Ensemble: Cooperative Proximity-based Authentication

Andre Kalamandeen
Computer Science
University of Toronto
andrek@cs.toronto.edu

Adin Scannell
Computer Science
University of Toronto
amscanne@cs.toronto.edu

Eyal de Lara
Computer Science
University of Toronto
delara@cs.toronto.edu

Anmol Sheth
Intel Research Seattle
anmol.sheth@intel.com

Anthony LaMarca
Intel Research Seattle
anthony.lamarca@intel.com

ABSTRACT

Ensemble is a system that uses a collection of trusted personal devices to provide proximity-based authentication in pervasive environments. Users are able to securely pair their personal devices with previously unknown devices by simply placing them close to each other (e.g., users can pair their phones by just bringing them into proximity). Ensemble leverages a user's growing collection of trusted devices, such as phones, music players, computers and personal sensors to observe transmissions made by pairing devices. These devices analyze variations in received signal strength (RSS) in order to determine whether the pairing devices are in physical proximity to each other. We show that, while individual trusted devices can not properly distinguish proximity in all cases, a collection of trusted devices can do so reliably. Our Ensemble prototype extends Diffie-Hellman key exchange with proximity-based authentication. Our experiments show that an Ensemble-enabled collection of Nokia N800 Internet Tablets can detect devices in close proximity and can reliably detect attackers as close as two meters away.

Categories and Subject Descriptors

C.5.3 [**Computer System Implementation**]: Microcomputers—*Portable devices*; D.4.7 [**Operating Systems**]: Organization and Design—*Distributed systems*

General Terms

Experimentation, Human Factors, Performance, Security

Keywords

Ensemble, Proximity, Authentication

1. INTRODUCTION

Computer and consumer electronics developers have long debated the relative merit of carrying a single, all-purpose mobile device versus a collection of special purpose devices. Despite the recent success of some excellent general purpose platforms, the market is clearly trending towards the latter, with an enormous proliferation of research and commercial devices spanning a wide range of functions including netbooks, smart phones, digital cameras, music players, implantable medical devices, fitness activity sensors, wearable (e.g. wrist worn) remote controls, wireless headsets, etc. With the miniaturization and improved power-efficiency of radio, these personal devices are increasingly networked, allowing them to provide more integrated services across applications.

The prospect of networked ensembles of personal devices creates an opportunity to use the collection of devices to enable a capability or service that no single device in the collection could provide alone. In this paper, we show that an ensemble of devices can be used to determine whether communicating devices are within physical proximity of each other, and that this information can be used to provide proximity-based authentication when pairing devices that lack a pre-existing shared secret to secure their communication – devices that are previously unknown to each other.

Proximity-based authentication offers an attractive way to bootstrap secure communication channels in pervasive environments. For example: users may place their laptop immediately next to the data projector in order to authenticate before presenting slides; or a portable video player may, when placed directly in front of a large display, show its output on the bigger screen with the touch of a button. Proximity-based authentication reduces the uncertainty inherent to pairing with a formerly-unknown device by offering assurances that communication is taking place between devices that are actually in close proximity, as opposed to with an attacker with a directional antenna some distance away.

We present Ensemble, a system that leverages a user's worn or carried trusted devices to enable proximity-based secure pairing in pervasive environments. Ensemble determines whether two devices are in close physical proximity by monitoring their communications and determining whether the variations in their received signal strength (RSS) are correlated. Ensemble leverages two key observations: first, given an environment of sufficient complexity, the nature of multipath fading implies that the channel between the user's trusted collection and the pairing devices will vary haphazardly over time and be very difficult to predict. Second, RSS variations of transmitters that are in close proximity (e.g., 10 cm apart) are highly correlated – paths between the transmitters and a monitoring device experience sim-

MobiSys'10, June 15–18, 2010, San Francisco, California, USA.

ilar changes – whereas RSS variations of devices that are separated by some distance are not. We show that, while individual trusted devices can not properly distinguish proximity in all cases, a collection of trusted devices can do so reliably.

Ensemble has several advantages over existing approaches to secure pairing. Foremost, it does not require extra hardware beyond the wireless interface already present on the mobile device. Moreover, because Ensemble only needs to run on the user's collection of trusted devices, but not necessarily on the other device that is being paired, it can be easily deployed by individual users without requiring widespread adoption to be useful. For example, a user could authenticate a device from another user that is not running Ensemble.

We developed an Ensemble prototype based on a collection of Nokia N800 devices. Our prototype, which extends Diffie-Hellman key exchange with proximity-based authentication, can reliably detect attackers as close as two meters away. Our experimental evaluation shows that the system works well in different environments without the need for recalibration and is also successful when pairing heterogeneous devices.

In this paper, we first formalize and motivate the problems of secure pairing and determining whether two devices are in close physical proximity. Next, we describe the Ensemble system. Following this, we evaluate Ensemble under different scenarios and conditions. We then discuss possible use cases and extensions to the system. Finally, we compare Ensemble to related work in the area and present our conclusions.

2. PROBLEM DEFINITION

We define the problem of secure pairing of devices in close proximity as follows: two devices that are located close to each other but do not know each other a priori need to establish a channel between them that is both secure and authentic. A secure channel implies that no eavesdropper may intercept and decrypt messages between the endpoints, while authenticity requires that both endpoints are able to confirm the identity of each other. The creation of a secure channel with another device is currently possible through well-known protocols such as Diffie-Hellman [5]. However, such secure channels are effectively anonymous and need to be authenticated given that an attacker can easily spoof a name or MAC address.

We define four types of devices in a typical pairing scenario. In the set of trusted devices, one device acts as a *pairing* device while the rest of the trusted ensemble act as *witness* devices; the other legitimate device with whom pairing is being attempted is the *candidate* device; *impostor* devices are rogue devices located some distance away trying to pair with a legitimate device. These devices are shown in a typical pairing scenario in Figure 1. Ensemble leverages information from trusted witness devices in order to infer whether a device is a candidate device or an impostor.

For simplicity, we limit the problem of secure pairing to the perspective of the *pairing* device. If necessary, each device (the candidate and the pairing device) may independently authenticate the other, but in many circumstances only one side requires authenticity. For example, a point-of-sale terminal may not care whose device is paying for a transaction (as long as they use a valid credit card number),

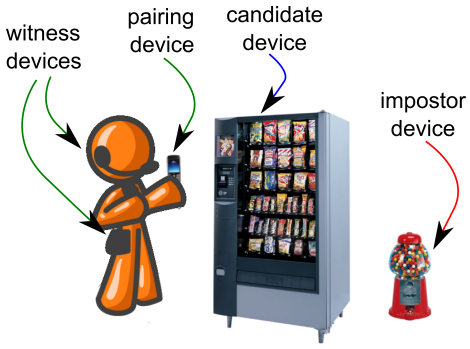


Figure 1: An example of a pairing scenario, where a user is authenticating with a vending machine.

but the credit card holder certainly cares about paying the correct terminal.

We assume that the devices can communicate over compatible wireless radios (e.g., WiFi) and that neither additional out-of-band communication channels are required (e.g., ultrasound) nor is additional hardware present on the devices (e.g., accelerometers). We assume that trusted devices (the pairing device and the witness devices) already share some pre-existing secret – an SSL certificate or a WPA key, for example. We also assume that the trusted devices have not been compromised a priori (e.g., malware).

