limited to a single certificate at a time, this approach provides sender validity, message integrity, and short-term linkability. An attacker is unable to violate sender validity with a stolen key since the authority can detect the theft and will refuse to return a certificate for the revoked vehicle. The ability to anonymously acquire new certificates for random ECDSA keys provides long-term unlinkability. The authorities ability to de-anonymize group signatures allows tracing of misbehavior. The drawback to using this approach is that vehicles require connectivity to request new keys from the authority. If connectivity is not available, vehicles will be unable to receive new certificates and unable to send V2V messages. Downloading multiple keys at a time improves availability, but delays detection of key theft and violates sender validity.

Chapter 5

Using a Visual Channel to Verify Senders in Vehicle-to-Vehicle Applications

For vehicle-to-vehicle safety and platooning applications to work (see Chapter 7 for more details on the applications), a vehicle needs accurate information about nearby vehicles. If the vehicle receives incorrect information due to malevolence or a malfunctioning sensor, the technology may cause more accidents than it prevents (e.g., a vehicle claims to be accelerating when in reality it is braking, resulting in a rear end collision).

Existing work on securing V2V communication and verifying messages (see Section 4.4) focuses on using a PKI and signed messages. However, a PKI does not ensure that a message corresponds to a physical vehicle. This is another scenario where the name of the wireless identity is irrelevant, especially since the certificates are anonymized to prevent privacy violations. The name in the certificate is a random string. What a receiver needs to verify is that the messages accurately describe the physical world. A signature and the certificate merely indicate that a party with knowledge of a not-yet-revoked private key pair generated a message; a vehicle can still broadcast false information or some other transmitter that has extracted valid keys can broadcast a message that is cryptographically valid. Only after such attacks are detected and reported does a key management scheme provide any mitigation by revoking the offending keys. In addition, revocation of keys that have been misused requires connectivity to the authority, which may be limited in a vehicular network.

In this chapter, we discuss how to verify that wireless messages accurately describe the physical source of the message. Instead of the traditional detect and response paradigm, our approach is to verify the correctness of the information to prevent attacks. In addition to fulfilling the security properties needed for V2V (see Section 4.3), we address two attacks where malicious parties broadcast messages that inaccurately describe the physical state of vehicles: ghost vehicles and impersonation attacks. A ghost vehicle occurs when a malicious party claims there is a vehicle at some coordinates, but there is no actual vehicle. A vehicle lying about its location or some other transmitter (e.g., a laptop on the side of the road) claiming to be a vehicle can create a ghost vehicle. Impersonation occurs when there is a vehicle at some location, but another transmitter broadcasts a beacon claiming the same location. By detecting these attacks, we also mitigate the threat of Sybil attacks (one physical vehicle claiming to be multiple vehicles) [31].

Our solution leverages relatively inexpensive cameras[1] in the front and back of the vehicle to monitor the physical world and see nearby vehicles. When a ghost vehicle attack occurs, the receiver can detect that no vehicle exists at the claimed location, mitigating the attack by ignoring the associated messages. However, impersonation attacks are still possible with only a camera and wireless messages. Specifically, another sender can still claim to be that vehicle in view. As such, we need a way to verify the physical vehicle is the source of the data. To achieve this, we leverage existing work on how to use blinking LEDs on a vehicle and a receiver's camera to establish a visual communication channel that also supports ranging and tracking [1, 81].

[1]These cameras may be standard in the near future (`http://articles.latimes.com/2010/dec/04/business/la-fi-autos-backup-camera-20101204`).

However, we still need to use the wireless channel because inexpensive cameras and long-distance communication limit the bandwidth of the visual channel. As such, we use the visual channel to help verify information sent on the wireless channel. Specifically, the LEDs transmit a representation of the sender's certificate and are arranged in such a pattern that the receiver can estimate the relative position of the sender. As such, receivers know that: 1) a physical vehicle at that location is using a specific key pair, 2) only messages containing that location correctly describe that sender, and 3) only messages digitally signed using the correct private key were crafted by that vehicle. A malicious party can still broadcast false information (e.g., claim to be braking), but now receivers can attribute it to the correct vehicle on the road. By preventing, rather than responding to the attacks, our solution operates without requiring frequent connectivity to the authority for key updates [96] or revocation [40, 8, 15]. In addition, verification that a certificate corresponds to a vehicle, enforces short-term linkability, and allows the authority to sign certificates with overlapping periods of validity. Under prior V2V security approaches, overlapping certificates allow malicious parties to launch Sybil attacks. However, our approach mitigates these attacks via detection. This means vehicles can use a certificate whenever they want. This added flexibility to when vehicles change keys can help maintain privacy (long-term unlinkability) by using mix-zones [14, 34] or any future proposals. This also means vehicles require fewer certificates since the certificates will not simply expire while the vehicle is not in use.

Of course this solution has three significant drawbacks: the visual channel's limited bandwidth requires vehicles to be in view for a long time to be verified, the visual channel limits verification to vehicles in view, and a verified sender can begin to broadcast false location information once out of view. We present extensions to our system to address each of these issues and analyze how they impact security and operation.

We simulate our solution and the various extensions under realistic network and vehicular traffic conditions to determine if vehicles will be able to verify enough senders to support VANET and platooning applications. We specifically examine the **verification range** of vehicles. Verification range is the minimum distance between a receiver and a vehicle that has not been verified. In other words, every vehicle within verification range has been verified. Related work indicates that a verification range between 20 and 60 meters is necessary for VANET safety applications [39]. Our results indicate that deployment of our solution provides sufficient verification range to support the V2V applications.

5.1 Problem Definition

A V2V security solution must allow verification of messages while protecting a driver's privacy. Here we present a summary of the V2V security requirements from Section 4.3.

Sender Validity A vehicle generated a wireless message.

Message Integrity A wireless message was not modified.

Short-term Linkability A receiver can link a series of messages back to an individual sender over a short period of time. This also means a single vehicle is unable to pose as multiple senders at the same time.

Long-term Unlinkability Receivers are unable to determine if the same vehicle generated 2 different messages broadcast at 2 locations at distant times. **Traceability** An authority is able to determine which vehicle generated a message based on a transcript of the messages broadcast.

The goals of these properties is as follows. We must verify that a physical vehicle (sender validity) broadcast a series of wireless messages (short-term linkability) that have not been modified by another party (message integrity). At the same time, protection of drivers' routes and endpoints (long-term unlinkability) is needed for privacy and ultimately the adoption by consumers. In addition to using the prior properties to detect false information, an authority should be able determine what vehicle broadcast a message (traceability). Once the authority identifies a party which abuses V2V communication, punishments can act as deterrents to prevent further abuse.

Prior work has relied heavily on detection and revocation of invalid parties to ensure messages correctly describe physical vehicles (see Section 4.4).

These represent reactive solutions where only after an incident negatively impacts V2V is anything done. This chapter focuses on how to proactively verify that wireless data originated from and accurately describes a physical vehicle at a specific location, preventing any type of negative impact from false information. With this information verified, ghost vehicles and impersonation attacks (i.e., messages that claim to be a specific vehicle on the road, but are from some other source) are prevented.

In the remainder of this section, we describe the goals and capabilities of an attacker that is trying to subvert safety or platooning applications and state any assumptions we make about the system.

5.1.1 Attacker Model

An attacker's goal is to subvert the performance of a VANET safety or platooning applications by crafting messages with false information. Specifically, the attacker wants a receiver to believe messages about vehicles that do not exist (i.e., ghost vehicles) or messages that are impersonating a valid vehicle (i.e., the attacker claims to be at the physical location of a victim vehicle). This can result in the generation of false safety alerts (e.g., blind spot alerts for vehicles that do not exist), erroneous platooning behavior (e.g., the vehicle slowing down after receiving a braking message from an impersonating vehicle), or other undesirable and possibly dangerous conditions.

Attacker's may also want to violate drivers' privacy by eavesdropping on wireless messages to track a vehicle. However, for this chapter, we assume the ability to periodically change between different anonymous certificates is sufficient to provide long-term unlinkability. See Section 4.4 for more discussion on this topic.

An attacker can own a legitimate vehicle with all of the necessary wireless equipment and keying material. An attacker can also build his own wireless hardware and use keying material extracted from a vehicle. During an impersonation attack, the attacker does not have the victim's keying material. Prior work has shown that once physical access is possible, the attacker can remotely control the vehicle [53]. Given full control of the victim's vehicle, the attacker could force the victim to broadcast arbitrary information, making key theft unnecessary. As such, protecting the vehicle from physical attackers and key theft is an important problem, but outside the scope of this work.

The attacker has sufficient computing power to perform basic cryptographic operations, but is unable to break different cryptographic primitives in order to forge signatures.

Finally, only a small number of attackers will ever work together to successfully launch an attack. We analyze how tolerance of more attackers impacts the operation of our system later in this work.

5.1.2 Assumptions

We make some assumptions about general key management and the hardware available to vehicles.

We assume the basic key distribution according to the IEEE standard [48] such that each vehicle has a large number of keys and some corresponding certificates. Unlike prior approaches [40, 96], these certificates can have overlapping periods of validity, reducing the number of keys per vehicle and the complexity of the CA's role.

For VANET and platooning operation, vehicles have a wireless antennae, processing unit, and some means to determine the vehicle's current location. VANET standards use IEEE 802.11p to facilitate vehicle-to-vehicle communication. Platooning can also use this standard to facilitate wireless communication. These vehicles also possess some type of processing unit to generate and process messages. We can leverage this unit to perform some computation in addition to craft and cryptographically sign or verify packets. Vehicles must know their current location as part of VANET or platooning operation. As such, each vehicle will have GPS and some dead reckoning capabilities (or other means to determine location when GPS signals are lost).

In addition to the hardware for typical VANET or platooning systems, we assume vehicles have cameras and LEDs that can help verify messages in addition to supporting other useful functions. The installation of cameras can provide useful functionality in addition to supporting a visual communication channel. Rear facing cameras help prevent back-over accidents where vehicles accidentally reverse over unseen people. Forward facing cameras can be used to help increase visibility at night via night-vision and to help detect pedestrians and alert drivers. Both cameras can also be used to detect position in a lane, replacing the need for the installation of magnets on roadways for platooning [73]. We use LEDs as the source to complete the visual channel we need to help verify which vehicle generated a message. Vehicles are already starting to use LEDs to replace traditional incandescent bulbs, given LEDs longer lifetime and higher efficiency. With expensive high-speed cameras, existing LED headlight and taillights can be used to transmit information without distracting drivers by blinking lights at imperceptible rates [1]. If inexpensive cameras are used, the blinking rate must be slower to accommodate the cameras' slower frame rates. As such, additional LEDs that operate outside of the visible portion of the spectrum (e.g., IR LEDs) are needed to prevent driver distraction. Manufacturers can add these LEDs to the front and back of the vehicle at minimal cost.

5.2 Visual Vehicle Verification

The goal of visual vehicle verification is to allow a receiver to verify the correctness and physical source of a wireless V2V message as part of a VANET or platooning application. We leverage both computer vision and cryptography to associate a vehicle in video of the road with that vehicle's messages. Specifically, the sender uses lights on the vehicle to transmit a representation of its current certificate. Receivers use frames from the video to decode the sender's certificate information and to associate the cryptographic information with the physical vehicle in the visual frame. This cryptographic information allows the receiver to associate digital signatures in messages with that physical vehicle. This allows receivers to verify that a wireless message originated from a specific physical vehicle on the road. The position and spacing of the lights in the frame also allows the receiver to estimate the location of the sender, and verify that the claimed location matches the actual location.

However, this approach has three drawbacks that we must also address: limited bandwidth of the visual channel, line of sight verification limits the range of the V2V applications, and verified vehicles can misbehave once out of view. To address these drawbacks we propose broadcasting a shortened hash on the visual channel, leveraging visually verified vehicles as witnesses to report legitimate out-of-view senders, and expiring verification to prevent already verified vehicles from claiming false information.

In the remainder of this section we discuss how a basic version of our system works, provide a summary of the drawbacks to the approach, and how each of our extensions alleviates those drawbacks. In the next section, we provide an analysis to describe why our system is secure and to provide guidance on setting important parameters.

5.2.1 Matching Wireless Messages to Vehicles in View

In the naive solution (i.e., without witnesses), communication over the wireless channel is unchanged. For VANET applications and platooning, each vehicle periodically broadcasts signed messages that describe the physical properties of the car (e.g., location, heading, speed, and acceleration) [16]. At a lower frequency, each vehicle also broadcasts its current certificate to allow receivers to verify the validity of the signatures. Our protocol only impacts how receivers verify the validity of a certificate. In addition to verifying the validity of the signature in the certificate, a receiver also verifies that the source of the message is a vehicle with the corresponding certificate, at the location claimed in the message. The sender and receiver have different tasks to facilitate this verification of the physical entity that corresponds to the wireless message. The sender broadcasts its certificate on the visual channel. The receiver uses the information from the visual channel to associate a cryptographic key and a physical vehicle to a wireless message.

