Looks Good to Me: Authentication for Augmented Reality

Ethan D. Gaebel

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science and Applications

Wenjing Lou, Committee Chair

Ing-Ray Chen

Guoqiang Yu

April 25, 2016

Falls Church, Virginia

Keywords:

Looks Good to Me: Authentication for Augmented Reality

Ethan D. Gaebel

(ABSTRACT)

Augmented reality is poised to become the next dominant computing paradigm over the course of the next decade. With three-dimensional graphics and interactive interfaces that rival the best science fiction novels. Users will want to have shared experiences in these rich mixed reality scenarios, but surely users will want to restrict who can see their content in certain scenarios. It is currently unclear how users of such devices will authenticate one-another, particularly in scenarios where access to the Internet is restricted or undesirable. Traditional authentication protocols rely on a trusted authority to bootstrap authentication between two users, but augmented reality content sharing will usually occur in face to face scenarios where it will be advantageous to keep communications and authentication localized. Looks Good To Me (LGTM) is an authentication protocol for augmented reality headsets that leverages the unique hardware and context provided with augmented reality headsets to solve an old problem in a more usable and more secure way. LGTM works over point to point wireless communications so users can authenticate each other in any circumstance and is designed with usability at its core, requiring users to only perform two actions: one to initiate and one to confirm. LGTM allows users to intuitively authenticate one another, seemingly only each other's faces. Under the hood LGTM uses a combination of facial recognition and wireless localization to ensure secure ridiculously simple authentication.

Security-based abstract and stuff!!!

This work was supported in part by the National Science Foundation under Grants CNS-1443889, CNS-1405747, CNS-1446478, and CNS-1343222.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Augmented reality is one of the most exciting new technologies on the horizon, promising three-dimensional interfaces that one or more users can directly interact with. These immersive interfaces are commonly provided via head-mounted displays with translucent lenses that render digital content in two or three dimensions on top of the real world. Many of these head-mounted displays will be stand-alone computers equipped with powerful processors, wireless communications, one or more high resolution cameras, and depth sensors. This sophisticated array of hardware is used to provide fully interactive augmented reality which requires: hand tracking, graphics processing, and wireless communication, since connectivity is a requirement for any consumer device today.

Users of augmented reality headsets have the same need that users of any other consumer electronic device have today: the need to send content to others. With augmented reality, this content will usually be far richer, and thus larger, than we've seen on other platforms, since users will be creating, viewing, and working with three-dimensional objects, which inherently have more information associated with them than their equivalent two-dimensional

representations. It is also expected that users will share a large amount of content face to face since this will provide 3D objects that both users can see and/or interact with in real time. The current analog to face to face digital sharing in augmented reality is using a smart phone to show a video or picture to someone. The sharing action is the same, but the sharing medium has switched from physically displaying one's screen to another to sharing the content across two users screens. We expect this type of sharing to increase with augmented reality as much of the physical world's in person content exchanges transition to the digital world, like so may other content streams in the past.

Content sharing between users in close proximity is an interesting and well-studied [$CITATIONS$] problem. In comparison, today's typical content sharing schemes take advantage of preexisting infrastructure such as cell towers, wireless access points, and the Internet. This makes sense considering the most common use case for digitally sharing content currently is to share it with those who aren't present. You wouldn't send someone standing next to you a video, you'd pull out your phone and show them. But with augmented reality you would send someone sitting next to you a video so that the two of you can watch it together through your respective headsets.

This type of local sharing deserves separate consideration from the general content sharing problem. Since we expect it to become so pronounced it will be worthwhile to explore and design for the specific conditions that separates local sharing with remote sharing. One of the new augmented reality headsets to hit the market, the Microsoft HoloLens, recognizes the value in this and has separate support for local sharing vs. remote sharing [1]. Local sharing is piped over a local area network instead of going through the Internet and back. Another way is with point to point wireless communication, which is a very small step from a local area network since most WiFi hardware can use WiFi Direct, a popular point to

point wireless protocol [2]. Also, point to point wireless communication is more dynamic and universally usable since it doesn't require any infrastructure.

Regardless of how it is accomplished, local network resources should be used over global ones for local content sharing, particularly when the content is stored locally on a user's device, as will often be the case with photographs, three-dimensional scene captures, videos, shared app interfaces, and user-created three-dimensional content. By sharing content locally, network resources are saved and money may be saved if charge-by-data-usage plans are being used for Internet access. On top of this, local content sharing is robust in the face of global infrastructure failures, and content sharing privacy details are kept between the two users doing the sharing instead of being shared with whatever entity is providing the sharing service, such as who is communicating with whom, what is being communicated, and when the communication is occurring.Our work in this paper focuses on specifically using point to point communication to facilitate local content sharing in augmented reality systems. This is largely due to preserve generality, since point to point communication does not have any infrastructure dependencies. As a result of this generality much of our work can be translated into the local area network approach to local content sharing.

When working in a localized communications scenario where there is no central authority, authentication can be tricky. It will often be the case that two users have not used their devices to communicate before, meaning there will be no preexisting security context between the two devices such as a pin, key, token, etc. This problem is commonly known as device pairing and it is well studied [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]. We go into more detail about existing device pairing work in the background section.

Using our specific instance of device pairing we have developed a novel solution to pair two

augmented reality headsets that is quick and simple while providing a high degree of security. This solution leverages the unique hardware capabilities required for augmented reality along with facial recognition and very recent advances in the area of wireless localization. The core idea of our scheme relates to how humans authenticate other human speakers. When trying to determine whom is speaking, our brains localize the source of the audio signal heard and then we match up that origin with a face that our brain has also recognized [$CITATION$, $FOR\,REAL$!]. We take a wave-signal source, localize it, then pair it with a face. Our work builds on this idea except instead of localizing an audio signal we localize a wireless signal that is adjacent to a face and recognize the face automatically using facial recognition. By doing this, we create an authentication scheme robust against man-in-the-middle attacks that reduces the problem of pairing for the user to simply looking at another user's face and indicating to their augmented reality headset that it "Looks Good To Me", thus establishing a secure connection. This system is aptly called "Looks Good To Me" or LGTM for short.

In this paper we present the two-party device pairing protocol LGTM along with a full open-source implementation of the protocol. The remainder of this paper is divided up as follows. Section II surveys several relevant areas including augmented reality and device pairing. Section III presents the system model, threat model, security objectives, and the LGTM protocol. Section IV presents analyses of LGTM covering security, usability, and privacy implications of LGTM. In section V we present the details of our implementation of LGTM, including a link to the complete open-source code-base. Section VI presents experimental results involving localization accuracy and overall protocol performance. In section VII we discuss research areas that could improve LGTM further. In section VIII we conclude.

1.1 Background

1.1.1 Augmented Reality

Augmented reality has been around as a research topic since as early as 1993 [15], but only recently have augmented reality headsets been pursued as consumer and business products [16, 17, 18, 19]. Augmented reality at its core is taking what we see in reality and overlaying additional digital objects on top of reality in the form of two-dimensional or three-dimensional renderings. Additionally, users can directly interact with the digital objects using their hands or other parts of their body. Additional peripheral devices may be used to increase immersion and responsiveness, but the most common case is for users to use their hands.

There are several different ways to render digital objects over reality. Augmented reality smart phone apps exist on every platform and these apps rely on taking in reality from the smart phone's camera and other sensors, displaying it on the smart phone screen, and rendering additional digital objects over reality on the screen of the smart phone [20, 21, 22]. More natural approaches to augmented reality rely on projectors, some of which require special glasses to view properly [23]. However the vision for future augmented reality experiences lies in the head-mounted display, a large pair of glasses with a mechanism to deliver light encoding specific digital objects directly into the user's field of vision. The first such head-mounted display to gain public notoriety was Google Glass [19]. Glass renders two-dimensional objects into the user's field of vision and really acted as an ever-present extension of the smart phone interface. This interface was used via voice commands and buttons on the side of the glasses. This version of Glass did not become a consumer product and indeed it may have partially sullied the reputation of augmented reality. Now however we see a

wave of augmented reality head-mounted displays coming to market. The Meta 1 and 2 [17] the Microsoft HoloLens [18], Magic Leap [16], several small startups focused on augmented reality, and recent confirmation of Google working on a next generation of Glass [24]. A common characteristic of these new headsets is that they are more immersive than Glass, all allowing users to directly interact with digital objects using natural hand movements, enabled by infrared depth sensors that capture user gestures and the environment around them. Depth information is combined with other sensory data to provide a service referred to as SLAM, Simultaneous Location and Mapping [$CITATION\ SLAM$]. This current and upcoming augmented reality platform is the target platform for LGTM.

Another trend we see in practical augmented reality headsets is towards them being stand-alone computers. The Microsoft HoloLens is already a fully functioning computer and the company Meta has stated that it intends to move towards a full stand-alone computer headset for consumer devices [17]. This trend implies that augmented reality users will be able to enjoy all the current advantages of computers with the addition of the powerful sensors and paradigm changing user interfaces that come with augmented reality. This allows us to look at traditional problems that arise in the context of computers and computing and reexamine these existing problems and their existing solutions in the context of augmented reality computers with an eye to improve upon the current state-of-the-art by using the additional functionality provided via the hardware and abilities from augmented reality. One of these problems is device pairing, which is a common procedure that must be done to connect two devices together such as a headset and a smart phone, a speaker and a smart phone, two smart phones, or any other wireless peripheral with a smart phone or computer.

1.1.2 Device Pairing

Device pairing is the area dedicated to authenticating devices without prior security context, and there have been many schemes introduced to address this problem on devices with myriad hardware features and constraints [7, 25, 10, 11, 12, 13, 14, 4]. Virtually all of these methods rely on communication over one or more out-of-band channels. An out-of-band channel is any channel which is not the primary communications channel being used. This secondary channel can be defined quite broadly and many schemes use human sensory capabilities or human activity as the out-of-band channel. The primary channel, that the secondary channel is used to authenticate, is a human imperceptible channel, most often the wireless channel. So most device pairing schemes proceed as follows. Some information is transmitted over the out-of-band channel which is then used for authentication in the primary channel via some specialized authentication procedure. These authentication procedures can vary widely depending on what sort of out-of-band channel is used and what sort of information is transmitted across it. The simplest and most common example is for a numerical pin to be exchanged or verified between two devices via one or more human actors. Depending on the scheme, the pin may need to be entered on both devices, transmitted between two devices and then verified as matching by one or more users, or some complex combination of the two [26, 27, 28]. When human actors are involved, the exchanged information is often short to improve usability [27], and thus possesses a low level of entropy, reducing its security. As a result a good deal of research has gone into how to improve the security short strings. The two dominant protocols that have emerged from this are known as short authenticated string (SAS) protocols [8] and so-called manual authentication (MANA) protocols [9], with SAS protocols enjoying more widespread use in practical systems such as Bluetooth [26]

Bluetooth versions 2.1 and beyond use secure simple pairing for pairing two devices and it includes four different pairing options [26, 29, 30]. Numeric comparison, where both devices display a 6-digit numeric string that must be confirmed on both devices. Passkey entry, where the user enters a 6-digit numeric string on one or both devices; in the case where the 6-digit numeric string is entered on one device it must be verified on the other. Out-of-band mode, where a secure key is agreed on using a non-human out-of-band channel, such as near-field communication (NFC). The fourth pairing option is called just works and unlike the other three options it does not have any protection against man-in-the-middle attacks. Just works prompts the user to confirm a device name which does not have security properties; the only security provided in just works mode is through timing and Bluetooth's relatively short range, neither of which protect a user from a determined attacker in a public space. Just works was included for compatibility and usability reasons, to maintain compatibility with Bluetooth 2.0 devices and to improve usability for end users [27] if they decide that the other three techniques are too much of a hassle. It should come as no surprise that the just works option has become a primary attack point on the secure simple pairing protocol [31]. This security compromise for usability and backwards compatibility, which is really another sort of usability, has opened up a multitude of devices to attack. But usability is central to device pairing since human interaction is almost always a requisite component for bootstrapping authentication between two devices that have no preexisting context.

