Radio-telepathy: Extracting a Secret Key from an Unauthenticated Wireless Channel

ABSTRACT
Securing communications requires the establishment of cryptographic keys, which is challenging in mobile scenarios where a key management infrastructure is not always present. In this paper, we address this challenge by presenting a protocol that allows two users to establish a common cryptographic key by exploiting the properties of the wireless channel: that the underlying channel response between any two parties is unique and that the channel response decorrelates rapidly in space. This established key can then be used to support security services such as encryption between two users. Our algorithm uses level-crossing and quantization of extract bits from correlated stochastic processes. The resulting protocol resists cryptanalysis by an eavesdropping adversary and a spoofing attack by an active adversary without requiring an authenticated channel, as is typically assumed in prior information-theoretic key establishment schemes. We evaluate our algorithm through theoretical and numerical studies, and provide validation through two complementary experimental studies. First, we use an 802.11 development platform with customized logic that extracts raw channel impulse response data from the preamble of a format-compliant 802.11a packet. We show that it is possible to practically achieve secret key establishment rates of 1 bit/sec in a real, indoor wireless environment. Next, to illustrate the generality of our method, we show that our approach is equally applicable to per-packet coarse signal strength measurements using off-the-shelf 802.11 hardware.

1. INTRODUCTION
Many of the risks associated with securing wireless systems stem from challenges associated with operating in a mobile environment, such as the lack of a guaranteed infrastructure or the ease with which entities can eavesdrop on communications. Traditional network security mechanisms rely upon cryptographic keys to support confidentiality and authentication services. However, in a dynamic mobile environment, with peer-to-peer associations being formed on-the-fly between mobile entities, it is difficult to ensure availability of a certificate authority or a key management center to support security needs. Since such scenarios are likely to become increasingly prevalent in the future, it is necessary to have alternative methods for establishing keys between wireless peers without resorting to a fixed infrastructure.

In this paper, we explore an alternative for building cryptographic services by exploiting an untapped resource - the wireless channel itself. The wireless medium can serve as a mechanism to share secrets between two wireless entities by exploiting the variability and dimensionality associated with wireless propagation. The specificity of the radio channel between two wireless devices, and its rapid decorrelation with distance, provide a basis for the creation of shared secret information, such as cryptographic keys, even in the presence of an eavesdropper. In typical multipath environments (see Figure 1), the wireless channel between two users, Alice and Bob, produces a random, time-varying, stochastic mapping between the transmitted and received signals. This mapping changes with time in a manner that is location-specific and reciprocal, i.e., the mapping is the same whether Alice is the transmitter with Bob as the receiver or vice-versa. The time-varying mapping, commonly termed fading, decorrelates completely over distances of the order of half a wavelength, \(\lambda\). Thus, an adversary, Eve, who is more than \(\lambda/2\) away from both Alice and Bob, experiences fading channels to Alice and to Bob that are statistically independent of the fading channel between Alice and Bob. These two properties allow us to generate a common, secret cryptographic key at Alice and Bob such that Eve gets no information about the generated key. For example, at 2.4 GHz, we only require that Eve be roughly \(\lambda/2 = 6.25\) cm away from Alice and Bob to ensure that she gets no useful information. Thus, while fading is typically considered harmful, it is profitably exploited by our technique to extract perfectly secret bits without leaking information to an adversary.

The extraction of secret bits from the wireless channel can be viewed as a 'black-box' that can be advantageously utilized in various ways putting to good use information that is already available from the channel. For example, in the current 802.11i system, session keys for communication between a station and an AP are derived by hashing together authentication credentials and randomly selected nonces which are exchanged by the two parties in the clear. This effectively ties the confidentiality of all future messages sent over the air to the authentication credentials. Therefore, if these credentials are compromised at some time in the future, an adversary will be able to derive the session keys and decrypt all past encrypted messages. If instead, the nonces are derived from the unique channel between the two users, making them information-theoretically secure, a passive adversary has no means to derive the session keys even if it learns the authentication credentials [1]. Similarly, the session keys can be updated by employing secret bits derived from the channel to deliver the new keys, instead of relying on previously existing keys [1]. This ensures that the confi-
dentiality of each new session is protected independently of earlier sessions.

Yet another vulnerability in 802.11i stems from the fact that during the establishment of a secure link between a station and an AP, all messages exchanged over the air, including management frames, are sent unencrypted till the point at which both parties have obtained the session key (called the temporal key or TK in 802.11 jargon) and are therefore susceptible to eavesdropping by unauthorized users sharing the wireless medium as well as to spoofing by other users. While the 802.11w amendment seeks to protect some management frames from such attacks, it too fails to protect messages exchanged before the establishment of TKs. The problem is that securing the initial exchanges between the parties requires them to share a key which is not established until later. Our key extraction mechanism provides a natural solution [2] by allowing the parties to generate a temporary key that protects the interim exchanges before the formal keys are in place.