In addition to broadcasting information on the wireless channel, a sender

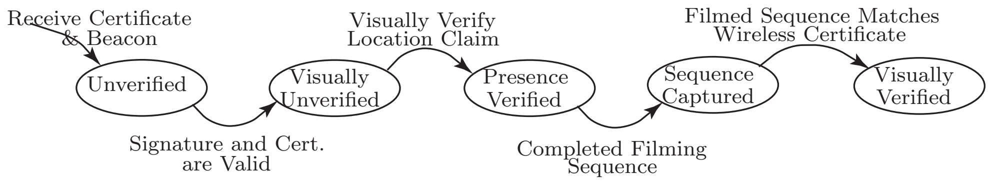


Figure 5.1: Receiver's State Diagram for the Verification of a Sender
(Untrusted State Removed for Clarity)

uses pairs of LEDs on the front and back of the vehicle to transmit the certificate it is currently using on the visual channel. By using a set of LEDs that are a standardized distance apart, receiving vehicles can use existing visual ranging techniques to estimate the relative position of the sender [81]. The sender only uses the visual channel to transmit its current certificate since the wireless channel is used to transmit the majority of VANET and platooning information.

Receiver operation is the same as traditional VANET or platooning once a sender has been visually verified. The steps taken to visually verify a sender with our system help prevent ghost or impersonation attacks. Specifically, a receiver maintains a database of potential senders (i.e., valid certificates) and updates the state of a sender based on the transitions in Figure 5.1. During verification, the sender transitions between six possible states: **Unverified**, **Visually Unverified**, **Presence Verified**, **Sequence Captured**, **Visually Verified**, and **Untrusted**. The **Untrusted** state is the starting state for any valid certificate and the default state if information is incorrect or the receiver is unable to complete the task needed to transition to the subsequent state in the verification process. Only once the sender reaches the **Visually Verified** state will the receiver process the messages as valid VANET or platooning messages.

Untrusted Any certificate that has been received and is valid is first considered as untrusted. To be valid a certificate must contain a valid signature from the certificate authority and be not-yet expired. In traditional VANET security models, a certificate that matches this criteria (and is not on a revocation list) is considered verified and any messages that are signed with the corresponding private key are used in the VANET applications. However, at this time there is no way for the receiver to know if the certificate corresponds to a valid vehicle on the road or is being abused by a malicious party.

When checks in the latter state transitions fail or the receiver is unable to complete the transitions (see subsequent paragraphs for descriptions of the checks), the sender transitions back to this state. A naive approach would be to discount a certificate associated with a failed check (i.e., an indication of misbehavior) and consider the sender as invalid. However, under that approach framing attacks could negatively impact operation. For example, a malicious party could repeat a valid certificate, but craft an invalid signature. The certificate will fail visual verification due to the invalid signature (**Unverified** to **Visually Unverified** transition). Instead, when one of these checks fail, we consider the certificate as untrusted, but not definitively malicious.

Unverified After receiving a beacon that is broadcast with a certificate (or matches a previously received certificate), the sender transitions to the **Unverified** state. If a beacon arrives before the receiver has heard the corresponding certificate,[2] the receiver simply discards the message and makes no changes to the state of any sender. The receiver's next step in the verification process is to check that the signature in the beacon can be verified using the public key from the certificate. If the signature is invalid, the sender transitions back to the **Untrusted** state.

Visually Unverified Once the receiver knows the claimed location of the sender from the beacon and knows the owner of the corresponding key generated the message (i.e., the signature on the beacon is valid), the sender is in the **Visually Unverified** state. The receiver's next step in the verification process is to scan the visual field and determine if the sender's

[2]Beacons without certificates include a hash of the corresponding certificate so receivers can determine what cryptographic information is needed to verify the included signature.

claimed location corresponds to any of the vehicles in view. If the vehicle is outside of the receiver's field of view (blocked by other vehicles, too far away, or to the side), the sender remains in the **Visually Unverified** state for one second. During this time, the receiver may acquire an updated location claim from more recent beacons with valid signatures. Each time this happens, the receiver will remain in the **Visually Unverified** state and reset the timer until the claimed location is in view. Eventually, the sender will exit radio range or the location will come into view. If the sender exits radio range, the timer will expire and the sender will transition back to **Untrusted**. Once the location is in view, the receiver can use ranging capabilities [81] to confirm that a physical vehicle is at or near the claimed location. If there is no vehicle at the location, the sender is **Untrusted**. If a vehicle is at the claimed location, the receiver has verified the presence of a sender, removing the possibility of a ghost vehicle attack. However, an attacker may be launching an impersonation attack, thus there are more steps to the verification process.

Presence Verified Once the receiver has verified a physical vehicle exists at the claimed location on the road, the sender has reached the **Presence Verified** state. The receiver's next step is to track the vehicle at the location to record the sequence from its blinking lights. If the sender drives out of visual range (due to an occlusion, moving besides the receiver, or driving too far away), the verification process fails and the sender returns to the **Untrusted** state. Occlusions due to other traffic seem short lived on the human time scale, but are several times longer than the time needed to record the sequence when using the techniques from Section 5.2.2. As such, we choose to restart the verification process once the occlusion has ended, rather than attempt to accommodate the obstruction. Once the receiver has completed filming the sequence, the sender transitions to the next state.

Sequence Captured A sender remains in the **Sequence Captured** state for a very short period of time while the receiver compares the visually received sequence to the certificate from the wireless channel. If the sequence and the certificate differ, this is an indication of an impersonation attack and the sender is moved to the **Untrusted** state. If the sequence and the certificate match, the receiver considers the sender as **Visually Verified**.

Visually Verified The final state in the verification process is the **Visually Verified** state. If a sender has reached this state, the receiver knows that the physical vehicle in its field-of-view is currently using the certificate from the wireless channel. The receiver also knows that any message that contains a signature that can be verified using the public key from that certificate was generated by that physical vehicle.

Our verification process ensures that the wireless V2V messages accurately describe the vehicles nearby, by using a combination of techniques to transmit data on the visual channel, track and range vehicles, and cryptographically bind the vehicle in view to the wireless messages it produces.

However, this visual verification technique suffers from drawbacks and vulnerabilities that negatively impact its performance and could allow malicious parties to successfully perform an attack. As such, the next few subsections describe these drawbacks and the extensions we propose to address them.

5.2.2 Visual Broadcast of a Shortened Hash

Prior work on using the visual channel for V2V communication assumes high speed cameras (100 fps or greater) to achieve high bandwidth communication at long range (100m) [1]. However, when constrained to the slower cameras with wider fields of view that are part of current cameras on vehicles (20 to 30 fps and 130 degree FoV), the effective bandwidth of the system is much less. As such, rather than broadcasting the entire certificate, senders can broadcast a shortened hash of the certificate to reduce the time needed to broadcast the information. Receivers then hash the certificate from the corresponding wireless messages and compare the first several bits of the hash to the sequence from the visual channel. In Section 5.3.2, we discuss how to select the minimum hash length to prevent attacks.

5.2.3 Leveraging Witnesses to Verify Out-of-View Senders

Even if only a small amount of information is ever sent on the visual channel, visual verification still requires the sender to be in the receiver's field of view for some period of time. This limitation reduces the efficacy of safety applications. However, if we remove the requirement of the visual verification for vehicles past a certain range, impersonation attacks and ghost vehicles are possible. Rather than giving up on strict verification for distant vehicles, receivers can use senders in the **Visually Verified** state to act as witnesses and help verify senders that are in the **Visually Unverified** state and out-of-view.

This extension is the only part of our system that requires additional communication on the wireless channel. To act as a witness, a vehicle forwards the hash of certificates for any senders that are currently in the **Visually Verified** state. By only forwarding senders that have been visually verified, senders that behave properly help isolate any ghost or impersonation attacks that do occur when W or malicious parties collude (Section 5.3 has more details).

Receivers have to be careful who they use as witnesses and how many witnesses are needed before trusting a non-visually verified vehicle. For example, a visually verified vehicle behaving maliciously could craft a ghost vehicle or impersonate another sender by claiming one of its keys represents a supposed vehicle out-of-view. If a trusted, but non-visually verified vehicle is not valid, then using it as a witness is likely to lead to the acceptance of more attacker controlled ghost or impersonation vehicles. Our solution is to use only visually verified vehicles as witnesses and to require at least a threshold number, W, of witnesses before a non-visually verified vehicle is trusted. Figure 5.2 includes the additions to the sender state machine to accommodate witness verified vehicles. Once W or more visually verified witnesses report a sender, the receiver transitions the sender to the **Trusted** state. If that sender enters the receiver's field-of-view, the sender can move towards the **Visually Verified** state. However, while a sender is in the **Trusted** or **Trusted Presence** states, a receiver will not consider

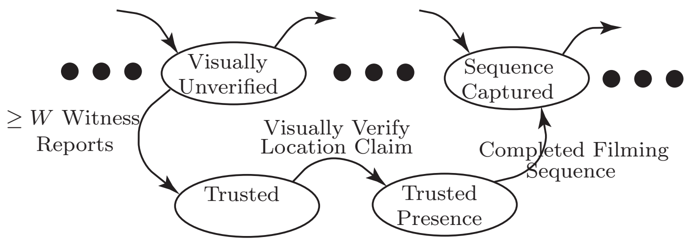


Figure 5.2: Extended State Diagram to Handle Witnesses

the sender as a valid witness. If a vehicle fails the check between states (e.g., the claimed location enters view, but no vehicle is there), the sender transitions to the **Untrusted** state. When using this extended state diagram, ghost or impersonation attacks are only possible if W or more real vehicles are colluding to fool nearby receivers.

One advantage to this technique is that it allows a receiver to trust vehicles that are on different segments of the road (e.g., on intersecting streets or merging onto the highway). This is very important for VANET safety applications like intersection collision warning [3]. By leveraging vehicles that have recently entered the same flow of traffic as the receiver, the receiver can begin to trust senders on intersecting roads and use V2V to prevent intersection collisions.

The drawback to requiring multiple witnesses is that in low vehicle density situations, there may not be enough witnesses present. If traffic density is high enough, our technique can work on a simple two lane road. With one lane of traffic in each direction, a receiver can leverage on-coming traffic as a stream of witnesses. However, if there is very little traffic, a lack of oncoming vehicles means the receiver will only have the vehicles directly in front or behind itself to act as witnesses, and is unlikely to have $\geq W$ reports.

5.2.4 Verification Expiration to Prevent Attacks

As presented our verification mechanism assumes that once a vehicle is verified or trusted, it is broadcasting correct information, and will continue to broadcast correct information. However, a malicious vehicle could behave at one time and later broadcast false information. For example, if a visually verified vehicle leaves the receiver's field of view, the receiver is unable to detect if the sender begins to send false information (e.g., claiming a physical location that is also out-of-view, but does not accurately describe the sender's location). A witness verified vehicle could also start sending false information after it has entered the **Trusted** state.

To address both of these threats, the receiver can require periodic evidence to support the validity of a sender. For vehicles in view, this means tracking a vehicle to verify the claimed location continues to match the physical location of the vehicle. As such, a vehicle will only act as a witness for a sender, if the witness is able to confirm the physical vehicle is still in the location from the most recent beacon and that the sender has not begun to use a new certificate. For vehicles out of view, this means a new report from a witness, which implies that the witness has verified that the sender out of view is broadcasting correct information. As such, a receiver sets a timer for a threshold time τ for each sender once the sender enters the **Trusted**, **Trusted Presence**, or **Visually Verified** states. Whenever, the receiver hears a new report from a witness or visually confirms that the sender's current location matches the information in a beacon, the timer is reset to τ. If the timer expires for a sender in any of these states, the sender transitions back to the **Visually Unverified** state. By requiring constant visual re-verification or reports to re-assert the validity of a sender, a receiver is able to detect ghost vehicle or impersonation attacks. This also allows a receiver to update what certificate is valid when a vehicle changes keys (i.e., the blinking sequence will change or witnesses will report new information). Finally, re-verification prevents Sybil attacks where the sender tries to use more than one identity at a time (more details in Section 5.3).

In this Section, we presented how visual verification of vehicles can help ensure that wireless V2V messages accurately describe the physical properties of nearby vehicles. We also discussed how to accelerate verification by transmitting a shortened hash on the visual channel. Rather than limiting receivers to senders that pass through the field of view and drastically limiting the range of V2V communication, we also propose using multiple visually verified witness vehicles to help verify the validity of further away vehicles. Finally, rather than exposing receivers to attacks by verifying a sender once and trusting all of the sender's subsequent messages, we discuss how using a timer to allow verification to expire. However, the receiver resets that timer whenever new evidence supporting the sender arrives. As we show in the next two sections, these extensions help maintain security (detect and prevent different attacks) while allowing receivers to maintain sufficient communication range for VANET or platooning operation.