Usability in device pairing schemes has accumulated its own body of research [32, 33, 7, 34, 35, 3] through protocol comparisons, usability studies, and analyses of security issues that usability can affect. Usability is important because it can affect protocol adoption and even the security of the the protocol. If users are prompted to confirm a numeric string that appears on a pair of devices and they confirm it without thorough checking it is easy for

incorrect devices to be authenticated, which is a huge security leak. One study investigating usability's impact on security in device pairing protocols found that users made mistakes with 4-digit numeric comparison-based pairing that lead to security errors 20% of the time [35]. The study found security errors to be alarmingly high for three other pairing schemes based on entering and comparing 4-digit numeric strings as well. The device pairing community has given a great deal of attention to inventing new pairing protocols which rely on new hardware capabilities [7, 25, 10, 11, 12, 13, 14, 4], but many of these schemes have been found to not be very usable when given to users [35, 3]. This was further examined in [33] where usability studies coupled with user interviews revealed that users prefer different pairing methods based on the context: location, time urgency, and sensitivity of the information being exchanged after pairing. The study also revealed that many non-technical users are interested in using new pairing techniques, but that their perceptions of security and the context in which the pairing was occurring greatly influenced which pairing technique was going to be used.

Pairing using augmented reality headsets is a greatly unexplored subset of device pairing, which is understandable due to the newness of practical augmented reality headsets. However it has recently been revealed that the Microsoft HoloLens can use QR codes that are displayed on a device, with either a screen or a sticker, to authenticate peripheral devices [36]. Beyond this we are unaware of any device pairing work having been done with augmented reality headsets, making our work on LGTM one of the first to specifically address pairing on augmented reality headsets.

Chapter 2

LGTM Protocol

2.1 Security Opportunities with Augmented Reality

Device pairing schemes have taken advantage of myriad interesting sensors and hardware, but many of these have involved unrealistic or awkward pairing scenarios. With the paradigm shift that is augmented reality we have an influx of new hardware coupled with a new sort of user and device context that includes how certain pieces of hardware are positioned. For instance, on smart phones the main camera is positioned on the back of the phone and is generally not facing the same direction as the user's gaze. But with augmented reality the camera is lined up with the user's line of sight. This context can be leveraged in interesting ways. With this setup the augmented reality headset can preprocess what the user is seeing. This preprocessing can lead to the user's vision being annotated in certain ways that the user finds useful. Another example of hardware positioning that we can utilize is that of wireless communications antennas. Wireless localization can now be used for more than just localizing a device, thanks to the close proximity of the wireless antennas to a person's face

we can localize a wireless signal and then pair that location with a user's face using proximity alone.

By combining the area occupied by a user's face with the area of a wireless signal's origin we can localize the signal, preprocess the area around the signal's origin by searching for a face matching some facial recognition model, and present matching faces to the user for verification, thus associating a specific user with a specific wireless signal and authentication both. We can use this authentication to secure key exchange mechanisms, like Diffie-Hellman, from man-in-the-middle attacks and provide an incredibly usable device pairing experience to users.

Consider this scenario; Alice and Bob meet for the first time and want to pair their augmented reality headsets to exchange some three-dimensional content. Alice and Bob both trigger the pairing protocol on their devices. Moments later they are both prompted to confirm one-another's faces, which are outlined on their respective augmented reality displays. Alice and Bob confirm one-another's faces and now they are free to communicate over an encrypted channel. What could be simpler than that? This is what LGTM promises.

2.2 System Model

First we must adequately understand the scenarios where the Looks Good to Me (LGTM) authentication protocol is applicable. This requires us to go down the list of: hardware requirements, security context, the users' context, and all other assumptions we make. LGTM is specifically designed for authenticating two users, Alice and Bob. Alice and Bob are assumed to be users equipped with head-mounted augmented reality displays which are also

stand-alone computers equipped with: wireless communications hardware with support for a point to point communications protocol, software and hardware support for wireless localization using the same wireless hardware used for communication, a high-definition video camera, and a translucent display directly in front of the users' eyes that is capable of displaying digital objects on top of the physical world (this last requirement is satisfied simply by the users possessing head-mounted augmented reality rigs, but we want to be thorough).

The two hardware devices are assumed to have no prior security context, which means the two devices do not possess any information that can be used to prove the authenticity of their claimed identity to the other. A pre-shared key is a common example of a piece of information which can be used as preexisting security context.

An important piece of information that each headset is assumed to possess is a facial recognition model capable of recognizing the user of the headset. That is, Alice's headset has a facial recognition model that can recognize her, and Bob's headset has a facial recognition model that can recognize him. Training these models can easily be done in a mirror.

As for the users themselves, Alice and Bob are assumed to be in sight of each other and would like to share some sort of digital content with one another, such as: a shared application experience, a shared video, shared music, shared holograms, or basic messaging. No assumptions are made of Alice and Bob's relationship with each other: they may be friends, acquaintances, or strangers.

No assumption regarding access to the Internet is made.

2.3 Security Objectives

The primary security objective is for Alice and Bob to correctly authenticate one another's wireless signals so that they can engage in secure point to point wireless communication. Recall from the system model section that Alice and Bob do not share any security context, so this authentication must be bootstrapped.

A secondary objective is for Alice and Bob to correctly select which user they want to communicate with. This objective is directly related to the first but it warrants distinction as a worthy problem on its own merit since there are many practical attacks that rely upon tricking users into selecting the wrong thing [$CITATIONS$, $DIVERSE ONES$]. We refer to this as the user-selection problem.

2.4 Threat Model

Attackers have quite a bit of power when dealing with a wireless channel since the communications medium is both public and localized. Attackers have the capability to: jam communications, perform denial-of-service attacks by flooding the channel with packets constructed to fit the protocol, eavesdrop on all packets transmitted, and replay packets collected from eavesdropping in any order. Further, the attacker may have multiple transceivers at his or her disposal, so attacks can be coordinated between more than one node. However it is necessary for an attacker to by physically present, either personally or via a device which they control remotely.

Attackers may use these capabilities to accomplish one or several goals. An attacker may want to impersonate Alice to send Bob falsified content. An attacker may want to imperson-

ate both Alice and Bob to perform a man-in-the-middle attack allowing them to eavesdrop on encrypted communications between Alice and Bob. The attacker may simply want to prevent communication between Alice and Bob. Or the attacker may want to do any of the above steps with the higher goal of exploiting some subsystem.

2.5 Protocol

To begin the protocol Alice presses a button which may be digitally rendered over reality or physically located on the augmented reality headset. At this point Alice's headset begins listening on the local wireless channel and broadcasts an initialization packet containing: Diffie-Hellman parameters g and p corresponding to a generator over a group and a prime number, respectively, Alice's Diffie-Hellman public key, and a randomly generated number. Since Bob is not listening yet Alice's message goes unanswered. Bob presses a button to activate the protocol, opening his device up to listen to the wireless channel and generating an initialization packet containing: Diffie-Hellman parameters g and p, Bob's private key b, Bob's public key computed as: $B = g^b \, mod \, p$, and a randomly generated number, R_{B1}, to serve as half of a session identifier. Bob broadcasts his message $g||p||B||R_{B1}$. Alice receives Bob's initialization packet and generates her private key a and computes her public key in relation to the received Diffie-Hellman parameters g and p as $A = g^a \, mod \, p$. Alice generates a random number R_{A1}, to serve as the second half of a session identifier and broadcasts her public key A concatenated with the random number R_{A1}, and $R_{B1} \oplus R_{A1}$, which is the session identifier for Alice and Bob's current session, where $\oplus$ denotes the exclusive-or operation. This makes Alice's full message: $A||R_{A1}||(R_{A1} \oplus R_{B1})$. Bob receives Alice's message and verifies that it is in response to his initial message by checking that $R_{B1} \oplus R_{A1}$

equals the final number in Alice's message. Alice and Bob both compute the shared key $K = B^a \, mod \, p = A^b \, mod \, p$. At this point Alice and Bob have established a shared symmetric key, but have not established the authenticity of each other's identities.

Bob retrieves facial recognition parameters that define a facial recognition model to recognize his face, F_B. Bob encrypts the facial recognition parameters using the shared key and an initialization vector, IV, which Bob concatenates with his encrypted facial recognition parameters, $IV||E(IV, K, F_B)$, and broadcasts this. Alice receives Bob's encrypted message with his facial recognition parameters and also records the channel state information, CSI_A, associated with the packets carrying Bob's message. Alice decrypts the message and retrieves F_B. Alice computes Bob's location using a wireless localization algorithm denoted by $G(\cdot)$ to get Bob's location: $L_A = G(CSI_A)$. Alice runs a facial recognition algorithm denoted by $H(\cdot)$, using Bob's facial recognition parameters to obtain a location for any faces that match F_B, denoted by $FL_A = H(F_B)$. Alice compares L_A and FL_A to see if the coordinates overlap. If they do not match then they are thrown out. If there are no matches among all the options for L_A and FL_A then the protocol is aborted. If there are matches then Alice's device renders a box around Bob's face, matched by F_B, and whose location, FL_A, overlaps with the corresponding L_A. Alice is prompted to verify that the protocol has selected the correct face. In practice there may be many pairs of faces and locations to check at once since multiple users may be carrying out LGTM at the same time. In this case Alice selects Bob's face from the available faces, if it is present. If Alice fails to select Bob's face then the protocol aborts at this point.

Otherwise, Alice retrieves facial recognition parameters that define a facial recognition model to recognize her face, F_A. Alice uses the shared key, K and the last several bytes of F_B as an initialization vector, IV, to compute the encryption, $E(IV, K, F_A)$ and broadcasts it. Bob

receives Alice's encrypted message with her facial recognition parameters and also records the channel state information, CSI_B, associated with the packets carrying Alice's message. Bob decrypts the message using the shared key, K, and the initialization vector derived from the last several bytes of F_B and retrieves F_A.

Bob computes Alice's location using the wireless localization algorithm to get: $L_B = G(CSI_B)$. Bob runs the facial recognition algorithm, using Alice's facial recognition parameters to obtain a location for any faces that match F_A, denoted by $FL_B = H(F_A)$. Bob compares L_B and FL_B to see if the coordinates overlap. If they do not match then they are thrown out. If there are no matches among all the options for L_B and FL_B then the protocol is aborted. If there are matches then Bob has a box drawn around Alice's face, matched by F_A, and whose location, FL_B, overlaps with the corresponding L_B. Bob is prompted to verify that the protocol has selected the correct face. In practice there may be many pairs of faces and locations to check at once since multiple users may be carrying out LGTM at the same time. In this case Bob selects Alice's face from the available faces, if it is present. If Bob fails to select Alice's face then the protocol aborts. Otherwise, the protocol has successfully completed, Alice and Bob have established a secure key K and they have authenticated each other's wireless signals ensuring that they are communicating with who they think they are. They are now free to send encrypted content back and forth.

2.6 Protocol Analysis

2.6.1 Components (This title sucks, get a better one)

LGTM uses several techniques to achieve user authentication. As a result it's constructive to unpack the purpose of each technique along with what LGTM gains by having that technique included.

LGTM uses Diffie-Hellman key exchange [37] for establishing a shared key between Alice and Bob. Using Diffie-Hellman provides fast public-private key pair generation, even more so if elliptic curve Diffie-Hellman is used. High-speed key generation allows us to generate a new key pair every time we perform pairing. Making the public-private key pairs ephemeral provides LGTM with perfect forward secrecy, meaning that if an attacker obtains one of the private keys used in a single pairing the other public-private key pairs along with the symmetric keys derived from them remain secure.

In practice Alice and Bob may be receiving several messages at once since it is not apparent which wireless signals can be ignored and which one should be focused on until the protocol is complete. To differentiate between wireless signals Alice and Bob use the random numbers, R_{A1} and R_{B1}, and combine them using the exclusive-or operation to generate a unique session key. Using this session key Alice and Bob can quickly determine which stream of messages an incoming message belongs to, but this session key does not have security guarantees, it is only used for message routing in non-malicious scenarios.

LGTM's security rests atop wireless localization's accuracy. Localization connects a wireless signal to a physical location by computing the signal's point of origin. This location can be matched with a device or person occupying that location and with human assistance

this device or person can be authenticated. Wireless localization remains a difficult and unsolved problem and many current schemes are too inaccurate to be suitable for security applications. Very recent work has show great strides in improving localization precision and in reducing the amount and cost of hardware required to perform localization [38, 39, 40]. The most recent and most promising of these methods [40] achieves $< insertaccuracy >$ with a single laptop equipped with the Intel 5300 wireless card and three antennas. Consider now that LGTM's localization scenario is a constrained one. If two users are performing LGTM they must have a line of sight between each other and in the most common use-case they will be quite close to one-another, likely within three meters. If we look at the results in [40] with these constraints in mind we see that this localization technique achieves accurate localization down to less than 20 cm of error, which is sufficient for security applications.