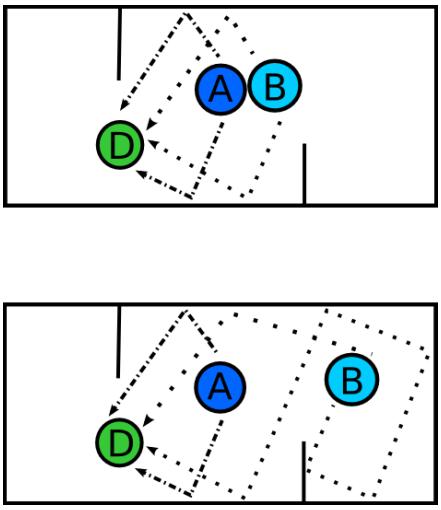
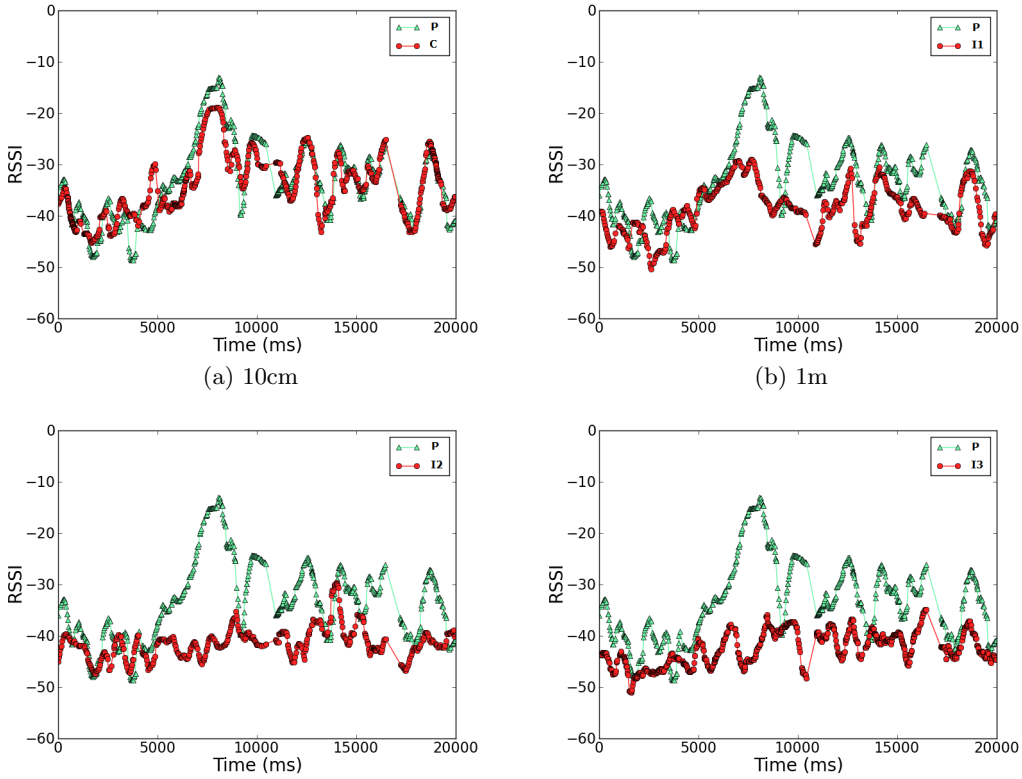


Figure 2: Different paths radio signals can take to a receiver. When A and B are close together, they share similar radio paths to D; conversely when they are separated, the paths are likely to be different.

2.1 Sensing Proximity

As a radio wave propagates from its source, it is affected by several factors such as multipath, fading, shadowing, and interference. These factors are both time and environment specific and are difficult to predict. Moreover, as movement is introduced into an environment, the unpredictability of the radio environment increases dramatically. However, when two devices are in close physical proximity, they often share a very similar radio path to other wireless devices. As



(a) 10cm (b) 1m

(c) 2m (d) 4m

Figure 3: RSS values for transmitters separated by 10cm, 1m, 2m and 4m as observed by a single device on the wrist. When the transmitters are close, the RSS values are highly correlated.

devices are separated by more space, this path quickly once again diverges and becomes difficult to predict. Figure 2 demonstrates this idea.

Ensemble is based on the assumption that witness devices will perceive similar disturbances and fluctuations in the radio path to the legitimate pairing devices, which are in close physical proximity. Conversely, if the pairing devices are not in close physical proximity, the fluctuations observed will not be similar. It is not required that trusted witness devices are nearby (only within wireless range), nor that any device is stationary.

We restrict ourselves to using only simple Received Signal Strength (RSS) information on a per-packet level to infer physical proximity. In the future, detailed physical layer information may be available from commodity wireless networking hardware. We imagine that such an enhancement could also be used to improve any system using channel information, including localization and proximity detection, simply because the information available is richer. However, for practical reasons, we limit ourselves in this paper to the information available on commodity WiFi networking devices, which is unlikely to change in the foreseeable future. This consists of a single signal strength estimate for each packet received, which must be taken during the preamble.

2.2 Attacker Model

We assume the presence of an attacker that will try to pair with the pairing device. The attacker can eavesdrop, spoof devices, transmit at varying power levels and may be equipped with directional antennas. The attacker may try to jam the channel of the pairing devices, a *jamming or DoS attack*; try to pair directly with the device, an *impostor attack*; mount a *man-in-the-middle attack* wherein two (legitimate) devices in close physical proximity are attempting to pair but in reality both have secure connections to the attacker; or retransmit the votes of the witnesses, a *replay attack*. We assume in all cases that the distance between the pairing devices is smaller than the distance from the legitimate device to the attacker. A user would easily spot an attacker lurking centimeters away.

3. ENSEMBLE

Ensemble has two major requirements. The first is to determine if the transmissions that the witnesses observed were emitted by proximal devices. Secondly, the system must be robust enough to ensure secure communication and be resilient to attacks. We start by describing how Ensemble determines proximity and then discuss a protocol that allows the witnesses and pairing devices to exchange this information securely.

3.1 Determining Proximity

Ensemble determines proximity in two stages. First, individual witnesses make a decision based on observed RSS variations. Next these decisions are combined to make a final vote on proximity.

3.1.1 Single Witness Algorithm

Given a set of received packets and their RSS values witness devices must determine whether the pairing devices are co-located. This is accomplished by examining the variance of RSS over a short period of time.

For example, Figure 3 shows 20 seconds of RSS values for five devices (smoothed over 200ms) - the pairing device (P), the candidate device (C) and three impostor devices (I1-I3) - as perceived by a single witness device located on a user's wrist. The pairing device P serves as a reference from which devices C, I1, I2, and I3 are separated by distances of 10cm, 1m, 2m and 4m respectively; each diagram shows the RSS values of P and another device. In Figure 3(a), we see that both RSS values are tightly correlated, with increases in P's RSS being followed by increases in C's RSS and vice versa. As the separation between the devices increases, the correlation falls off. As discussed in Section 2, this tight correlation holds for pairing devices in proximity because the path that the signal will take to the receiver will be similar and subjected to the same environmental factors.

To determine if devices are co-located, we analyze the temporal variation in RSS values of each pairing device as observed by a witness device. This process has three distinct steps: First, we transform the set of RSS values into workable *signalprints*. Second, the signalprints are then broken into a number of distinct *segments* and a correlation value is computed between pairs of segments. Finally, we apply a sliding window over a collection of segments to reach a decision about proximity.

Signalprints

The broadcast nature of WiFi restricts two transmitters in range of each other from utilizing the medium simultaneously. This means that transmitters will only send packets when the channel is free which results in a constant swapping between the transmitters and other devices that also share the channel. As such, the packets collected by the witnesses will be interspersed with each other. To be meaningful, we need to analyze the RSS values for the pairing devices in close succession. First, we define a time line that is common to both transmitters, any packets before and after this defined time line are ignored as shown in Figure 4. We refer to this time restricted set of packets as our *signalprint*. To reduce the effect of random noise, we apply a smoothing function that averages RSS values over a short period of time, in our case, 200 milliseconds. The smoothing filters out random noise and highlights recent fading. Next we apply interpolation to each signalprint to generate pairs of RSS values - one value for each signalprint - that correspond to the same instant. This assists in highlighting the short term RSS variation. Figure 5 shows this technique being applied to two signalprints. The blue circle represents RSS values from the pairing device, while the red square represents RSS values from the candidate device (as observed from a witness). The end result is two *interpolated signalprints* that contain RSS values for the same instances in time.