Ad-hoc networks present another avenue where our technique can be useful. Alice may not care to establish Bob’s identity if she merely wishes to employ his services to forward her data over another hop but may like to establish a secure link layer with him nevertheless so as to add link-level encryption to her data.

Prior work in information theory has noted the potential of using the wireless channel for generating shared secret bits, but most of this work has been aimed at computing theoretical limits and neither provides practical algorithms, nor a demonstrable and quantifiable impact on security. Our contribution in this paper may be summarized as follows:

1. We translate prior information-theoretic ideas into a practical protocol applied to wireless channels;

2. We build a new algorithm for key extraction that, unlike prior schemes, does not require an authenticated channel, and we evaluate its performance for typical fading channels;

3. We validate our algorithm using channel impulse responses measured using the 802.11a packet preamble on a customized FPGA-based 802.11 development platform and a second study that uses only coarse per-packet RSSI information readily available to off-the-shelf 802.11 platforms.

We emphasize that existing mobile radio platforms already provide the information we need, but such data are normally discarded after physical layer processing and can be profitably exploited to benefit security services. It is important to recognize that the new approach we present is intended to augment, rather than to replace existing cryptographic security mechanisms. It provides a new approach to establishing keys that is useful when there is no key management infrastructure (e.g. in peer-to-peer wireless systems).

In Section 2 we summarize the related work, in Section 3 we describe our system model and the design issues relevant to our problem, in Section 4 we describe our key-extraction algorithm in detail, in Section 5 we evaluate the performance of our algorithm and in Section 6 we present two experimental studies that validate our algorithm on 802.11a hardware. We conclude in Section 7 with a discussion on the practical implications of our proposed system and the broader impact it may have on securing wireless systems.

2. RELATED WORK

Figure 1: The multipath fading for a signal from Alice to Bob is different from that for the signal reaching Eve.

An extensive body of information-theoretic work during the past several decades has explored the use of information from the physical layer in deriving security benefits. In [3,4], the authors introduced the problem of generating identical bits based on correlated information available to two users such that a third eavesdropping user does not learn anything about the generated key. They showed, provided Alice and Bob already share an authenticated public channel, that it is possible to generate identical keys at the two users. The standard method for generating secret keys at Alice and Bob under this assumption consists of three basic steps and has been utilized by a number of proposed systems [5–7]. In advantage distillation [3,8], the legitimate users, Alice and Bob obtain correlated information while Eve is allowed to eavesdrop, so that Alice & Bob share greater information1 than that shared between Alice & Eve or Bob & Eve. Alice and Bob then convert their information into bits. In the information reconciliation stage [6], Alice and Bob exchange error-correcting messages over an authenticated public channel that allow them to agree on an identical string of bits. However, the publicly exchanged messages reveal a certain amount of information about the bit strings to Eve. In privacy amplification [10], Alice and Bob diminish the partial information revealed to Eve by systematically discarding some of their common bits. Efficient protocols have since been designed [6,11] to allow key generation without leaking out information to an eavesdropping adversary.

However, a central assumption in this entire body of work is that Alice and Bob have an authenticated channel available to them even before key generation begins. This is an unrealistic assumption in practice because the availability of an authenticated channel implies that Alice and Bob already share a secret key to begin with! Therefore, the purpose of generating a common secret key is defeated.

In [13], Maurer and Wolf showed that secret key extraction without an authenticated channel is possible only if Eve cannot possibly transmit a signal to Bob that is statistically indistinguishable from signals coming from Alice (and vice-versa). This provides an important insight that has not been translated into a practical algorithm. Our work is the first to build upon this result: we use use the wireless channel to guarantee that Eve does not possess the required information to prevent key generation.

More recently, [14] examined PHY-layer based authentication and confidentiality in wireless systems. The work in [15,16] looked at authentication using channel signatures between the transmitter and receiver(s). Our work is perhaps most closely related to the work of [17], which proposes

1In information-theoretic terms, the amount of information between two observations $X$ and $Y$ is measured by the mutual information $I(X;Y)$ between them [9].

2Much of this work was done in the context of quantum key distribution. For a more complete bibliography on quantum key distribution see [12].
a scheme for generating secret bits from correlated observations of deep fades by two users communicating via a TDD link. This work focuses on the theoretical construction for extracting randomness through use of universal hash families. However, they do not demonstrate or evaluate the amenability of the wireless channel to detection of deep fades by both users using TDD to the degree of precision required by their scheme. Their work does not provide a quantification of the secret key rate versus parameters associated with the underlying fading process or parameters involved in their algorithm. Additionally, we note that their method focuses primarily on a passive adversary. The reliance on deep fades can be easily exploited by an active adversary that produces greater interference power at one legitimate user than the other so that a deep fade for one user may not be a deep fade for the other. In [18], a method exploiting channel reciprocity using ultra-wideband (UWB) channels to generate secret bits was presented. In [19], specialized electronically steerable antennas were proposed for use in generating key bits by exploiting channel reciprocity. The methods in [17–19] all rely on conventional reconciliation for correcting bit-errors, and thus require an authenticated channel. In [20,21], a method for secret key generation based on phase reciprocity of frequency selective fading channels was proposed. While this is attractive, it is difficult to implement or evaluate in practice because accurate phase information is difficult to harvest from existing hardware platforms.