This brings us to facial recognition. We state above that wireless localization is what allows LGTM to authenticate a device or person, so what use is facial recognition? Having users authenticate a physical location, identified by say, a sphere, requires users to search for the sphere and verify that it intersects with the person they're attempting to communicate with. The facial recognition parameters exchanged in LGTM serve to make it easier for the user to solve the user-selection problem. Humans are phenomenal at facial recognition. It was found in [41] that infants as young as 1-3 days old are able to recognize faces. LGTM only asks for facial verification, not even full recognition. Verifying that an encircled face belongs to the person being authenticated is a far simpler task than verifying that a sphere intersects with the person's face. This is a great win for usability. In practice, users may be performing LGTM with several parties at once since they do not know which party is actually the one they wish to communicate with until the protocol is complete. This could result in several sphere's being scattered across a user's vision, which may or may not match

up with anyone's current location since the original source of the signal may have moved. By using facial recognition to effectively preprocess what the user is currently seeing we are able to eliminate locations obtained via localization that do not match up with the location of the face identified by the facial recognition parameters. This reduces the chances of a user selecting an invalid location either accidentally or through an attacker's careful misdirection. Once the invalid selections are eliminated there may still be multiple valid face-location pairs to choose from, but selecting a face remains a far simpler task than picking out a sphere or another sort of location representation.

Using facial recognition parameters to aid in the user-selection problem does not just affect usability, it also affects security. While the facial recognition parameters are not considered private, they are not instantly obtainable either. Training a facial recognition model for someone requires several photographs of a person, which can be obtained either online, by performing LGTM with the person, or in person by taking photos of a person. This is perfectly feasible for a targeted attacker, but this work in obtaining facial recognition parameters serves to significantly dampen the powers of spontaneous random attackers.

Consider a small node with wireless capabilities hidden in a discrete location that is designed to exploit every opportunity to attack wireless protocols. This node will be unable to attack LGTM since it would have to obtain facial recognition parameters as well. Furthermore, LGTM is specifically designed so that such a node cannot begin LGTM with a user and obtain their facial recognition parameters for use later. Notice in $< figurereference >$ the user who initiates LGTM sends their facial recognition parameters and is authenticated first, meaning a malicious user cannot use LGTM to harvest other user's facial recognition parameters without providing valid ones themselves, along with a valid human face that matches them. This makes the spontaneous random attacker's task significantly more difficult.

The final step in the protocol is for the actual human users to select a face from the options LGTM provides. As mentioned before, humans are excellent at identifying other human faces. On top of this, humans are great at recognizing the difference between a mask and a true human face. Without the human verification step LGTM could be fooled by an attacker wearing a mask. Human verification serves as a final check against spoofing attempts. On top of this, as stated before, a user may be unintentionally performing LGTM in concert with several users at once. At the end of the protocol, the user may have several faces left to select from, and so the user must select one face from several potential choices, solving the user-selection problem. In most cases the selections will be very limited since the two users are expected to be facing one another and to have a very short time window to initiate the protocol.

2.6.2 Security Against Attacks

We have spoken at length about the security properties the techniques used in LGTM possess, now we discuss how LGTM performs against attacks.

Man-in-the-middle attacks plague virtually every pairing protocol [3]. The very nature of pairing opens these protocols up to man-in-the-middle attacks since the purpose is to bootstrap authentication when there is no prior authentication information. This absence makes it possible for an attacker to insert themselves in-between the two devices and trick them both. LGTM, however, has protection against man-in-the-middle attacks not afforded to other pairing protocols thanks to wireless localization. Localization pairs a signal to a location so attackers must occupy the same physical location as both users to successfully launch a man-in-the-middle attack. This is theoretically possible using two coordinated

devices, although very difficult.

An attacker, Eve, can use two small devices physically attached to Alice and Bob's augmented reality headsets that mirrors the messages between Alice and Bob, replacing the Diffie-Hellman public keys and random numbers with her own. These devices could forego localization since they do not have any way to verify the locations of Alice and Bob and pushing through the protocol, "verifying" every response they get. This will result in two nearly identical choices for Alice and Bob. This sort of man-in-the-middle attack can be mitigated by notifying the user that two devices appear to be occupying almost the same space. However users often rush during authentication stages [42] so users may complete LGTM before the second choice is presented and it is unclear which device will be presented first. This entire attack procedure may sound rather preposterous, but as electronics shrink more and more this attack will become more and more feasible. While not a large threat today, it may be a significant one in the future, perhaps even by the time augmented reality headsets become popular consumer products.

A more realistic man-in-the-middle attack for the present is one reliant on inaccurate localization results. As mentioned above, localization can not be called a solved problem yet, and any implementation of LGTM will have to deal with the currently accepted localization techniques, errors and all. The procedure would proceed as in the attack above, except Eve's devices will not need to be as close. We assume that implementors of LGTM will not rely on security through obscurity, so Eve will be aware of the localization abilities of Alice and Bob's device's implementations. Eve can position her devices behind, adjacent to, or in front of Alice and Bob with less precision and run her man-in-the-middle attack as above.

Inaccurate localization results can also be used in one-way impersonation as shown in 2.2.

Eve might want to impersonate Alice to Bob, so she sets up a device near enough to Alice to fool the localization procedure and transmits Alice's facial recognition parameters. Eve will also need to obtain a facial recognition model for Alice, which she can do by participating in LGTM with Alice, searching for photos of Alice on the Internet, or by taking photos of Alice herself and then training a facial recognition model. As mentioned in the previous subsection, this attack will be partially defended against in the case where Eve is a spontaneous random attacker since the facial recognition parameters, while not secret, do take some work to acquire, work which an attacker is most likely unable to do if they are attacking random users.

User impersonation will usually be protected against even further by context. LGTM's most likely use-case is that Alice and Bob want to share digital content via their augmented reality headsets and that they are already interacting and conversing. If Eve impersonates Alice to Bob, then Alice will not be sharing anything with Bob and vice-versa, which will not go unnoticed. Bob will likely terminate the connection and try again since it will be apparent that the protocol failed. This is slightly similar to pairing protocols which use an out-of-band channel to transmit authentication information, except in this case the out-of-band channel is being used for verification of a successful pairing. It is possible though to imagine scenarios where Bob might not notice that he is paired with someone besides Alice. Perhaps for some reason Alice and Bob are only pairing their devices and not communicating outside of that interaction. In this scenario Eve's impersonation will succeed. It is our hope that wireless localization will continue to grow more and more precise, as it has over its entire history, to the point where inaccurate localization is no longer a valid attack vector.

Denial-of-service attacks are a common attack across pretty much all devices with network connectivity. Here we discuss two different ways that denial-of-service attacks can be

launched against LGTM. The first is common to all wireless devices, jamming the wireless channel so that messages are not successfully received at all. There aren't any practical defenses against jamming, and thus it succeeds against LGTM. It is worth noting though that jamming requires expensive equipment and that it is illegal in many countries.

The second avenue for denial-of-service is through the protocol itself. LGTM has fairly expensive steps, particularly the localization step. If Eve spams Alice or Bob's device with LGTM requests they will waste significant time processing these requests. But this attack goes further. Since these augmented reality headsets are stand-alone and mobile they will be battery-powered. By spamming Alice or Bob's device and forcing heavy computation Eve can execute a battery drain attack. The best defense against an attack like this is to have some sort of monitoring on LGTM protocol messages and if too many are received the protocol can be aborted. This is similar to TCP's congestion control mechanism [43]. This prevents battery drain attacks, but does nothing to stop the denial-of-service attack. There is no way to really defend against this attack either.

When implemented properly LGTM is very secure against the most common attacks on pairing protocols: man-in-the middle and user impersonation attacks. This security is however reliant on the accuracy of the implementation's wireless localization technique, but we expect the accuracy of wireless localization techniques to continue to increase. In fact, in the time between the start of work on this research and now a new technique was published [40] bringing another leap forward in wireless localization.

2.6.3 Usability

Usability is a great thing to have for any piece of technology. It increases the likelihood of adoption, decreases avoidable user error, and reduces user frustration, which is linked to the other two points.

The International Standards Organization defines usability as "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use" [44]. Besides being good for user satisfaction, usability can be an important factor in increasing security by reducing user errors that lead to security issues. It is no surprised that usability is increasingly becoming a focus for security schemes [$CITATIONS$], especially in pairing [32, 33, 7, 34, 35, 3]. One study of device pairing methods [35] found that increasing the quality and usability of a user interface in a security scheme decreased errors that lead to a security failure in one pairing method by 20 percentage points and another by 7.5 percentage points. These two pairing methods were not seemingly complex schemes either. The first one required comparing two short alphanumeric strings on each device and confirming that they matched. The second one required selecting matching alphanumeric strings from a list on each device. It's also important to note that in this study the devices were under the control of a single user. One might expect that if the devices were controlled by different users then the error rates could be even higher.

So to improve user satisfaction and increase security, LGTM is designed to be a highly usable security scheme that is both a pleasure to use and difficult to make security errors with. Combining facial recognition and wireless localization to reduce user-device authentication to two actions is very user-friendly. By adding facial recognition on top of wireless localization

LGTM can auto-remove choices whose wireless signal location and face location do not match up. Users outside the device user's field of view are also auto-eliminated by context since the user is not facing that direction. Reducing the number of choices available to a user is a fantastic way to increase usability [$FINDACITATIONFORTHISCOMMENT$]. When we consider the device pairing usability study mentioned above, we see that reducing the number of user choices should also reduce user error, since there are fewer wrong choices. Consider further that instead of recognizing a string of numbers users are recognizing human faces, which humans are adept at [41]. LGTM's combination of facial recognition and wireless localization makes for a very usable scheme which will also help prevent users from making security errors during authentication.

Usability is also impacted by the communications method used for LGTM. Point to point device communication is always available, barring local device failures, which inherently makes it more usable since it can be used in a larger variety of scenarios. Consider the common case of being on an airplane with no reliable wireless connection or the even more common case of being in an area with no WiFi and no cellular connection. This second case is present every day for many people in both rural and undeveloped areas with slow or nonexistent Internet connections and as a result sharing content with people who are present by going over the Internet would range from prohibitively time consuming to impossible.

2.6.4 Privacy

A common misconception is that most people are not concerned about digital privacy. However a 2015 study by the Pew Research Center [45] found that 86% of Internet users have taken steps to remove or mask their digital footprints online and 55% of Internet users

have taken steps to avoid surveillance by specific individuals, organizations, or governments. When it comes to user privacy, LGTM delivers. Since LGTM relies on point to point wireless communication, attackers must be co-located with their targets, significantly diminishing the reach that arbitrary attackers can exert. Prolonged digital surveillance of an individual's point to point content sharing would require something akin to stalking, which we believe is unlikely to occur as often as hacking does on the Internet.

Furthermore, by sending data from point to point we avoid empowering a central authority with hordes of user data. Who users talk to, who users are close to, what users are sending to one another, and when. This is data that many users would like to keep private, but that many large companies see as a gold mine. The safest way to keep this data from being abused is not to make it available to third-parties. With LGTM this data is as private as any untapped conversation between two people.

Figure 2.1: Protocol diagram.