Segmentation and Correlation

Next, we break up the signalprints into overlapping segments representing a fixed amount of time. The overlapping feature of segments acts as a weight for the RSS variations. For example, in the event that the variations are highly uncorrelated, this feature would act as a punishment and subsequently make it more difficult for the connection to be authenticated. The segment must contain enough packets to highlight recent RSS events, but must be short enough so as not to prolong the authentication process. In our experience, segments of 30 seconds taken every 10 seconds work best. In this approach, the first segment accounts for the first 30 seconds of the signalprint, the second segment for seconds 10 through 40, and so on and so forth.

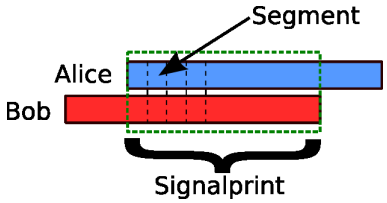
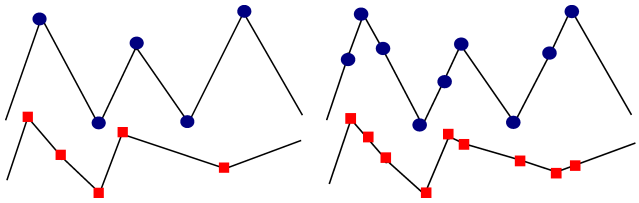


Figure 4: Selecting packets with a common time range. The resulting signalprint is broken down into segments for analysis.



(a) Before interpolation (b) After interpolation

Figure 5: Applying interpolation to signalprints to align RSS values.

The Pearson Coefficient

We use the Pearson correlation coefficient to determine if variations in RSS values in two segments (one from each of the two signalprints) are correlated. This metric returns a value that indicates the linear relationship between the two data sets. The range of values are from -1 to 1: a value of -1 means that the data sets vary inversely, while 1 indicates that the sets are perfectly correlated. While we do not expect to have a perfect correlation coefficient $r = 1$, we do expect to have highly correlated changes for devices in close proximity. Naturally, with a high r value we would be able to conclude that pairing devices are in proximity, while a low r value would lead to the conclusion that the pairing devices are not. In Section 4 we analyze the effect of choosing a threshold for r.

We apply the Pearson correlation coefficient as follows:

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

Where :
X_i is the RSS of the ith packet;
$\bar{X}$ is the mean of the RSS values in the segment;
Y_i is the RSS of the ith packet;
$\bar{Y}$ is the mean of the RSS values in the segment;
and n is the number of packets in the segment.

The Pearson Coefficient has the interesting property of measuring the variations in RSS and *not* the absolute RSS values themselves. That is, if we consider two exact signalprints (in terms of variation) but one is offset by some fixed number, then the correlation coefficient between these two signalprints would be $r = 1$ since the variations match exactly. Thus, this metric works well even in the presence of heterogeneous devices that transmit at different signal strengths.

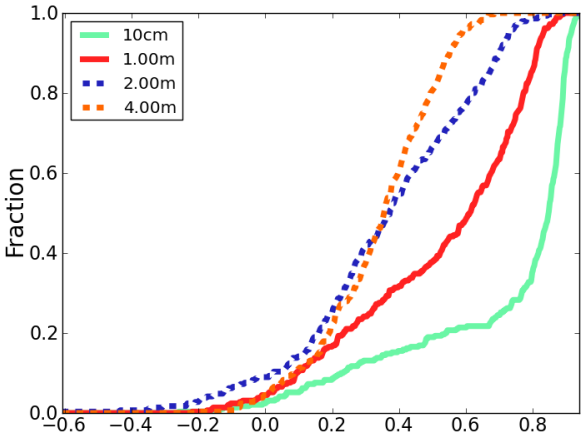


Figure 6: Cumulative distribution function of the Pearson correlation coefficient for the RSS values of a single witness device and several transmitters separated by different distances.

Figure 6 shows a cumulative distribution function (CDF) of the Pearson correlation coefficient on a sample experiment done in our lab. The details of the setup are similar to those of Figure 3. This CDF shows that in this experiment, there is a 80% chance that devices located within 10cm have an r value higher than 0.6. Conversely for devices located 4m apart, there is only a 4% chance that the r value will be above 0.6. For 1m, this probability is 52%, while for 2m, it drops to 23%. This figure clearly shows that Pearson coefficient is a strong indicator of the physical proximity of two transmitters. Judging from the figure, a sensible threshold for the r value would be between 0.6 and 0.8. For our purpose we use $r = 0.7$, although we explore this choice further in Section 4.

Windowing

Each witness device uses a sliding window on the signalprint and computes an r value for each segment in the window. The witness then determines how many of the computed values are at least *0.7*, and if there is a majority, concludes that the pairing devices are in proximity. Otherwise, the witness concludes that the pairing devices are not in proximity. Figure 7 shows a simple example with a window size of 3 segments. In our experience a window size of 5 segments works well in practice. We explore this choice further in Section 4.

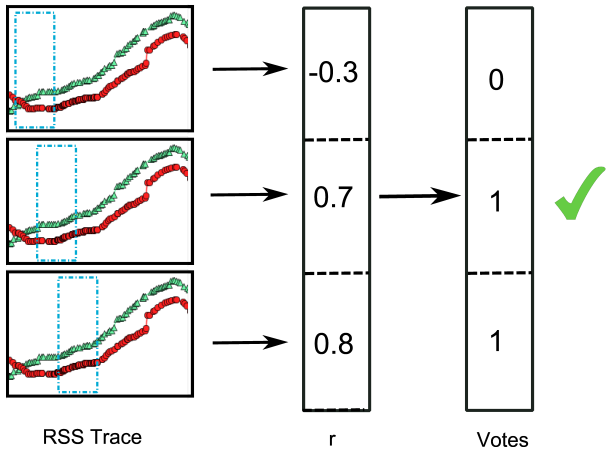


Figure 7: The process of determining proximity for each witness device. Correlation coefficients are computed for the RSS values using sliding windows. A decision is then reached based on a majority vote.

Detecting Static Environments

The Pearson coefficient represents the co-variation of two sequences. If RSS values perceived by a witness device are excessively static or constant, the Pearson coefficient may not be an accurate indicator of when the pairing devices are in physical proximity. We define a static environment as one where the channel between two devices are constant and is not subjected to multipath and fading. From a witness's perspective, the RSS values will only vary within a few dB. In this case, the witness device does not conclude anything about a particular signalprint (does not cast a vote), and the pairing device must rely on other trusted witnesses. In our experiments, if the variance is less than 5dB, we assume that the environment is too static to allow for pairing.

3.1.2 Multi-Witness Algorithm

In the final step, the pairing device collects decisions from its trusted witnesses (or alternatively one of the witness devices can do this on its behalf). In a simple voting scheme executed by the pairing device, if at least 50% of these devices have concluded that the pairing is legitimate, then the connection is authenticated. Otherwise, it is rejected.

There may be cases where Ensemble cannot securely determine proximity – when fewer than half of the witnesses cast a vote due to lack of RSS variability. In such situations, Ensemble prompts the user to reattempt pairing in a different location or to introduce some variation into the environment. Fortunately, such cases do not occur often in typical dynamic environments, such as those containing people. However, if an environment happens to be particularly static, it is simple for a user to introduce variation into the radio channel by moving either the pairing device and the candidate device or the witnesses (in our evaluation, it is the witnesses that are moved to vary the channel). The movement creates entropy in the environment and allows the witness to determine if the variations are correlated.

3.2 Ensemble Protocol

Suppose a user, Alice, needs to pair her phone with Bob's phone. She has a trusted ensemble of devices that includes,

for example, her portable music player and her netbook in her backpack. Similarly, Alice could also use as witnesses any other trusted devices that are in wireless range, not just her body-worn devices, such as her colleague's phone in the office next door, the access point in her office or her desktop computer.

