ProxiMate: Proximity-based Secure Pairing using Ambient Wireless Signals

Suhas Mathur[1]*, Rob Miller[2], Alexander Varshavsky[3], Wade Trappe[2], and Narayan Mandayam[2]
[1]AT&T, 33 Thomas Street, New York, NY, suhas@att.com
[2]WINLAB, Rutgers University, 671 Route 1 South, North Brunswick, NJ, USA
[2]{rdmiller, trappe, narayan}@winlab.rutgers.edu
[3]AT&T Labs, Florham Park, NJ, varshavsky@research.att.com

ABSTRACT

Forming secure associations between wireless devices that do not share a prior trust relationship is an important problem. This paper presents ProxiMate, an algorithm that allows wireless devices in proximity to securely pair with one another autonomously by generating a common cryptographic key directly from their shared time-varying wireless environment. The shared key synthesized by ProxiMate can be used by the devices to authenticate each others' physical proximity and then to communicate confidentially. Unlike traditional pairing approaches such as Diffie-Hellman, ProxiMate is secure against a computationally unbounded adversary and its computational complexity is linear in the size of the key. We evaluate ProxiMate using an experimental prototype built using an open-source software-defined platform and demonstrate its effectiveness in generating common secret bits. We further show that it is possible to speed up secret key synthesis by monitoring multiple RF sources simultaneously or by shaking together the devices that need to be paired. Finally, we show that ProxiMate is resistant to even the most powerful attacker who controls the public RF source used by legitimate devices for pairing.

1. INTRODUCTION

The number of devices with wireless interfaces is growing at an increasingly rapid pace. This growth fuels the need for devices to interact as they move about and come in proximity of one another. As an example, two people meeting for the first time may wish to exchange data between their mobile devices, or a passenger at a train station may wish to pay for a ticket by having their phone interact with an electronic ticket booth.

Securing such interactions from malicious adversaries is an important and challenging problem. Due to the broadcast nature of the wireless medium, how does one device know that it is really interacting with the device it intends to communicate with, if it has never encountered this device before? As a result, setting up a secure link between wireless devices in proximity is presently a surprisingly cumbersome procedure that often requires significant human intervention in the form of entering a shared key on both devices. Moreover, with the global trend toward miniaturization and increased variety of device form factors, the devices may not have a common set of hardware components required for setting a human-supported secure association. For instance, the devices may or may not have a screen, buttons, LEDs, accelerometers, RFIDs and NFC chips. The only hardware component that is guaranteed to be present on all interoperable devices is the wireless radio.

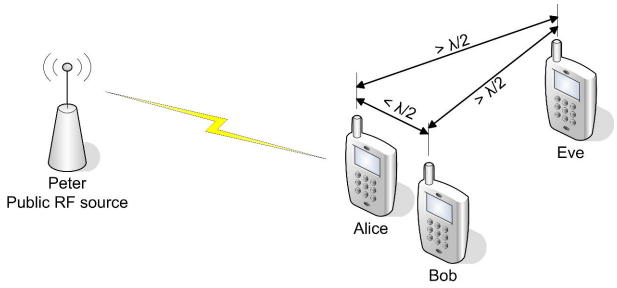


Figure 1: Temporal variations in the wireless RF channel from a public source (Peter) can be used by parties in physical proximity (Alice and Bob) to extract a random cryptographic key. An adversary (Eve) who is not within the proximity of Alice and Bob cannot extract the same key. Here, λ is the wavelength of the public RF transmission.

In this paper, we show that devices in close physical proximity can derive a shared secret key directly from their common but continuously fluctuating radio environment. Our technique, called ProxiMate, is based on the observation that wireless devices in close proximity perceive the same small scale temporal variations in their wireless channels, known as *small-scale fading*. We show that these common temporal variations can be used by the co-located devices to derive a shared secret key directly, without the need for an expensive Diffie-Hellman key exchange [1, 2]. In contrast, as shown in

*This work was done while the primary author was at WINLAB, Rutgers University.

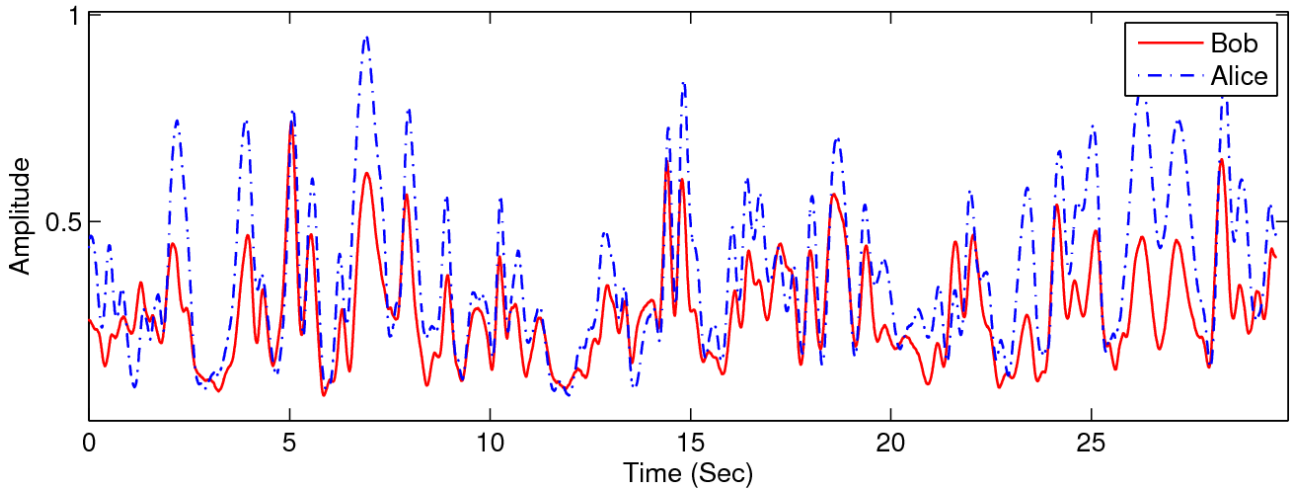


Figure 2: A 30 second trace of the temporal channel variations observed at two receivers tuned to a TV broadcast frequency (584.31 MHz) when the receivers are 8 cm apart.

Figure 1, an adversary that is not in close proximity to the co-located devices will perceive different small-scale fading effects and won't be able to derive the same secret key. Figure 2 illustrates that the temporal variations as perceived by two co-located devices are indeed correlated.

The proximity at which the two co-located devices must reside in order to perceive common small scale fading effects depends on the wavelength of the RF transmission. As a general rule of thumb, the devices located closer than half a wavelength will perceive similar signal fluctuations, whereas a device that is located further than half a wavelength will perceive different small scale fading effects. For example, the wavelengths of an 90 MHz FM Radio, 624 MHz TV and 2.4 GHz WLAN transmissions are 3.3m, 47cm and 12.5cm, respectively.

The main idea of using common radio environment as a proof of physical proximity was first presented in Amigo [2]. Amigo relies on the Diffie-Hellman protocol for a secret key exchange between two devices and then utilizes the received signal strength indicator (RSSI) from WiFi packets for a proof of physical proximity. ProxiMate has four advantages over Amigo. First, ProxiMate doesn't use the Diffie-Hellman key exchange protocol, but instead derives the shared key directly from the radio environment. Removing the reliance on the Diffie-Hellman key exchange has two benefits: (a) Diffie-Hellman involves discrete modular exponentiation which is $O(n^3)$ in the size n of the desired key, whereas ProxiMate uses a linear code, whose complexity is $O(n)$. This makes ProxiMate particularly useful in domains where the entities wishing to form a key have limited computational resources, such as sensor networks [3]; (b) reliance on Diffie-Hellman implicitly assumes that the adversary is computational bounded, whereas ProxiMate extracts secret bits from the wireless channel itself and is therefore secure against even a computationally unbounded adversary. Second, instead of relying just on WiFi, ProxiMate can utilize any available radio technology. For example, in this paper we show how ProxiMate utilizes FM radio and TV signals for secret key extraction. Third, instead of relying on individual packets, ProxiMate uses a continuous signal to measure the time-varying channel, thereby improving the speed at which common randomness can be harvested. Finally, ProxiMate does not rely on a coarse-grained RSSI metric that measures the average received power in a packet preamble and is, therefore, relatively simple for an attacker to manipulate in a controlled manner, as shown in [4]. Instead, it uses a finer grained 2-dimensional (amplitude and phase) channel measurements, which aren't as easy to manipulate in a controlled manner [4].