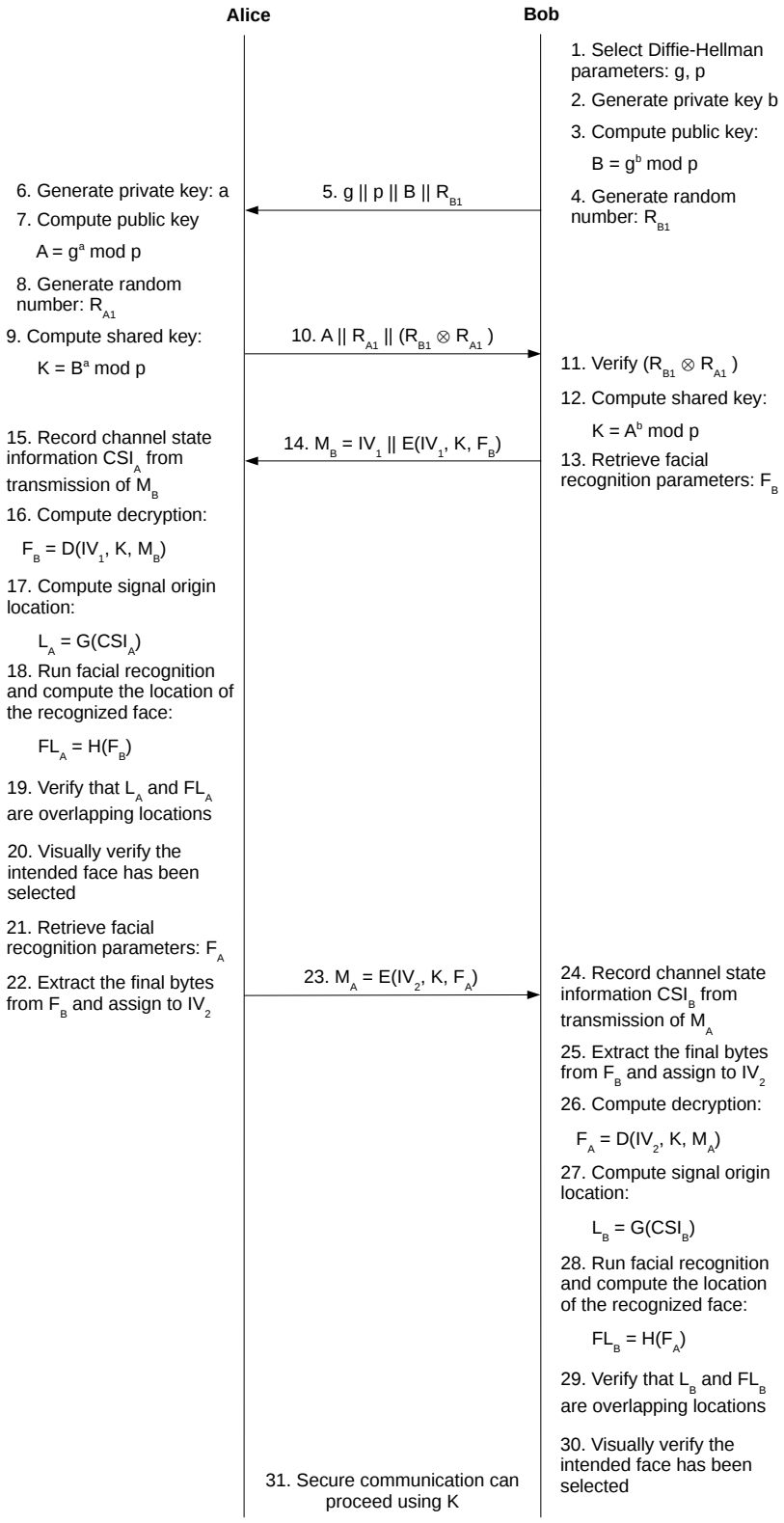
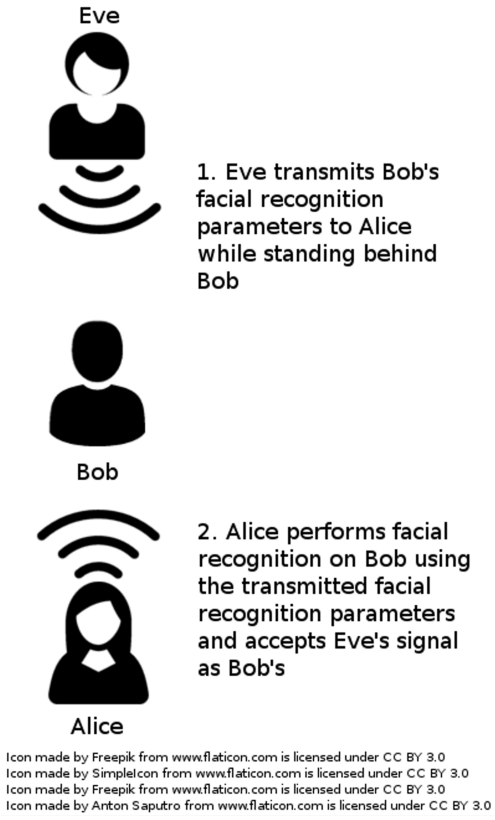


Figure 2.2: Eve performs a user-impersonation attack by broadcasting from behind Bob, thus impersonating Bob to Alice.



Chapter 3

Implementation

3.1 Testbed

Real world augmented reality headsets are still fresh on the market and are very rare. On top of this the few headsets on the market do not provide sufficient access to information about the wireless channel that is necessary for wireless localization. These two factors make implementing LGTM on real augmented reality headsets infeasible.

However, we have stripped down LGTM's hardware requirements to the bare minimum leaving us in need of: a high-definition video camera, wireless point to point communication capabilities, wireless localization capabilities, a display, reasonable computational abilities, and close proximity of a face to wireless antennas. We satisfy all of these requirements by equipping two Fujitsu LifeBook T900 laptops running Linux Mint 17 Qiana with: a Logitech C270 720p web cam, an Intel 5300 wireless card with custom firmware for Linux from Halperin et al. [46] that enables retrieval of channel state information and point to

point wireless communication using 802.11n, an array of three antennas affixed to the back of the laptop screen at 10 centimeter intervals, and finally by attaching printed out pictures of faces from the Yalefaces dataset [47] to the back of the laptop over the antenna array. This last point is important since augmented reality headset users will have their face directly adjacent to the wireless transmitter which LGTM exploits to couple the wireless signal and the user together. This context is an extremely important enabler of LGTM. To be sure, our testbed itself is not a practical setup, but it perfectly emulates the requisite hardware that real augmented reality headsets will be equipped with.

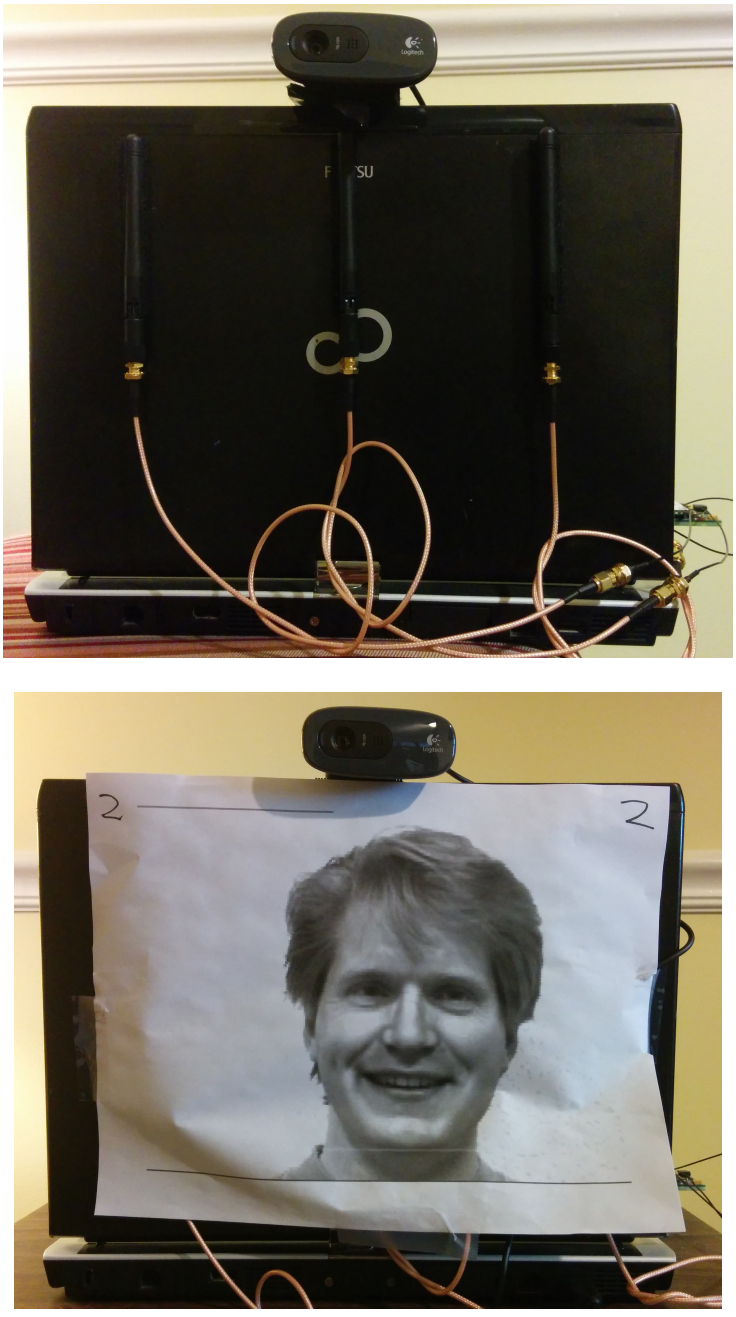
3.2 Technologies and Software

LGTM employs many technologies in service of bootstrapping secure communication and these technologies have several different implementation choices.

3.2.1 Wireless Communication

We use the 5 GHz channel of 802.11n as our wireless channel. There are known issues that arise with the Intel 5300 wireless card and the channel state information the custom firmware provides when it is operated on the 2.4 GHz channel [48, 39]. To listen to the channel and send and receive packets we use the existing supplementary software for the Intel 5300 CSI Tool [46]. We make some minor modifications to the existing code to handle packet decoding in addition to channel state information recording and to facilitate reading data from files to be transmitted. The supplementary software is written in a combination of MATLAB and C. To configure the wireless card and OS settings we use the Linux networking 'ip' command

Figure 3.1: Testbed laptop shown without a printed out picture of a face and with a printed out picture of a face.



and wireless networking 'iw' command extensively [$CITATIONS\ FOR\ THESE$??].

3.2.2 Cryptography

Technologies

In our implementation we employ elliptic curve Diffie-Hellman key exchange instead of standard Diffie-Hellman key exchange to reduce the number of bits required for the public-private key pairs, thus increasing performance [49]. Elliptic curve Diffie-Hellman requires an appropriate elliptical curve to perform computations over and we use curve secp256r1, specified in [50]. This curve is used by default in our implementation so there is no agreement step to come to a consensus between the devices involved and the parameters g and p in 2.1 are not transmitted.

For symmetric encryption we use the advanced encryption standard (AES) block cipher with a 128 bit key [51] in Galois Counter Mode (GCM) [52]. Galois counter mode provides both authentication and encryption for messages by using a keyed hash as a message authentication code or HMAC for short. GCM only operates over block ciphers with 128-bit blocks, hence our choice of AES-128 over AES-256.

Software

To implement LGTM's cryptographic components we use the popular Crypto++ C++ library, which provides functions and classes to support elliptic-curve Diffie-Hellman, random number generation, AES, and Galois counter mode for various symmetric key algorithms.

Using these crypto-primitives I built up two main files: one to run the cryptographic building

blocks of LGTM and one to use these building blocks to form the messages sent back and forth between Alice and Bob. The cryptographic building blocks for LGTM are separated into 6 functions:

- generateRandomNumber
- generateDiffieHellmanParameters
- diffieHellmanSharedSecretAgreement,
- generateSymmetricKeyFromSharedSecret
- encryptFile
- decryptFile

The function names should be mostly self-explanatory, they serve to generate a random number and return it, generate a public-private key pair for Diffie-Hellman and return them, agree on a shared key using Diffie-Hellman, generate a symmetric key using the shared secret, and encrypt and decrypt files. In my implementation these functions can be found under the cryptography folder in the lgtm_crypto.cpp file.

These functions are used by the higher level functions that construct messages to send at each step of the protocol. These messages can all be mapped back to steps in the protocol diagram in figure 2.1. Their names are:

- firstMessage
- replyToFirstMessage

- secondMessage
- replyToSecondMessage
- decryptThirdMessageReply

FirstMessage generates a Diffie-Hellman public-private key pair, generates a random number that's 256 bits long, and writes the random number and public key to a single file as well as writing all the generated data to separate files for retrieval later. ReplyToFirstMessage splits a received file into the random number and public key from firstMessage, generates a public-private key pair, derives the shared secret and symmetric key using the other public key, generates a random number, exclusive-ors the generated random number and the received random number together, then writes the generated parameters to separate files and the generated public key, random number, and exclusive-or of both random numbers to a single file.

SecondMessage splits a received file into the random number, public key, and exclusive-or of both random numbers. Then it derives the Diffie-Hellman shared secret, computes a symmetric key using the Diffie-Hellman secret, writes all the newly generated parameters to separate files, and encrypts and saves a file using the shared symmetric key, that holds data to be used for facial recognition.

ReplyToSecondMessage decrypts and saves a received message, exiting with a failure status if the encrypted data is found to have been tampered with. Then it encrypts a file using the shared symmetric key holding data to be used for facial recognition and saves it.

Finally, decryptSecondMessageReply decrypts and saves a received file, exiting with a failure status if the encrypted data is found to have been tampered with.

These various methods are are invoked from the command line by passing the following string arguments to the program:

- first-message
- first-message-reply
- second-message
- second-message-reply
- decrypt-third-message-reply

In my implementation these functions can be found under the cryptography folder in the lgtm_crypto_runner.cpp file.