In contrast to prior work, the algorithm we propose transcends the requirement of an authenticated channel, does not require specialized hardware and is not limited to UWB channels. We provide both a fundamental analysis as well as a real-world experimental implementation of our algorithm and show that existing mobile platforms already provide sufficient information for producing secret bits using our algorithm. We evaluate the randomness of the bit-sequences produced by our algorithm, a generally overlooked aspect in prior work on secret key generation, and show that they are suitable for use as cryptographic keys. Lastly we note that our technique may be compared with classical key establishment techniques such as Diffie-Hellman, which also use message exchanges to establish keys. However, classical cryptographic key establishment techniques rely upon the (unproven) arguments of computational complexity such as the hardness of inverting discrete logarithms or factoring a product of large prime numbers, while our algorithm provides information-theoretic security, which does not rely on these computational hardness assumptions and builds a practical methods to achieve this type of security. In this way we provide an analog of quantum cryptography for wireless networks.

3. SYSTEM MODEL & DESIGN ISSUES

The crucial insight that allows the wireless channel to be amenable for generating a secret key is that the received signal at the receiver is modified by the channel in a manner that is unique to the transmitter-receiver pair. It is the modification of the transmitted signal by the wireless multipath channel that introduces information into the received signal from which we can extract secret bits. This distortion captures information that depends critically upon the location of the transmitter, the receiver, and the placement of scattering surfaces in the wireless environment. Typically, such distortion is estimated at the physical layer of the receiver and the associated distortion information dealt with to ensure reliable physical layer decoding of the communication signal. However, since this information is always present and uniquely corresponds to the transmitter-receiver pair, it also provides our transmitter (Alice) and receiver (Bob) a means to privately establish secret bits associated with this distortion. In this section, we focus on the challenges of using the stochastic nature of the wireless channel to secretly establish bits. We break down our discussion to include a description of: (1) the underlying channel model associated with multipath fading; (2) the tools needed to obtain bits from the channel response; and (3) the design goals that need to be addressed in order to reliably establish these bits. To assist the reader, we provide notation in Table 1.

Table 1: A summary of the notation used

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h$</td>
<td>Stochastic channel parameter of interest</td>
</tr>
<tr>
<td>$h(t)$</td>
<td>Value of the stochastic process $h$ at time $t$</td>
</tr>
<tr>
<td>$s(t)$</td>
<td>Probe signal transmitted to estimate $h(t)$</td>
</tr>
<tr>
<td>$f$</td>
<td>Maximum Doppler frequency (Hz)</td>
</tr>
<tr>
<td>$r$</td>
<td>Rate at which each user sends probes (Hz)</td>
</tr>
<tr>
<td>$q_-$</td>
<td>Quantizer boundaries (Upper and lower resp.)</td>
</tr>
<tr>
<td>$m$</td>
<td>Req'd. no. of estimates in a excursion</td>
</tr>
<tr>
<td>$n$</td>
<td>Length of key in bits</td>
</tr>
<tr>
<td>$p_b$</td>
<td>Probability of a bit error</td>
</tr>
<tr>
<td>$p_m$</td>
<td>Probability of key mismatch $= 1 - (1 - p_b)^m$</td>
</tr>
</tbody>
</table>

3.1 Channel model

Let $h(t)$ be a stochastic process corresponding to a time-varying parameter that describes the wireless channel between Alice and Bob. Although there are many choices for $h(t)$, for our discussion, we shall assume that $h(t)$ is the magnitude of the transfer function of the multipath fading channel evaluated at a fixed test frequency, $f_0$. Implicit in this formulation is the observation that the system transfer function of the channel is the same in the Alice→Bob direction as in the Bob→Alice direction at a given instant of time. This follows from reciprocity, which is a fundamental property of electromagnetic wave propagation [22, 23] in a medium and must not be confused with additive noise or interference, which may be different for different receivers. To distinguish between the channel parameter of interest, and its value at a given instant of time, we denote the parameter by $h$ and refer to its value as $h(t)$. To estimate the parameter $h$, Alice and Bob must
The bit strings they generate must be suitable for use as meaningful estimates of the Alice-Bob channel, \( h \). Therefore, despite possessing knowledge of the probe signals sent by each of the users, the signals received by each party are completely different: the probe signals sent by each of the users, the signals received by Alice and Bob due to successive probes may be written as

\[
\begin{align*}
r_a(t_1) &= s(t_1)h_a(t_1) + n_a(t_1) \quad (1) \\
r_a(t_2) &= s(t_2)h_a(t_2) + n_a(t_2), \quad (2)
\end{align*}
\]