The Ensemble pairing process consists of the following steps:

1. Alice sets up her trusted ensemble (the witnesses and the pairing device) with shared key *KEY*. The key is used to encrypt messages amongst her devices. She only needs to do this once when she is configuring her ensemble. We assume that ensemble setup is done in a reasonable secure environment (e.g., the user's home) and is not subject to attack.

2. Alice and Bob initiate pairing and their phones establish a secure channel - though it may not be authentic. This can be accomplished using a known cryptographic method (for example, DSA or RSA) to generate a shared key that will be used to encrypt their communication. In our prototype, we use the Diffie-Hellman key exchange (though any other cryptographic algorithm that is available to the pairing devices can be used). The result is the session key *SKEY*.

3. Alice's phone (the pairing device) informs the witnesses about SKEY. It does this by encrypting messages with *KEY*. This message includes a nonce that prevents replay attacks. It also informs them about the set of packets that it will be sending to Bob's phone (the candidate) by relaying the set of nonces it plans to use.

4. The pairing device and candidate send multiple messages to each other. That is, the pairing device sends messages to candidate encrypted with SKEY; the candidate sends messages to the pairing device encrypted with SKEY. The above messages are overheard by the witnesses.

5. Witnesses inform the pairing device about their vote.

6. The pairing device tallies the votes and determines whether the connection with the candidate device is authentic.

Figure 8 illustrates the message exchanges in steps 3 to 5 of the Ensemble protocol. Steps 1 and 2 rely on standard cryptographic techniques, such as Diffie-Hellman or WPA, and are therefore not shown.

The pairing device (P) sends a packet to the witness (W) encrypted with KEY, containing the session key *SKEY*. The packet also includes a collection of nonces which are used to verify packets sent by P and to detect replay attacks. Witness W decodes the packet and monitors the channel for packets encrypted with the session key (SKEY). The witness does not send any acknowledgments (ACKs). This is done so that it remains unknown to attackers. The only time the witness transmits data is at the end of the protocol when the votes are sent.

Device P and candidate C exchange packets encrypted with SKEY. Each of the packets sent by P also includes a nonce from the collection that was previously communicated

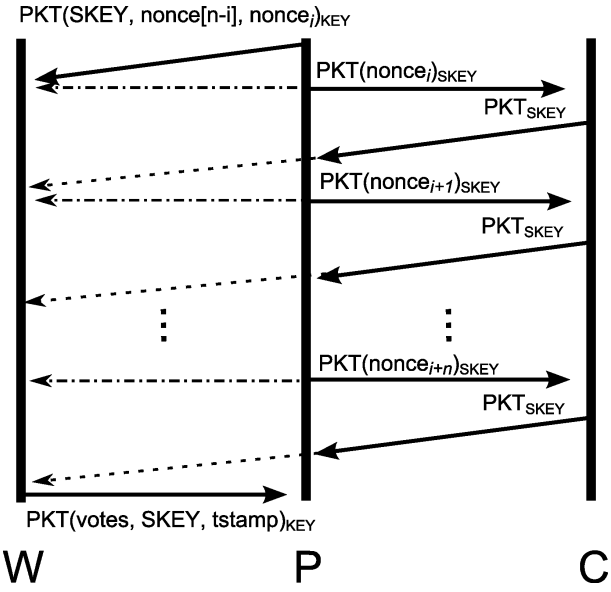


Figure 8: Steps taken in the Ensemble protocol. P sends packets to W encrypted with KEY and to C encrypted with SKEY. The solid lines represent messages being sent to a device while the broken lines represent messages overheard by W.

to W. In turn W records the RSS of an overheard packet only if it is encrypted with SKEY, and for a packet originating from P if it also contains a non-duplicated nonce from the sequence that P sent earlier.

At the end of the process, W sends its vote to P encrypted with KEY, along with the session key (SKEY), and a time stamp. This helps to prevent against replay attacks.

3.2.1 Attacks

We consider several types of attacks during the different stages of the Ensemble protocol including: jamming attacks (or DoS attacks); impostor attacks; man-in-the-middle attacks; and replay attacks.

Session Key Setup Phase

In the initial setup phase when P and C are negotiating on a session key an attacker can either mount a man-in-the-middle attack or impersonate C. In a man-in-the-middle attack, a malicious device attempts to pair simultaneously with two legitimate devices that are in close physical proximity, and leverages this fact to successfully authenticate. In this scenario, the attacker starts two independent pairing sessions for P and C. The resulting session key will be compromised and the pairing device and candidate device will each have an unauthenticated channel with the attacker.

In an impostor attack, the attacker successfully disables one of the pairing devices and tries to pair with the other legitimate device. In this case the attacker can impersonate C by jamming its messages and spoofing its identity. In both attacks described above, the attacker still has to circumvent the proximity detection stage in order to compromise the pairing process.

Transmission of SKEY and nonce to W

The message that P sends to W that conveys the session key and the nonce sequence cannot be forged as it requires knowledge of KEY. The nonce sequence in the message also prevents a replay attack, in which the attacker saves these messages and rebroadcasts them at a later time in order to achieve a successful authentication.

Messages transferred between P and C

At this stage the attackers have already compromised the session key, have two channels set up with P and C, or are impersonating C. This is the most complex part of the protocol to spoof as it requires the attacker to prove their proximity before the channel is authenticated.

The impostor may try to vary her transmission power to compensate for environmental changes and mimic the RSS variation that she observes from P. However, the RSS variation that the attacker observes is the variation of the channel between herself and P. It is not the variation that each witness observes from P. In its present configuration, this attack will not be successful.

To be successful, the impostor must determine the variation of all of the channels between the witnesses and P, account for the natural fading that occurs in the environment, and adjust her transmission power to be in sync with the RSS variations that each witness perceives. The attacker will need several directional antennas each pointed to a witness to simulate the RSS variation of that channel and P. This is further complicated since witnesses only reveal themselves at the end of the protocol when they cast their votes.

For the man-in-the-middle attack to be successful the attacker needs to intercept the entire packets that P and C transmit, jam them before they reach the witnesses, and rebroadcast them to the witnesses by either using two radios that are physically close, or one radio that is capable of simulating the two pairing devices. While theoretically possible, jamming all the witnesses is in practice an extremely difficult and complex task that requires multiple directional antennas, specifically since the attacker may not know where these devices are located. The environment may complicate this task even further if it is crowded or if the user is moving.

Finally, the nonces in the messages sent by P prevent a replay attack.

Message from W to P with Votes

At this stage a replay attack is possible on the message containing the votes. Consider the case where an attacker has successfully disabled C and is now impersonating it. At the end of the protocol, the attacker jams the witnesses' votes to P and retransmits the votes of an earlier session where two devices were authenticated. To negate this attack the message with the votes contains the session key used by the pairing devices. This ties the votes to a particular session and makes it difficult to use in a replay attack. If the attacker manages to spoof the current session key from one that was reused, the time stamp field comes into effect. Device P can discard votes that take longer than a fixed threshold to arrive.

3.2.2 DoS Attack

If the attacker is able to jam messages to W and messages between P and C, Ensemble will be unable to detect that an attack is taking place. In response to this, Ensemble can employ some of the strategies suggested by Xu [23] to detect when these attacks are taking place. One strategy includes having the pairing device keep a ratio of messages that are actually sent to messages that need to be sent. If this ratio goes above a set threshold, then the pairing device can infer that it cannot use the channel and that a DoS attack is taking place.

4. EVALUATION

In this section, we first present our experimental setup, followed by the results of our evaluation.

4.1 Experimental Setup

We collected data for our experiments using ten Nokia N800 Internet Tablets. The tablets were running a modified 2.6.29-omap Linux kernel with the open source stlc45xx WLAN driver. We configured five of these devices as witnesses, one as the pairing device, one as the candidate device and three as impostors (see Figure 9). The witness devices were placed on the body of the experimenter, W1 on the upper arm, W2 on his wrist, W3 in his back pocket, W4 in his front pocket, and W5 by his ankle. Witnesses W1 - W5 are used as representations of where we expect mobile devices to be found. For example, a WiFi/GPS watch, an iPod on the shoulder, a phone (in either right or back pockets) and Nike+ shoes. We note that in our evaluation all of the witness devices are homogeneous, and in reality a user will have different witnesses. However, since each witness is responsible for providing its own vote based on its observation of RSS values, we believe that differences amongst witnesses will not be an issue. The two pairing devices (P and C) were positioned within 10 cm of each other. Finally, the three impostor devices (I1, I2 and I3) were positioned 1, 2 and 4 meters away from P.