This paper makes four contributions. First, we describe ProxiMate, a mechanism for wireless devices in physical proximity to convert their correlated wireless channel measurements into identical sequences of bits, which can then be used as a shared encryption key. As part of the algorithm, we introduce a novel encoding scheme, which we call list-encoding. Second, we evaluate the performance of this algorithm through measurements of ambient wireless signals. Third, we propose to simultaneously monitor multiple RF sources and show that concurrent monitoring significantly increases the rate at which secret bits can be extracted. As an example, ProxiMate can generate the equivalent of a 4-digit PIN used in Bluetooth systems within 0.34 sec with 10 TV sources (see Section 4.6 for details). Finally, we describe how the *differential-phase* of the received signals can be used to protect against a very powerful adversary who controls the public wireless source that is being used by co-located devices for the secret bit

extraction.

2. SYSTEM MODEL

In this section, we define the problem of proximity-based secure association, describe our threat model and present an overview of wireless channel characteristics that are important for this paper.

2.1 Problem Definition and Threat Model

We assume that two legitimate wireless devices, *Alice* and *Bob*, are located in close physical proximity and are interested in exchanging private information via compatible radios, without an adversary *Eve* being able to decrypt their communication. This requires an authenticated and secure channel. Alice and Bob do not know each other a priori. Therefore, they cannot prove their identities to each other. However, Alice and Bob know that they are located in close physical proximity and can use this information for authentication. Note that Alice and Bob cannot simply use the Diffie-Hellman key exchange [1] protocol to establish a secure channel since Eve can easily masquerade as Alice to Bob or vice-versa, or launch a man-in-the-middle attack. We further assume that Alice, Bob and Eve can overhear a public source of radio waves *Peter* that transmits a signal with a wavelength of λ. Peter can be an FM radio station, a TV station, a WiFi AP or a cellular tower.

We consider two threat models. First, we assume that Eve can eavesdrop on all communications, and can try to authenticate with Bob, pretending that she is Alice (or vice versa). Second, we show how our technique can deal with the most powerful attacker who controls the public source of radio waves that Alice and Bob are using for authentication - that is Eve is Peter! In both cases, we assume that Eve is located at a further distance from both Alice and Bob than they are from each other. In Section 4, we show that to reliably derive a shared secret key, Alice and Bob need to be located closer than 0.1λ from each other, whereas the attacker needs to be located further than 0.4λ from Alice and Bob.

2.2 Wireless Channel

We assume that the wireless channel between Peter and the legitimate devices is a multi-path channel, consisting of many reflectors and scatterers. Many of the paths taken by the signals transmitted by Peter are time-varying due to the movement of people and objects and varying atmospheric conditions. Even slight perturbations in the environment can generate significant signal fluctuations at the receiver. This is the case for all terrestrial wireless transmissions such as FM radio, TV and cellular networks. The state of the wireless channel at any given time instant can be expressed as a complex number, representing amplitude and phase

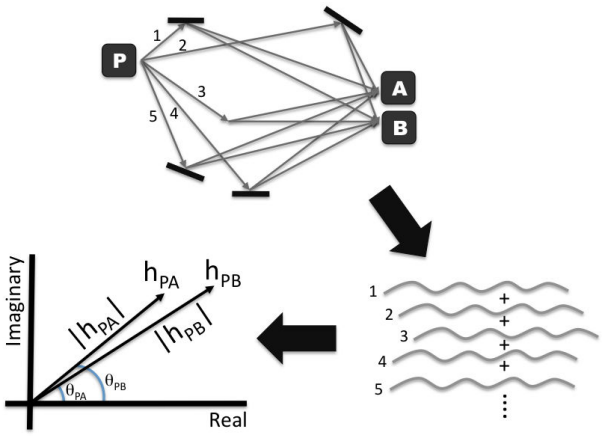


Figure 3: Wireless signals travel along multiple paths, adding up to random amplitude and phase at the destination. Thus, a wireless channel can be represented as a 2-D vector resulting from the sum of many random 2-D vectors, each corresponding to a separate path from transmitter to receiver. If Alice and Bob are in proximity, the 2-D vectors corresponding to the channels between Peter and them are correlated.

of the composite channel (see Figure 3). The received signal has an amplitude given by the product of the transmit signal's amplitude and the amplitude of the channel at that instant. The phase of the received signal is the sum of the phase of the transmit signal and the phase of the channel at that instant.

ProxiMate leverages two properties of the wireless channel. First, wireless channels at two locations in close proximity are highly correlated. However, this correlation diminishes fast as the distance between the two locations increases, and typically vanishes completely beyond $\lambda/2$. Second, both the strength and the phase of the wireless channel vary randomly in time due to changes in reflectors and scatterers. However, the signal is highly correlated at times t and $t+\delta$ if δ is small. The correlation vanishes as δ increases. The smallest δ for which the channel is statistically independent at times t and $t+\delta$ is termed the *coherence time* T_c of the channel. In other words, the coherence time of the channel indicates how often the channel appears random again. Thus, samples of the signal at times t and $t+T_c$ are statistically independent [5].

3. PROXIMATE

In this section, we describe how Alice and Bob, two devices in close proximity, can derive a shared secret key by tuning into an ambient wireless signal. ProxiMate is based on the observation that measurements of the wireless *channels* between an RF source and devices in proximity are highly correlated. Channel measurements

Symbol	Meaning
$s(t)$	Signal transmitted by the public source, Peter.
$\underline{X}_U$	Complex samples collected by user U
$\underline{h}_{PU}$	Complex channel measurements between Peter and U
Q_U	Quantizer threshold of user U
$\underline{S}_U$	U's estimate of $s(t)$ gotten by demodulating $\underline{X}_U$
$\underline{K}_U$	Result of quantizing $\lvert\underline{h}_{PU}\rvert$ w.r.t. Q_U.
$\mathcal{C}$	An (n, k) linear block error correcting code.
$f_c(x)$	n-bit Codeword in $\mathcal{C}$ that is closest to n-bit sequence x
P	Offset of $\underline{K}_a$ w.r.t. closest codeword in $\mathcal{C}$

Table 1: Summary of notation. Underlined symbols indicate vectors. In subscripts, the letter 'U' refers to user and can be Alice, Bob or Eve.

consist of amplitude and phase components (see Figure 3 and Section 2.2), and since the two components vary independently of one another but are correlated at Alice and Bob, secret bits can be extracted from both separately. Extracting bits from phase has the advantage of being secure against a strong attacker who *controls* the public wireless source that is being used by Alice and Bob, as we show in Section 4.7.

Below, we first provide an outline of the logical steps involved in extracting secret-bits from a general correlated quantity at Alice and Bob. We will then specialize this algorithm for the specific cases when the correlated quantity is amplitude of the wireless channel (Section 3.8), and when it is the phase (Section 3.9).

1. Alice and Bob periodically sample an ambient wireless signal coming from Peter. Each user uses the samples to obtain a sequence of channel measurements.

2. Each user also uses samples of the received signal to demodulate the signal transmitted by Peter.

3. Each user uses its channel measurements to estimate the coherence time T_c of its channel.

4. Alice obtains a sequence of bits from her channel measurements. The exact mechanism for obtaining bits from channel measurements depends upon whether amplitude of phase is used for extracting bits. Each successive bit is obtained from channel measurements separated in time by at least T_c, as calculated in step 3.

5. Alice sends Bob a snippet of her demodulated signal from step 2. This is meant to help Bob synchronize to Alice's time-frame so that he can extract similar bits as Alice. After synchronizing to Alice's time frame, Bob extract bits from his channel measurements, just like Alice did in step 4.

6. Along with the snippet in step 5, Alice sends Bob, in the clear, a function of the bit-sequence that she obtains from her measurements in step 4. Bob uses this information along with his own bits, which he obtains in step 5, to arrive at Alice's bits with high probability. This step is called *reconciliation* [6].

7. Alice and Bob discard part of their bits to eliminate any information that leaked out to Eve during step 6. This step is called *privacy amplification* [7].

Below we expand and elaborate on each of the steps described above. Refer to Table 1 for the summary of notation used. Underlined variables represent arrays (i.e. sequences of values rather than a single value). Subsections 3.1 to 3.7 correspond to the numbering of the 7 steps outlined above.

3.1 Collecting samples and channel measurements

Every T seconds, each user samples the signal received from Peter. T is a fixed parameter of the protocol and is chosen to be much smaller than the expected approximate coherence time T_c. These samples are stored into a time series $\underline{X}_u$, where u = Alice/Bob/Eve, and are used to measure the channel time series $\underline{h}_{PU}$. Since the wireless channel has both amplitude and phase, each element of the channel measurement vector $\underline{h}_{PU}$ is represented by a complex number. It is the temporal variation of the amplitude portions of $\underline{h}_{PA}$ and $\underline{h}_{PB}$ that is shown in Figure 2. It is important to understand that the samples $\underline{X}_u$ of the received signal are not the same as channel measurements $\underline{h}_{PU}$. Instead, channel measurements are *derived* from the samples of the received signal $\underline{X}_u$. For example, if it is known that the transmit signal is of constant amplitude (fixed power), then the variation in the amplitude of the received signal $|\underline{X}_u|$ must be purely due to the variations in the wireless channel and can therefore be treated as $|\underline{h}_u|$.