Data is passed and stored primarily using files. The reasoning behind this is that the cryptography software for LGTM is run periodically as part of the higher-level LGTM program. As a result we must maintain state variables between runs where the program exits as well as reading in data generated by other separate programs. The simplest way to do this is through files.

3.2.3 Facial Recognition

Facial recognition is a multi-step process. Before recognizing a face the faces must be detected. Our implementation detects all faces in the current field of view and then checks each one with facial recognition. Our implementation's facial detection relies upon Haar feature-based cascade classifiers [53]. This method was designed specifically for performance. It

uses many feature detectors which are run in a tree-like fashion, with simpler, faster feature detectors being run first, followed by more complex, more expensive, more discriminating feature detectors.

To recognize the faces detected we require robust facial recognition under sometimes unideal circumstances. Most facial recognition algorithms are tested using image files directly fed into the algorithm. Oftentimes these images are taken under idealized conditions which are not necessarily representative of common use-cases. As a result, we found that several facial recognition algorithms are not sufficiently robust to serve our purposes. In our testbed we are running facial recognition on images from the Yalefaces dataset attached to laptops, but our training set of images is comprised of image files from the Yalefaces [47] dataset. So we need a facial recognition algorithm that can generalize from faces in photos taken in idealized conditions to the unideal case we face in our testbed of recognizing faces in photos affixed to laptops. To serve this need we use linear binary pattern histograms (LBPH) for facial recognition [54, 55]. This technique has been found to be robust in many conditions where facial recognition techniques based on more elegant methods, such as Eigenfaces [56] and Fisherfaces [47], fail. Our cursory evaluation of facial recognition techniques indeed found that Eigenfaces and Fisherfaces were insufficient for our applications, recognizing approximately half of the faces in Yalefaces, while LBPH succeeds in recognizing all the faces in Yalefaces.

We used the OpenCV 3.0 [57] C++ library in our implementation to compare Eigenfaces, Fisherfaces, and LBPH for facial recognition and also for facial detection using Haar feature-based cascade classifiers. OpenCV provides out-of-the-box implementations for all of these methods.

I put together these components, and others, to implement LGTM. The Yalefaces dataset is preprocessed before the training sets for each individual is put together. This preprocessing step crops the image to include the face and as little else as possible. This was done using the Haar cascade classifier on each image and then cropping on the detected face. I experimented with data augmentation techniques to increase the size of the dataset. Specifically, I perturbed each training image by many combinations of angles along all three axes. Unfortunately this did not create a better classifier for my use-case, in fact, it made it worse. This was true for Eigenfaces, Fisherfaces, and LBPH.

When LGTM is being run I retrain an LBPH classifier using a newly received set of photos of one of the individuals from the Yalefaces dataset. This retraining takes less than 1 second, so it is not a significant contributor to the computational cost of running LGTM.

3.2.4 Wireless Localization

The most intricate of the technologies involved in LGTM is by far wireless localization. The recent advances in wireless localization [38, 39, 40] are indeed what make LGTM feasible. For our testbed we implemented a modified version of SpotFi [39], adapted to run from a single device instead of coordinated devices. SpotFi is an angle of arrival based localization technique that relies on the classical MUSIC algorithm [58] to compute the angle that a wireless signal originates from relative to an array of antennas.

MUSIC

The key to MUSIC is the existence of a phase difference between each signal as it arrives at each antenna in an array. These phase differences occur because the antennas are at different

Figure 3.2: The steering vector form used in music. Each element indicates the phase offset from the first antenna in an array, which is why the first element is 1. Since the antennas in the array are spaced at a constant distance, d, the phase offset at each array element is a multiplicative increase.

$$\begin{bmatrix} 1 \\ e^{-j2\pi d\frac{sin(\theta_k)f}{c}} \\ e^{2*(-j2\pi d\frac{sin(\theta_k)f}{c})} \\ \dots \\ e^{(M-1)*(-j2\pi d\frac{sin(\theta_k)f}{c})} \end{bmatrix}$$

locations, meaning a single wireless signal must travel different distances to arrive at each antenna in an antenna array, this causing a phase difference. $< MAKEFIGUREILLUSTRATINGTHIS! >$ When expressed as a complex exponential we can write the signal at the Nth antenna as:

$$\Phi(\theta_k) = e^{-j2\pi d\frac{sin(\theta_k)f}{c}}$$

where j is $\sqrt{-1}$, f is the wireless signal's frequency, d is the distance between antennas in the array, c is the speed of light, and θ_k is the angle of arrival for path k. The vector of these expressions is referred to as the steering vector, denoted $a(\theta)$, and is a function of θ since the remaining parameters are constant.

The MUSIC paper models a received wireless signal, the column vector X with M elements, in terms of a matrix, A, with D columns, each of which is a steering vector parameterized with a different θ. The D steering vectors are for the D different signal paths resulting from multipath effects. Each of these steering vectors has M elements, one for each antenna,

making A an $M \times D$ matrix. This matrix is multiplied multiplied by a column vector of D complex gains, and this expressions has Gaussian white noise added to it. All this is written as:

$$X = AF + W$$

For the purpose of localization we seek to compute A from the information we have about the physical layer through our setup and the measured signal, X. To start with, MUSIC computes the covariance matrix $S = XX^H = AFF^HA^* + WW^H$, where H denotes the conjugate transpose. This expression is decomposed using eigendecomposition so that we get: $S = AFF^HA^H + \lambda S_0$, where S_0 represents noise and λ is an eigenvalue of S. When the elements of the noise vector W have mean 0 and variance σ^2 $\lambda S_0 = \sigma^2 I$, where I is the identity matrix. When the number of paths, D, is less than the number of sensors, or in our case antennas, M, then AFF^HA^H is singular, making its determinant 0. This insight yields the expression:

$$|AFF^HA^H| = |S - \lambda S_0| = 0$$

If A is a full rank matrix and FF^H is positive definite then AFF^HA^H must be a positive semi-definite matrix, meaning that the λ in the expression above must be the minimum eigenvalue $\lambda_{min} \geq 0$.

The rank of AFF^HA^H is D and this can be shown by computing the eigenvalues of S, since λ_{min} will be repeated $P = M - D$ times. Using this we can compute the number of paths, D, from measurements by determining P by counting the number of times the minimum eigenvalue is repeated for S.

Back to our steering vectors. Recall that there are D actual angle of arrivals from multipath. When these true angle of arrivals are input to the steering vector it will be orthogonal to what is referred to as the noise sub-space. MUSIC computes the noise sub-space by computing the eigenvectors of the signal covariance matrix and constructing a new matrix using the eigenvectors whose eigenvalues are at a minimum as columns. There are P such eigenvectors. This matrix is denoted E_N.

To use MUSIC for computing the angle of arrival we compute $P_{MUSIC}(\theta) = \frac{1}{a^*(\theta)E_N E_N^* a(\theta)}$ for a range of θ values and see where P_{MUSIC} peaks. The largest peak is the direct path angle of arrival.

SpotFi

Due to multipath effects there will usually be multiple peaks. Multipath is resolved in MUSIC by having more sensors, which in our case are antennas. Specifically, the number of sensors in MUSIC must be more than the number of multipath components. However average indoor conditions usually have five paths resulting from multipath effects and having five antennas on a consumer device is currently impractical and expensive.

The key insights in SpotFi to overcome limitations from multipath and a limited number of antennas are using channel state information measurements from each subcarrier in 802.11n combined with a method of creating more sensors using mathematical tricks while only requiring three antennas.

To understand this first insight I will briefly explain what subcarriers are. Many modern communications systems, including 802.11n, rely on a scheme called orthogonal frequency division multiplexing (OFDM). From a high level, OFDM divides up a band of

frequency into separate channels and transmits different data using each channel. These separated channels are referred to as subcarriers since each has a carrier frequency which is a small part of a different central frequency which defines the section used for OFDM. $< GET\ A\ GOOD\ FIGURE\ FOR\ THIS >$.

To use these subcarriers first requires a modification to the model of a wireless signal presented in the original MUSIC paper. Instead of modeling X as a column vector of M measurements, from M antennas, we model X as an $M \times N$ matrix, where N equals the number of subcarriers used. This results in the channel gain matrix F being modeled as a $D \times N$ matrix and the noise matrix W being modeled as a $M \times N$ matrix.

The authors of SpotFi perform their manipulations on the expression, minus the noise matrix, W, for simplicity, so I will do the same for the remainder of my explanation of SpotFi.

SpotFi modifies MUSIC to use steering vectors that consider both time of flight and angle of arrival. As stated above, MUSIC uses phase differences across sensor arrays, phase differences between subcarriers are too small to be useful since the difference is $2\pi(m-1)d(f_i - f_j)\sin(\theta_k)/c$, where f_i and f_j denote the frequencies of two subcarriers. This phase difference is rendered insignificant by the speed of light term, c, in the denominator. The maximum spacing between subcarriers in 802.11 is 40 MHz. At this distance, with antennas spaced at half a wavelength any angle of arrival introduces a phase shift of at most 0.002 radians across the subcarriers, which can easily be a result of noise. The authors of SpotFi introduce time of flight to obtain more discriminating phase differences. Time of flight between subcarriers can also be expressed as a complex exponential:

$$\Omega(\tau_k) = e^{-j2\pi f_\delta \tau_k}$$

where f_δ is the frequency difference between subcarriers and τ_k is the time of flight for the kth path. The phase difference for time of flight across two adjacent subcarriers can be computed with: $-2\pi(n-1)f_\delta\tau_k$, where f_δ is the frequency difference between two adjacent subcarriers. Since there is no speed of light term in the denominator the phase difference that exists between subcarriers is significant. SpotFi uses these phase differences between subcarriers for time of flight the same way classical MUSIC uses the phase differences between antennas.

However, to properly utilize the new time of flight phase shifts some additional work must be done. Recall that MUSIC requires that the number of measurements from the sensor array that can be written as a linear combination of steering vectors must be greater than the number of paths. To satisfy this requirement the SpotFi authors construct a new signal measurement matrix X using the recorded signal information.

Recall that in the classical MUSIC algorithm the steering vector is a vector of signals obtained across an antenna array that is parameterized only by an angle θ. SpotFi's modification in adding time of flight makes the steering vector a two parameter function, $a(\theta, \tau)$, where τ is time of flight. However, the new steering vector and signal measurement matrix must be carefully manipulated to fit the constraints put forward by MUSIC. The trick to doing this is to construct the new X such that the sensors are overlapping sub-arrays of sensors in terms of time of flight and angle of arrival.

Since MUSIC is a mathematical technique, the authors of SpotFi move $\Phi_{\theta k}$ and $\Omega_{\tau k}$ terms into the matrix of complex gains, F so that A can be expressed in terms of steering vectors whose only difference is the values of θ and τ that they're parameterized by.

See figure 3.3 to see exactly how this manifests itself. This altered signal matrix is referred to as the smoothed channel state information matrix and this signal matrix can be said

to have 30 sensors total, which is achieved by creating all possible shifts of 15 subcarriers between two antennas. Now that we have the new steering vector $a(\theta, \tau)$ and the smoothed channel state information matrix we can input this into the classical MUSIC algorithm, with minor modifications to account for both angle of arrival and time of flight. These minor modifications amount to simply plugging in different values for θ and τ in the expression $P_{MUSIC}(\theta, \tau) = \frac{1}{a^H(\theta,\tau)E_N E_N^H a(\theta,\tau)}$ instead of only plugging in different θ values. We search for peaks in the same way that we would with only θ, except expanded to two-dimensions.

SpotFi runs the modified MUSIC algorithm for several packets and then clusters the results using each packet's computed normalized time of flight and normalized angle of arrival as two-dimensional coordinates. See figure 3.4 for an example. The original SpotFi paper used Gaussian mean clustering to achieve this, which requires as input the expected number of clusters to generate, which equates to the number of multipath components. To make our implementation more robust to changing conditions we used agglomerative clustering, which only takes qualitative cluster measurement parameters such as: the inner squared distance from point to cluster and the distance between the centers of clusters before the clusters are merged into one. These parameters can be easily determined prior to deployment. To determine these parameters I used collected channel state information from an ASUS 5 GHz router by pinging it when it was at several known locations. Afterwards I ran SpotFi on this data several times and tuned the clustering parameters until the clusters seemed correct. This tuning should hold across different deployments of LGTM since the time of flight and angle of arrival used in clustering are normalized. Therefore the distances are over normalized values which do not change.