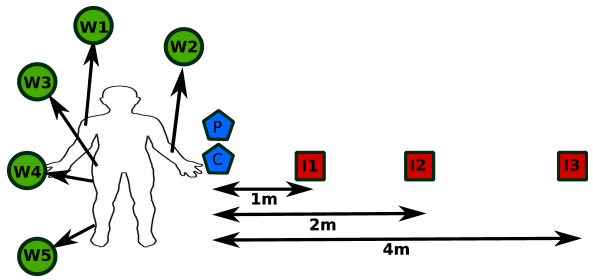


Figure 9: Sample setup used for experiments. Devices W1 - W5 are the trusted devices that act as witnesses, P is the pairing device, C is the candidate device and I1 - I3 are impostors. The experimenter is about 20cm from the pairing device P.

We configured the pairing devices (P and C) and the impostor devices (I1-I3) to broadcast at a rate of 40 packets per second. Simultaneously, the witness devices (W1-W5) were put in promiscuous mode so that they could monitor the transmissions of the other devices.

All of the transmitting devices were in the same orientation (facing the experimenter) and had a clear line of sight to him. For Ensemble, this actually represents the worst scenario as the lack of obstacles limits the degree of multi path and reduces channel variability.

We took samples in three different configurations: *static*, in which the experimenter was standing still facing the paring devices and trying to remain motionless; *back*, in which the motionless experimenter faced away from the transmitting devices; and *moving*, in which the pairing and impostor devices remain fixed, but the experimenter was moving.

We collected measurements in four different locations: our Lab, in the Lounge and Atrium of our University building, and in a basement of a house. The Lab consisted of a conference room with a large conference table and several chairs. The devices used for transmitting were placed on the conference table. The lab represents an area with high network traffic, with an average of 23 access points (APs) being discovered. The basement represents a location with minimal interference from WiFi APs and on average has 2 APs. It is a small space, with a lot of furniture and many other small obstacles. The lounge is a large open room with several chairs and desks, the devices were placed on the chairs. The Atrium is an open area on the ground floor of our University building and generally has people walking nearby. This is the only experiment where other persons were present during sampling, for all other experiments, only the experimenter was present.

In each of the above four locations and for each of the three configurations, we collected samples for ten minutes. We then divided each of the ten minute traces into 50 runs of continuous RSS measurements. In our evaluation, we treat each run as an independent opportunity to determine proximity. Unless otherwise indicated, the results we present use runs of 70 seconds in length; this corresponds to a 30 second segment size, and a decision window of size five.

4.2 Results

We answer the following questions:

1. How does Ensemble perform?
2. What is the effect of the ensemble size on system performance?
3. How does Ensemble perform when the two pairing devices are different?
4. What is the effect of varying the Pearson's threshold on system performance?
5. How sensitive is performance to the *segment* size and the *decision window* size?
6. Does access to a directional antenna give any benefits to an attacker?

4.2.1 Performance

We characterize the performance of Ensemble in terms of its acceptance rate, the fraction of time that the system identifies devices as being in proximity. Ideally, the acceptance rate for pairing P with C, which we will refer to as $acceptance_p$, would be 100%, while the acceptance rate for pairing P with any of the impostor devices, which we will refer to as $acceptance_i$, would be 0%.

Ensemble determined that the traces collected for the *static* and *back* experiments lacked sufficient variation in the environment to allow secure proximity detection (i.e., both $acceptance_p$ and $acceptance_i$ were 0%). Thus, for these environments, Ensemble would prompt the user to induce some variation in the channel (e.g., by moving) before making a decision. We believe that this represents a minor inconvenience to the user that is more than offset by Ensemble's success in preventing pairing with impostor devices under these conditions. In the rest of this section we focus our discussion on experiment traces collected from the *moving* configuration.

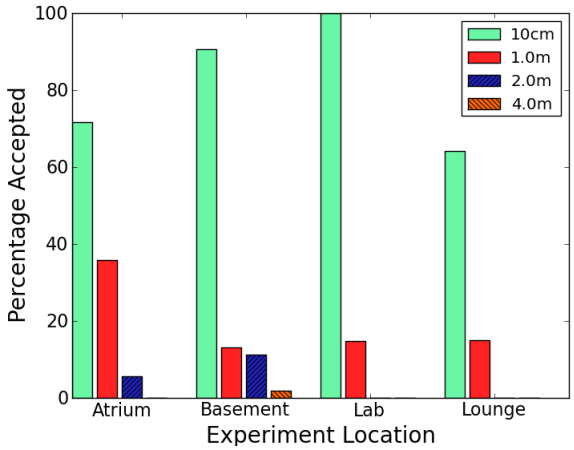


Figure 10: Acceptance rate of Ensemble across all environments.

Figure 10 shows how Ensemble performs in different environments when the channels are being varied.

On average, Ensemble correctly identifies the two pairing devices as being in close proximity in 81% of cases. The system's average acceptance rate for impostor devices located 1, 2 and 4 meters away is 7.5%, 2.4% and 0.5%. The acceptance rate for devices located 5 meters away or more (not shown) was 0%. From the above, we note that Ensemble is incapable of correctly rejecting devices located within 1m; however, realistically, we do not expect that an attacker would be within this range. In contrast, Ensemble is reliable at detecting impostors that are 2 or more meters away.

4.2.2 Ensemble Size

Figure 11 shows the average acceptance rate with varying number of devices across all environments that we sampled. The bars in the figure show the distribution of the data; the horizontal line on each bar represents the median value. The part of the bar above the median value shows the 75th percentile of the data, while the part below shows the 25th percentile of the data, and the white circle shows the mean of the distribution. We found that on average, a single device is incapable of making a reliable decision of proximity. This is reflected in the height of the bars for single devices for all locations. However, as devices combine their observations, we see that they become more accurate in determining proximity. From this figure we can see that on average, three devices are capable of achieving close to the performance of the five devices, and seems to be the optimal point for balancing number of devices and performance achieved.

4.2.3 Device Heterogeneity

The devices used for the previous experiments were identical and had similar configurations. In a real setting, this is

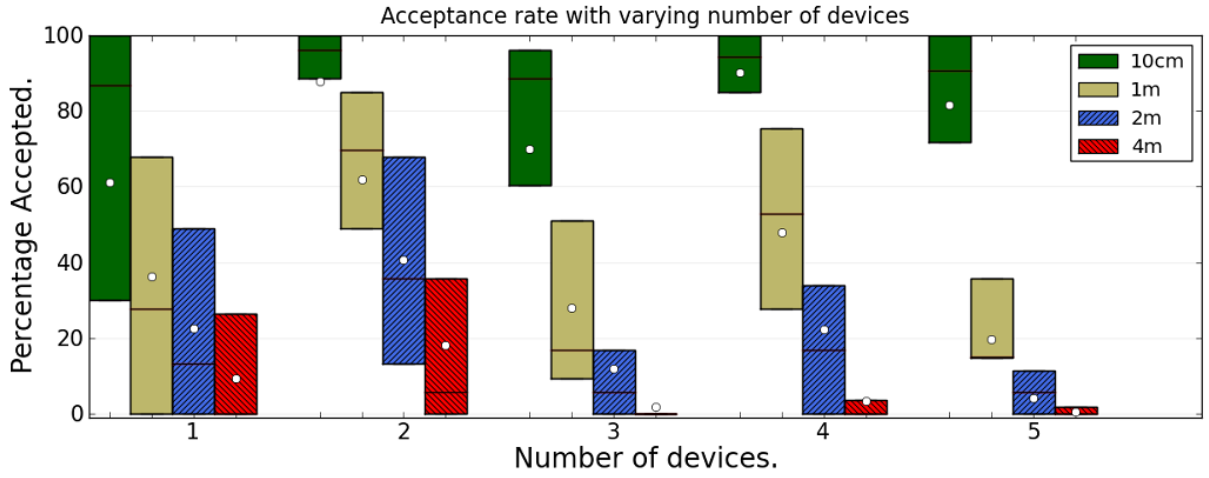


Figure 11: Acceptance rate across all environments with different number of devices.