3.2 Demodulation

From the sequence of received samples $\underline{X}_a$, each user separately demodulates $s(t)$, the signal transmitted by Peter, to obtain the demodulated signal $\underline{S}_u$ (u = Alice or Bob). For example, if Peter is an FM radio transmitter, the demodulated signal is simply the audio signal on the FM station. The purpose of demodulating Peter's signal is to use it to time-synchronize Alice and Bob.

3.3 Estimating the coherence time

Each user separately estimates the coherence time of the channel, T_c, using its sequence of channel measurements $\underline{h}_{PU}$. In estimating T_c, we assume a Rayleigh fading channel [8]. We tested this assumption in practice and found it to be correct [9]. That is, the distribution of channel amplitude measurements matches closely a Rayleigh distribution.

The coherence time T_c of a Rayleigh fading channel is related to the level crossing rate, LCR, of the channel, i.e. simply the rate at which channel amplitude crosses a specified level. For a Rayleigh fading channel, this

is [8]

$$LCR = \sqrt{2\pi} f_d \rho e^{-\rho^2} \quad (1)$$

where ρ the ratio between the level-crossing threshold considered and the root mean square amplitude, and f_d is the Doppler spread. From this, T_c can be estimated as [8]:

$$T_c \approx \frac{1}{4 f_d} = \frac{3\rho e^{-\rho^2}}{2\sqrt{2\pi} LCR} \quad (2)$$

For example, the coherence time of the channel being monitored in Figure 2 is estimated to be 0.22 sec.

3.4 Obtaining bits

The process of translating channel measurements $\underline{h}_{PA}$ and $\underline{h}_{PB}$ to bits lies at the heart of ProxiMate. Here we will describe a general method to obtain bits from a given correlated quantity at Alice and Bob and then we will specialize it to the case of amplitude in Section 3.8 and to phase in Section 3.9. Suppose the quantity that Alice and Bob are attempting to obtain bits from, is stored as an array $\underline{A}$ at Alice and an array $\underline{B}$ at Bob. For the case of amplitude, $\underline{A} = |\underline{h}_{PA}|$ and $\underline{B} = |\underline{h}_{PB}|$, and for phase, $\underline{A} = angle(\underline{h}_{PA})$ and $\underline{B} = angle(\underline{h}_{PB})$. Alice and Bob separately compute the median Q_u of their measurements $\underline{A}$ and $\underline{B}$. Alice quantizes $\underline{A}$ once per T_c, using Q_a as a quantizer threshold to extract one bit per coherence time, to obtain a bit-sequence $\underline{K}_a$. That is, her quantizer output is a bit "1" if the value of an element in $\underline{A}$ is $< Q_a$ and "0" otherwise. Bob performs the exact same steps using his own measurements $\underline{B}$, after synchronizing to Alice's time-frame of reference (see next step - Section 3.5) and obtains the array of bits $\underline{K}_b$.

3.5 Synchronization

Alice sends Bob a snippet of her demodulated signal $\underline{S}_a$, e.g. demodulated audio in the case of FM-radio, with the understanding that the start of this snippet corresponds to time $t = 0$ in her frame of reference. Bob can thus easily synchronize to Alice's time-frame of reference by correlating his own demodulated signal $\underline{S}_b$ with Alice's snippet. By synchronizing, the bits $\underline{K}_b$ obtained by Bob correspond to the same time instants as Alice's bits $\underline{K}_a$.

3.6 Reconciliation

After synchronization and quantization, Alice and Bob end up with n-bit sequences, $\underline{K}_a$ and $\underline{K}_b$ respectively, which may differ at any given bit position with a probability ϵ. Reconciliation is the process of 'repairing' the difference between $\underline{K}_a$ and $\underline{K}_b$ so that Alice and Bob end up with the same bit-sequence and can then use it as a key. The reconciliation process drastically reduces the errors between Alice's and Bob's bit sequences (hopefully eliminating all errors), but it also reduces the number of common bits that Alice and Bob share. Hence, reconciliation trades off bit-error rate ϵ with the key bit-rate.

A crucial observation that allows reconciliation is the following: suppose we treat Alice's bit sequence $\underline{K}_a$ as the key and would like Bob to somehow deduce $\underline{K}_a$. Then Bob's bit-sequence $\underline{K}_b$ can be thought of as a distorted version of $\underline{K}_a$. How can Alice convey to Bob which bits in $\underline{K}_b$ are different from $\underline{K}_a$ without actually sending him $\underline{K}_a$ (that would reveal $\underline{K}_a$ to Eve)? The answer lies in treating both $\underline{K}_a$ and $\underline{K}_b$ as distorted versions of 'some' n-bit codeword of an (n, k) error correcting code[1] $\mathcal{C}$. An (n, k) code $\mathcal{C}$ consists of a one-to-one encoding function that maps any k-bit string to a n bit string ($n > k$), and a many-to-one decoding function that maps any n-bit string to one of 2^k n-bit strings called 'codewords' of $\mathcal{C}$. The code $\mathcal{C}$ to be used is known to all parties, including Eve.

Let $f_c(\cdot)$ be the decoding function of $\mathcal{C}$ that maps any n-bit sequence to the closest codeword in $\mathcal{C}$. Alice first computes $f_c(\underline{K}_a)$, i.e. the codeword in $\mathcal{C}$ that is closest to $\underline{K}_a$ and then computes the offset $P = \underline{K}_a - f_c(\underline{K}_a)$, i.e. the bit-by-bit difference between this codeword and $\underline{K}_a$. Alice then sends P to Bob in cleartext. If the value of ϵ is roughly known, a code with a suitable error correcting ability can be chosen to allow Bob to decode $\underline{K}_a$ using $\underline{K}_b$ and P by the following operation: $P + f_c(\underline{K}_b - P)$ which is equal to $P + f_c(\underline{K}_a)$ with high probability, which in turn equals $\underline{K}_a$ by the definition of P. Therefore, at the end of this step, both Alice and Bob possess $\underline{K}_a$ with high probability. We refer to this construction as a *quantization-based construction*. We will see below, that this construction is not suitable for use for extracting a secret key from the amplitude of channel measurements $\underline{h}_a$, but is suitable for extracting the key from phase.

3.7 Privacy amplification

Since the offset P is sent by Alice in the clear, Eve obtains *partial information* about Alice's and Bob's shared key, $\underline{K}_a$. That is, although she cannot guess $\underline{K}_a$ by observing P alone, she knows something about $\underline{K}_a$ – in particular she knows that $\underline{K}_a - P =$ a valid codeword of $\mathcal{C}$. It can be formally shown that Eve's learns $n - k$ bits of information about Alice's and Bob's n-bit strings by observing P. Therefore, in order to ensure that the key used by Alice and Bob is not even *partially* known to Eve, in the privacy amplification step, Alice and Bob reduce the size of their bit-string by $n - k$ bits to obtain k-bit strings, about which Eve has absolutely no information. The simplest way to accomplish this is to use the k-bit pre-image of the n-bit codeword $f_c(\underline{K}_a)$,

[1]For simplicity, we will assume the $\mathcal{C}$ is a linear block error correcting code (we use the (12, 23) Golay code in Section 4 in our experimental evaluation.)

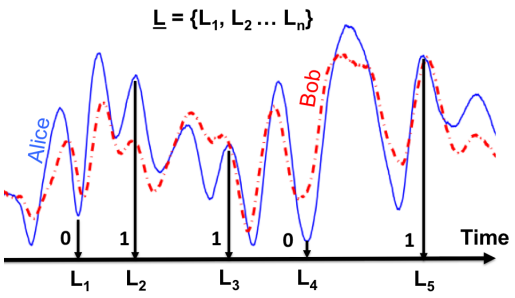


Figure 4: An illustration of list-encoding. Alice uses relative maxima and minima to generate $\underline{K}_a$. Discret-time indices $L_i, i = 1, \ldots n$ of the selected extrema are collected into the list $\underline{L}$ to be sent to Bob.

which Alice and Bob possess.

We have now described how Alice and Bob can extract an identical secret key using ProxiMate from a corellated set of channel measurements. Next, we tweak the basic framework of ProxiMate above to make it suitable for extracting bits from amplitude (Section 3.8) and phase (Section 3.9).