A cluster is selected as the correct choice using a weighted likelihood computed using: number of points per cluster, average angle of arrival for the cluster, average time of flight for the

cluster, and mean time of flight.

$$w_C C_k - w_\theta \sigma_{\theta_k} - w_\tau \sigma_{\tau_k} - w_s \mu_{\tau_k}$$

Note that C_k is the number of points in cluster k, σ_{θ_k} is the variance over the angle of arrival for cluster k, σ_{τ_k} is the variance over the time of flight for cluster k, and μ_{τ_k} is the mean of the time of flight. The main idea behind this likelihood is that the cluster with the most points and the least variance (tightest clustering) should be selected since the direct signal path should have less variance in its measurements. The term with the mean time of flight for cluster k is included since the signal with shorter time of flight should be more likely to be the direct path signal in line of sight scenarios.

To compute the weights for each of the likelihood terms I used the collected channel state information from the ASUS 5 GHz router again and then fed cluster information labeled with the correct cluster choice into a linear support vector machine [59].

The SpotFi system runs the modified version of MUSIC on multiple wireless access points and then combines the results from each to come up with a precise location. In our implementation however we are working from a single laptop, and so we do no such combination, relying upon the angle of arrival computed on our single receiver. This certainly decreases our localization accuracy in comparison to the original SpotFi system, however for this LGTM proof of concept I think that is acceptable, especially since there has been very little work that can localize a wireless signal from a single point using commodity hardware. Furthermore, there are unique properties of LGTM that should mitigate localization errors.

We find ourselves working in a constrained scenario in comparison to the SpotFi paper however. SpotFi is a general wireless localization technique designed to work in non-line of

sight and line of sight scenarios. In our application we will always have a line of sight between the two users engaging in LGTM. Furthermore the distance between two users is expected to generally be quite small and SpotFi's localization performs better over shorter distances [39]. These constraints make us confident that this adapted SpotFi will be suitable enough for a proof of concept of LGTM. To validate this claim we report localization accuracy results in the experiments section.

Since SpotFi is the most expensive operation in LGTM we performed some optimizations to improve performance. In SpotFi we compute the angle of arrival for each packet using the modified MUSIC algorithm, which is the most expensive part of the entire LGTM protocol. Since the packets are independent of one-another, we parallelize the entire packet processing loop and improve the overall performance significantly. The speed of our processing is thus heavily linked to the number of cores used in processing. Adding additional cores will result in large speed ups.

We implement our altered SpotFi along with the altered MUSIC algorithm from scratch in MATLAB. To our knowledge there is no open-source implementation of SpotFi yet, so we also release the source code for our implementation of SpotFi and the entire LGTM implementation, making it available at: `www.github.com/egaebel/lgtm`.

3.2.5 Bringing It All Together

The technologies described above are implemented in a combination of: C, C++, MATLAB, and Linux shell commands. To tie all this software together and construct a data pipeline for transmission, reception, processing, and user input we use a Bash script to coordinate the flow of control. For passing state variables and data between separate programs we use files

for simplicity. The script first configures the wireless card to listen to the wireless channel for packets containing a special LGTM token and sets up a prompt for the user to initiate LGTM with a keystroke. When one user performs the keystroke on one of the testbed laptops a public key and random number are generated, concatenated together with an LGTM protocol string indicating it is the first message, and broadcasted over the wireless channel. The LGTM protocol proceeds on autonomously as put forward in the protocol diagram in figure 2.1 with one important difference. Instead of sending a facial recognition model for each test device's assigned face, our implementation transmits a folder of images from the Yalefaces dataset, which is tarred (using the program tar) into an archive file. This folder contains a training set which is used by the recipient to train a facial recognition model. We found this method to be more efficient in terms of network usage since OpenCV's facial recognition models were very, very large. After both devices have received the images in the training set along with the channel state information associated with those packets our modified version of SpotFi is run and LGTM trains an LBPH facial recognition model using the training set. Now we have reached the human verification step. At this point a window pops up displaying the real-time stream from the web cam attached to the laptop. Rendered atop the stream is a box around the recognized face (if there is one) and text below the box prompting the user to press space to confirm the face. If both users press space then LGTM is a success, otherwise the users may press escape to terminate LGTM.

Figure 3.3

(a)

$$X = \begin{bmatrix} csi_{1,1} & csi_{1,2} & \dots & csi_{1,16} & csi_{2,1} & csi_{2,2} & \dots & csi_{2,16} \\ csi_{1,2} & csi_{1,3} & \dots & csi_{1,17} & csi_{2,2} & csi_{2,3} & \dots & csi_{2,17} \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ csi_{1,15} & csi_{1,16} & \dots & csi_{1,30} & csi_{2,15} & csi_{2,16} & \dots & csi_{2,30} \\ csi_{2,1} & csi_{2,2} & \dots & csi_{2,16} & csi_{3,1} & csi_{3,2} & \dots & csi_{3,16} \\ csi_{2,2} & csi_{2,3} & \dots & csi_{2,17} & csi_{3,2} & csi_{3,3} & \dots & csi_{3,17} \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ csi_{2,15} & csi_{2,16} & \dots & csi_{2,30} & csi_{3,15} & csi_{3,16} & \dots & csi_{3,30} \end{bmatrix}$$

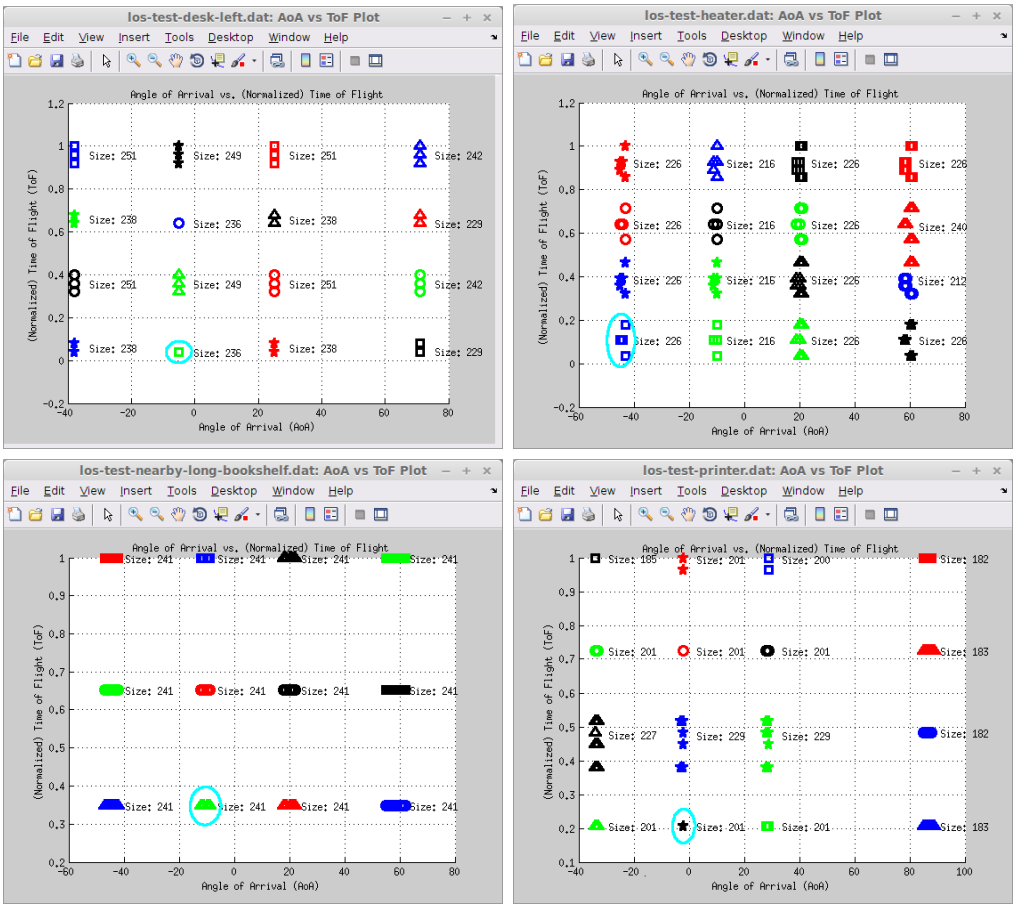
(b)

$$A = \begin{bmatrix} 1 & 1 & \dots & 1 & 1 & 1 & \dots & 1 \\ \Omega_{\tau 1} & \Omega_{\tau 2} & \dots & \Omega_{\tau \frac{L}{2}} & \Omega_{\tau(\frac{L}{2}+1)} & \Omega_{\tau(\frac{L}{2}+2)} & \dots & \Omega_{\tau L} \\ \Omega_{\tau 1}^2 & \Omega_{\tau 2}^2 & \dots & \Omega_{\tau \frac{L}{2}}^2 & \Omega_{\tau(\frac{L}{2}+1)}^2 & \Omega_{\tau(\frac{L}{2}+2)}^2 & \dots & \Omega_{\tau L}^2 \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ \Omega_{\tau 1}^{14} & \Omega_{\tau 2}^{14} & \dots & \Omega_{\tau \frac{L}{2}}^{14} & \Omega_{\tau(\frac{L}{2}+1)}^{14} & \Omega_{\tau(\frac{L}{2}+2)}^{14} & \dots & \Omega_{\tau L}^{14} \\ \Phi_{\theta 1} & \Phi_{\theta 2} & \dots & \Phi_{\theta \frac{L}{2}} & \Phi_{\theta(\frac{L}{2}+1)} & \Phi_{\theta(\frac{L}{2}+2)} & \dots & \Phi_{\theta L} \\ \Omega_{\tau 1}\Phi_{\theta 1} & \Omega_{\tau 2}\Phi_{\theta 2} & \dots & \Omega_{\tau \frac{L}{2}}\Phi_{\theta \frac{L}{2}} & \Omega_{\tau(\frac{L}{2}+1)}\Phi_{\theta(\frac{L}{2}+1)} & \Omega_{\tau(\frac{L}{2}+2)}\Phi_{\theta(\frac{L}{2}+2)} & \dots & \Omega_{\tau L}\Phi_{\theta L} \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ \Omega_{\tau 1}^{14}\Phi_{\theta 1} & \Omega_{\tau 2}^{14}\Phi_{\theta 2} & \dots & \Omega_{\tau \frac{L}{2}}^{14}\Phi_{\theta \frac{L}{2}} & \Omega_{\tau(\frac{L}{2}+1)}^{14}\Phi_{\theta(\frac{L}{2}+1)} & \Omega_{\tau(\frac{L}{2}+2)}^{14}\Phi_{\theta(\frac{L}{2}+2)} & \dots & \Omega_{\tau L}^{14}\Phi_{\theta L} \end{bmatrix}$$

(c)