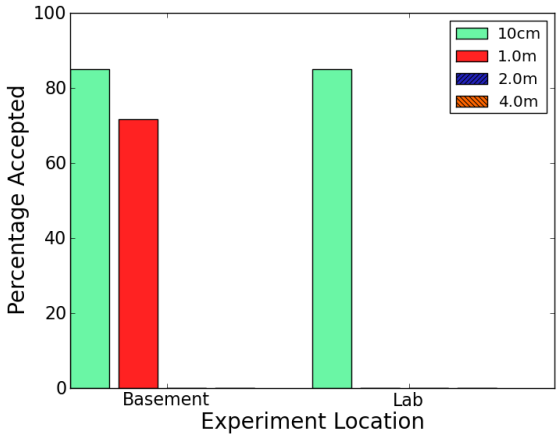


Figure 12: Acceptance rate of Ensemble with USB dongles and N800s.

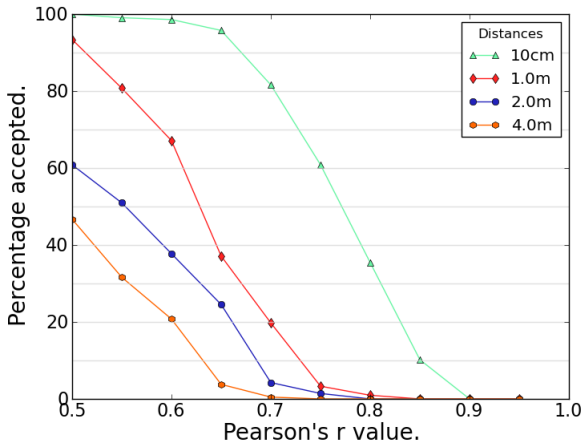


Figure 13: Performance of Ensemble with a varying r threshold.

not likely to be the case. To test Ensemble's robustness to device heterogeneity, we replaced the candidate device (C) and the three impostors (I1-I3) with Belkin USB WiFi dongles that were connected to a laptop. We set the dongles to transmit at 40 packets per second, and collected data for 10 minutes.

Figure 12 shows that Ensemble is robust to hardware variations and performs acceptably across different hardware. We choose two environments for this experiment, the lab, which so far gave us the best results, and the basement, which has given us the worst results from the set. Our criteria for best and worst results are based on the false acceptance rates for the different environments. For example, the Lab had the least amount of false positives while the basement had the most number of false positives. A user can easily re-pair in the case of a false negative, but pairing with an attacker has serious implications. The figure shows that while the acceptance rate for devices in proximity goes down somewhat, Ensemble actually improves in its ability to detect impostors located 2 or more meters away.

4.2.4 *Pearson's Coefficient Threshold Sensitivity*

Figure 13 shows the effect of varying the Pearson's Coefficient threshold r on Ensemble's mean acceptance rate across all environments. A high r value results in the system not accepting any devices, while a low r value will accept most devices in spite of proximity. We found that a value of $r = 0.7$ (this is the value of r that we use throughout this paper unless otherwise stated) gives us a good balance in all environments accepting the majority of instances where devices are within 1m and rejecting most instances when devices are further apart.

4.2.5 *Segment Size Sensitivity*

In our configuration, we use a segment size of 30 seconds to calculate an r value. Recall that a segment is a part of the signalprint consisting of consecutive RSS values within a specific time range. That is, all RSS values received within a 30 second period. In figure 14 we show the average effect of varying the segment size on the performance of Ensem-

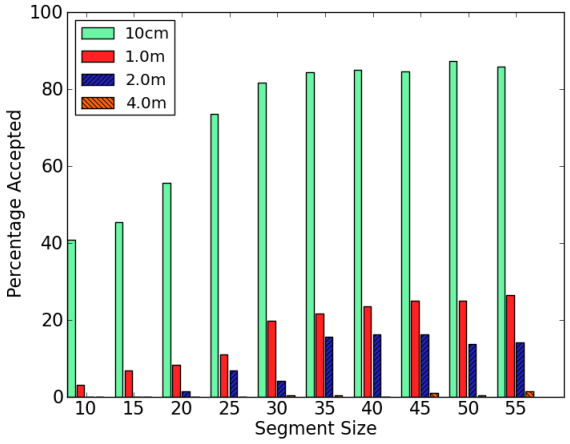


Figure 14: The effect of varying segment size (in seconds) on the performance of Ensemble. As the segment size increases, performance of Ensemble improves up to a threshold.

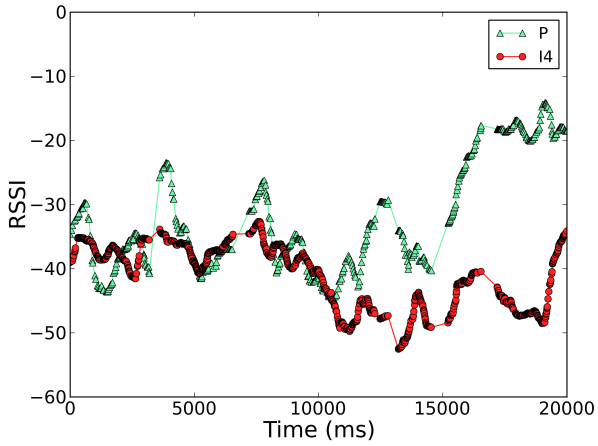


Figure 16: RSS of P and impostor with directional antenna.

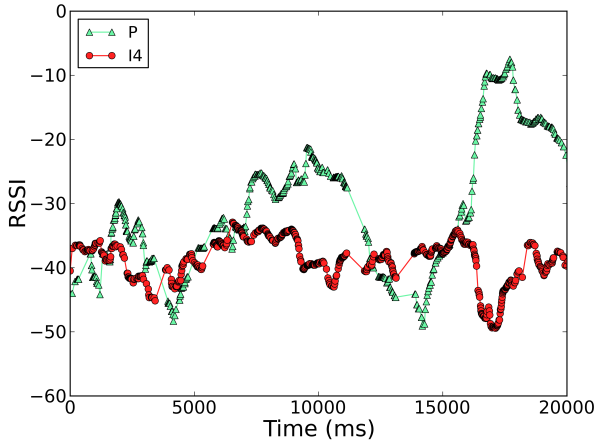


Figure 17: RSS of P and impostor with directional antenna and hand waving.

ble across all tested environments. As the segment size increases, the acceptance rate of Ensemble improves, until a size of about 30 - 35 seconds. After this threshold, there is no considerable increase in performance and the acceptance rate for devices in proximity begins to level off. Beyond this point Ensemble's performance actually decreases as the system begins to accept more devices located further apart. We believe this to be the correlation of large scale fading coming into effect.

4.2.6 *Decision Window Size Sensitivity*

Recall that the decision window is the number of overlapping segments a witness device uses in order to make a decision, and that we use a decision window of five to determine proximity. In this Section we evaluate the effect of varying the decision window on the performance of Ensemble. With more windows available to make a decision, Ensemble's performance should improve. The drawback of using more windows is the user will have to wait a longer period for a decision. Figure 15 shows the performance of Ensemble in all environments as the decision window is varied. As the window is increased across all environments, Ensemble is able to detect almost all attackers within 1m. In our worst environment, the basement, increasing the window size to 10 (this represents 120 seconds) improves the performance of Ensemble significantly, as only pairing devices within proximity are accepted, and a fraction of impostors at 2m away.