3.8 List-encoding

Unfortunately, the quantization-based construction described in Section 3.6 has the problem that unless the bit-error rate ϵ, between the bits of $\underline{K}_a$ and $\underline{K}_b$ is extremely small, the code $\mathcal{C}$ needed to do reconciliation successfully, must be of a very large block-length, i.e. n must be very large. The Bit-error rate ϵ is related to the distance between Alice and Bob d/λ (we evaluate this relationship in Section 4), and for typical values of d/λ in our problem, ϵ is moderately large ($\sim 0.1 - 0.2$). For example, when using an ATSC TV signal at 584.31 MHz (channel 33) and when Alice and Bob are only 1.5 inches apart, we have $\epsilon \approx 0.15$. Using a code $\mathcal{C}$ with a very large block-length n requires that ProxiMate be run for a long time in order for Alice and Bob to collect n-bit long $\underline{K}_a$ and $\underline{K}_b$. In this subsection, we show how it is possible to lower ϵ for a given d/λ, by altering the way in which Alice and Bob obtain bits from their channel measurements.

We now present list-encoding, an alternative to the quantization-based construction above for extracting bits from the amplitude of channel measurements, that significantly lowers ϵ to a value suitable for use of error correcting codes with a reasonably small block length n. Short codes, in turn, enable ProxiMate to pair Alice and Bob in a shorter time-interval.

List-encoding uses the relative minima and maxima in the temporal variations of $|\underline{h}_{PU}|$ to create bits at Alice and Bob (see Figure 2) instead of quantization with a threshold Q_u in the quantization-based construction. It involves the following steps (see Figure 4)

1. Alice locates sharp upward peaks and deep downward fades in the sequence of her channel measurements.

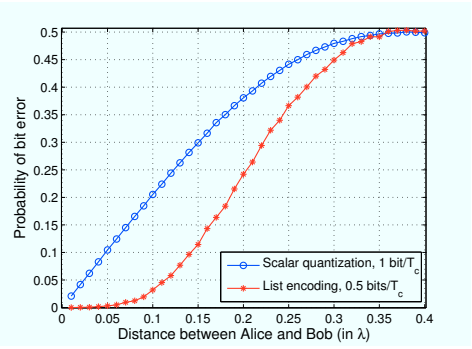


Figure 5: Bit-error rate between raw bits at Alice & Bob for a simple quantizer and for list-encoding (Section 3.8).

2. Alice maps upward peaks to a bit "1" and downward fades to a bit "0". In this way Alice obtains a bit-string $\underline{K}_a$.

3. Each successive extremum chosen by Alice is separated from the previous extremum by at least T_c to make sure that bits in the extracted key are independent.

4. Alice collects the time-indices at which these extrema occur into a list $\underline{L}$.

5. Alice demodulates her received signal to obtain $\underline{S}_a$, just like in the ProxiMate's basic algorithm.

6. Alice computes a code-offset P of $\underline{K}_a$ with respect to a known linear block code $\mathcal{C}$ as before.

7. Alice sends Bob a package consisting of $\{P, \underline{S}_a, \underline{L}\}$

8. Bob synchronizes to Alice's time-frame of reference using $\underline{S}_a$ as before.

9. Bob determines for each time-index in $\underline{L}$, the location of the nearest extremum in his own sequence $|\underline{h}_{PA}|$ of channel measurements. If his nearest extremum is a maximum, he assigns the bit "1", and for a minimum, he assigns a "0", and thus Bob builds his bit-sequence $\underline{K}_b$.

10. Bob performs reconciliation using P and $\underline{K}_b$ and knowledge of the code $\mathcal{C}$ as before.

11. Alice and Bob locally perform privacy amplification as before.

Even though the values of the channel measurements h_{PA} and h_{PB} at Alice and Bob are not necessarily better correlated at the location of extrema in Alice's process, the *type* of extrema (i.e. minimum or maximum) at Alice and Bob at these locations are in much better agreement, which lowers ϵ.

Tradeoff with rate. List-encoding has a subtle tradeoff, which we will describe here but will defer detailed results to [9] due to space limitations. While in the basic version of the algorithm, Alice and Bob contribute 1 bit to $\underline{K}_a$ and $\underline{K}_b$ respectively per T_c, using extrema in the manner described above, reduces this to less than one bit per T_c. Therefore, the drop in ϵ comes as the cost of a drop in key bit-rate. However, it can be shown [9] that the overall effect of list-encoding on key bit-rate after reconciliation is that the improvement in ϵ due to list-encoding more than makes up for the drop in key bit-rate. Figure 5 shows, via simulation, the error bit-rate (ϵ) due to list-encoding and that due to the basic version of ProxiMate, for different values of the distance between Alice and Bob. Notice how list-encoding results in a significantly lower ϵ, but produces half the number of bits per T_c generated by the basic version.

3.9 Extracting bits from phase

The phase of the signal alone cannot be directly used for extracting bits at Alice and Bob because the measurement of phase also depends upon the phase of the local oscillators (LO) at both the transmitter and the receiver. Since it is not pragmatic to assume phase synchronization between the LOs of Alice and Bob, we use the *change in phase* or *phase differential* over a fixed time interval Δ, instead of the actual value of the phase to extract bits. Δ must be at least as large as the coherence time T_c so as to ensure that successive bits are independent. We use the differential phase values in place of the amplitude of $\underline{h}_{PU}$ in the quantization-based construction of ProxiMate and the rest of the algorithm remains the same.

4. EVALUATION

In this section, we describe our experimental setup, discuss our evaluation metrics and present results from our prototype implementation of ProxiMate. Specifically, we first experimentally evaluate what must the distance between Alice and Bob be in order to successfully generate a shared key, and how far Eve must be in order for this key to be secret. Then, we determine how many secret-bits per second Alice and Bob can generate and we explore the use of multiple sources in parallel in order to increase this rate. Finally, we experimentally study the case of a malicious public RF source.

4.1 Experimental Setup

We evaluated ProxiMate using an experimental prototype built on top of GNUradio [10]. GNUradio is an open source software toolkit, written in C++ and Python, that allows the creation of custom modular blocks for receiving, processing and transmitting RF signals. GNUradio interfaces via USB with a piece of hardware called the USRP (Universal Software Radio Peripheral), which acts as the bridge between the RF and software worlds. While software-based manipulation of radio signals at the level of detail needed by ProxiMate is not currently possible using commercial off-the-shelf wireless devices such as cellphones or Wifi cards, the rapid adoption of software-defined radio systems will likely make this possible in the near future. For any given scenario in our evaluations below, each user was mapped to a USRP radio front-end, connected to a laptop running GNUradio (see Figure 6 showing Alice and Bob). This setup allows each user to collect complex samples (i.e. each sample has amplitude and phase) of the RF frequency band it is tuned into.

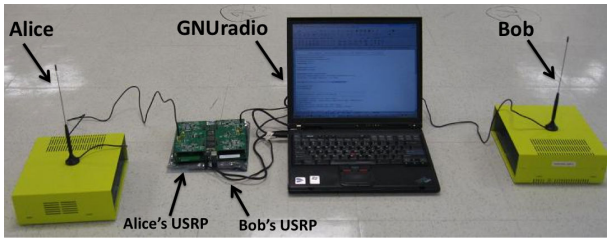


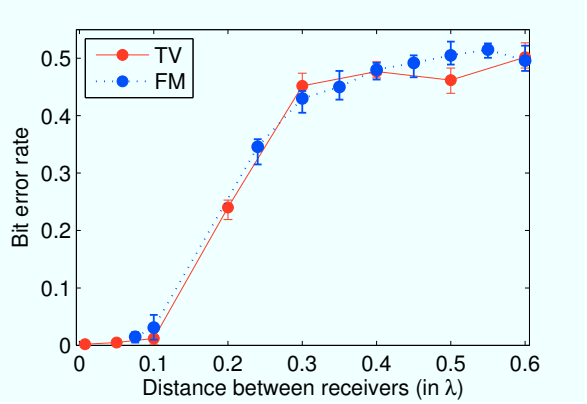
Figure 6: The GNUradio setup corresponding to two devices (Alice and Bob).

In evaluating ProxiMate we used real RF signals in the television broadcast band between 512 and 608 MHz (channels 21-36 in the US), and the FM-radio broadcast band between 88 and 108 MHz. Each broadcast TV channel contains an always-on constant-amplitude pilot tone, which we tune into, and track its amplitude to obtain channel measurements (i.e. $|\underline{h}_{PU}|$). For FM-radio signals, we make use of the fact that the FM radio signal has a very narrow spectrum (200 kHz) and is transmitted at constant power by the transmitter. Thus, we measure the total received power over the 200 KHz band for a given FM radio channel as our channel measurement $|\underline{h}_{PU}|$. For extracting bits using phase, we retain the phase of each sample, which is then used to compute the phase-differential across a time interval $\Delta > T_c$. Overall, we collected close to 1.5 hours of data from the TV bands and 1 hour of data from the FM bands, spread over 6 days of measurements.