$$F = \begin{bmatrix} \alpha_1 & \alpha_1\Omega_{\tau 1} & \alpha_1\Omega_{\tau 1}^2 & \dots & \alpha_1\Omega_{\tau 1}^{15} & \alpha_1\Phi_{\theta 1} & \alpha_1\Omega_{\tau 1}\Phi_{\theta 1} & \alpha_1\Omega_{\tau 1}^2\Phi_{\theta 1} & \dots & \alpha_1\Omega_{\tau 1}^{15}\Phi_{\theta 1} \\ \alpha_2 & \alpha_2\Omega_{\tau 2} & \alpha_2\Omega_{\tau 2}^2 & \dots & \alpha_2\Omega_{\tau 2}^{15} & \alpha_2\Phi_{\theta 2} & \alpha_2\Omega_{\tau 2}\Phi_{\theta 2} & \alpha_2\Omega_{\tau 2}^2\Phi_{\theta 2} & \dots & \alpha_2\Omega_{\tau 2}^{15}\Phi_{\theta 2} \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ \alpha_{\frac{L}{2}} & \alpha_{\frac{L}{2}}\Omega_{\tau\frac{L}{2}} & \alpha_{\frac{L}{2}}\Omega_{\tau\frac{L}{2}}^2 & \dots & \alpha_{\frac{L}{2}}\Omega_{\tau\frac{L}{2}}^{15} & \alpha_{\frac{L}{2}}\Phi_{\theta\frac{L}{2}} & \alpha_{\frac{L}{2}}\Omega_{\tau\frac{L}{2}}\Phi_{\theta\frac{L}{2}} & \alpha_{\frac{L}{2}}\Omega_{\tau\frac{L}{2}}^2\Phi_{\theta\frac{L}{2}} & \dots & \alpha_{\frac{L}{2}}\Omega_{\tau\frac{L}{2}}^{15}\Phi_{\theta\frac{L}{2}} \\ \alpha_{(\frac{L}{2}+1)} & \alpha_{(\frac{L}{2}+1)}\Omega_{\tau(\frac{L}{2}+1)} & \alpha_{(\frac{L}{2}+1)}\Omega_{\tau(\frac{L}{2}+1)}^2 & \dots & \alpha_{(\frac{L}{2}+1)}\Omega_{\tau(\frac{L}{2}+1)}^{15} & \alpha_{(\frac{L}{2}+1)}\Phi_{\theta(\frac{L}{2}+1)} & \alpha_{(\frac{L}{2}+1)}\Omega_{\tau(\frac{L}{2}+1)}\Phi_{\theta(\frac{L}{2}+1)} & \alpha_{(\frac{L}{2}+1)}\Omega_{\tau(\frac{L}{2}+1)}^2\Phi_{\theta(\frac{L}{2}+1)} & \dots & \alpha_{(\frac{L}{2}+1)}\Omega_{\tau(\frac{L}{2}+1)}^{15}\Phi_{\theta(\frac{L}{2}+1)} \\ \alpha_{(\frac{L}{2}+2)} & \alpha_{(\frac{L}{2}+2)}\Omega_{\tau(\frac{L}{2}+2)} & \alpha_{(\frac{L}{2}+2)}\Omega_{\tau(\frac{L}{2}+2)}^2 & \dots & \alpha_{(\frac{L}{2}+2)}\Omega_{\tau(\frac{L}{2}+2)}^{15} & \alpha_{(\frac{L}{2}+2)}\Phi_{\theta(\frac{L}{2}+2)} & \alpha_{(\frac{L}{2}+2)}\Omega_{\tau(\frac{L}{2}+2)}\Phi_{\theta(\frac{L}{2}+2)} & \alpha_{(\frac{L}{2}+2)}\Omega_{\tau(\frac{L}{2}+2)}^2\Phi_{\theta(\frac{L}{2}+2)} & \dots & \alpha_{(\frac{L}{2}+2)}\Omega_{\tau(\frac{L}{2}+2)}^{15}\Phi_{\theta(\frac{L}{2}+2)} \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . \\ \alpha_L & \alpha_L\Omega_{\tau L} & \alpha_L\Omega_{\tau L}^2 & \dots & \alpha_L\Omega_{\tau L}^{15} & \alpha_L\Phi_{\theta L} & \alpha_L\Omega_{\tau L}\Phi_{\theta L} & \alpha_L\Omega_{\tau L}^2\Phi_{\theta L} & \dots & \alpha_L\Omega_{\tau L}^{15}\Phi_{\theta L} \end{bmatrix}$$

Figure 3.4: Clustering results from running SpotFi on four different sets of wireless data. The cluster selected by SpotFi is circled in each figure. Time of flight is on the y-axis, angle of arrival is on the x-axis. Notice that angle of arrival is displayed in degrees, but the calculations for clustering are performed over normalized angle of arrivals. The representation in degrees is for readability.



Chapter 4

Experiments

We found it prudent to run a few experiments to validate our implementation laid out in the previous section, particularly our wireless localization technique, as a sort of sanity check. In addition to validating our implementation, we also wanted to identify and validate potential performance enhancements since LGTM's core promise is usable security and slow systems are less usable. Our primary point of evaluation focuses on the performance and accuracy of our implementation of SpotFi adapted for point to point use since this is the most time consuming operation in LGTM, the primary component responsible for the security of this LGTM proof of concept, and our modifications are completely untested.

We also briefly discuss our validation of OpenCV's LBPH facial recognizer since we are using it in a non-traditional scenario. Most research in facial recognition deals with a known dataset of photos and feeds these photos directly into the facial recognition algorithms, often with some preprocessing first. In LGTM we train on photos and test on faces captured via web cam in real-time. This introduces additional noise since the faces are not perfectly centered in the x, y, or z axes and there is inherent noise in picking up a printed out photo

Table 4.1: Base data results.

Distances:	1 m	2 m	3 m
Correct in Top 1:	47.0 %	27.0 %	24.0 %
Correct in Top 2:	77.0 %	41.0 %	36.0 %
Correct in Top 3:	90.0 %	66.0 %	48.0 %
Correct in Top 4:	100.0 %	74.0 %	49.0 %
Correct in Top 5:	100.0 %	74.0 %	49.0 %
Mean Error:	0.838 m	1.543 m	2.374 m
Median Error:	0.803 m	1.282 m	2.368 m

from a camera. For these reasons, we see fit to briefly mention our evaluation methodology and mention the performance timing for training, if only to leave the reader without a doubt of our system's abilities.

4.1 Wireless Localization – SpotFi (FIX THIS TITLE HERE!! THIS ISN'T GREAT!)

4.1.1 Accuracy

Localization accuracy is front and center in LGTM. We rely on wireless localization to provide us with guarantees about a transmitter's location which we then use to bootstrap authentication. We found that our system performs modestly in comparison to other localization works, albeit not up to the high standards that a secure system would require.

Table 4.2: Sampling data results.

Distance:				1 m				
Number of Packets Used in SpotFi:	1025 (no sampling)	750	500	250	100	50	25	10
Correct in Top 1:	47.0 %	47.0 %	49.0 %	47.0 %	51.0 %	49.0 %	58.0 %	60.0 %
Correct in Top 2:	77.0 %	77.0 %	77.0 %	80.0 %	81.0 %	82.0 %	88.0 %	83.0 %
Correct in Top 3:	90.0 %	90.0 %	90.0 %	90.0 %	94.0 %	90.0 %	95.0 %	95.0 %
Correct in Top 4:	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %
Correct in Top 5:	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %	100.0 %
Mean Error:	0.838 m	0.838 m	0.829 m	0.796 m	0.768 m	0.722 m	0.790 m	0.745 m
Median Error:	0.803 m	0.803 m	0.789 m	0.751 m	0.703 m	0.703 m	0.757 m	0.745 m

Distance:				2 m				
Number of Packets Used in SpotFi:	1025 (no sampling)	750	500	250	100	50	25	10
Correct in Top 1:	27.0 %	27.0 %	25.0 %	25.0 %	23.0 %	20.0 %	27.0 %	30.0 %
Correct in Top 2:	41.0 %	41.0 %	40.0 %	45.0 %	42.0 %	46.0 %	50.0 %	47.0 %
Correct in Top 3:	66.0 %	66.0 %	63.0 %	65.0 %	68.0 %	67.0 %	65.0 %	65.0 %
Correct in Top 4:	74.0 %	74.0 %	74.0 %	74.0 %	76.0 %	74.0 %	74.0 %	75.0 %
Correct in Top 5:	74.0 %	74.0 %	74.0 %	76.0 %	76.0 %	74.0 %	74.0 %	75.0 %
Mean Error:	1.543 m	1.543 m	1.487 m	1.392 m	1.565 m	1.547 m	1.484 m	1.644 m
Median Error:	1.282 m	1.282 m	1.213 m	1.041 m	1.278 m	1.407 m	1.233 m	1.520 m

Distance:				3 m				
Number of Packets Used in SpotFi:	1025 (no sampling)	750	500	250	100	50	25	10
Correct in Top 1:	24.0 %	24.0 %	22.0 %	20.0 %	20.0 %	19.0 %	24.0 %	23.0 %
Correct in Top 2:	36.0 %	36.0 %	36.0 %	35.0 %	34.0 %	35.0 %	33.0 %	38.0 %
Correct in Top 3:	48.0 %	48.0 %	45.0 %	44.0 %	43.0 %	42.0 %	42.0 %	45.0 %
Correct in Top 4:	49.0 %	49.0 %	48.0 %	48.0 %	49.0 %	47.0 %	50.0 %	50.0 %
Correct in Top 5:	49.0 %	49.0 %	49.0 %	50.0 %	51.0 %	49.0 %	50.0 %	50.0 %
Mean Error:	2.374 m	2.374 m	2.405 m	2.446 m	2.444 m	2.224 m	2.477 m	2.396 m
Median Error:	2.368 m	2.368 m	2.289 m	2.330 m	2.367 m	1.915 m	2.739 m	2.425 m

To account for acceptable errors in angle of arrival computations I introduce a slight error tolerance of 40 cm $< I may make this smaller... >$ on either side of the face. Since we control the distance and angle between test devices in our experiment we use the error tolerance and distance to compute the tolerance for the angle in degrees.

There are many lessons to gain from the data we collect for LGTM's localization. LGTM uses a likelihood technique to make the final selection of angle of arrival. This can lead to an incorrect angle of arrival being selected over the correct one, sometimes by a very slim margin. To evaluate LGTM's localization accuracy with this in mind we focus on the accuracy of the top-5 angle of arrival selections as well as the mean error and median error.Top-5, mean error, and median error accuracy reports for distances of: 1 meter, 2 meters, and 3 meters. The data we used was gathered with our two testbed laptops at angles of: -20, -10, 0, 10, and 20 degrees, where 0 degrees is when the laptops are directly facing one another, +10 degrees indicates that one laptop is 10 degrees to the right from the center and -10 degrees indicates that one laptop is 10 degrees to the left from the center. The +/- 20 degree cases are the same as the +/- 10 degree cases. For each angle I performed LGTM 10 times and saved off the channel state information used on each laptop. So this yields 20 samples for each angle at each distance and a total of 200 samples for each distance.

The localization accuracy results are reported in table 4.1. As expected, localization accuracy is strongly coupled with the distance between devices. Percent correct in the top-1 position is almost halved when moving from 1 meter to 2 meters and beyond 2 meters the percent correct in the top-5 drops below 50%.

4.1.2 Performance

Wireless localization performance is primarily bounded by the number of packets whose channel state information is used in computing the angle of arrival with our modified SpotFi implementation. As mentioned in the implementation section we parallelize this since processing independent packets is an "embarrassingly parallel" problem. However we still incur high performance penalties at this step due to the large number of packets. We collect channel state information for the packets carrying the facial recognition parameters, which in our evaluation is actually just a tarred folder of photos which are used to train a facial recognition model. The number of packets to transmit this comes to approximately 1025 with minor variations due to slightly different image file sizes. Processing all these packets in the modified SpotFi takes up precious time, but using a large number of packets should intuitively provide better localization results. To see how the trade-off between localization accuracy and processing time when varying the number of packets considered we sample packets uniformly from the packet stream using sample sizes of: 750, 500, 250, 100, 50, 25, and 10. Our findings are presented in table 4.2 in comparison to the baseline localization results with no sampling.

Here we find something very surprising, reducing the number of packets increases our localization accuracy. This could be explained by the presence of bursty noise which occurs in large chunks and throws off the clustering step in SpotFi, but this is simply a speculation at this point. Regardless of what is causing this unexpected effect, I can certainly say that it is very positive. What I expected to be a trade-off between computation time and localization accuracy turned into an increase for both, a most pleasant surprise!

4.2 Facial Recognition

In deploying facial recognition in LGTM we utilized the Yalefaces [47] dataset to provide the faces to be recognized. This dataset is not large, it includes photos of 15 subjects with 12 photos per subject. Most of each subject's set of photos contained one photo of the subject wearing glasses. A couple of subject's photo sets had the subject wearing glasses in all but a few photos. We removed from each subject's set of photos the least common eyewear choice. That is, if a subject was not wearing glasses in a majority of the photos, we removed the glasses photos, if the subject was mostly wearing glasses, we removed the photos where they were not wearing glasses. Recognizing users both with and without glasses makes facial recognition slightly more difficult and in all LGTM use cases both users will need to be wearing augmented reality headsets, so keeping eye wear uniform makes sense for our use case.

For evaluating facial recognition accuracy we train on all images but one, the "happy" photo in each subject's set of photos. We then test recognition on this one left out photo. Using this methodology we were able to accurately recognize every single subject in the Yalefaces dataset.

Facial recognition is not a performance bottleneck at all. We train a new classifier using the training photos on each run which takes less than one second [$GETSOMEREALTIMESFORTHIS....the$ Recognition itself is virtually instant, there is no noticeable lag time.

We can safely say, just from our cursory validation experiments on LBPH facial recognition for use-cases fitting LGTM that facial recognition is an efficient and accurate part of the whole LGTM system.

4.3 Full System Timing (BETTER TITLE)

Overall, our implementation of LGTM takes $< INSERTTIMETORUN >$ seconds to run. Consider on top of this that there are various unindeal procedures that I used to workaround various issues and this timing is very good for a device pairing procedure.