5.3 Security Analysis

In this section, we discuss how our solution detects ghost vehicles or impersonation attacks and fulfills the basic security properties needed for V2V. We begin by discussing how the basic approach detects the attacks, and what a malicious party may be able to do to violate those properties. After that we discuss how each extension impacts the verification process and the attacker's capabilities. We conclude this section with a discussion on how visual verification fulfills the V2V requirements from Section 4.3.

5.3.1 Attack Detection with the Basic Solution

Without any of the extensions, a receiver only trusts vehicles that have been visually verified. The process of visual verification ensures that a receiver has used its camera to verify that there is a set of LEDs at an estimated location on the road that are transmitting the certificate that verifies the corresponding V2V beacons. As such, the only way to circumvent visual verification and create ghost vehicles or impersonate vehicles is to influence the location estimation, attach LEDs to non-vehicle bodies, know the same cryptographic information as a sender, or transmit data on the visual channel such that it appears to come from the victim. In addition to discussing more details about these possible attacks and their mitigation, we also discuss how prevention of ghost and impersonation attacks implies the prevention of Sybil attacks.

By verifying that LEDs exist at an estimated location (i.e., the check performed between the **Visually Unverified** and **Presence Verified** states), our technique prevents the majority of ghost vehicle attacks. However, if an attacker is able to influence visual location estimate or attach LEDs to non-vehicle objects, additional actions are needed to prevent ghost vehicle attacks. For example, an attacker could move the LEDs further apart to convince a receiver that the attacker's vehicle is closer than it really is. However, a small population of vehicles with additional sensors (e.g., police with laser ranging as part of speed monitoring technology) could help detect such abuses. Another way to create a ghost vehicle is to have LEDs where a vehicle is absent. For example, an attacker could affix the LEDs to roadside infrastructure or onto the road. In that scenario simple sanity checks will help detect the false information. For example, maps included with GPS will let receivers know where the road is and ignore lights that are not on the road, but claiming to be vehicles that are part of traffic. Policing can also help detect and remove these lights.

The comparison between the certificate from the wireless messages and the transmission on the visual channel prevents an attacker from impersonating another vehicle on the road. To circumvent this check and successfully perform an impersonation attack, the attacker needs to extract the victim's key information, which is outside the scope of this work, or try to create messages on the visual channel that encode the attacker's certificate, but appear to come from the victim's LEDs. One way to impersonate a vehicle on the visual channel is to reflect light off of the victim. When performing this attack, the attacker can try to bounce light off of the victim's own LEDs or the body of the vehicle. When using a simple on/off encoding (i.e., LED on means 1, LED off means 0), the attacker can only flip the victim's 0s to 1s. Prior work has shown how to encode a certificate such that any addition of 1s to the message is detectable [17]. If used, the receiver can detect the additional 1s and simply discard the sequence from the visual channel. This solution stops the attack, but also prevents the sender from being verified, since the receiver will discard the sequence from the visual channel. Of course, if attackers want to stop communication, they could jam the wireless channel. The alternative impersonation attack bounces lights off of the victim such that the receiver acquires two different sequences from the same vehicle (one from the victim's LEDs and another from the reflected light). In that scenario, the receiver will detect conflicting sequences from the same physical space and ignore both certificates.

Our solution prevents Sybil attacks (one physical vehicle claiming multiple radio identities) by detecting ghost vehicles and impersonation attacks. By preventing these attacks, receivers will only accept a wireless identity if there is a physical vehicle at the corresponding location, broadcasting the appropriate certificate on the visual channel. Under these constraints an attacker is unable to convince a receiver to accept more than one wireless identity. If the attacker broadcasts more than one identity, the attacker must claim its physical location in both identities' beacons. As such, receivers will only believe the identity with a matching certificate on the visual channel, and discard the other identities, preventing Sybil attacks.

5.3.2 Impact of Using a Shortened Hash

Rather than broadcasting an entire certificate on the visual channel, senders can broadcast a truncated cryptographic hash of the certificate. This reduces the amount of time needed to film the sequence from the visual channel, but increases the chance of impersonation attacks if vehicles possess several keys. Specifically, if an attacker's certificate produces the same truncated hash value as a victim's certificate, an impersonation attack is possible. As such, we would like to select a shorter hash to reduce filming time, but keep it long enough that the chance an attacker's hash matches the victim's hash is small. To further complicate the issue, each vehicle is issued N certificates at a time for privacy reasons [41]. If the authority assigns certificates such that each certificate given to an individual vehicle is different, that means the attacker has N chances to match a victim's hash. In the remainder of this subsection, we calculate the probability of an attacker having a matching hash based on the number of certificates given to each vehicle N and the number of different possible truncated hashes ($H = 2^{\text{Length of the Sequence}}$).

When assigning certificates to a vehicle, the authority selects certificates that produce N different hashes. As such, the authority can assign hashes to a vehicle in the following number of ways:

$$\binom{H}{N} \tag{5.1}$$

After the victim is assigned N certificates, the number of possible selections that leave an attacker without a matching hash is the same as choosing N hashes from the remaining $H - N$ hashes or :

$$\binom{H-N}{N} \tag{5.2}$$

The probability that an attacker is unable to impersonate a sender (i.e., no matching hashes exists) is simply the fraction of assignments where no match exists to the the total number of assignments or:

$$\frac{\binom{H-N}{N}}{\binom{H}{N}} = \frac{(H-N)!^2}{H!(H-2N)!} \tag{5.3}$$

The chance of impersonation is simply 1−Eq.5.3. We plot the chance of impersonation due to overlapping hashes versus hash length in Figure 5.3. For this plot, we assume vehicles have 512 or 1024 different certificates at a time. With 30 bits or more, the probability of a successful attack is less than 0.1%. In addition, this is the probability at least one match exists. If a victim notices another vehicle in radio range is claiming a certificate with the same hash value, the victim can change to another one of its certificates to thwart the impersonation attack, since the probability of having more than one matching hash is many times smaller. Based on this analysis, a hash length of 30 bits provides strong protection from impersonation attacks due to using a shortened sequence on the visual channel.

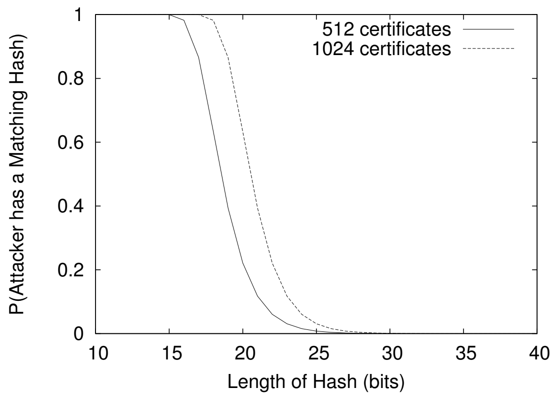


Figure 5.3: Probability of Impersonation Attack Due to Hash Collisions

5.3.3 Impact of Using Witnesses

By relying on witnesses to forward information about potential senders, receivers can begin to accept communication from vehicles that are outside of their field of view, but are exposed to potentially invalid senders if the witnesses forward information about invalid senders. In this section, we discuss how requiring W distinct witnesses help protect receivers from parties that forward information about ghost vehicles or impersonating vehicles. We also discuss why limiting witnesses to forwarding information about visually verified vehicles (as opposed to forwarding information from other witnesses) can help isolate attacks.

When W witnesses are required, a receiver will only begin to trust an invalid source if $\geq W$ colluding malicious vehicles report the invalid source. Since visual verification prevents a vehicle from successfully launching a Sybil attack, this means that $\geq W$ actual vehicles on the road have to collude to successfully fool other vehicles. During a ghost vehicle or impersonation attack, legitimate senders will not forward information about a vehicle they have not visually verified. This leaves only colluding malicious vehicles to disseminate information about invalid vehicles. With one report per physical vehicle, W or more attackers are needed to convince nearby receivers that the invalid vehicle is legitimate.

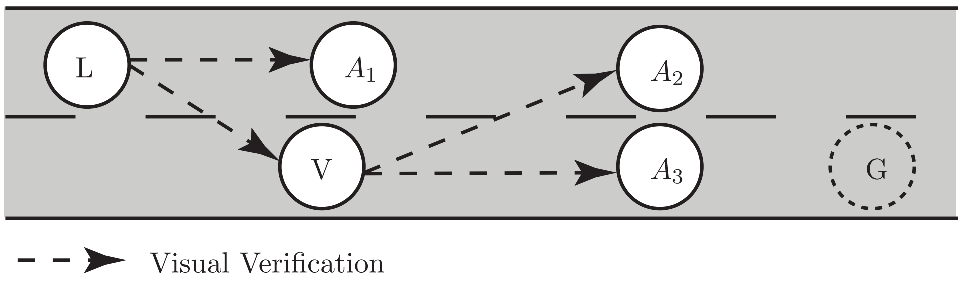


Figure 5.4: Isolation of Invalid Vehicles

If legitimate parties only forward reports for vehicles they have visually verified, we can help isolate the invalid sources colluding attackers support. Specifically, this forces the requirement of W or more colluding vehicles in the victim's field of view, rather than just W in radio range. For example, consider the scenario in Figure 5.4 where a number of vehicles are driving together on a road and $W = 2$. L and V are legitimate vehicles. A_1, A_2, and A_3 are colluding attackers trying to forward information about the ghost vehicle G. V has visually verified A_2 and A_3 and thus accepts their reports that G actually exists. However, since A_1 and V block L's view of the other attackers, L has to rely on A_1 and V as witnesses. Since V never forwards information about G, L correctly leaves the ghost vehicle in the **Visually Unverified** state, instead of the **Trusted** state, despite the presence of 3 colluding attackers in radio range.

5.3.4 Impact of Expiring Verification

Expiring verifications help prevent attacks by forcing periodic re-verification, via visual checks or reports from witnesses. This also addresses the situation where vehicle's change certificates, since continuing to trust the old certificate could enable a Sybil attack. After discussing why requiring re-verification is necessary to prevent attacks, we provide a discussion about the appropriate value of τ, the time between the last re-verification of a sender and the expiration of its validity.

If a receiver were to verify a sender once (via witnesses or visually) and trust that sender forever, the sender could easily launch an attack. The sender would simply wait until it heard W or more witness reports about itself or a witness report from the desired victim for itself (an indication the victim verified the attacker) and then begin to broadcast false information. Once the victim accepts that false information, the attacker could begin to use a new certificate at the same time as the old one. Once the second certificate is accepted, the attacker has successfully completed a Sybil attack and may be able to introduce even more invalid senders into the system if W is small.

Expiring verification prevents these attacks by having receivers discard the old identity after the old certificate is used to claim invalid information. While claiming invalid information, other vehicles will detect that the location in the beacon does not match the location of the vehicle. Without this visual verification, witnesses will stop reporting the certificate is valid. Without witness reports for the sender and without a way for receivers to visually verify the sender, the sender's old certificate will expire. If the sender simply changed certificates for privacy reasons [41], receivers will expire the old certificate and begin to verify the new certificate.

When expiring certificates, it is important to select an appropriate value for τ. If τ is too small, receivers may discard valid vehicle's information simply because of lost witness reports or temporary obstructions in the field of view. However, if τ is too large, an attacker could abuse the infrequent re-verification to switch between different identities or temporarily claim an invalid location. Intuitively, a good value for τ is the amount of time needed to film the hash sequence on the visual channel. With this value, if the vehicle has changed its current certificate, the old certificate will expire just as receivers begin to complete verification of the new certificate. However, if a vehicle tries to constantly switch between multiple certificates, only a single certificate will be trusted at a given time.

5.3.5 Fulfillment of V2V Security Requirements

We have discussed how visual verification with various extensions allows a receiver to verify a vehicle sent a message that contains correct information. In this section, we discuss how our solution fulfills the remaining V2V security requirements of: sender validity, message integrity, short-term linkability, long-term unlinkability, and traceability.

Sender Validity Visual verification allows a vehicle to determine which specific physical vehicle on the road generated a message. This fulfills a stricter property than sender validity, which implies that the message came from a vehicle.

Message Integrity The digital signature in a message allows a receiver to detect if the message has been modified. Given the attacker does not know the victim's key, verifying a signature allows a receiver to verify message integrity.

Short-term Linkability The continued use of a single certificate for some period of time provides short-term linkability for a well-behaved sender. However, we also need to ensure that receivers can link several consecutive messages from a malicious party back to the same vehicle. A malicious party may try to cycle through certificates, using different keys at different times to make it appear as though different senders generated different messages, to violate short-term linkability. However, associating the certificates with the vehicle allows receivers to link the messages back to the same physical vehicle. If an attacker tries to rapidly change between multiple certificates, receivers will be unable to decode the hash from the visual channel. Without visual verification, the receivers will never trust any of the certificates and ignore the messages, preventing the attacker from violating short-term linkability.

Associating a physical vehicle with a wireless message prevents Sybil attacks (one vehicle creating multiple simultaneous identities); either the multiple simultaneous identities are associated with the same vehicle and are detected, or the identities are never associated with a physical vehicle and are ignored.