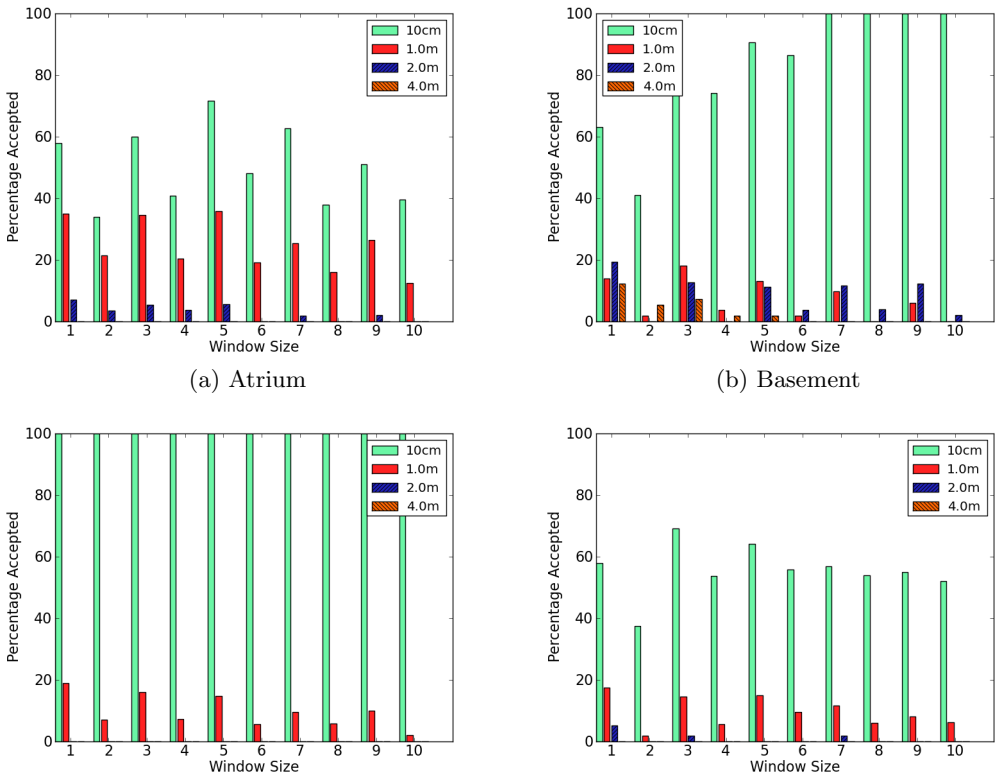
4.2.7 *Directional Attackers*

We evaluate whether an attacker gains an advantage by using a directional antenna by adding a new impostor to the setup shown in Figure 9. The new impostor is a laptop positioned 5m away from P and equipped with an Orinoco Gold WiFi card with an external directional antenna. The directional attacker also has clear line of sight to the experimenter and the trusted devices.

We test two scenarios: the first in which the attacker is just pointing the directional antenna at the trusted devices (while the experimenter is moving); and the second where the attacker creates random variation in the channel between the directional antenna and the receivers by moving her hand in front of the antenna in an attempt to dupe the system. This randomly varies the signal strength between the receivers and the attacker.

Figures 16 and 17 show the RSS for packets sent by P and the directional impostor (shown as I4). The differences between the two pairs of signals are clear even to the naked eye. Thus, it is not surprising that Ensemble's acceptance rate for the directional impostor in both configurations is 0% as Ensemble can easily detect that the random variation does not correlate with the pairing device.

It would be difficult for an attacker to infiltrate Ensemble if trying to impersonate a trusted device. To be successful, the attacker would either need to be in proximity of the pairing device, or be able to predict how the wireless channel will vary over the period and compensate by adjusting her signal strength accordingly. The former is clearly easier to accomplish.



(a) Atrium (b) Basement

(c) Lab (d) Lounge

Figure 15: The acceptance rate of Ensemble in different environments as the decision window is varied. As the decision window increases, Ensemble is able to detect impostors within 1m.

5. DISCUSSION AND EXTENSIONS

We see promise in the use of channel variations as a means for detecting proximity. The simple classifier used for Ensemble worked effectively in different environments. We envision that with richer information from the PHY layer, groups of devices could predict with a higher degree of confidence whether two transmitters are in close physical proximity.

The high-level techniques employed by Ensemble suggest several interesting extensions and applications of secure proximity detection. In this section, we briefly touch on two such extensions.

Activity Recognition

The proximity information inferred by Ensemble can be used for purposes other than secure pairing. A smart-home or smart-office could monitor the proximity relationships of an occupant's personal devices and those in the smart-space. As personal devices are observed in proximity to successive non-mobile devices, coarse location, occupancy and paths can be inferred. Proximity to sequences of semantically similar devices (e.g. stove, refrigerator, dishwasher) can be used to infer high-level activity (e.g. preparing food). While the observations offered by Ensemble are coarse, they have the advantage that they require no additional software or hardware for wirelessly networked devices to be detected as proximal.

Human-based Authentication

Throughout this paper, Ensemble has been evaluated from the context of a device. In reality however, a mobile device often acts only as an agent of a user; we would ideally like to provide authentication of users themselves. By automatically attempting to pair with each different on-body device, we could allow users to authenticate whenever *any* of their devices are in close physical proximity with a previously unknown device, aligning more naturally the idea of authenticating a user. The authentication of Alice and Bob's phones for example, by a chain-of-trust authenticates all of Alice and Bob's respective ensembles. Another example is transparently authenticating the point-of-service terminal that Alice is using without concern for the specific devices involved.

6. RELATED WORK

The research that is closest to Ensemble is the Amigo project [22]. Amigo also authenticates co-located devices by using knowledge of their common radio environment as proof of physical proximity, but uses a dramatically different approach. In Amigo, devices interested in pairing determine their relative proximity by passively monitoring the transmissions of ambient radio sources (e.g., available WiFi access points). Based on their observations, the pairing devices independently derive a signature, which they then securely compare to determine whether there is enough similarity to conclude that they are in close proximity. Ensemble takes

the opposite approach, with the pairing devices doing the transmission and the trusted body-worn collection of personal devices acting as infrastructure that determines proximity by monitoring these transmissions.

Ensemble has several advantages over Amigo. Foremost, Ensemble can be used to authenticate non-participating devices. That is, Ensemble requires that only one of the devices involved in the pairing has a trust relationship set up with the witnesses. One can verify that a device is in close proximity without requiring any involvement from the device beyond transmitting packets and generating a shared key (via a known protocol). In contrast, Amigo requires both devices involved in the pairing process to run special purpose software. In addition, Ensemble renders the task of faking proximity much more difficult. In Amigo an attacker in control of the ambient radio sources needs to predict the channel variations between these sources and the victim device. In contrast, in Ensemble, the attacker has to predict the channel between the victim device and the user's trusted collection of devices; arguably a much harder challenge. Furthermore, the Ensemble system is designed to use generic cryptographic methods and is not tied to any specific algorithm (such as WPA, SSL, Diffie-Hellman etc.).

Upon initial examination, fingerprint based localization systems seem to offer a promising way of inferring proximity. These systems generally provide a location estimate within a global co-ordinate system [2, 13, 16, 12]. Most of these approaches are however not secure as an attacker can easily obtain an RSS signature associated with a given position. Moreover, whereas Ensemble only needs to be calibrated once to operate in different environments, fingerprint based localization systems require expensive and tedious training in every target location. This is possible because Ensemble simplifies the problem. Instead of the more complex question *"Where am I?"*, Ensemble aims to answer only *"Are we close?"*.

Similarly, approaches such as location proofs [18], designed to prove presence at a location at a particular time by transmitting cryptographically signed timestamped location aware hashes, are susceptible to attacks by remote devices with sensitive directional antennas – with the appropriate hardware a remote attacker can boost their transmission range and obtain a location proof from far away.