where \( s(t) \) is the known probe signal, \( n_a \) \& \( n_b \) are the independent noise processes at Alice and Bob and \( t_1 \& t_2 \) are the time instants at which successive probes are received by Alice and by Bob respectively. Using the received signal, Alice and Bob, each compute (noisy) estimates of \( h \):

\[
\begin{align*}
\hat{h}_a(t_1) &= h(t_1) + z_a(t_1) \\
\hat{h}_a(t_2) &= h(t_2) + z_a(t_2),
\end{align*}
\]

where \( z_a \) \& \( z_b \) represent the noise terms due to \( n_a \) and \( n_b \) respectively after having been processed by the function that estimates \( h \). We refer the reader to [24] for designing good estimator functions for \( h \). The estimates \( \hat{h}_a \) and \( \hat{h}_b \) are in all likelihood unequal, due in part to the independent noise terms and in part to the time lag \( \tau \), however they can be highly correlated if Alice and Bob send probes to one another at a fast enough \(^3\) rate, i.e. if \( \tau \approx t_2 - t_1 \) is small. By repeatedly sending probes in an alternating manner over the time-varying channel, Alice and Bob can generate a sequence of \( n \) estimates \( \hat{h}_a = \{\hat{h}_a[1], \hat{h}_a[2], \ldots, \hat{h}_a[n]\} \) and \( \hat{h}_b = \{\hat{h}_b[1], \hat{h}_b[2], \ldots, \hat{h}_b[n]\} \) respectively, that are highly correlated, as in Figure 2. Although Eve can overhear the probe signals sent by each of the users, the signals received by Eve are completely different:

\[
\begin{align*}
r_e(t_1) &= s(t_1)h_e(t_1) + n_e(t_1) \quad (5) \\
r_e(t_2) &= s(t_2)h_e(t_2) + n_e(t_2), \quad (6)
\end{align*}
\]

where \( h_e \) and \( h_a \) denote the channel between Bob \& Eve and between Alice \& Eve respectively and \( n_e \) is the noise added at Eve. If Eve is more than \( \lambda/2 \) away from Alice and Bob, then \( h_e \) \& \( h_a \) are uncorrelated with \( h \) [25]. Therefore, despite possessing knowledge of the probe signal \( s(t) \), Eve cannot use her received signals to compute meaningful estimates of the Alice-Bob channel, \( h \).

### 3.2 Converting the channel to bits

Alice and Bob must translate their respective sequences of channel estimates to extract identical bit-strings. Further, the bit strings they generate must be suitable for use as cryptographic keys. This means that they must be:

1. **Suitably long:** Keys of length 128 to 512 bits are commonly used in symmetric encryption algorithms. So they should be able to generate at least these many bits in a reasonable amount of time.

2. **Statistically random:** The bits should be random with equal probability of a ‘0’ and a ‘1’. Also, the bit-sequences must not suffer from statistical defects that could be exploited by an attacker.

\(^3\)Fast enough’ here is in relation to the coherence time of the channel, which is inversely proportional to the maximum Doppler frequency \( f_d \).

The second requirement guarantees that the generated key has desirable security properties. That is, an \( N \)-bit key must really provide \( N \) bits of uncertainty to an adversary who only knows the key generation algorithm and nothing else.

We now briefly describe how to obtain bits from the channel estimates \( \hat{h}_a \) and \( \hat{h}_b \) to provide the intuition behind our algorithm, while postponing a formal description to Section 4. The sequence of channel estimates \( \hat{h}_a \) and \( \hat{h}_b \) are sequences of random variables, drawn from an underlying probability distribution that characterizes the random channel parameter \( h \). We assume, for the sake of our discussion, that \( h(t) \) is a Gaussian random variable and the underlying stochastic process \( h \) is a stationary Gaussian process. A Gaussian distribution for \( h \) may be obtained, for example, by taking \( h \) to be the magnitude of the in-phase component of a Rayleigh fading process between Alice and Bob [22]. It must be noted that the assumption of a Gaussian distribution on \( h \) is for ease of discussion and our algorithm is equally valid in the general case.

Since the channel estimates computed by Alice and Bob are continuous random variables, it is necessary to quantize their estimates using a quantizer \( Q(\cdot) \) to obtain bits. However, a straightforward quantization of the vectors \( \hat{h}_a \) and \( \hat{h}_b \) is not sufficient because it does not guarantee that an identical sequence of bits will be generated at the two users. In our scheme, Alice and Bob use the channel statistics to determine scalars, \( q_+ \) and \( q_- \) that serve as reference levels for the quantizer \( Q(\cdot) \) as follows:

\[
Q(x) = \begin{cases} 
1 & \text{if } x > q_+ \\
0 & \text{if } x < q_- 
\end{cases}
\]