Long-term Unlinkability Each vehicle is assigned a large number of keys and certificates and is allowed to change keys at any time. Visual verification places no limits on when a vehicle changes keys. The vehicle simply stops using one key and begins to sign messages with the new key and broadcast the hash of the certificate on the visual channel. This flexibility means a vehicle can use more advanced privacy techniques like mix-zones [14, 34] or silent periods [82]. Older schemes with non-overlapping certificates make use of these techniques difficult since a vehicle can only change certificates when one certificate expires and the new one becomes valid.

Traceability Under our approach, the certificate authority assigns a set of unique keys to each vehicle. Even though shortened certificate hashes may collide, no two vehicles own the same certificates. This means that the authority can use a database to store key to vehicle mappings and the signature and certificate from messages to determine which vehicle sent a message.

In this section, we described how visual verification and our different extensions help detect or prevent the dissemination of false information in V2V systems and fulfill the basic properties needed for V2V. We also provided an analysis to describe how various parameters should be set in the system. In the next section, we use several of those parameters to simulate visual verification to determine if the system will support VANET safety or platooning applications.

5.4 Simulation

In this section, we try to determine if our visual verification techniques allow a receiver to verify enough nearby vehicles to support V2V applications. Prior work demonstrates that a visual channel between vehicles can successfully communicate, determine the range between vehicles, and support tracking [1, 81]. The work of Ashok et al. indicates that a visual channel is capable of nearly 10 kbps at a distance of 100 meters when using a 1000fps camera and LEDs. Other work by Saito et al. indicates that LED rear lights allow tracking, distance estimation with an error of $\pm$ 2 meters, and bit error rates of less than 10^{-6} in daylight (with better performance at night) [81]. These results provide strong evidence that a visual channel between vehicles is possible. As such, the focus of our simulations is to determine if vehicles following realistic traffic patterns are in view long enough to support visual verification with or without our extensions. We also want to investigate if the additional bandwidth associated with broadcasting witness reports (the additional hashes for visually verified vehicles) negatively impacts wireless network performance.

To answer these questions, we use NS 2.34 simulations [102] and realistic vehicle traces from the NGSIM project [32]. These traffic traces allow us to simulate traffic on both one side of 6 lane highways (I-80 in Emeryville, CA and US-101 in Los Angeles, CA) and city streets with bidirectional traffic with cross streets (Lankershim Blvd. in Los Angeles, CA and Peachtree St. in Atlanta, GA).

5.4.1 Simulation Description

The NGSIM data sets provide vehicle traces that we can use in NS2. In addition to realistic mobility, we used NS2 to accurately model V2V communication and modified mobile nodes behavior to simulate the visual channel.

For each of the 4 different locations, NGSIM provides multiple vehicle traces from different times in the day. We summarize the traces we use to evaluate visual verification in Table 5.1. We concatenate consecutive trace files to make a single longer trace. For example, we concatenate the 8:30am - 8:45am and 8:45am to 9:00am traces files for Lankershim Blvd. to generate a single long trace. For non-consecutive trace files, we present the results from the trace file with the worst performance, instead of concatenating unrelated traffic patterns.

We used NS2 with the recent extensions to the physical and MAC layer

Location	Start Time	End Time	Total Vehicles	Vehicles a Minute	Road Length (m)
US-101	7:50am	8:20am	4186	140	640
I-80	5:00pm	5:15pm	1836	122	503
Lankershim Blvd.	8:30am	9:00am	2442	81.4	487
Peachtree St.	4:00pm	4:15pm	1222	81	640

Table 5.1: Vehicle Trace Information

to accurately model V2V communication [23]. To simulate normal VANET wireless communication, each vehicle broadcasts a signed beacon according to the IEEE [48] and SAE [91] format standards every 100ms and a copy of its own certificate in every 10th beacon (i.e., once a second). Without the certificate, a beacon message is 259 bytes, including the V2V headers, application data, ECDSA signature, and a digest of the sender's certificate. Beacons with a certificate are 368 bytes. This helps simulate realistic V2V communication and network contention. All broadcasts use 10dBm power. In scenarios with witnesses, each vehicle will add a maximum of 5 certificate hashes to a beacon if the owner of the certificate was in view since the last beacon. We assume SHA-1 hashes are used. Each hash is 20 bytes long, for a maximum of 100 additional bytes added to a message for witness values.

We made pessimistic assumptions about the visual capabilities of the vehicles to ensure our simulations represented realistic worst case performance. We assumed each vehicle had 2 cameras (front and back) that operated at 20 frames per second with a 130 degree field of view and were able to receive information at a rate of 2 frames/bit (10 bps) with a maximum range of 100m. These values were used to reflect inexpensive low resolutions cameras used for back-up cameras trying to film at long ranges, rather than the short ranges or more capable cameras used in prior works [1, 81]. Our simulated LEDs only have a viewing angle of 45 degrees. As such, a receiver's camera must be within 22.5 degrees from the center of the LED's beam to see the LED. To simulate vehicles as obstructions, we assumed each vehicle was an orb with a radius of 1.5m. A receiver had to see the entire orb to receive information on the visual channel and any vehicle between the two was considered an obstruction (i.e., we did not assume the ability to see through the glass of vehicles).

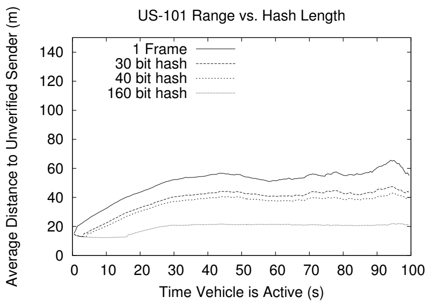
As a metric we consider the average verification range, the distance to the nearest unverified sender. Prior work by Haas and Hu [39] show that VANET safety applications require between a 20 and 60 meter communication range, as such our technique must allow verification to that radius or more to support those applications. Platooning applications require vehicles to share information between two adjacent vehicles [89]. With this limited requirement, we just have to check that receivers are able to verify their neighbors within a short amount of time.

To determine the impact of our scheme on wireless performance, we compare the normalized packet reception rate under the different configurations. We consider visual verification without witnesses as baseline performance for packet reception since the vehicles only send normal VANET traffic. We compare the normalized packet reception rate with witnesses and with witnesses and verification expiration. To calculate the normalized packet reception rate, we take the number of packets received under a given configuration and divide that by the number of packets received under the baseline configuration (without witnesses). If the reception rate is significantly less than 1, that is an indication that our approach causes contention on the wireless channel.

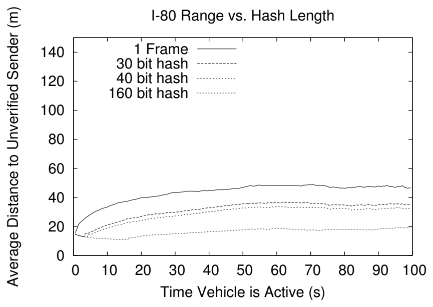
5.4.2 Simulation Results

We simulate visual verification with and without each of our extensions. We present the average verification range from these simulations in Figures 5.5, 5.6, and 5.7. We present the packet reception rate versus distance in Figure 5.8.

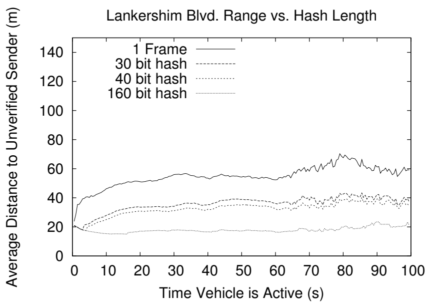
The x-axis of each verification range plot indicates how long a vehicle has been driving in the simulation and the height of the line indicates how far away the nearest unverified sender is on average. We truncated the plots to only the first 100 seconds since most vehicles exit the simulation area in less than a minute and vehicles that remain longer maintain sufficiently



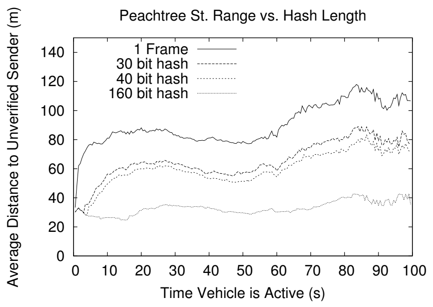
(a)



(b)



(c)



(d)

Figure 5.5: Impact of Hash Length on Verification Range

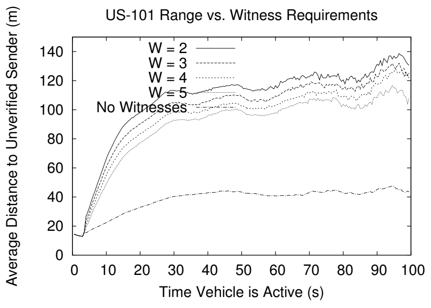
large verification ranges. For all of these plots, vehicles start with a small verification range (i.e., no vehicles visually verified), but the range grows as a vehicle visually verifies neighbors and uses those vehicles as witnesses to verify further away vehicles.

Visual Sequence Length Figure 5.5 presents the impact of hash length on verification range. Instead of simulating the transmission of the entire certificate (117 bytes or 234 frames), our largest sequence is the SHA-1 hash of the certificate. In addition to 30 and 40 bit hash lengths, we also simulate the performance of the system if the visual channel requires a single frame to transfer the vehicle's certificate. Such a rapid transfer is unrealistic, but provides an upper bound on verification range when only direct visual verification is used.

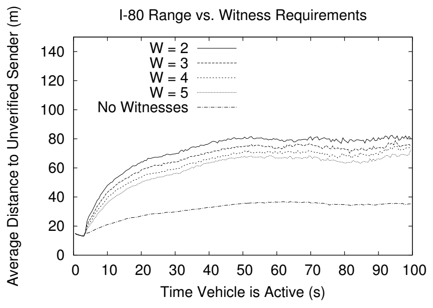
These results indicate that even if the visual channel supports high bandwidth communication, vehicles will have insufficient range when first driving. The results also indicate that a shortened hash significantly increases the verification range, when compared to visually transmitting the entire 160 bit hash value. Even with the high bandwidth visual channel, only vehicles in the Peachtree trace were able to visually verify vehicles within 60 meters in less than a minute. Simulation of more realistic bandwidth for the visual channel also supports the need for witnesses. A full length SHA-1 hash provides very strong protection against impersonation attacks (the authority must generate more than 2^{80} certificates before a collision occurs). However, our pessimistic visual channel bandwidth of 2 frames per bit more than halves the range of the application when compared to using a 30- or 40-bit hash that provides sufficient protection. We use a 30-bit hash for the remainder of the simulations since a 30-bit hash provides sufficient security and an improved verification range compared to the 40-bit hash.

Number of Witnesses Figure 5.6 contains plots of how the value of W impacts the verification range when vehicles transmit a 30-bit hash on the visual channel. Every scenario benefits significantly from using witnesses to verify vehicles not currently in view. In the highway scenarios with one-way traffic, the difference in speed between lanes allows passing vehicles to act as witnesses for vehicles in front of or behind of a receiver. In the city scenarios, on-coming traffic acts as witnesses for vehicles.