4.2 Evaluation Metrics

In evaluating ProxiMate, we employ three metrics: key bit-rate, bit-error rate and mutual information.

Key bit-rate reflects the rate at which two devices can generate secret bits. For example, if a key bit-rate is 1 bit/sec, then a 128 bit key can be generated in 128 seconds. The higher the key bit-rate, the faster two devices can generate a key of a given length. Key bit-rate depends on the encoding scheme and the coherence time of the channel. Recall from Section 2.2 that the coherence time is an indicator of how often the channel



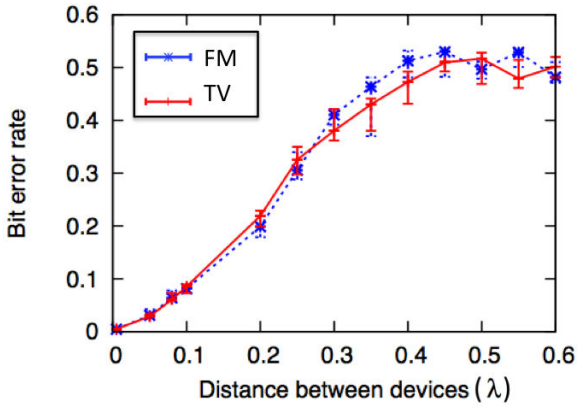


Figure 7: The bit-error rate ϵ between the bits obtained (prior to reconciliation) by two devices as the distance between them is varied, using (a) list-encoding with channel amplitudes, and (b) phase-differentials, for a TV signal (584.31 MHz) and an FM signal (88.7 MHz). Errorbars indicate the min and max values in each case across 20 experiments of 120 seconds each.

can provide fresh random secret bits. Once the coherence time of a channel is estimated (using the method in Section 3.3), we can calculate the key bit-rate.

Bit error rate (BER) is the number of bit-errors divided by the total number of generated bits or, in other words, the fraction of bits extracted by the two devices that differ. For instance, random selection of bits by two devices will result in a bit-error rate of ~ 0.5, since, on average, the devices will generate different bits roughly half the time. The lower the BER, the higher the likelihood that two devices will generate exactly the same key at the end of ProxiMate. Given a wireless channel, BER depends on the distance between two devices. The larger the distance between two devices, the more likely they are going to observe different variations of the signal, and, therefore, the more likely they are to generate different bits. Fixing the distance between two devices, fixes their BER.

Mutual information [11] between two random quantities X and Y, denoted $I(X;Y)$, is a non-negative measure of the amount of information in bits, that observing the value of either of the quantities, say Y, conveys about the value of the other, X. If mutual information between X and Y is zero, then observing the value of one does not convey any information about the other. We use mutual information to measure the similarity between observations of two legitimate devices and the dissimilarity between observation of an adversary and either of the legitimate devices. A mutual information of close to 0 between the channel measurements of an adversary and those of the legitimate devices would indicate adversary's inability to obtain any useful information about the legitimate device's measurements from its observations. Further, since bits are extracted from channel measurements (whether amplitude or phase), this would also necessarily indicate the adversary's inability to get any information about the bits generated by the legitimate devices. Such bits can therefore be considered secret bits. Mutual information between two quantities X and Y can be empirically computed [12] if a large number of data pairs $(X_i, Y_i), i = 1, \ldots$ is available.

4.3 What should be the distances between Alice, Bob and Eve?

In order to evaluate: (a) the ability of Alice and Bob to extract the same bits, as a function of the distance between them, and (b) the inability of Eve to extract the same bits as Alice or Bob, we run instances of ProxiMate for various distances between two USRP receivers. Figure 7(a) shows the bit-error rate ϵ between bits obtained using list-encoding at two receivers for the TV and FM bands as a function of physical distance between the receivers. Figure 7(b) shows the BER for bits obtained from phase alone, using the simpler quantization procedure described in Section 3.9. The BERs shown in Figure 7 are before the application of error correcting code.

The first insight from these plots is that the BER approaches 0.5 as the distance between the receivers approaches $\sim 0.4\lambda$. Second, in the case of amplitude, the BER between receivers that are within $d < 0.1\lambda$ is within ~ 0.03, but in the case of phase, the error performance is worse ~ 0.03 at $d = 0.05\lambda$. However, phase-based extraction has the additional quality of being resistant to a malicious public source, as we show in Section 4.7. Such low bit-error rates can be handled by error correcting codes with short block lengths. Our results suggest that adversary should be located farther than 0.4λ, and the legitimate devices should be within $d = 0.1\lambda$ or less for amplitude-based bit extraction and $d = 0.05\lambda$ for phase-based bit extraction.

In order to characterize the information available to Eve about the channel measurements of Alice and Bob,

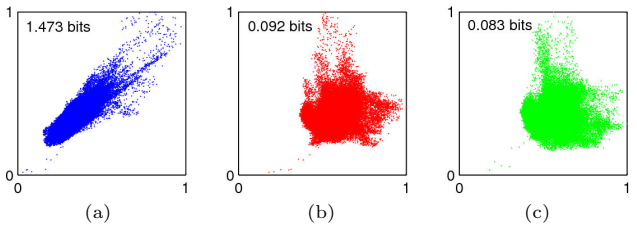


Figure 8: Scatterplots of the channel estimates of (a) Alice Vs. Bob, (b) Alice Vs. Eve and (c) Bob Vs. Eve, made using a three-user measurement on an FM radio channel. Eve is at a distance of $\lambda/2$ from both Alice & Bob, who are $d = 0.05\lambda$ apart. Each plot also shows an estimate of the mutual information per pair of channel estimates, computed using the algorithm in [12].

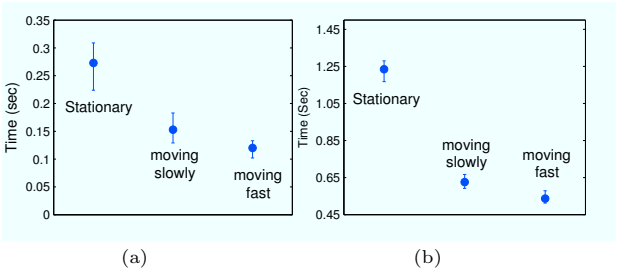


Figure 9: Average estimates of the coherence time made using equation 2 for (a) TV (584.31 MHz) and (b) FM (98.7 MHz) signals. Each estimate is computed using six one-minute traces. Error bars indicate the min and the max estimates in each case.

we conducted measurements with three GNUradio receivers representing Alice, Bob and Eve, simultaneously tuned in to the same FM-radio channel. Figure 8 shows pairwise scatterplots of the amplitude of channel measurements of each pair of users when Eve is at a distance of 0.5λ from both Alice and Bob and Alice and Bob are 0.05λ apart. The upper left corner of each plot also shows the empirically computed mutual information between the measurements of the two users being compared.

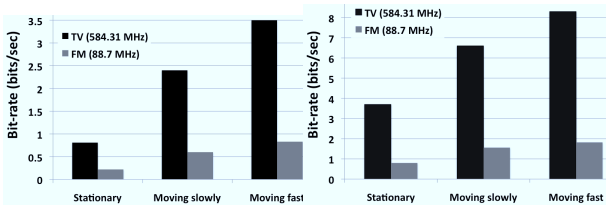
The plots and the mutual information computations show that Eve obtains only a negligible amount of information about Alice's and Bob's measurements. One channel measurement provides Bob with 1.473 bits of information about Alice's corresponding measurement. In contrast, one channel measurement provides Eve with only 0.092 bits of information about Alice's corresponding measurement and 0.083 bits of information about Bob's measurement. This means Alice and Bob have $15 \sim 16$ times more information about each other's measurements than what Eve knows about their measurement. The intuitive meaning of this result is that if Alice and Bob generate a 15 bit key, then the theoretical maximum that Eve can know about this key is 1 bit and no more. If Alice and Bob use this 15 bit key, it is exactly equivalent to Alice and Bob generating a *perfectly secret* 14 bit key about which Eve has completely no (0 bits) information.