Alice then parses through her channel estimates \( \hat{h}_a \) to determine the locations of excursions\(^4\) of her channel estimates above \( q_+ \) or below \( q_- \) that are of a duration \( \geq m \) estimates, i.e., \( m \) successive channel estimates in \( \hat{h}_a \) are \( > q_+ \) or \( < q_- \), where \( m \) is a protocol parameter. She sends Bob a message over the public channel containing the locations of \( k \) such excursions in the form of an array of indexes \( L = \{l_1, l_2, \ldots, l_k\} \). Bob then checks his own sequence \( \hat{h}_b \) at the locations specified by the indexes in \( L \) to determine whether it contains an excursion above \( q_- \) or below \( q_+ \) for a duration greater than or equal to \( m - 1 \) samples, i.e., whether \( \hat{h}_b(l_i) \) is \( > q_+ \) or \( < q_- \) for a duration that spans \( m - 1 \) or more estimates, for each \( i = 1, \ldots, k \). Bob identifies ‘good’ indexes by finding all index values \( i \) in \( L \) that produce such an excursion in \( \hat{h}_b \). He places these indexes into an array \( L' \) to be sent to Alice publicly. Indexes in \( L \) but not in \( L' \) are dropped from consideration by each party. The indexes in \( L \) are used by each user to compute a sequence of bits by performing the quantization: \( Q(\hat{h}_a(L)) \) and \( Q(\hat{h}_b(L)) \). If the bit-vectors \( Q(\hat{h}_a(L)) \) and \( Q(\hat{h}_b(L)) \) are equal, then Alice and Bob succeed in generating \( |L| \) identical bits. We show later that provided the levels \( q_+ \) \& \( q_- \) and the parameter \( m \) are properly chosen, the bit sequences generated by the two users are equal with very high probability. A variation of the protocol that copes with the spoofing attack is detailed in Section 4.1.

### 3.3 Design goals

An important quantity of interest will be the rate of generation of secret bits, expressed in secret-bits per second or ‘s-bits/sec’. Naturally, it is desirable that Alice and Bob achieve a high secret-bit rate. According to 802.1x recom-

\(^4\)A stochastic process is in an excursion when it lies above or below a specified level for a given duration of time.
mendations, it is generally desirable for master keys to be refreshed at intervals of one hour [26]. Using these examples and AES key sizes of 128 bits as a guideline, a conservative key rate of roughly 0.1 bits per second is needed, though it is desirable to achieve higher secrecy rates. However, we are especially wary of bit errors. If the sequence \( Q(h_a(L)) \) is different from \( Q(h_b(L)) \) even by a single bit, then the two bit-strings cannot be used as cryptographic keys and consequently the entire batch of bits must be discarded. Therefore, we would like the bit error probability \( p_e \), to be extremely low, so that the probability \( p_n \) that the keys generated by the two users do not match is acceptably small. For example, in order to have a key-mismatch probability \( p_n = 10^{-6} \), assuming keys of length 128 bits, we must target a bit-error probability of \( p_e \) where
\[
p_e = 1 - (1 - p_e)^{128}, \tag{8}
\]
which gives \( p_e \approx 10^{-8} \). A bit-error is defined as the event that Alice and Bob agree to use a certain index \( l \), contained in the list \( L \) for generating a bit, but they end up generating different bits, i.e. \( h_a(l) \) and \( h_b(l) \) both lie in excursions at the index \( l \), but the excursions are of opposite types.

The rate at which secret bits can be extracted from the channel is fundamentally limited by the rate of time-variation in the channel. We quantify this variation by the maximum Doppler frequency, \( f_d \). In a fading channel, \( f_d \) determines the both the rate at which the channel varies and the magnitude of the swings produced. A simple measure of the maximum Doppler frequency in a given wireless environment is given by \( f_d = \frac{\nu}{\lambda} \), where \( \nu \) is a measure of the effects of user mobility and the dynamic environment around the users, expressed in meters/sec and \( \lambda \) is the wavelength of the carrier wave. In our case \( \lambda = \frac{c}{v} \), where \( c \) is the speed of light. It can be seen that increasing the value \( m \) or the magnitudes of the quantizer boundaries \( q_+ \) & \( q_- \) would not only result in a lower rate but also a lower probability of error. Intuitively, this is because larger magnitudes of \( q_+ \) & \( q_- \), or a larger value of \( m \) makes it less likely that Alice’s and Bob’s channel estimates lie in opposite type of excursions, thereby reducing the error rate. However both types of excursions also become less frequent, thereby decreasing the number of secret bits that can be generated per second. Thus, there is a tradeoff between rate and probability of error, and the parameters \( q_+, q_- \), and \( m \) provide convenient controls to select suitable operating points over this tradeoff. Apart from generating bits at a reasonable rate and achieving robustness to errors, we also require the bits to be random and free from statistical defects. This aspect is discussed in Section 5.3.