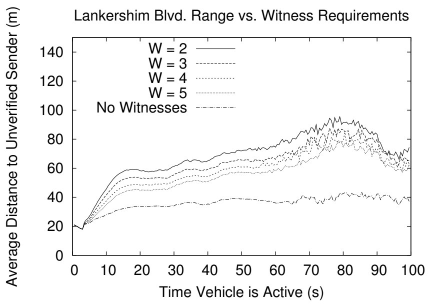
The large difference in vehicle speed between lanes allows vehicles in the US-101 trace to reap the most benefits from the use of witnesses. On average, without witnesses, vehicles are only able to verify senders within 32 meters after driving for 20 seconds. However, with witnesses, a vehicle is able to verify senders within 98 meters in the same time period. The reason for this gain is the large variance in speed between lanes of traffic (average speeds of 20mph in one lane and 40mph in another). As vehicles pass each other new senders enter and exit visual range and are verified. Once visually verified, these senders can report information about vehicles further ahead or behind depending on what they saw recently. In the I-80 trace, the difference in lane speed is smaller (15mph versus 30mph), reducing



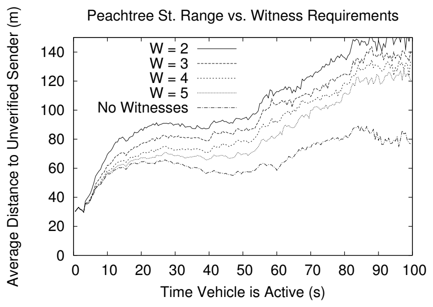
(a)



(b)

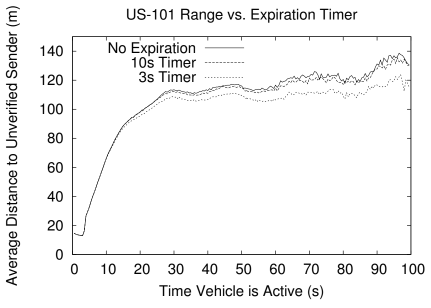


(c)

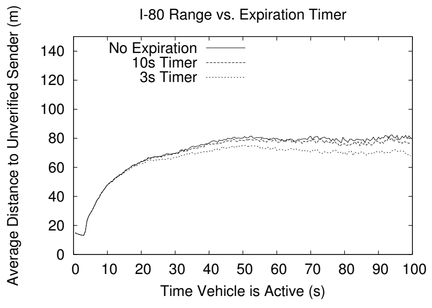


(d)

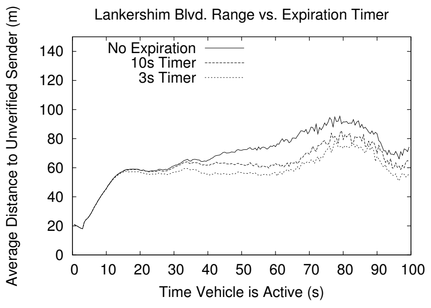
Figure 5.6: Impact of Witnesses on Verification Range
(Hash Length = 30 bits)



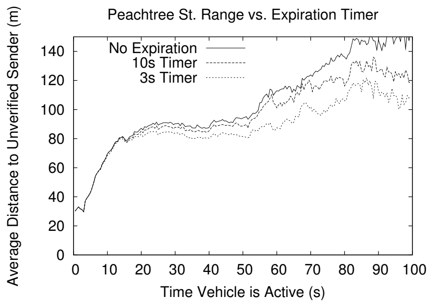
(a)



(b)



(c)



(d)

Figure 5.7: Impact of Verification Expiration on Verification Range (Hash Length = 30 bits, $W = 2$)

the mixing speed of traffic and thus the impact of witnesses.

Witnesses improve the verification range in the city scenarios because of bi-directional traffic and the mix of traffic at intersections. In addition to passing traffic, vehicles entering the flow of traffic help receivers verify senders that were on contributing roads. Peachtree street has better performance due to the decreased vehicle density (a 640 meter road versus a 487 meter road segment on Lankershim blvd.).

We choose to use $W = 2$ for the remainder of the simulations because it prevents attacks from a single malicious vehicle and provides the greater verification range. Of course, if colluding vehicles become common, a larger value of W is needed to handle the increased threat, but will not drastically impact verification ranges.

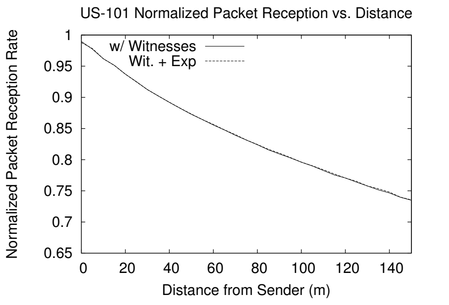
Expiration Timer Figure 5.7 contains plots of how the addition of verification expiration impacts range. For these simulations, we tested no expiration ($W = 2$), $\tau = 3$, and $\tau = 10$. $\tau = 3$ provides the best protection against attacks and has a limited impact on verification range in the highway settings. However, the city scenarios do suffer more because of the expiration of the verification of oncoming traffic.

In the highway scenarios, $\tau = 3$ only reduces the range by 15 meters on average, when compared to no expiration. Given the limited impact, the average verification range remains large enough to support the relevant V2V applications. The requirement of only a single report to re-verify a vehicle and the frequent mix of vehicles limits the impact of verification expiration.

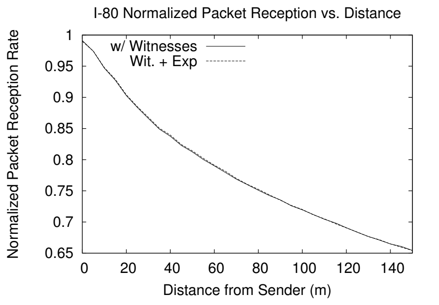
In the city scenarios, $\tau = 3$ has a larger impact on verification range. In the Peachtree scenario, $\tau = 3$ reduces the verification range by a maximum of 50 meters. Verification expiration also negatively impacts the verification range in the Lankershim scenario with a maximum decrease of 24 meters. In both scenarios, the range is still enough to support the safety applications. The reason for the larger decrease in verification range for the cities is the introduction of merging traffic to oncoming traffic. After a vehicle passes a receiver and has been visually verified, it remains visually verified until the receiver stops hearing reports about that vehicle from verified traffic. When a new vehicle joins the oncoming traffic flow, the new vehicle blocks the visual path between two verified vehicles. Without this visual path, the two verified vehicles stop reporting each other, the verification expires, and the receiver's verification range decreases.

Despite the decrease in verification range, our approach still provides the 20 to 60 meters of communication range to support the VANET safety applications and platooning.

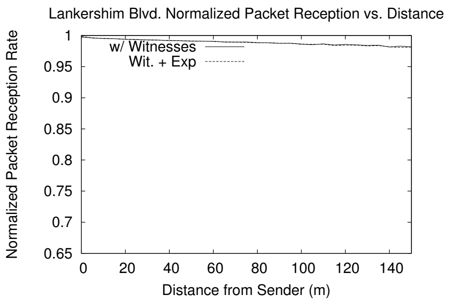
Impact on the Wireless Chanel Figure 5.8 summarizes the impact of broadcasting witness values on the wireless channel. These results indicate that expiration has little impact on the wireless channel. In the city scenarios, vehicle density is low enough that the addition of 100 bytes or less has little impact on the packet reception rate. However, broadcasting wit-



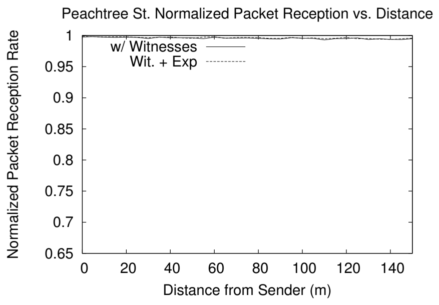
(a)



(b)



(c)



(d)

Figure 5.8: Impact of Visual Verification on Packet Reception
(Hash Length = 30 bits, $W = 2$, $\tau = 3s$)

ness values causes enough contention and interference to lose 35% of packets when compared to not using witnesses.

This is a significant impact on the wireless channel and we simulate different radio and application configurations to determine how to improve radio performance. To determine if we can alleviate channel contention, we use two simulation configurations: 1) the lower transmission power of 0dBm and 2) the same transmission power (10dBm), but a reduced beacon broadcast rate of 5 beacons per second. Our results indicate that to maintain verification range and improve wireless network performance, reducing the beacon broadcast rate is the better approach. The lower transmission power suffers from both shorter verification ranges in all of the scenarios and lower packet reception rates. The impact of decreased beacon broadcast rates has a negligible impact on verification range and drastically improves the packet reception rate. In the US-101 simulation, the normalized packet reception rate with witnesses and verification expiration at 150 meters improves from 73% to 93%. In the I-80 simulation, rates improve from 65% to 88%.

These results indicate that visual verification with out extensions can support various V2V applications under realistic traffic patterns with a low bandwidth visual channel. However, care must be taken to ensure that the addition of witness values does not negatively impact the already strained wireless channel.

5.5 Summary

Visual verification of senders helps secure VANET safety and platooning applications. By using the visual channel to confirm a source's location and associate a physical vehicle with cryptographic information, we are able to prevent a class of attacks against V2V systems that try to disseminate false information about the physical world and cause false alarms, erroneous behavior, and even accidents. In addition to direct visual verification of a source, the system also benefits from the visual broadcast of shortened hash values, the use of witness vehicles, and verification expiration to expedite visual verification and extend the range of visual verification, while maintaining a high level of security.

Simulations indicate that visual verification and our extensions can support V2V applications and allow the detection of invalid senders and false information, without requiring the connectivity needed for timely revocation.

Chapter 6

Discussion on Endpoint Verification

This thesis proposes three different protocols to perform endpoint verification for three different scenarios. We use devices' sensors and actuators to establish a secondary communication channel, one in addition to the wireless channel, with physical properties that enable verification of the location of communication endpoints. For each scenario, we propose the use of hardware that already exists on the devices to establish this channel. However, endpoint verification is also possible with other hardware that enables different communication channels with the same physical properties. Instead of using a secondary channel with specific physical properties to solve the problem of endpoint verification, a number of prior works propose techniques that use the time-of-flight of messages on the wireless channel to verify the location of a device.

We begin this chapter with a discussion of the general properties a secondary channel must provide to enable the use of our protocols for endpoint verification. We also give some examples of alternative communication channels that use different hardware to achieve the same goal. We then discuss existing work that relies solely on the wireless channel to verify the location of a device and contrast that approach with our solutions.

6.1 Secondary Channel Requirements for Endpoint Verification

The different scenarios we discuss in this thesis have different requirements for source and/or destination verification, which require different solutions. In this work, we constrained ourselves to hardware that is already available or is likely to be available in the near future on commodity systems. However, other systems may have similar endpoint verification requirements with different hardware. For the remainder of this section, we discuss the basic channel requirements that each of our protocols require. These requirements provide guidelines to facilitate adoption of our protocols to new settings.

6.1.1 Source and Destination Verification

In SHOT, we use phones' vibrators and accelerometers as a channel that allows users to verify that only the two phones in physical contact are communicating. Users are able to observe the vibrations and detect if another device is trying to transmit on this channel. Information exchanged on the vibration channel allows the phones' to bootstrap communication through a server and cryptographically verify that the two phones successfully received each others' information. In general, the protocol requires a channel between the two phones that:

1. Allows the users to identify the communicating endpoints.
2. Allows detection of an unwanted third party's attempts to communicate on the channel.

Vibrators and accelerometers are one of many ways to implement such a channel. In Chapter 7.1.3, we discuss protocols that use cameras to capture a visual encoding of data (barcodes or sequences of blinking lights) or use microphones to record information encoded as an audio signal. Such visual and audio channels fulfill the properties necessary to execute SHOT and verify the endpoints of the exchange. If Near Field Communication (NFC) is available and secure, SHOT can use NFC to exchange information and verify the endpoints of the exchange. NFC's limited range of 10 cm or less allows users to infer that the two devices held together are communicating. NFC devices in active mode are able to detect attempts to inject data onto the channel [44]. This ability to detect additional communication with a nearby device allows automatic detection of attacks, instead of relying on humans to monitor the communication channel.

6.1.2 Destination Verification

MULE leverages a constrained channel [52] so that only a device within a trusted physical space is able to receive a decryption key. The inability to receive messages from the constrained channel when outside of the space provides the secrecy we leverage to provide location-based access control. For our implementation of a constrained channel, we use the laptop's webcam to receive information from an infrared LED. However, any signal that is contained to a trusted physical space (i.e., unable to pass through walls or other boundaries that humans use to delineate a physical space) is a viable constrained channel. In addition to infrared, Kindberg et al. also suggest using wireless communication in the 60GHZ range or ultrasound as constrained channels that are unable to pass through walls. Surprisingly, NFC does not meet the constrained channel requirements. Normal NFC communication is only possible within tens of cm. However, an attacker's ability to eavesdrop on NFC communication from up to 10 meters away [44] means an attacker outside of the physical space can access the channel.

If a system lacks a constrained channel with secrecy to verify the destination of communication, it is possible to use source verification to verify the receiver of digital information as part of a multi-step protocol. First, the system has to verify that the device in question is a source of a public key within the physical space. After verifying the public key corresponds to a device within the space, the system can use the public key to encrypt the important information (e.g., the information controlled with location-based access control). This encryption ensures that only the device, which was verified as in the physical space, is able to receive the information.

6.1.3 Source Verification

We propose the use of LED headlights and tail lights, and cameras to verify the source of messages in V2V applications. This channel provides two important properties:

1. Verifies that a device (i.e., vehicle) exists at some position relative to the receiver.
2. Allows the receiver to associate data with that vehicle.

RADAR or LIDAR, technologies already available in luxury vehicles, are able to fulfill the first property. However, those technologies fail to fulfill the second property. Without a way to confirm that a public key corresponds to a vehicle, the system will expose receivers to impersonation attacks. Sastry et al. [83] propose using the time of flight of soundwaves to determine the location of a device. However, ambient sound on the highway would be too noisy to allow successful communication over an audio channel (audible frequencies or ultrasound). In addition, their protocol requires multiple parties to verify the location of a device. With only two devices (the sender and a receiver trying to verify the location of the sender), the receiver is only able to determine if the source is within a radius, but lacks a way to verify the relative direction to the sender. As such, no other communication channel exists that supports both verification of the presence of a vehicle and a way to associate digital information with the vehicle.