4.4 How many secret bits can be extracted per unit time?

We would like Alice and Bob to be able to establish an identical secret key of a given size in the shortest possible time. The rate at which secret bits can be extracted is determined by 3 factors: **(i)** The rate at which the Peter-Alice and Peter-Bob channels vary with time. The faster these channels vary, the faster, fresh randomness becomes available to Alice and Bob. This notion is captured by the coherence time T_c, which we will evaluate in this section. **(ii)** The distance between Alice and Bob. The greater the distance between Alice and Bob, the greater the bit-error rate ϵ between the bits extracted by them. Making their secret bits identical by the process of reconciliation necessarily requires a reduction in the number of bits, because the use of an error-correcting code trades off one for the other. Therefore, in evaluating the effect of channels with different T_c's, we will use the same distance between Alice and Bob to keep the BER roughly the same. **(iii)** The number of separate sources of randomness (different public sources) that Alice and Bob are monitoring. As long as the channels between different public sources and Alice/Bob are independent, the sources can be treated as independent sources of randomness. We will explore multiple sources further in Section 4.5.

The coherence time T_c of a given wireless channel tells us how often a time-varying channel can provide fresh secret random bits. It is important to understand that T_c has to do with the multiple paths between a transmitter (in our case Peter) and the receiver (in our case Alice/Bob), i.e. the propagation environment, rather than the time-varying nature of the transmit signal $s(t)$ itself. It is possible to *create* an increased rate of random temporal variation in a wireless channel by physically moving or shaking the receivers Alice and Bob together. Moving Peter, or reflectors and scatterers in the propagation path would also have the same effect, but these entities are not under our control. Since a lower T_c enables a higher secret bit-rate, it is useful to investigate the extent to which physical movement changes the T_c.

To evaluate the effect of physical movement on T_c, we collected channel measurements at Alice and Bob for three cases: stationary, moving slowly, and moving fast. For the latter two cases, we physically waved Alice's and Bob's antennas together in the air, first slowly, and then vigorously. Figure 9 plots the coherence time for each of three cases for both TV and FM signals. The



(a) Amplitude (b) Phase

Figure 10: (a) Average secret-bits per second before performing reconciliation, using (a) list-encoding on amplitude measurements, and (b)phase-differentials. Alice and Bob were separated $d = 0.1\lambda$.

figure shows that shaking devices together vigorously can lower T_c by a factor or 2 to 2.5, and hence increase the rate at which secret bits can be extracted by the same factor. For example, a stationary Alice and Bob can extract a new bit from a TV signal once every 0.27 sec (1.25 sec for FM), but if shaken vigorously, they can get a new bit once per 0.12 sec (0.55 sec for FM).

Figure 10 plots the average key bit-rate at which secret bits were extracted from a single RF source for the three types of environments we have considered (stationary, moving slowly and moving fast). Alice and Bob were positioned 0.1λ apart, which translates to 5 cm for the TV signal and 34 cm for the FM radio signal. The figure reveals two important results: **(i)** The key bit-rate is roughly proportional to the frequency of the RF signal used, for both amplitude-based and phase-based extraction. **(ii)** Fast physical movement can improve the key bit-rate by a factor of $3 \sim 4$ compared to the stationary case.

The results in this plot are prior to reconciliation. For reconciliation, we used the $(23, 12)$ error-correcting Golay code because it is a linear short-blocklength code, with very simple linear encoding and decoding algorithms. The Golay code takes 23-bit segments of bits, $\underline{K}_a$ and $\underline{K}_b$, and reduces them to 12-bit segments (i.e. it reduces the bit-rates, shown in Figure 10, by a factor of $\frac{12}{23}$), reducing the bit-error rate ϵ drastically in the process. For instance, applying the $(23, 12)$ Golay code on the bits obtained from the TV signal when Alice and Bob were moved together fast results in a final secret bit rate of 1.8 secret bits per sec, with a bit-error rate of 2.4×10^{-3}. When the same code is applied to bits obtained from phase-differential measurements, the final bit-rate is 4.3 secret-bits/sec with a bit-error rate of 5×10^{-2}, which is a very high bit-error rate. However, if the code is applied two times, i.e. if another round of reconciliation is performed, then the rate is 2.2 secret-bits/sec, with an error rate of 10^{-5}. Note that each time reconciliation is performed, there is an accompanying privacy amplification step to eliminate

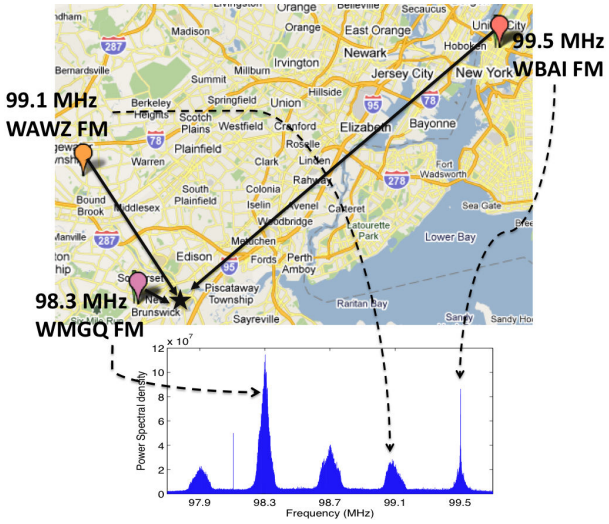


Figure 11: Locations of three FM radio stations we tested in the NY/NJ area, with respect to the location of Alice and Bob (marked by a star). The plot below shows five FM radio stations being monitored simultaneously by Alice's GNUradio, including the three FM-stations we used for evaluating independence.

information leaked out to Eve during reconciliation. As a result, one round of reconciliation and privacy amplification produces a reduction in the rate (secret-bits/sec) by a factor of $\frac{12}{23}$ (i.e. approximately half) when the (23,12)-Golay code is used.

4.5 Monitoring multiple sources

Our analysis thus far has only considered a single RF source. By simultaneously monitoring multiple RF sources, Alice and Bob can increase the amount of common randomness available to them per unit time. In fact, the number of secret bits per second achievable, scales linearly with the number of added sources, so long as the sources themselves are physically separated by a distance of at least $\lambda/2$. In such a scenario, Alice and Bob can treat each signal source as an *independent* source of common randomness and run parallel instances of our secret bit extraction algorithm. In our evaluation of multiple sources, we focus on FM signals, because the narrow bandwidth of each FM radio station allows us to use the GNUradio platform to receive multiple FM stations simultaneously. With a bandwidth of 200 KHz per FM station, the maximum number of FM sources that Alice and Bob can receive with a single GNUradio+USRP setup is 40. While not all channels in the FM band may carry an FM radio station with a strong signal, most regions in the country have an adequate FM radio coverage. Although we could observe signals on all 40 channels, we sampled the channel responses for five radio stations within 1 MHz of block

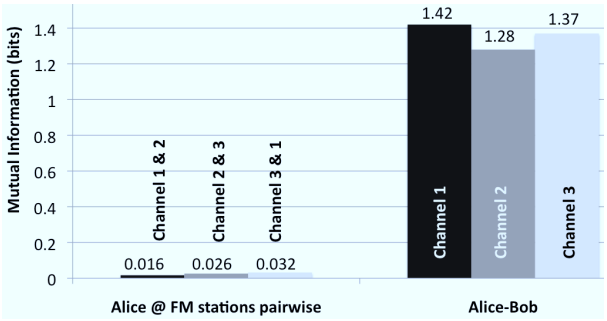


Figure 12: Left: Mutual Information between measurements of Alice on the three FM radio stations, taken two at a time. Right: Mutual information between channel measurements of Alice versus those of Bob on three different FM radio stations.

of spectrum (at 97.9, 98.3, 98.7, 99.1, and 99.5 MHz) for this study. Figure 11 shows the locations of three of these local FM stations from the NJ/NY area on a map, and the inset shows the five successive FM radio stations on the frequency dial within 1 MHz block of spectrum.

To determine the extent to which wireless channels corresponding to different FM-radio stations are independent of one another, we employ Mutual Information as a metric. Recall that if the mutual information $I(X;Y)$ between two quantities X and Y is ~ 0 then X and Y are nearly independent because observing one gives no information about the other. In particular, we evaluate the mutual information between Alice's channel measurements at a given instant of time, across three of these FM stations, taken two at a time. Figure 12 shows the mutual information values for Alice's channel measurements for the three stations, taken pairwise on the left. For comparison, on the right are comptuations of mutual information between the measurements of Alice and Bob for each of the three stations. Our results suggest that temporal variations in the channels from different FM radio stations are in fact fairly independent. We ran our list-encoding algorithm on the 5 FM channels taken as parallel, independent sources of randomness, and observed a combined average secret bit rate of 1.08 bps, with an average error rate of 0.039 when Alice and Bob were stationary and 0.1λ apart, and 4.27 bps with an average error rate of 0.042 when Alice and Bob were moved together in unison. We conclude that, when compared to the results of a single FM channel, sampling the abundance of public sources could significantly increase key bit-rate.