Chapter 5

Conclusion

5.1 Future Work on LGTM

There remains room to further improve LGTM by using more of the unique hardware and abilities that come with augmented reality. Simultaneous location and mapping (SLAM) is the field which explores mapping out a user's surroundings in terms of depth to objects and user location in the environment [$CITATIONS$]. No doubt many techniques from this field can be used to bolster LGTM's usability and make selecting users to share with even easier.

Of course improved wireless localization accuracy will be a large contributor to improving LGTM. We fully expect the field to continue advancing the accuracy of wireless localization as they have over its entire existence. Indeed, since work on LGTM initially began wireless localization has taken a huge step forward thanks to the works of Vasisht et al. in [40], which describes a technique for performing decimeter level localization from a single access point with three antennas.

Finally, there remains to be a great deal of work done on examining augmented reality attacks in general, particularly ones involving spamming a users vision or tricking users into accepting something inadvertently.

5.2 Conclusion

In this work we presented LGTM, an authentication protocol for pairing two augmented reality headsets. Pairing is known to be a very difficult problem since it involves assigning trust to wireless signals and users and not assumed to have any prior security context. The protocol uses the same intuition behind how humans identify sound waves and adapts this technique to instead work with electromagnetic waves.

Bibliography

[1] W. H. D. F. moderator. (2016, April) Hologram sharing. [Online]. Available: http://forums.hololens.com/discussion/comment/1135#Comment_1135

[2] *Wi-Fi Peer-to-Peer Services (P2Ps) Technical Specification*, Wi-Fi Alliance Std., 2014.

[3] A. Kumar, N. Saxena, G. Tsudik, and E. Uzun, "A comparative study of secure device pairing methods," in *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, March 2009, pp. 1–10.

[4] N. Saxena, J. E. Ekberg, K. Kostiainen, and N. Asokan, "Secure device pairing based on a visual channel," in *2006 IEEE Symposium on Security and Privacy (S P'06)*, May 2006, pp. 6 pp.–313.

[5] S. Gollakota, N. Ahmed, N. Zeldovich, and D. Katabi, "Secure in-band wireless pairing," in *Proceedings of the 20th USENIX Conference on Security*, ser. SEC'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 16–16. [Online]. Available: http://dl.acm.org/citation.cfm?id=2028067.2028083

[6] F. Stajano and R. J. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Proceedings of the 7th International Workshop on Security*

Protocols. London, UK, UK: Springer-Verlag, 2000, pp. 172–194. [Online]. Available: http://dl.acm.org/citation.cfm?id=647217.760118

[7] A. Gallego, N. Saxena, and J. Voris, "Playful security: A computer game for secure wireless device pairing," in *Computer Games (CGAMES), 2011 16th International Conference on*, July 2011, pp. 177–184.

[8] S. Vaudenay, "Secure communications over insecure channels based on short authenticated strings," in *Proceedings of the 25th Annual International Conference on Advances in Cryptology*, ser. CRYPTO'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 309–326. [Online]. Available: http://dx.doi.org/10.1007/11535218_19

[9] C. Gehrmann and K. Nyberg, "Manual authentication for wireless devices," *RSA Cryptobytes*, vol. 7, p. 2004, 2004.

[10] R. Mayrhofer and H. Gellersen, "Shake well before use: Intuitive and secure pairing of mobile devices," *IEEE Transactions on Mobile Computing*, vol. 8, no. 6, pp. 792–806, June 2009.

[11] J. M. McCune, A. Perrig, and M. K. Reiter, "Seeing-is-believing: using camera phones for human-verifiable authentication," in *2005 IEEE Symposium on Security and Privacy (S P'05)*, May 2005, pp. 110–124.

[12] D. B. Smetters, D. Balfanz, D. K. Smetters, P. Stewart, and H. C. Wong, "Talking to strangers: Authentication in ad-hoc wireless networks," 2002.

[13] M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun, "Loud and clear: Human-verifiable authentication based on audio," in *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, 2006, pp. 10–10.

[14] C. Soriente, G. Tsudik, and E. Uzun, *Information Security: 11th International Conference, ISC 2008, Taipei, Taiwan, September 15-18, 2008. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, ch. HAPADEP: Human-Assisted Pure Audio Device Pairing, pp. 385–400. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85886-7_27

[15] P. Wellner, W. Mackay, and R. Gold, "Back to the real world," *Commun. ACM*, vol. 36, no. 7, pp. 24–26, Jul. 1993. [Online]. Available: http://doi.acm.org/10.1145/159544.159555

[16] Magic leap website. Magic Leap. [Online]. Available: https://www.magicleap.com/

[17] (2016) Augmented reality meta website. Meta. [Online]. Available: https://www.metavision.com/

[18] (2016) Microsoft hololens website. Microsoft. [Online]. Available: https://www.microsoft.com/microsoft-hololens/en-us

[19] Google glass website. Google. [Online]. Available: https://www.google.com/glass/start/

[20] (2016, April) Layar homepage. Layar. [Online]. Available: https://www.layar.com/

[21] (2015, October) Google translate. Play Store. Google. [Online]. Available: https://play.google.com/store/apps/details?id=com.google.android.apps.translate&hl=en&gl=us

[22] (2016, April) ionroad homepage. iOnRoad. [Online]. Available: http://www.ionroad.com/

[23] The cube. Virginia Tech. [Online]. Available: https://www.icat.vt.edu/content/cube-0

[24] J. Temperton. (2015, December) This is the next generation of google glass. Wired. [Online]. Available: http://www.wired.co.uk/news/archive/2015-12/29/google-glass-enterprise-edition-confirmed

[25] C. Soriente, G. Tsudik, and E. Uzun, "Beda: Button-enabled device association," 2007.

[26] *BLUETOOTH SPECIFICATION Version 4.0*, Bluetooth Std., 2010.

[27] U. E. Group, "Bluetooth user interface flow diagrams for bluetooth secure simple pairing devices," Bluetooth, Tech. Rep. v1.0, September 2007.

[28] A. Kumar, N. Saxena, G. Tsudik, and E. Uzun, "A comparative study of secure device pairing methods," *Pervasive Mob. Comput.*, vol. 5, no. 6, pp. 734–749, Dec. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.pmcj.2009.07.008

[29] R. C.-W. Phan and P. Mingard, "Analyzing the secure simple pairing in bluetooth v4.0," *Wireless Personal Communications*, vol. 64, no. 4, pp. 719–737, 2010. [Online]. Available: http://dx.doi.org/10.1007/s11277-010-0215-1

[30] R. Chang and V. Shmatikov, "Formal analysis of authentication in bluetooth device pairing."

[31] K. Haataja and P. Toivanen, "Practical man-in-the-middle attacks against bluetooth secure simple pairing," in *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, Oct 2008, pp. 1–5.

[32] R. Kainda, I. Flechais, and A. W. Roscoe, "Usability and security of out-of-band channels in secure device pairing protocols," in *Proceedings of the 5th Symposium on Usable Privacy and Security*, ser. SOUPS '09. New York, NY, USA: ACM, 2009, pp. 11:1–11:12. [Online]. Available: http://doi.acm.org/10.1145/1572532.1572547

[33] I. Ion, M. Langheinrich, P. Kumaraguru, and S. Čapkun, "Influence of user perception, security needs, and social factors on device pairing method choices," in *Proceedings of the Sixth Symposium on Usable Privacy and Security*, ser. SOUPS '10. New York, NY, USA: ACM, 2010, pp. 6:1–6:13. [Online]. Available: http://doi.acm.org/10.1145/1837110.1837118

[34] A. Kumar, N. Saxena, and E. Uzun, "Alice meets bob: A comparative usability study of wireless device pairing methods for a "two-user" setting," *CoRR*, vol. abs/0907.4743, 2009. [Online]. Available: http://arxiv.org/abs/0907.4743

[35] E. Uzun, K. Karvonen, and N. Asokan, *Financial Cryptography and Data Security: 11th International Conference, FC 2007, and 1st International Workshop on Usable Security, USEC 2007, Scarborough, Trinidad and Tobago, February 12-16, 2007. Revised Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, ch. Usability Analysis of Secure Pairing Methods, pp. 307–324. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-77366-5_29

[36] N. Wadhwa. (2016, April) Visual pairing question. On a Microsoft GitHub repository issues page as a question. Microsoft. [Online]. Available: https://github.com/Microsoft/HoloToolkit/issues/4

[37] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theor.*, vol. 22, no. 6, pp. 644–654, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/TIT.1976.1055638

[38] D. K. D. R. Swarun Kumar, Stephanie Gil, "Accurate indoor localization with zero start-up cost," in *Proceedings of the 20th annual international conference on Mobile computing and networking (Mobicom)*, 2014.

[39] D. B. Manikanta Kotaru, Kiran Joshi and S. Katti, "Spotfi: Decimeter level localization using wifi," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 269–282, Aug. 2015. [Online]. Available: http://doi.acm.org/10.1145/2829988.2787487

[40] D. Vasisht, S. Kumar, and D. Katabi, "Decimeter-level localization with a single wifi access point," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*. Santa Clara, CA: USENIX Association, Mar. 2016, pp. 165–178. [Online]. Available: https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/vasisht

[41] S. F. L. I. Turati C, Macchi Cassia V, "Newborns' face recognition: role of inner and outer facial features," *Child Development*, vol. 77, pp. 297–311, March 2006.

[42] N. Saxena and M. B. Uddin, "Secure pairing of "interface-constrained" devices resistant against rushing user behavior," in *Proceedings of the 7th International Conference on Applied Cryptography and Network Security*, ser. ACNS '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 34–52. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-01957-9_3

[43] I. E. B. M. Allman, V. Paxson, *TCP Congestion Control*, Network Working Group Std. [Online]. Available: https://tools.ietf.org/html/rfc5681

[44] *ISO 9241-11: Guidance on Usability*, International Standards Organization (ISO) Std., 1998.

[45] R. K. LEE RAINIE, SARA KIESLER and M. MADDEN. (2015, September) Anonymity, privacy, and security online. Web article. Pew Re-

search Center. [Online]. Available: http://www.pewinternet.org/2013/09/05/anonymity-privacy-and-security-online/

[46] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Tool release: Gathering 802.11n traces with channel state information," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, pp. 53–53, Jan. 2011. [Online]. Available: http://doi.acm.org/10.1145/1925861.1925870

[47] P. N. Belhumeur, J. a. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 711–720, Jul. 1997. [Online]. Available: http://dx.doi.org/10.1109/34.598228

[48] J. Gjengset, J. Xiong, G. McPhillips, and K. Jamieson, "Phaser: Enabling phased array signal processing on commodity wifi access points," in *Proceedings of the 20th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '14. New York, NY, USA: ACM, 2014, pp. 153–164. [Online]. Available: http://doi.acm.org/10.1145/2639108.2639139

[49] D. J. Bernstein, *Public Key Cryptography - PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, April 24-26, 2006. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, ch. Curve25519: New Diffie-Hellman Speed Records, pp. 207–228. [Online]. Available: http://dx.doi.org/10.1007/11745853_14

[50] D. B. C. K. Y. M. R. H. V. S. T. P. N. S. Turner, IECA, *Elliptic Curve Cryptography Subject Public Key Information*, Network Working Group Std., March 2009. [Online]. Available: https://www.ietf.org/rfc/rfc5480.txt

[51] *Announcing the ADVANCED ENCRYPTION STANDARD (AES)*, National Institute of Standards and Technology Std., November 2001. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[52] D. A. McGrew and J. Viega, "The galois/counter mode of operation (gcm)," National Institute of Standards and Technology, Tech. Rep., 2005.

[53] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, 2001, pp. I–511–I–518 vol.1.

[54] T. Ahonen, A. Hadid, and M. Pietikäinen, *Computer Vision - ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, May 11-14, 2004. Proceedings, Part I.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, ch. Face Recognition with Local Binary Patterns, pp. 469–481. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24670-1_36

[55] S. Liao, X. Zhu, Z. Lei, L. Zhang, and S. Z. Li, "Learning multi-scale block local binary patterns for face recognition," in *Proceedings of the 2007 International Conference on Advances in Biometrics*, ser. ICB'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 828–837. [Online]. Available: http://dl.acm.org/citation.cfm?id=2391659.2391754

[56] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, Jan. 1991. [Online]. Available: http://dx.doi.org/10.1162/jocn.1991.3.1.71

[57] G. Bradski, "Opencv," *Dr. Dobb's Journal of Software Tools*, 2000.

[58] R. Schmidt, "Multiple emitter location and signal parameter estimation," *Antennas and Propagation, IEEE Transactions on*, vol. 34, no. 3, pp. 276–280, Mar 1986.

[59] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998. [Online]. Available: http://dx.doi.org/10.1023/A:1009715923555