6.2 Using Wireless Time-of-Flight to Verify Endpoints

Instead of using hardware that allows the establishment of an additional communication channel with some physical properties to verify the source or destination of communication, a number of prior works propose ways to use the wireless channel to determine the location of the source of digital data. After verifying the source of data as some device at a location, we can use the data from the source to bootstrap secure communication. If this channel is secret, we are able to verify that only that device at the verified location can receive the data. Unfortunately, many of these location verification solutions require hardware that is unavailable for the scenarios we address. We begin with a discussion of distance bounding, which allows a receiver to verify a device is within some radius of the receiver based on the round trip delay for wireless exchanges. We then discuss location verification schemes which often use distance bounding from multiple vantage points to determine the location of a device with respect to multiple .

6.2.1 Distance Bounding

Distance bounding uses the finite speed of wireless messages to determine the maximum distance to an entity. In these protocols, one radio acts as a verifier, trying to determine the maximum distance to the prover radio. When the protocol begins, the prover commits to a series of bits. The verifier then sends a series of challenges and measures the time the prover needs to return the response. The time needed to respond to the challenges allows the verifier to determine the distance between the two devices. Finally, the verifier compares the responses to the challenges to the commitment to verify that the prover did not respond to a challenge before receiving the challenge.

Brand and Chaum were the first to propose using radio messages to perform distance bounding [9]. Unfortunately, the delay associated with the prover receiving, decoding, processing the challenge, and generating a response means there is a large potential error in the distance estimate based on the processing speed of the prover. To address this shortcoming, Rasmussen and Capkun propose the use of special hardware to perform the processing in the analog domain, reducing the time associated with receiving and responding to the challenge to 1 nanosecond [74]. However, the widespread deployment of this specialized hardware is unlikely. As such, a commodity device without the hardware will take additional time to respond and will suffer inflated bounds on distance estimation. If the verifier assumes the processing is slow, an attacker could use Rasmussen's specialized hardware to appear closer to a verifier than she really is. Other work has shown an attacker can generate responses to a verifier's challenges that appear valid, but lack redundant symbols at the physical layer encoding to appear up to 140 meters closer [33].

6.2.2 Location Verification

Location verification allows a group of verifiers to determine the approximate location of a prover. Existing work uses multiple executions of distance bounding protocols [83, 97] or packet reception times [18, 19] with multiple verifies to verify location claims. However, these schemes require multiple verifiers and without extremely accurate timers or secure underlying distance bounding protocols, these schemes provide limited accuracy.

Under both approaches, each verifier uses the timing of the prover's broadcasts to create a circle that contains all potential locations for the verifier, with the prover the center. The verifiers take the intersections of these different circles to determine an overlapping area that represents the region where the prover is located. Even if multiple verifiers are available, the location verifications schemes lack accuracy. If the underlying distance bounding scheme suffers from vulnerabilities, each verifier's measurement has a large margin of error, producing inaccurate location verification. In schemes that compare the reception time of messages, each verifier requires nanosecond accuracy. This may be possible, but is prohibitively expensive for location-based access control in the home or V2V networks. Users will not want to spend the money to deploy multiple verifies with such accuracy in their home. The need to deploy verifiers along all roadways to detect V2V sources that broadcast false location information is prohibitively expensive even for a nation.

6.3 Summary

Work on verifying the physical location of a communication endpoint relies on either the physical properties of a secondary communication channel or the time-of-flight of wireless messages. In this thesis, we focus on the former approach as an inexpensive way to fulfill the requirements of endpoint verification with hardware already present on commodity devices. However, if custom radios with high resolution timers and the deployment of numerous verifiers is feasible, distance bounding and location verification allow accurate endpoint verification with only the wireless channel.

Chapter 7

Related Work

We now discuss work related to our three different projects on endpoint verification. We first discuss protocols to exchange information between two devices (Section 7.1). We discuss alternatives to location based access control to protect information on stolen laptops in Section 7.2. We conclude with a discussion of related V2V work which aims to detect false information (Section 7.3). Background on basic V2V operation or prior key management schemes is in Chapter 4. We discuss distance bounding and location verification protocols in Chapter 6.

7.1 Exchange of Information Between Devices

Exchanging information between two devices, without allowing an unwanted party to insert additional data, is a long standing problem with numerous potential solutions. Another solution to this problem is to allow two devices to establish a secret key, aka device pairing. Once two devices share a symmetric key, they can exchange information, without having to worry about a third party injecting information. In this section, we discuss exchange or pairing protocols based on the different physical actions performed: human assisted comparison, the exchange of secrets, or observable communication between the devices.

7.1.1 Human Assisted Comparison

A number of prior works require the user to perform a comparison after an exchange or pairing to detect an attack [38, 49, 55, 71, 101]. Given an attacker is unable to change the output on devices' screens or the value the user enters into a device, this technique can successfully detect MitM attacks. Many works focus on comparing a string of hexadecimal digits [49, 55, 101]. However, Uzun et al. [100] found usability issues with string comparisons (i.e., users failed to detect attacks or indicated the strings were different when they were the same). Other works propose encoding the comparison value into a sentence [38] or image [70] to improve usability. Instead of using more complex comparison methods, Prasad and Saxena consider how to pair devices without a display [71]. Their solution is to have users compare a sequence of beeps or blinking lights from the devices. A difference between the two devices' sequences is an indication of an attack, and the user should press the correct button to abort the pairing. However, all of these schemes still require users to perform a comparison and are vulnerable to attacks when users click "Accept" without comparing the strings, sentences, images, blinking sequence, or beeping sequence.

Soriento et al. propose Button Enabled Device Pairing (BEDA) [92] as an alternative to the user performing the comparison at the end of an exchange. Instead, the user presses a button on one device whenever the other device blinks, beeps, or vibrates. If the user makes an error (e.g., pressing the button at the wrong time), the protocol fails and communication remains safe. The downside to this approach is the protocol requires more than 50 seconds on average. A variant of BEDA allows the exchange of secrets between devices, which we discuss in the next section.

7.1.2 Exchange of Secrets

A number of works assume the attacker is unable to observe the users talking in the room (e.g., sharing a password [49]), pressing buttons on the devices [92], motion of the phones [21, 45, 56, 60], communication over the vibration channel [26, 86], or communication over an electrical connection between the devices [94]. The threat model from Chapter 2 includes a powerful attacker who may be able to violate the secrecy of all but Stajano and Anderson's protocol [94]. Nearby parties can eavesdrop on spoken communication to recover a password or watch the user to determine the secret entered into the devices (both the timing and the keys pressed).

If the attacker is unable to see the users, moving the phones together provides an intuitive way to identify endpoints in an exchange. However, our attack on Bump demonstrates how attackers can observe and imitate simple movements of phones. Mayrhofer and Gellersen show that it is possible to tune these movement-based exchange algorithms to be resistant to attackers simply watching how users shake devices, but at significant cost to the usability of the system. Their system experiences a 10% failure rate for legitimate exchanges, and 16% of participants are unable to ever successfully complete the protocols [60]. Beyond sacrificing usability to provide security against a simple attacker, these protocols may be vulnerable to more sophisticated attacks, such as those that use high-speed cameras and real-time motion tracking [58] to reconstruct movements that an unaided human could not reconstruct.

When the phones' vibration function is used, user involvement is limited since users only have to hold the phones together, but it is still possible for an attacker to eavesdrop on the vibration channel to acquire the secret [43].

Stajano and Anderson use a wire or other electrical connection between the two devices [94]. In that scenario, the only way an attack will succeed is if the attacker can control communication on the wire without the users noticing. The drawback to this approach is that users have to carry around cables, and possibly adapters, in order to perform the exchange.

7.1.3 Observable Channel Between Devices

Much like SHOT, other works examine the use of a channel between two devices that allow users to infer which parties are communicating. Prior works considered using IR [5], sound [93], or light in the visual spectrum [61, 84, 85] as authentic human-observable channels.

If hardware exists, IR provides a user-friendly way to bootstrap secure communication between two devices [4]. However, the popularity of IR is decreasing and often not available on devices. For example, IR is not available on Blackberry, iPhone, or Android phones, which account for over 75% of all smartphones.[1]

HAPADEP: Human-Assisted Pure Audio Device Pairing [93] foregoes the use of wireless communication, and relies on the audio channel to exchange cryptographic information and data. Under this approach, a user simply has to listen for interference. This is a simple task and allows for timely exchange, but limits use to locations with limited background noise. Noise can also negatively impact SHOT, but only when it is so loud that the sound vibrates the phones.

Seeing-is-Believing (SiB) [61] and Saxena's follow-up work [84, 85] use the phone's camera to photograph barcodes or film a blinking light. However, SiB is a directional exchange, so users have to execute the protocol twice. Saxena's protocol allows a complete exchange with a single execution of the protocol. However, in addition to filming on P_A, the user has to confirm the exchange by pressing a button on P_B. SHOT allows phones to exchange authentic information in both directions, without involving the users to reposition the phones or press a button.

7.2 Laptop Protection

Encrypting files is a common solution to the stolen laptop problem [25, 63, 87]. The challenge with encrypting information on a laptop is how to store the key, without burdening the user. If the required user effort is too much, they will become annoyed with the system and disable the protections. If the user does not disable a system that requires significant effort, the user may find a way to reduce the burden that reduces the security of the solution. The simplest solution to implement is to use a password to derive a file decryption key. Users are willing to accept the task of password entry to

[1]`http://blog.nielsen.com/nielsenwire/online_mobile/mobile-snapshot-smartphones-now-28-of-u-s-cellphone-market/`

act as a key, but often use relatively weak passwords. Once a laptop is in the attacker's possession, an attacker can brute force passwords to discover the key. In addition to passwords, prior works on user authentication utilize what the user is (biometrics) or what the user has (tokens). Once the laptop has access to sensitive data or the decryption key, another challenge is removing any traces of that information before the laptop falls into the wrong hands.

Users cannot forget their biometrics, but most systems require additional user interaction (e.g., speech or finger prints [64, 99]) and extra hardware for biometric entry (e.g., fingerprint reader). Some mechanisms also require the system to store a template used to verify that the appropriate biometric was entered, rather than deriving a key from the biometric. Often the template is encrypted on the hard drive and requires an additional key to decrypt the template [50].

Cryptographic tokens provide greater entropy than user's passwords [25, 79], but users may keep the token with the laptop to ensure effortless access [68]. Once an attacker steals the laptop with the cryptographic token, the attacker has access to all of the files.

Another approach is to use an online service to store keys for a user [59, 79]. Unlike our CKD, these schemes use per-laptop secrets and require significantly more administrative overhead, reducing the chance of widespread adoption. In addition, when the laptop is offline, for example during a flight, users cannot access their files.

Related to harddrive encryption is how to securely store and erase keys such that anyone that later acquires the laptop is unable to recover the key from a hard drive, memory, or a solid state drive [42, 105]. When a laptop sleeps or hibernates, the current state of the machine is stored to memory or the permanent storage (a hard drive or solid state disk), respectively. As such, the system must delete any un-encrypted copies of sensitive information before the laptop sleeps or hibernates. Otherwise a thief can extract that information. However, memory and solid state drives make the task more complicated than deleting keys and decrypted files before leaving the laptop unattended. Halderman et al. [42] show that an attacker can physically extract memory from a laptop that a user put to sleep, and recover any information stored there. Recent work by Wei et al. [105] indicates that the controllers in current solid state drives fail to truly remove information when a file is deleted. Based on these results, a complete laptop protection system must take care to protect any copy of sensitive information that may reside in memory when the system is put to sleep or hibernates or is ever swapped out to a drive.

7.3 V2V Related Work

In Chapter 5, we propose visual verification to secure V2V applications. Section 4.4 contains a summary of related work on how to manage cryptographic identities as part of a V2V security solution. Here we discuss different schemes which try to detect different types of misbehavior in V2V including: detecting false information, detecting Sybil attacks in V2V networks, or detecting abuses of V2V applications.

Our prior work on weak verification of location claims in VANET [95] and work by Golle et al. [37] try to answer the same question as our visual verification of vehicles: where is the source of this message? In our prior work, we use comparison of beacon reception times from multiple vehicles to determine an approximate ordering of vehicles on the road. Under such an approach, a receiver only knows the order of vehicles and not the distance. This is a significant shortcoming given that our prior approach is unable to differentiate a braking alert from a sender 50 meters away from a braking alert for a sender 5 meters away. Multi-path effects can also reduce the accuracy of this approach. If the radio messages does not travel a straight line between a source and a destination, the receivers will use the wrong information to estimate the ordering. Golle et al. [37] assume vehicles have a way to associate wireless messages to physical vehicles and analyze what kind of attacks that can detect. However, they never describe exactly how to perform that binding how network dynamics (i.e., the movement of vehicles) or the limited range of such a binding would impact their misbehavior detection.

Xiao et al. [109] propose using signal strength to detect Sybil attacks in VANETs. The idea is that if the same physical vehicle is broadcasting messages as different identities, a radio will receive all of those messages with roughly the same power since the messages will suffer the same path loss. However, multi-path effects and attackers that vary their transmission power can negatively influence the accuracy of such an approach.