Finally, the correlated information obtained by Alice and Bob can be utilized to build a secret key in a number of different ways and it is important to make sure the method employed does not allow Eve to infer any useful information. An alternative bit extraction scheme is to have each user estimate a statistical measure of the channel (e.g. the mean signal-strength, or variance in the estimates) using \( \hat{h}_a \) and \( \hat{h}_b \) respectively. If the channel is stationary\(^5\), then their respective statistical measures would each converge to the true value with time. In this way, Alice and Bob will each possess knowledge about a numerical quantity, without having actually sent messages over the air containing this quantity. They could then quantify their estimates of the statistical measure to generate bits. However, the trouble with using a statistical measure is that knowledge of the locations of Alice and Bob and their environment may allow Eve to infer the statistics of the channel between them. Indeed publicly available tools, such as the WISE ray-tracer [27], make it easy to predict the signal statistics at a receiver given the knowledge of the locations of the transmitter and receiver and the building’s layout. Thus, it is important to recognize that using a statistical measure for key generation can be perilous. Our algorithm avoids statistical measures from influencing the secret-key generation process by relying on specific instantiations of the fading process.

4. LEVEL-CROSSING ALGORITHM

We now detail our key-extraction algorithm based on level crossings and quantization of Alice’s and Bob’s channel estimates. It is assumed that when the algorithm is run, Alice and Bob have collected a sufficiently large number of channel estimates \( h_a \) and \( h_b \), by alternately probing the channel between themselves. Further, it is assumed that the vectors \( \hat{h}_a \) and \( \hat{h}_b \) are of equal length and their \( j \)th elements \( h_a(j) \) and \( h_b(j) \) correspond to successive probes sent by Bob and Alice respectively, for each \( j = 1, \ldots, \text{length}(\hat{h}_a) \). Algorithm 1 describes the procedure and consists of the following steps:

1. Alice parses the vector \( \hat{h}_b \), containing her channel estimates to find instances where \( m \) or more successive estimates lie in an excursion above \( q_+ \), or below \( q_- \).
2. Alice selects a random subset of the excursions found in step 1 and for each selected excursion, she sends Bob the index of the channel estimate lying in the center of the excursion, as a list \( L \). Therefore, if \( h_a(i) > q_+ \) or \( q_- < h_b(i) \) for some \( i = \text{k}_{\text{start}}, \ldots, \text{k}_{\text{end}} \), then she sends Bob the index \( L_{\text{center}} = \left\lfloor \frac{\text{k}_{\text{start}} + \text{k}_{\text{end}}}{2} \right\rfloor \).
3. For each index received from Alice, Bob checks whether his own vector of channel estimates \( \hat{h}_b \) contains at least \( m - 1 \) channel estimates centered around that index in an excursion above \( q_+ \) or below \( q_- \), i.e. whether \( h_b(i) > q_+ \) or \( q_- < h_b(i) \) for each index \( \{l - \left\lfloor \frac{\text{k}}{2} \right\rfloor, \ldots, l + \left\lfloor \frac{\text{k}}{2} \right\rfloor \} \), for each \( l \in L \).
4. For some of the indexes in \( L \), Bob’s channel estimates do not lie in either excursion. Bob makes a list \( \tilde{L} \) of all indexes that lie in excursions and sends it to Alice.
5. Bob and Alice compute \( Q(\hat{h}_a) \) and \( Q(\hat{h}_b) \) respectively at each index in \( \tilde{L} \), thus generating a sequence of bits. Since Eve’s observations from the channel probing stage do not provide her with any useful information about \( \hat{h}_a \) and \( \hat{h}_b \), the messages \( L \) and \( \tilde{L} \) exchanged between Alice and Bob do not provide her with any useful information either. This is because they contain time indexes only whereas the generated bits depend upon the values of the channel estimates at those indexes. Further, the selection of a random subset in Step 2 from the set of eligible excursions found in Step 1, guarantees that Eve cannot use \( L \) and \( \tilde{L} \) to infer the values of the channel estimates of Alice or Bob at those time indexes.

4.1 Preventing a spoofing attack

Since Alice and Bob do not share an authenticated channel, Eve can impersonate Alice in Step 2, or Bob in Step 4 above, allowing for a spoofing attack. If successful, such an attack would allow Eve to insert her own ‘fake’ \( \hat{L} \) or \( \tilde{L} \) messages, thereby spoofing a legitimate user to the other and effectively disrupting the protocol without revealing her presence to the two users.

\(^5\)By stationary, we mean a channel whose stochastic characteristics do not vary significantly over the time duration of the key extraction protocol.
Our protocol provides a mechanism to detect the adversary in each of the two cases above. We first focus on the possibility of Eve inserting a fake $L$-message. Since Eve has no information about the locations of channel excursions apart from $L$, she can only make random guesses about which indexes to place into a fake $L$-message to Bob (apart from the ones Eve learns from $L$). For each guess, she has a very low probability of choosing an index that lies in an excursion spanning $(m - 1)$ or more estimates at Bob. Of these, the indexes that do not lie in an excursion in $\hat{h}_L$ are discarded by Bob while those that do happen to lie in an excursion are considered eligible for quantization and placed into the $L$-message sent to Alice. Therefore, an unsuccessful guess provides no benefit to Eve, while a successful guess, albeit improbable, causes $L$ to contain an index that was not present in $L$, thereby alerting Alice. Thus, Eve must modify the message $\tilde{L}$ by deleting this index before it reaches Alice.