6.1 Leveraging Radio Information

Manufacturing differences in the radio hardware have been used to identify individual devices [4]. Patwari [17] and Faria [6] leverage PHY information in order to distinguish between devices in different physical locations. These techniques apply when communicating with a previously known device, i.e., the device's hardware profile is known in advance, or when communication takes place between devices located at well known positions, i.e., the channel between the two devices is known ahead of time. Unfortunately, this information (which in effect is equivalent to a shared secret between the two devices) is not available in pervasive environments where users spontaneously pair with previously unknown devices.

Techniques exist for deriving a secret directly from variations in the radio channel [14, 9]. These techniques are complementary, as they are often focused on the task of *securing* a channel between two communicating devices as opposed to authenticating a previously unknown device. Both Ensemble and the techniques deriving a secret from the radio channel benefit from a dynamic environment. Their combination could be applied to establish a channel between devices in close physical proximity which is both secure and authentic without the need for any heavy-weight public key cryptography.

6.2 Other Device Pairing Approaches

The most common pairing mechanism in-use, Bluetooth pairing, has users provide a secret PIN number to two devices. This approach is simple, but requires user involvement and a keypad. As a result, many devices without a keypad use a default PIN of 0000 or 1234, completely defeating the purpose of the PIN. Additionally, Bluetooth pairing has been shown to be susceptible to eavesdroppers [19], who may be equipped with sensitive directional antennas. In contrast, Ensemble does not require users to type a PIN and is not susceptible to eavesdropping.

Using common readings from accelerometers to establish an association between devices shaken at the same time has been proposed by [8] and [15]. While accelerometers are becoming more common on mobile devices, some scenarios may not be appropriate for shaking. For example, authenticating with a public display, a vending machine or a laptop computer would demand a different approach.

The use of physically-constrained channels have also been explored as a means of establishing authentic connections between devices. Examples include the use of a direct electric contact [21], infrared beacons [3, 20], ultrasound [10], and near field communication (NFC) [1]. Unfortunately, physically constrained channels often require extra hardware (e.g., a special cable), and can be susceptible to attacks by sensitive receivers that can detect dim signal refractions and reflections.

The NearMe Wireless Proximity Server [11] gave coarse notions of proximity using signal strength estimates. The granularity of the proximity information in this case was limited to rooms, which was application-appropriate. Generally, specialized hardware is needed to provide relative distance information at a finer granularity. A good example of such a system is Relate [7], which could provide accurate distance and orientation information within centimeters.

A public key infrastructure can side-step the problem of authentication. However in this case, every mobile device must have a well-known name and certificate a priori; simply shifting the authentication problem to one of naming.

7. CONCLUSION

We introduced Ensemble, a system that determines if two devices are in close physical proximity by taking advantage of the similarity of the channel between these devices and a third, observing device. This system leverages the many devices that users already possess to aid in this process. We showed that while it may be difficult for one device to determine proximity, an ensemble of devices is able to do so reliably.

Our system is not environment specific, does not require any recalibration, works for different hardware and is robust against several types of attacks, including impostor and man-in-the-middle attacks. We showed that Ensemble is capable of detecting attackers located at least 2m away. We found that it is difficult to determine proximity in static environments, but we can reliably detect these cases and

prompt the user accordingly. We evaluated Ensemble with WiFi enabled mobile devices and showed that we can accurately determine when devices are in close proximity.

8. REFERENCES

[1] Near field communication (nfc). http://www.nfc-forum.org/resources/faqs.

[2] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *INFOCOM: Proceedings of IEEE Conference on Computer Communications*, volume 2, pages 775–784, Tel-Aviv, Isreal, March 2000.

[3] D. Balfanz, D. Smetters, P. Stewart, and H. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proc. Network and Distributed Systems Security Symposium*, San Diego, CA, 2002.

[4] V. Brik, S. Banerjee, M. Gruteser, and S. Oh. Wireless device identification with radiometric signatures. In *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 116–127, New York, NY, USA, 2008. ACM.

[5] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, pages 644–654, November 1976.

[6] D. B. Faria and D. R. Cheriton. Detecting identity-based attacks in wireless networks using signalprints. In *WiSe '06: Proceedings of the 5th ACM workshop on Wireless security*, pages 43–52, New York, NY, USA, 2006. ACM.

[7] M. Hazas, C. Kray, H. Gellersen, H. Agbota, G. Kortuem, and A. Krohn. A relative positioning system for co-located mobile devices. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 177–190, New York, NY, USA, 2005. ACM.

[8] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.-W. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *UbiComp '01: Proceedings of the 3rd International Conference on Ubiquitous Computing*, September 2001.

[9] S. Jana, S. N. Premnath, M. Clark, S. K. Kasera, N. Patwari, and S. V. Krishnamurthy. On the effectiveness of secret key extraction from wireless signal strength in real environments. In *MobiCom '09: Proceedings of the 15th ACM international conference on Mobile computing and networking*, pages 321–332, New York, NY, USA, 2009. ACM.

[10] T. Kindberg and K. Zhang. Validating and securing spontaneous associations between wireless devices. In *ISC '03: Proceedings of the 6th Information Security Conference*, Bristol, UK, 2003.

[11] J. Krumm and K. Hinckley. The nearme wireless proximity server. In *UbiComp '04: Proceedings of the 8th International Conference on Ubiquitous Computing*, pages 283–300, Notthinham, UK, September 2004. Springer.

[12] A. Ladd, K. Bekris, G. Marceau, A. Rudys, L. Kavraki, and D. Wallach. Robotics-based location sensing using wireless ethernet. In *MobiCom '02: Proceedings of the 8th ACM International Conference on Mobile Computing and Networking*, Atlanta, GA, USA, 2002.

[13] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit. Place lab: Device positioning using radio beacons in the wild. In *Proceedings of the Third International Conference on Pervasive Computing*, Lecture Notes in Computer Science. Springer-Verlag, May 2005.

[14] S. Mathur, W. Trappe, N. Mandayam, C. Ye, and A. Reznik. Radio-telepathy: extracting a secret key from an unauthenticated wireless channel. In *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 128–139, New York, NY, USA, 2008. ACM.

[15] R. Mayrhofer and H. Gellersen. Shake well before use: Authentication based on accelerometer data. In *Proceedings of the 5th International Conference on Pervasive Computing*. Springer, 2007.

[16] V. Otsason, A. Varshavsky, A. LaMarca, and E. de Lara. Accurate gsm indoor localization. In *UbiComp '05: Proceedings of the 7th International Conference on Ubiquitous Computing*, Tokyo, Japan, September 2005.

[17] N. Patwari and S. K. Kasera. Robust location distinction using temporal link signatures. In *MobiCom '07: Proceedings of the 13th annual ACM international conference on Mobile computing and networking*, pages 111–122, New York, NY, USA, 2007. ACM.

[18] S. Saroiu and A. Wolman. Enabling new mobile applications with location proofs. In *HotMobile '09: Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, February 2009.

[19] Y. Shaked and A. Wool. Cracking the bluetooth pin. In *MobiSys '05: Proceedings of the 3rd International Conference on Mobile Systems, Applications and Services*, Seattle, WA, June 2005.

[20] D. Smetters, D. Balfanz, G. Durfee, T. Smith, and K. Lee. Instant matchmaking: Simple, secure virtual extensions to ubiquitous computing environments. In *UbiComp '06: Proceedings of the 8th International Conference on Ubiquitous Computing*, Irvine, CA, September 2006.

[21] F. Stajano and R. J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proceedings of the 7th Security Protocols Workshop*, Cambridge, UK, 1999.

[22] A. Varshavsky, A. Scannell, A. LaMarca, and E. de Lara. Amigo: Proximity-based authentication of mobile devices. In J. Krumm, G. D. Abowd, A. Seneviratne, and T. Strang, editors, *Ubicomp*, volume 4717 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2007.

[23] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *MobiHoc '05: Proceedings of the 6th International Symposium on Mobile Ad-hoc Networking and Computing*, New York, NY, USA, 2005. ACM.