Other VANET misbehavior detection tries to detect false information in reports that does not directly relate to the position of a vehicle. These works often consider wireless reports of ice or debris on the road, a disabled vehicle on the side of the road, or other information that our proposal is unable to detect. To provide complete detection of abuses in VANET, a vehicle can use visual verification to determine the source of a message and these misbehavior detection systems to evaluate the validity additional information in the message. Raya et al. [75] consider the scenario where a vehicle receives multiple, possibly conflicting reports, and uses a framework with Dempster-Shafer logic to determine the belief in a report based on the evidence from multiple vehicles. Kim et al. [51, 75] proposes a framework that uses multiple sources of information to determine whether or not the likelihood or certainty of an event reaches a threshold that indicates if a driver should be alerted. In addition to multiple vehicles, the multiple sources include local sensors, road side infrastructure, and the reputation of the sender to assign a certainty to a given report. Ghosh et al. [36] propose a system to determine if a notification about a crash on the road is real. The system assumes a driver will maneuver the vehicle to avoid wreckage on the road, and uses the driver's actual response to determine if an accident really happened. This is helpful when trying to assign reputations to a sender. However, Ghosh's system only has value after a vehicle has encountered the potentially dangerous situation (i.e., an accident on the road), and not before when an early warning could help the driver change lanes to avoid the accident site.

Chapter 8

Conclusions and Future Work

Before discussing opportunities for further investigation, we state the conclusions of this thesis.

8.1 Conclusions

The addition of inexpensive sensing, actuation, computing, and communication to devices enables a broad range of applications in a variety of Cyber Physical Systems (CPS). For correct operation, these systems need correct information about the physical world they monitor and control. This means that the devices need to know the source and destination of information. In a system where devices' roles are fixed, deployment of the traditional security approach of a Public Key Infrastructure to associate a device with a specific role with a cryptographic value can verify endpoints of communication. However, in a CPS where devices move and the physical orientation of devices matters, a static name given to a device fails to describe the relevant physical properties of the device. Instead, the system needs a way to determine what physical device is the source of a message or to enforce that only devices with certain physical properties are able to receive some information. This thesis proposes solutions that allow devices to verify the physical source and/or destination of wireless messages, with only the use of inexpensive hardware and the commodity hardware that already exists in the system. Our solutions enable this type of verification in three different scenarios: information exchange between smartphones, location-based access control for laptops, and verification of vehicles in V2V networks. These solutions provide useful properties that traditional certificate-based security solutions are unable to provide, and will hopefully convince other researchers to investigate new techniques to secure systems that rely on information from the physical and digital realms.

We have shown how mobile phones can exchange information based on physical interactions. Our approach prevents Man-in-the-Middle (MitM) attacks where an unwanted third party injects unwanted information into the exchange. Our solution leverages the vibrators and accelerometers that are built into the majority of smartphones to establish a human-observable channel between the two devices. The devices use communication on this human-observable channel to bootstrap communication via a remote server and to verify the authenticity of the information that the phones exchange through this server. A malicious party can remotely vibrate the users' phones in an attempt to launch a MitM attack. However, users can detect the additional vibrations and cancel the exchange, preventing the attack. Without a way to covertly inject information into the vibration channel, only the two devices held together can exchange information, allowing verification of the physical source and receiver of the information exchange.

We have shown how to use location-based access control with minimal setup and maintenance to protect sensitive information on laptops as part of Mobile User Location-specific Encryption (MULE). MULE uses reception of a signal from a constrained channel to verify the physical location of the laptop. The physical properties of the constrained channel ensure that only devices within a physical space are able to receive the message. Given many users may keep non-digital copies of sensitive information in the trusted location, location-based access control allows users to align access policies for digital and paper copies of their information (i.e., access to the space implies access to both).

We have shown that V2V networks can utilize visual verification to determine the validity of a sender and the content of its beacons. Instead of relying on detection of key theft or abuse and subsequent revocation to ensure only vehicles produce V2V messages, visual verification allows a receiver to determine which vehicle generated a wireless message and if the message accurately describes the position of the vehicle. This reduces the need for a vehicle to frequently connect to an authority to download revocation information. This also adds flexibility to how or when a vehicle changes cryptographic information in order to avoid tracking, compared to prior work on key verification that requires certificates with fixed non-overlapping lifetimes.

It is encouraging that our implementations and implementations in prior work show that these solutions are able to work with the addition of little or no new hardware. By leveraging existing sensing capabilities, these devices are able to verify the physical source or destination of wireless messages. Endpoint verification is a first step towards ensuring the correct interpretation of data and, as a result, the correct operation of cyber physical systems. We hope that our work will provide a reference for how to verify entities in systems that blend the digital and physical realms. In addition, we hope to serve as inspiration for other researchers to pursue additional verification methods that take into account the physical embodiment of a device, instead of relying on the traditional security approach of a Public Key Infrastructure where a name is associated with a key.

8.2 Opportunities for Future Work

While this thesis explores several techniques to associate physical entities with wireless endpoints, there exist some issues that warrant further investigation. We consider two in particular: user studies and investigation of more cyber physical systems.

8.2.1 User Studies

We propose SHOT and MULE as user friendly security solutions that provide simple to understand security paradigms. For SHOT, the two phones in physical contact are communicating. For MULE, access to a trusted location implies access to sensitive files. As security centric researchers, these ideas seem intuitive and easy to use. However, prior work has shown that users often fail to successfully complete tasks that designers think are easy [100, 106]. As such, a formal user study is needed to verify that SHOT and MULE are beneficial and simple to use.

8.2.2 Verification for Other Cyber Physical Systems

In this work, we limited the hardware we used for endpoint verification to inexpensive secondary equipment (i.e., a shared server for SHOT or a TLD for MULE) and sensors or actuators that are readily available on the devices in the cyber physical system: vibrators and accelerometers on smartphones, web cameras on laptops, and LEDs and cameras on vehicles. However, a large number of CPSs exist that have different hardware and/or verification requirements. As such, further work is needed to analyze other CPSs to determine their exact verification requirements and how to fulfill those requirements based on the hardware that is already available within those systems.

Bibliography

[1] A. Ashok, M. Gruteser, N. Mandayam, J. Silva, M. Varga, and K. Dana. Challenge: Mobile optical networks through visual MIMO. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2010. 88, 93, 98, 111, 112

[2] R. Baheti and H. Gill. Cyber-physical systems. *The Impact of Control Technology, T. Samad and A.M. Annaswamy (eds.), IEEE Control Systems Society*, 2011. 1

[3] F. Bai, T. Elbatt, G. Hollan, H. Krishnan, and V. Sadekar. Towards characterizing and classifying communication-based automotive applications from a wireless networking perspective. In *Proceedings of the IEEE Workshop on Automotive Networking and Applications (AutoNet)*, December 2006. 1, 74, 100

[4] D. Balfanz, G. Durfee, R. E. Grinter, D. K. Smetters, and P. Stewart. Network-in-a-Box: How to set up a secure wireless network in under a minute. In *Proceedings of the USENIX Security Symposium*, 2004. 132

[5] D. Balfanz, D. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proceedings of the Network and Distributed System Security Conference (NDSS)*, 2002. 10, 11, 25, 131

[6] M. Barth. An emissions and energy comparison between a simulated automated highway system and current traffic conditions. In *Proceedings of the IEEE Conference on Intelligent Transportation Systems*,

2000. 1, 74

[7] S. M. Bellovin and M. Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1992. 49

[8] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *Proceedings of the ACM Conference on Computer and communications security (CCS)*, 2004. 81, 89

[9] S. Brands and D. Chaum. Distance-bounding protocols. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, 1993. 2, 126

[10] D. Brumley and D. Boneh. Remote timing attacks are practical. In *Proceedings of the USENIX Security Symposium*, 2003. 64

[11] bump Technologies. Bump. `http://bu.mp`. 1, 8, 9, 23

[12] Bureau of Transportation Statistics. National transportation statistics. `http://www.bts.gov/publications/national_transportation_statistics/`. 73

[13] W. E. Burr, W. T. Polk, and D. F. Dodson. Recommendation for electronic authentication. Special Publication SP 800-63, NIST, 2004. 71

[14] L. Buttyan, T. Holczer, and I. Vajda. On the effectiveness of changing pseudonyms to provide location privacy in vanets. In *Proceedings of the European Workshop on Security and Privacy in Ad hoc and Sensor Networ (ESAS)*, 2007. 79, 89, 110

[15] G. Calandriello, P. Papadimitratos, A. Lloy, and J.-P. Hubaux. Efficient and robust pseudonymous authentication in VANET. In *Proceedings of the ACM International Workshop on VehiculAr InterNETworking (VANET)*, 2007. 81, 85, 89

[16] CAMP Vehicle Safety Consortium. Vehicle safety communications project - final report: Identify intelligent vehicle safety applications enabled by dsrc. Technical Report DOT HS 809 859, National Highway

Traffic Safety Administration, March 2005. 75, 94

[17] S. Capkun, M. Cagalj, R. Rengaswamy, I. Tsigkogiannis, J.-P. Hubaux, and M. Srivastava. Integrity codes: Message integrity protection and authentication over insecure channels. *IEEE Transactions on Dependable and Secure Computing*, 5(4), 2008. 103

[18] S. Capkun, M. Cagalj, and M. Srivastava. Secure localization with hidden and mobile base stations. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2006. 2, 127

[19] S. Capkun and J. Hubaux. Secure positioning of wireless devices with application to sensor networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 2005. 2, 127

[20] M. Carbone and L. Rizzo. Dummynet revisited. *SIGCOMM Computer Communications Review*, 40, April 2010. 19

[21] C. Castelluccia and P. Mutaf. Shake them up! A movement-based pairing protocol for cpu-constrained devices. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2005. 9, 11, 23, 130

[22] D. Chaum. Blind signatures for untraceable payments. In *Proceedings of the International Cryptology Conference (CRYPTO)*, 1982. 49, 54

[23] Q. Chen, F. Schmidt-Eisenlohr, D. Jiang, M. Torrent-Moreno, L. Delgrossi, and H. Hartenstein. Overhaul of ieee 802.11 modeling and simulation in ns-2. In *Proceedings of the ACM Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWiM)*, 2007. 112

[24] CNN. Agency chief: Data on stolen VA laptop may have been erased. `http://www.cnn.com/2006/US/06/08/vets.data/`, 2006. 42

[25] M. D. Corner and B. D. Noble. Zero-interaction authentication. In *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2002. 39, 132, 133

[26] D. Halperin et al. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2008. 9, 130

[27] M. L. Damiani, E. Bertino, B. Catania, and P. Perlasca. Geo-rbac: A spatially aware rbac. *ACM Transactions on Information and System Security*, 10(1), 2007. 42

[28] D. E. Denning and P. F. MacDoran. Location-based authentication: Grounding cyberspace for better security. In *Internet Besieged: Countering Cyberspace Scofflaws*. 1997. 1

[29] R. Dhamija, D. Tygar, and M. Hearst. Why phishing works. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2006. 22

[30] T. Dierks and C. Allen. The TLS protocol version 1.0. Internet Request for Comment RFC 2246, Internet Engineering Task Force, January 1999. Proposed Standard. 15, 57

[31] J. R. Douceur. The sybil attack. In *Proceedings of the International Workshop on Peer-to-Peer Systems (IPTPS)*, 2002. 6, 77, 88

[32] Federal Highway Administration. Next generation simulation. `http://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm`. 111

[33] M. Flury, M. Poturalski, P. Papadimitratos, J.-P. Hubaux, and J.-Y. Le Boudec. Effectiveness of distance-decreasing attacks against impulse radio ranging. In *Proceedings of the ACM Conference on Wireless Network Security (WiSec)*, 2010. 127

[34] J. Freudiger, M. Raya, M. Flegyhzi, P. Papadimitratos, and J.-P. Hubaux. Mix-zones for location privacy in vehicular networks. In *Proceedings of the ACM Workshop on Wireless Networking for Intelligent Transportation Systems (WiN-ITS)*, 2007. 79, 89, 110

[35] R. M. Gerdes, T. Daniels, M. Mina, and S. Russell. Device identification via analog signal fingerprinting: A matched filter approach. In

Proceedings of the Network and Distributed System Security Symposium (NDSS), 2006. 78

[36] M. Ghosh, A. Varghese, A. Gupta, A. A. Kherani, and S. N. Muthaiah. Misbehavior Detection Scheme With Integrated Root Cause Detection in VANET. In *Proceedings of the ACM International Workshop on VehiculAr InterNETworking (VANET)*, 2009. 77, 135

[37] P. Golle, D. Greene, and J. Staddon. Detecting and correcting malicious data in vanets. In *Proceedings of the ACM International Workshop on VehiculAr InterNETworking (VANET)*, 2004. 77, 134