Our protocol can be made to resist modification of the $L$-message as follows. Once Bob computes the list $\tilde{L}$, he can proceed with the quantization stage and compute $K_\tilde{b} = Q(\hat{h}_L(\tilde{L}))$ of length $N$ bits. The first $N_{au}$ bits of $K_\tilde{b}$ are kept aside for use as an authentication key $K_{au} = K_{\tilde{b}}(1, \ldots, N_{au})$, to prevent a spoofing attack and the remaining $N - N_{au}$ bits are kept as the extracted secret key $K_\tilde{b} = K_{\tilde{b}}(N_{au} + 1, \ldots, N)$. Bob computes a message authentication code for $L$ using $K_{au}$ to obtain $MAC\left(K_{au}, \tilde{L}\right)$. Finally, he sends the package $\{\tilde{L}, MAC\left(K_{au}, \tilde{L}\right)\}$ to Alice. Upon receiving this package, Alice uses $L$ to form the sequence of bits $K_a = Q(h_a(\tilde{L}))$. She uses the first $N_{au}$ bits of $K_a$ as the authentication key $K_{au} = K_a(1, \ldots, N_{au})$, and using $K_{au}$ she verifies the message authentication code to confirm that the package was indeed sent by Bob. Since Eve does not know the bits in $K_{au}$ generated by Bob, she cannot modify the $L$-message without failing authentication at Alice. Finally, if Eve inserts a significant number of random guesses into a fake $L$-message, Bob can detect her presence by computing the proportion of indexes in $L$ that lead to excursions in $\hat{h}_L$. Since Eve can only make random guesses, this quantity would be much lower than one resulting from a legitimate $L$-message from Alice. Therefore, even without an authen-
Another type of active attack involves Eve impersonating Alice or Bob during the channel-probing stage itself, i.e. Eve may begin sending probes to Bob pretending to be Alice or vice-versa. Such an attack can be detected using a hypothesis testing approach on the sequence of probes received at each legitimate user, and this has been extensively studied in [15, 16]. Essentially, legitimate users compare each newly received probe against the recent history of received probes from the other legitimate user using a hypothesis test. The technique relies on the insight that a given sufficiently fast probing rate, successive probes received at a legitimate user are most likely to differ by a small amount. In practice the hypothesis test is implemented by comparing a likelihood ratio with a threshold and is based on a Neyman-Pearson criterion. We do not present further details on this type of attack here. Detection of Eve using the hypothesis test can only be accomplished if Alice and Bob have already exchanged some probes. If Eve impersonates either or both legitimate users from the very beginning of the channel probing stage, then the method is susceptible to a man-in-the-middle attack wherein Eve can establish two keys, one each with Alice and Bob, while giving Alice and Bob the impression that they have established a key with themselves. We note that this problem can be alleviated using techniques similar to those used to secure the Diffie-Hellman protocol.

5. PERFORMANCE EVALUATION

The central quantities of interest in our protocol are the rate of generation of secret bits, the probability of error and the randomness of the generated bits. The controls available to us are the parameters: $q_+$, $q_-$, $m$ and the rate at which Alice and Bob probe the channel between themselves, $f_d$. We assume the channel is not under our control, and as explained in Section 3.3, the rate at which the channel varies can be represented by the maximum Doppler frequency, $f_d$. The typical Doppler frequency for indoor wireless environments at the carrier frequency of 2.4 GHz is $f_d = \frac{v}{c} \sim \frac{2 \times 10^6}{3 \times 10^8} = 8$ Hz, assuming a velocity $v$ of 1 m/s. We thus expect typical Doppler frequencies in indoor environments in the 2.4 GHz range to be of the order of about 10 Hz and 20 Hz in the 5 GHz range. For automobile scenarios, we can expect a Doppler of $\sim 200$ Hz in the 2.4 GHz range.

5.1 Probability of error

The probability of error, $p_e$, is critical to our protocol. As explained in equation (8), in order to achieve a robust keystream-mismatch probability $p_k$, the bit-error probability $p_e$ must be much lower than $p_k$. A bit-error probability of $p_e = 10^{-7} \sim 10^{-8}$ is desirable for keys of length $N = 128$ bits. We have explained in Section 3.3 that there is a fundamental trade-off in the selection of parameters $m$, $q_+$ and $q_-$ that affects the rate and probability of error in opposing ways.