4.6 Putting it all together

A large number of factors affect the performance of ProxiMate: the distance between Alice and Bob, the wavelength of the public source, whether Alice and Bob are held stationary or moved, the length of the desired key, the amount of time available, the number of RF sources being monitored, and the probability that the final keys do not match in the end. In order to obtain a simplified view into ProxiMate's performance, we decided to hold all but two of the factors constant.

	Stationary	Moving slow	Moving fast
TV	33	10.2	7.5
FM	102.5	41.2	31.5

(a) Amplitude, Prob(key mismatch) $\leq 10^{-3}$

	Stationary	Moving slow	Moving fast
TV	16.4	7	6
FM	50.2	33	29

(b) Phase, Prob(key mismatch) $\leq 10^{-5}$

	Stationary	Moving slow	Moving fast
TV	11	4.2	3.3
FM	33.62	18.3	15

(c) Both Amplitude and Phase
$4.3 \times 10^{-4} \leq$ Prob(key mismatch)$\leq 6.2 \times 10^{-4}$

Table 2: Number of seconds needed to form a 128-bit key with 10 sources, when Alice and Bob are $d = 0.05\lambda$ apart.

Table 2 shows the amount of time needed by Alice and Bob to form a 128-bit key, when they are $d = 0.05\lambda$ apart, and use 10 sources in parallel, such that the probability that the final 128 bit keys (i.e. after reconciliation) extracted by them do not match is fixed (see Table 2). Table 2(a) is computed using the secret-bits/sec rates in Figure 10(a), the bit-error rates shown in Figure 7(a) and assuming the application of one round of reconciliation and privacy amplification using the (23,12)-Golay code. Table 2(b) for phase-based bit-extraction is computed using the secret-bits/sec rates in Figure 10(b), the bit-error rates shown in Figure 7(b) and assuming the application of *two* rounds of reconciliation and privacy amplification using the (23,12)-Golay code, because one round of reconciliation brings the key-mismatch probability to only $\sim 10^{-1}$ (bit-error rate of $\sim 10^{-3}$), which is too high to be useable. Finally, we combine the results of Table 2(a) and (b), in creating Table 2(c) assuming that Alice and Bob run parallel instances of ProxiMate on amplitude and phase. The key mismatch probability is computed by first computing the average bit-error rate, assuming that the 128-bit key consists of bits contributed by the amplitude and phase instances of ProxiMate in the proportion of their secret-bit rates. Instead of providing the key mismatch probability for each of the six cells in Table 2, we summarize its range below it.

Table 2(c) shows that the overall performance of ProxiMate is impressive, given the extremely low key mismatch probabilities. Further, the amount of time needed can simply be scaled down linearly if more RF sources are used. As a comparison, when both amplitude and phase are used by ProxiMate, we deduce from the results in Table 2(c), that the amount of time needed to establish a key the size of the Bluetooth 4-digit PIN (13.2 bits) is 0.34 sec with 10 TV sources shaken fast and 3.5 sec with 10 FM sources when Alice and Bob are stationary.

4.7 Coping with an adversarial source

In this section, we first show analytically and then validate empirically that our differential phase secret bit extraction can tolerate a powerful attacker who controls the transmission source.

4.7.1 Analytical Examination

A powerful attacker is able to manipulate the amplitude of the signal received by Alice and Bob in a controlled manner because the effect of the wireless channel on the transmit signal's amplitude is multiplicative – the amplitude of the received signal can be anywhere between 0 and a very large number. That is, by increasing or decreasing the amplitude of a transmitted signal, an adversary can increase or decrease the amplitude of the received signal, respectively. In contrast, the effect of the wireless channel on the transmit signal's phase is additive, and the phase of the resultant received signal must lie between 0 and 2π, as phase wraps around after 2π. In this case, the adversary has no control over changes in the phase of the received signal. We next illustrate this observation with a concrete example.

Suppose the adversary source transmits the signal $A(t)\cos(2\pi f_c t + \phi(t))$ instead of transmitting a signal $\cos(2\pi f_c t)$. That is, the adversary inserts the multiplicative amplitude term $A(t)$, and the additive phase term $\phi(t)$, both of which can be arbitrary functions of time. The receiver's channel estimate is then

$$h(t) = H(t)A(t) \cdot e^{j(\phi(t)+\theta(t))} \quad (3)$$

where $H(t)$ is the true channel amplitude and $\theta(t)$ is the true channel phase at time t if the transmitter is not adversarial. The adversary has inserted $A(t)e^{j\phi(t)}$ in front of the true channel state $H(t)e^{j\theta(t)}$, causing the channel to appear to have amplitude of $A(t)H(t)$ and a phase of $\phi(t) + \theta(t)$.

If Alice and Bob use only *amplitude* of their channels to extract bits, then an active Eve can vary $A(t)$ to influence the bits extracted by Alice and Bob in a manner chosen by her, and would therefore possess some information about the extracted key. However, the phase $\theta(t)$ added by the channel, and hence the phase of the signal received at Alice & Bob $\theta(t)+\phi(t)$ is not controllable by Eve, because phase is a number in the range 0 to 2π radians (i.e. 360 degrees), as phase wraps around 0 after going beyond 2π.

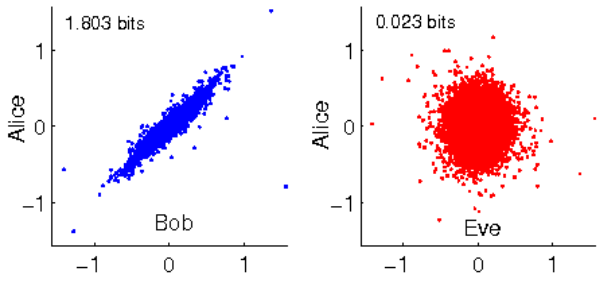


Figure 13: Differential phase measurements of Alice vs Bob (left), compared with those of Alice vs Eve (right). The axes represent the interval $[-\pi, \pi]$ radians. Each plot shows a mutual information estimate between the differential phases of the corresponding two users, computed using the algorithm in [12].

4.7.2 Empirical Validation

We evaluated our differential phase method for extracting secret bits by creating our own FM source (Eve) using the GNUradio as a transmitter this time. We generates a transmit signal at Eve's GNUradio with arbitrarily varying amplitude and phase. Alice and Bob were located in a different room $d = 0.04\lambda$ apart. Each user, Alice, Bob and Eve is made to collect phase-differential across a time interval Δ, which is conservatively chosen to be greater than the T_c, known from our previous evaluations (Section 4.4). We compared Alice's and Bob's phase-differentials against the phase differentials in Eve's transmit signal.

Figure 13(a) shows a scatterplot between Alice's and Bob's phase-differentials, and Figure 13(b) shows a scatterplot of Alice's phase-differentials versus the phase-differentials in Eve's transmit signal. Each plot also shows an estimate of the mutual information (in bits per phase-differential). These results indicate that phase-differential in the received signals at Alice and Bob are highly correlated, and can therefore be used to extract bits using the basic quantization procedure of ProxiMate. On the other hand, the phase-differentials in Eve's transmit signal are nearly independent of those of Alice or Bob. Therefore, Eve has no useful information about the change in phases at Alice and Bob, or to be precise, she has $\frac{1.803}{0.023} \approx 78$ (from the mutual information estimates in Figure 13) times less information about Alice's and Bob's measurements than they do about each others' measurements. Therefore if Alice and Bob generate a 78-bit key, it is equivalent to a 77-bit *secret* key.

5. DISCUSSION AND LESSONS LEARNED

In this section, we discuss the security aspects of

ProxiMate, its limitations, and lessons learned during our work.

Spatial security. Our experimental evaluation in Section 4.3 shows that if an adversary is more than 0.4λ away from both Alice and Bob, then the fraction of Eve's bits that differ from those of Alice or Bob, is very close to 0.5, which is the same as having Eve randomly guess Alice's and Bob's bits. For an FM and TV signals, 0.4λ corresponds to $\sim 1-1.3$ m and $20-50$ cm, respectively. Further, mutual information between the measurements of Eve and those of Alice or Bob, provide an *upper bound* on the amount of information available to Eve about Alice's & Bob's measurements. For example, from the mutual information numbers in Figure 8, we see that when Alice & Bob are 0.05λ apart and Eve is 0.5λ away from each other, the amount of information that Eve has about Alice's or Bob's measurements is more than an order of magnitude lower than the information Alice and Bob have about each other's observations. This implies that for each sequence of 10 bits generated by Alice & Bob, Eve can guess *at most* 1 bit.

Active adversary. An active Eve may transmit a signal in addition to Peter's public signal in an attempt to influence Alice's and Bob's estimates of their channels from Peter. This is a weaker form of the attack in which Eve has full control of the signal transmitted by Peter (Section 3.9). As before, using phase differentials in place of amplitudes of the channel measurements allows Alice and Bob to form a key without revealing any bits to Eve.