[38] M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun. Loud and clear: Human-verifiable authentication based on audio. In *Proceedings of the International Conference on Distributed Computing (ICDCS)*, 2006. 11, 25, 130

[39] J. J. Haas and Y.-C. Hu. Communication requirements for crash avoidance. In *Proceedings of the ACM International Workshop on VehiculAr InterNETworking (VANET)*, 2010. 90, 113

[40] J. J. Haas, Y.-C. Hu, and K. P. Laberteaux. Design and analysis of a lightweight certificate revocation mechanism for vanet. In *Proceedings of the ACM International Workshop on VehiculAr InterNETworking (VANET)*, 2009. 81, 82, 89, 92

[41] J. J. Haas, Y.-C. Hu, and K. P. Laberteaux. The impact of key assignment on vanet privacy. *Security and Communication Networks*, 3(2), 2010. 79, 83, 104, 108

[42] J. A. Halderman, S. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. Felten. Lest we remember: cold boot attacks on encryption keys. In *Proceedings of the USENIX Security Symposium*, 2008. 133

[43] T. Halevi and N. Saxena. On pairing constrained wireless devices based on secrecy of auxiliary channels: The case of acoustic eavesdropping. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2010. 9, 10, 131

[44] E. Haselsteiner and K. Breitfuss. Security in near field communication (nfc). In *Proceedings of the Workshop on RFID Security (RFIDSec)*, 2006. 124

[45] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.-W. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *Proceedings of the International Conference on Ubiquitous Computing (Ubicomp)*, 2001. 9, 11, 23, 130

[46] IBM. Ibm pci-x cryptographic coprocessor. `http://www-03.ibm.com/security/cryptocards/pcixcc/order4764.shtml`, February 2007. 82

[47] IEEE. 802.11p: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments. IEEE Standards, `http://standards.ieee.org/getieee802/download/802.11p-2010.pdf`. 75

[48] IEEE. 1609.2: Trial-use standard for wireless access in vehicular environments-security services for applications and management messages. IEEE Standards, 2006. 76, 92, 112

[49] J. Linksky et al. Simple Pairing Whitepaper, revision v10r00. `http://www.bluetooth.com/NR/rdonlyres/0A0B3F36-D15F-4470-85A6-F2CCFA26F70F/0/SimplePairing_WP_V10r00.pdf`, August 2006. 8, 9, 11, 25, 130

[50] A. Jain, A. Ross, and U. Uludag. Biometric template security: Challenges and solutions. In *Proceedings of the European Signal Processing Conference (EUSIPCO)*, 2005. 133

[51] T. H.-J. Kim, A. Studer, R. Dubey, X. Zhang, A. Perrig, F. Bai, B. Bellur, and A. Iyer. Vanet alert endorsement using multi-source filters. In *Proceedings of the ACM International Workshop on VehiculAr InterNETworking (VANET)*, 2010. 6, 77, 135

[52] T. Kindberg, K. Zhang, and N. Shankar. Context authentication using

constrained channels. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, 2002. 44, 45, 124

[53] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage. Experimental security analysis of a modern automobile. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2010. 92

[54] A. Kumar, N. Saxena, G. Tsudik, and E. Uzun. Caveat emptor: A comparative study of secure device pairing methods. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2009. 11

[55] S. Laur and K. Nyberg. Efficient mutual data authentication using manually authenticated strings. In *Proceedings of the International Conference on Cryptology and Network Security (CANS)*, 2006. 11, 25, 130

[56] J. Lester, B. Hannaford, and B. Gaetano. Are you with me? - using accelerometers to determine if two devices are carried by the same person. In *Proceedings of the International Conference on Pervasive Computing (Pervasive)*, 2004. 9, 11, 23, 130

[57] Y.-H. Lin, A. Studer, H.-C. Hsiao, J. McCune, K.-H. Wang, M. Krohn, P.-L. Lin, A. Perrig, H.-M. Sun, and B.-Y. Yang. SPATE: Small-group PKI-less Authenticated Trust Establishment. In *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2009. 11

[58] A. Lipton, H. Fujiyoshi, and R. Patil. Moving target classification and tracking from real-time video. In *Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV)*, 1998. 23, 131

[59] P. MacKenzie and M. K. Reiter. Networked cryptographic devices resilient to capture. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2001. 133

[60] R. Mayrhofer and H. Gellersen. Shake well before use: Intuitive and secure pairing of mobile devices. *IEEE Transactions on Mobile Com-*

puting, 8(6):792–806, 2009. Submitted 2008-06-15, accepted 2009-02-10. 9, 11, 23, 130, 131

[61] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2005. 11, 25, 131, 132

[62] R. Mears and L. Ponemon. Enterprise@risk: Privacy & data protection survey. `http://www.deloitte.com/dtt/article/0,1002,cid=182733,00.html`, 2007. 39

[63] Microsoft. BitLocker drive encryption: Technical overview. `http://technet.microsoft.com/en-us/windowsvista/aa906017.aspx`. 39, 63, 132

[64] F. Monrose, M. K. Reiter, Q. Li, and S. Wetzel. Cryptographic key generation from voice. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2001. 133

[65] National Highway Traffic Safety Administration. Fatality analyis reporting system. `http://www.nhtsa.gov/FARS`. 73

[66] OpenSSL. The OpenSSL project. `http://www.openssl.org`. 43

[67] R. OReilly, A. Khenkin, and K. Harney. Sonic nirvana: Using mems accelerometers as acoustic pickups in musical instruments. *Analog Dialogue*, 43, February 2009. 31

[68] A. Papadimoulis. Security by oblivity. `http://thedailywtf.com/Articles/Security_by_Oblivity.aspx`. 133

[69] PayPal. Paypal mobile payments. `https://personal.paypal.com/us/cgi-bin/?&cmd=_render-content&content_ID=marketing_us/mobile_payments`. 1, 4, 9

[70] A. Perrig and D. Song. Hash visualization: A new technique to improve real-world security. In *Proceedings of the International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC)*, 1999. 11, 25, 130

[71] R. Prasad and N. Saxena. Efficient device pairing using "human-

comparable" synchronized audiovisual patterns. In *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS)*, 2008. 130

[72] Privacy Rights Clearinghouse. A chronology of data breaches. `http://www.privacyrights.org/ar/ChronDataBreaches.htm`. 38

[73] R. Rajamani, H.-S. Tan, B. K. Law, and W.-B. Zhang. Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons. *IEEE Transactions on Control Systems Technology*, 8(4), July 2000. 76, 93

[74] K. B. Rasmussen and S. Capkun. Realization of rf distance bounding. In *Proceedings of the USENIX Conference on Security*, 2010. 126

[75] M. Raya, P. Papadimitratos, V. D. Gligor, and J.-P. Hubaux. On data-centric trust establishment in ephemeral ad hoc networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, 2008. 6, 77, 135

[76] M. Raya, P. Papadimitratos, and J.-P. Hubaux. Securing vehicular communications. *IEEE Wireless Communications Magazine, Special Issue on Inter-Vehicular Communications*, 2006. 81, 82

[77] K. Regan. No end in sight: Data breach tally approaches 100 million. `http://www.technewsworld.com/story/53222.html`. 39

[78] B. Rosenberg. Chronology of data breaches 2006: Analysis. `http://www.privacyrights.org/ar/DataBreaches2006-Analysis.htm`. 38

[79] RSA Labortories. RSA SecurID. `http://www.rsa.com/rsalabs/node.asp?id=1156`. 133

[80] A.-R. Sadeghi, M. Selhorst, C. Stüble, C. Wachsmann, and M. Winandy. TCG inside?: a note on TPM specification compliance. In *Proceedings of the ACM Workshop on Scalable Trusted Computing (STC)*, 2006. 70

[81] T. Saito, S. Haruyama, and M. Nakagawa. Inter-vehicle communication and ranging method using led rear lights. In *Proceedings of the*

IASTED International Conference on Communication Systems and Networks (CSN), 2006. 88, 95, 97, 111, 112

[82] K. Sampigethaya, L. Huang, M. Li, R. Poovendran, K. Matsuura, and K. Sezaki. CARAVAN: Providing location privacy for vanet. In *Proceedings of Conference on Embedded Security in Cars (ESCAR)*, 2005. 79, 110

[83] N. Sastry, U. Shankar, and D. Wagner. Secure verification of location claims. In *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, 2003. 2, 125, 127

[84] N. Saxena, J.-E. Ekberg, K. Kostiainen, and N. Asokan. Secure device pairing based on a visual channel. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2006. 29, 30, 131, 132

[85] N. Saxena and M. B. Uddin. Automated device pairing for asymmetric pairing scenarios. In *Proceedings of the International Conference on Information and Communications Security (ICICS)*, 2008. 131, 132

[86] N. Saxena and J. Watt. Authentication technologies for the blind or visually impaired. In *Proceedings of the USENIX Workshop on Hot Topics in Security (HotSec)*, 2009. 9, 130

[87] SecureStar. DriveCrypt: Disk encryption and data encryption software. `http://www.securstar.com/products_drivecrypt.php`. 39, 132

[88] S. Sheikholeslam and C. A. Desoer. Longitudinal control of a platoon of vehicles. In *Proceedings of the American Control Conference*, 1990. 76

[89] S. Sheikholeslam and C. A. Desoer. Longitudinal control of a platoon of vehicles with no communication of lead vehicle information. In *Proceedings of the American Control Conference*, 1991. 76, 113

[90] K. Small. Data breaches caused by human error, hardware theft. http://www.itnews.com.au/News/87188,data-breaches-caused-by-human-error-hardware-theft-survey.aspx. 38

[91] Society of Automotive Engineers SAE. J2735 - Dedicated Short Range Communications (DSRC) Message Set Dictionary. SAE Standards. 75, 112

[92] C. Soriente, G. Tsudik, and E. Uzun. BEDA: Button enabled device pairing. In *Proceedings of the International Workshop on Security for Spontaneous Interaction (IWSSI)*, 2007. 130

[93] C. Soriente, G. Tsudik, and E. Uzun. Hapadep: Human-assisted pure audio device pairing. In *Proceedings of the International Conference on Information Security (ISC)*, 2008. 131, 132

[94] F. Stajano and R. J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proceedings of the Security Protocols Workshop*, 1999. 131

[95] A. Studer, M. Luk, and A. Perrig. Efficient mechanisms to provide convoy member and vehicle sequence authentication in VANETs. In *Proceedings of the International Conference on Security and Privacy in Communication Networks (SecureComm)*, 2007. 134

[96] A. Studer, E. Shi, F. Bai, and A. Perrig. TACKing together efficient authentication, revocation, and privacy in vanets. In *Proceedings of the IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks (SECON)*, 2009. 77, 81, 85, 89, 92

[97] N. O. Tippenhauer and S. Capkun. Id-based secure distance bounding and localization. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, 2009. 2, 127

[98] Trusted Computing Group. TPM main specification. Main Specification Version 1.2 rev. 103, Trusted Computing Group, July 2007. 42, 63

[99] U. Uludag, S. Pankanti, and A. K. Jain. Fuzzy vault for fingerprints. In *Proceedings of the International Conference on Audio- and Video-Based Biometric Person Authentication (AVBPA)*, 2005. 133

[100] E. Uzun, K. Karvonen, and N. Asokan. Usability analysis of secure pairing methods. In *Proceedings of the Workshop on Usable Security (USEC)*, 2007. 11, 48, 130, 139

[101] S. Vaudenay. Secure communications over insecure channels based on short authenticated strings. In *Proceedings of the International Cryptology Conference (CRYPTO)*, 2005. 11, 25, 130

[102] VINT Project, University of Berkeley/LBNL. NS-2:network simulator. `http://www.isi.edu/nsnam/ns/`. 111

[103] Volvo Car Corporation. First demonstration of SARTRE vehicle platooning. `https://www.media.volvocars.com/global/enhanced/en-gb/Media/Preview.aspx?mediaid=36000`. 74

[104] M. Waibel. SARTRE: Autonomous car platoons. `http://spectrum.ieee.org/automaton/robotics/industrial-robots/sartre-autonomous-car-platoons`. 76

[105] M. Wei, L. M. Grupp, F. E. Spada, , and S. Swanson. Reliably erasing data from flash-based solid state drives. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, 2011. 133, 134

[106] A. Whitten and J. D. Tygar. Why johnny can't encrypt: a usability evaluation of pgp 5.0. In *Proceedings of the USENIX Security Symposium*, 1999. 139

[107] M. S. Wood and S. M. Schultz. *Three Mile Island*. Greenwood Press, Westport, CT, USA, 1988. 2

[108] Y. Xi, K. Sha, W. Shi, L. Schwiebert, and T. Zhang. Enforcing privacy using symmetric random key-set in vehicular networks. In *Proceedings of the International Symposium on Autonomous Decentralized Systems (ISADS)*, 2007. 81, 83

[109] B. Xiao, B. Yu, and C. Gao. Detection and localization of sybil nodes in vanets. In *Proceedings of the Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks (DIWANS)*, 2006. 135