Adversarial channel-manipulations. If an attacker can manipulate the wireless channel by controlling scatterrers and reflectors in the path of the signal between Peter and Alice/Bob (see Figure 3), she can, in principle, influence the temporal variations observed by Alice and Bob. However, this is hard to accomplish in a *controlled* manner (i.e. to produce desired temporal variations in the channel measurements at Alice and Bob), especially when the public source is distant, as is likely to be the case with public broadcast sources such as FM and TV transmitters.

Adversarial channel-modeling. What if a passive Eve attempts to model the temporal variation in the channels between the public source and Alice/Bob by placing herself at the location of Alice/Bob, before or after the key extraction protocol? Such an adversary cannot get any useful measurements about the measurements made by Alice and Bob, provided that at least one coherence time interval (hundreds of milliseconds, in the case of TV signals) had elapsed between the instance of the key-generation protocol and the adversary arriving at location of Alice/Bob. This is because the wireless channel decorrelates over a coherence time interval, and channel models are only a statistical description of the channel.

Statistical randomness of secret bits. The bits in the secret key must be statistically independent in order to be suitable for use as a cryptographic key. This is ensured in list-encoding by having Alice wait for at least one coherence time interval between successive bit extractions. As explained in Section 2.2, the coherence time, is by definition, the amount of time needed for the channel state to become random after an observation, and is estimated in ProxiMate by Alice using an empirical estimate of the level crossing rate.

Access to low level samples. The current generation of radio device (e.g. cellphones, WiFi cards) do not provide access to sample level information in software. This presents a challenge to the immediate adoption of security mechanisms such as ProxiMate as well as earlier efforts [4, 13–15] that propose the use of radio signals for improving security primitives in software alone. It is of this reason that we have had to use the USRP/GNUradio platform for our experimental evaluation. However, we believe that with sufficiently beneficial use-cases, and increasing adoption of software-defined highly programmable radios, such functionality is likely to become available in future generations of devices.

Tradeoffs. In evaluating the system through real radio signal measurements, we learnt that ProxiMate has a number of tradeoffs. Longer wavelength (λ) signals make ProxiMate more useable than small wavelengths, considering the typical distances at which Alice and Bob might want to pair in typical pervasive computing scenarios. However, longer wavelengths also increase the coherence time of the channel, almost proportionally, thus requiring longer to form the same size key. Further, a longer wavelength also implies that the minimum distance for an adversary, *viz.* $\lambda/2$ is larger. Also, it is easier to monitor multiple FM-radio stations, than it is to monitor multiple TV signals because many FM-radio stations can fit into a given block of monitored spectrum due to the much narrower bandwidth of FM radio (200 KHz) as compared to TV signals (6 MHz).

6. RELATED WORK

Amigo [2] was the first to propose the use of common radio environment as a proof of physical proximity. Ensemble [16] extended Amigo by having the pairing devices transmit instead of receive packets, and showed how an ensemble of trusted body-worn devices can determine proximity of the pairing devices by monitoring their transmissions. Both these techniques used the Diffie-Hellman protocol to exchange secret keys and used RSSI values in WiFi packets to detect proximity. In contrast, ProxiMate doesn't require the Diffie-Hellman protocol, but derives the secret key directly from the radio environment. In addition, ProxiMate

works with any radio technology and it extracts bits from the raw signal, as opposed to the more limited averaged RSSI values. Finally, by using the differential phase, ProxiMate is secure against a powerful adversary who controls the transmission source.

Several prior works have shown how two devices can derive a secret key based on fluctuations of the radio environment between them [4, 13, 15]. These techniques focus on securing the wireless link and assume that the two communicating devices have been authenticated before, whereas ProxiMate provides both authentication and security. In addition, the reliance on random fluctuating channel *between* communicating devices makes these techniques unsuitable in a setting where two communicating devices are in close physical proximity. This is because the randomness in a wireless channel arises from RF propagation along multiple time-varying paths, and in a proximity setting, the direct path dominates, making the channel predictable, rather than random.

Recent work shows how shaking of devices equipped with accelerometers can be used to authenticate devices [17, 18]. Unfortunately, this require the presence of accelerometers on devices, they might be susceptible to attacks where an adversary replicates the shaking movement, and may not be appropriate in certain scenarios, such as an authentication between a public display and a laptop. Finally, one cannot rely solely on the short-range nature of technologies such as Bluetooth and near-field communication (NFC) [19] because they are susceptible to attacks by eavesdroppers with directional antennas [20, 21].

7. CONCLUDING REMARKS

This paper shows how wireless devices in proximity can pair autonomously by using their correlated channel measurements from ambient wireless signals to form a shared crypto-key. The speed with which users can securely pair depends on their physical separation, and on the rate of temporal variation in the chosen channels. Pairing can be accelerated by monitoring multiple sources simultaneously, or by manually shaking the legitimate devices together. We showed that using changes in phase, in place of amplitude variations proves to be robust against active attacks. Finally, ProxiMate can easily be extended to allow establishing a common secure association for more than two devices.

We believe that a number of useful optimizations of ProxiMate can improve its functionality and performance even further. For example, if Alice and Bob do not end up with identical bits after using an error correcting code, they do not need to discard them and start over, but can instead employ further rounds of error correction. Another direction for improvement would be to explore the use of non-binary quantization for improved extraction rates.

8. REFERENCES

[1] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[2] A. Varshavsky, A. Scannell, A. LaMarca, and E. de Lara, "Amigo: Proximity-based authentication of mobile devices," in *In Proceedings of UbiComp 2007: Ubiquitous Computing*, 2007, pp. 253–270.

[3] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of the 10th ACM conference on Computer and communications security*, ser. CCS '03. New York, NY, USA: ACM, 2003, pp. 52–61. [Online]. Available: http://doi.acm.org/10.1145/948109.948119

[4] 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 *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking (MobiCom '09)*, September 2009, pp. 321–332.

[5] A. Molisch, *Wireless Communications*. Wiley-IEEE Press, 2005.

[6] G. Brassard and L. Salvail, "Secret key reconciliation by public discussion," *Advances in Crytology Proc. - Eurocrypt '93, Lecture Notes in Computer Science*, vol. 765, pp. 410–423, 1994.

[7] C. H. Bennett, G. Brassard, and J.-M. Robert, "Privacy amplification by public discussion," *SIAM J. Comput.*, vol. 17, no. 2, pp. 210–229, 1988.

[8] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Prentice Hall PTR., 2001.

[9] "Technical report WINLAB, Rutgers University, strongly secure pairing of wireless devices in physical proximity, http://www.winlab.rutgers.edu/∼ suhas/CliqueTR2010.pdf."

[10] "http://gnuradio.org/trac."

[11] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley, 1991.

[12] Q. Wang, S. R. Kulkarni, and S. Verdu, "A nearest-neighbor approach to estimating divergence between continuous random vectors," in *Int. Symp. on Inform. Theory*, 2006, pp. 242–246.

[13] S. Mathur, W. Trappe, N. Mandayam, C. Ye, and A. Reznik, "Radio-telepathy: Extracting a secret key from an unauthenticated wireless channel," in *Proc. of the 14th annual conference on mobile computing and systems (MobiCom 2008)*, San Francisco, CA, Sept 2008.

[14] C. Ye, S. Mathur, A. Reznik, Y. Shah, W. Trappe, and N. Mandayam, "Information-theoretically secret key generation for fading wireless channels," *Accepted for publication at the IEEE Trans. on Information Forensics and Security, June 2010, preprint available at http://arxiv.org/abs/0910.5027*, 2010.

[15] B. Azimi-Sadjadi, A. Kiayias, A. Mercado, and B. Yener, "Robust key generation from signal envelopes in wireless networks," in *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, 2007, pp. 401–410.

[16] A. Kalamandeen, A. Scannell, E. de Lara, A. Sheth, and A. LaMarca, "Ensemble: cooperative proximity-based authentication," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. New York, NY, USA: ACM, 2010, pp. 331–344. [Online]. Available: http://doi.acm.org/10.1145/1814433.1814466

[17] 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, 2009.

[18] D. Bichleri, G. Stromberg, and M. Huemer, "Key generation based on acceleration data of shaking processes," *Ubiquitous Computing 2007*, pp. 304–317, 2007.

[19] "Near field communications forum, http://www.nfc-forum.org/."

[20] J. Wright, *Dispelling Common Bluetooth Misconceptions*. SANS Technology Institute Security Laboratory, 2007.

[21] E. Haselsteiner and K. Breitfuss, "Security in near field communication (NFC)," *Workshop on RFID Security RFIDSec*, 2006.