The probability of bit-error, $p_e$ is the probability that a single bit generated by Alice and Bob is different at the two users. The symmetry of the distribution of $h$ allows us to consider just one type of bit error in computing $p_e$. Consider the probability that Bob generates the bit “0” at an index given that Alice has chosen this index but she has not generated the bit “1”. As per our Gaussian assumption on the parameter $h$ and estimates $\hat{h}_a$ and $\hat{h}_b$, this probability can be expanded as

\[
P(B = 0|A = 1) = \frac{P(B = 0, A = 1)}{p(A = 1)} = \frac{\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \exp \left\{ -\frac{1}{2} \mathbf{Z}^{-1} \mathbf{K}_{m}^{-1} \mathbf{Z} \right\} d(2m-1)x}{\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \exp \left\{ -\frac{1}{2} \mathbf{Z}^{-1} \mathbf{K}_{m}^{-1} \mathbf{Z} \right\} d(m)x}
\]

where $\mathbf{K}_m$ is the covariance matrix of $m$ successive Gaussian channel estimates of Alice and $\mathbf{K}_{m-1}$ is the covariance matrix of the Gaussian vector $(\hat{h}_a[1], \hat{h}_b[1], \hat{h}_a[2], \ldots, \hat{h}_b[m-1], \hat{h}_a[m])$ formed by the combining $m$ channel estimates of Alice and the $m - 1$ estimates of Bob in between, in chronological order. The numerator in (9) is the probability that of $2m - 1$ successive channel estimates ($m$ belonging to Alice, and $m - 1$ estimates in between belonging to Bob), all $m$ of Alice’s estimates lie in an excursion above $q_+$ while all $m - 1$ of Bob’s estimates lie in an excursion below $q_-$. The denominator is simply the probability that all of Alice’s $m$ estimates lie in an excursion above $q_+$ while all $m - 1$ of Bob’s estimates do not lie in an excursion at a given index, a bit error is avoided because that index is discarded by both users.

5.2 Secret-bit rate

The correct way to address the tradeoff between probability of error and rate of generation of secret bits is to upper bound the acceptable probability of error and then attempt to derive the greatest possible rate. How many s-bits/second can we expect to derive from a time-varying channel? An approximate analysis can be done using the

![Figure 3: Timing diagram for the key-extraction protocol.](Image)

![Figure 4: Probability of bit error $p_e$ for various values of $m$ at different SNR levels ($q_{\pm} = mean \pm 0.8\sigma$) ](Image)
level-crossing rate for a Rayleigh fading process, given by
\[ LCR = \sqrt{2f_D}\sigma_r^2 \] [22], where \( f_D \) is the maximum Doppler frequency and \( \rho \) is the threshold level, normalized to the root mean square signal level. Setting \( \rho = 1 \) in this expression gives \( LCR = f_D \).

The above calculation tells us that we cannot expect to obtain more s-bits per second than roughly the order of \( f_D \).

In practice, the rate of s-bits/sec depends also on the channel probing rate \( f_s \), i.e., how fast Alice and Bob are able to send each other probe signals. In Figure 5 (a) and (b), we plot the rate in s-bits/sec as a function of the channel probing rate for a wireless channel with maximum Doppler frequencies of \( f_D = 10 \) Hz and \( f_D = 100 \) Hz respectively. As expected, we find that the number of s-bits the channel yields increases with the probing rate, but saturates at a value that is of the order of \( f_D \).

A more precise analysis of the secret-bit rate can be performed as follows: The number of s-bits/sec equals the number of s-bits per observation times the number of observations per second, i.e. the probing rate. Therefore

\[ R_s = H(bins) \times p(A = B) \times \frac{f_s}{m} \] (10)

\[ = 1 \times (p(A = 1, B = 1) + p(A = 0, B = 0)) \times \frac{f_s}{m} \] (11)

\[ = 2 \frac{f_s}{m} \times p(A = 1, B = 1) \] (12)

\[ = 2 \frac{f_s}{m} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2} \frac{x^2 + (\alpha \cdot \sigma) q_m^2}{\sigma_m}} \, dq_m \, dx \] (13)

where \( H(bins) \) is the entropy of the random variable that determines which bin (\( > q_+ \) or \( < q_- \)) of the quantizer the observation lies in, which in our case equals 1 assuming that the two bins are equally likely\(^6\). The probing rate \( f_s \) is normalized by a factor of \( m \) because a single 'observation' or 'bin' lies in, which in our case equals 1 assuming that the number of s-bits per observation times the number of observations per second, i.e. the probing rate.

\[ R_s = \frac{H(bins) \times p(A = B) \times f_s}{m} \] (10)

\[ = 1 \times (p(A = 1, B = 1) + p(A = 0, B = 0)) \times \frac{f_s}{m} \] (11)

\[ = 2 \frac{f_s}{m} \times p(A = 1, B = 1) \] (12)

\[ = 2 \frac{f_s}{m} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2} \frac{x^2 + (\alpha \cdot \sigma) q_m^2}{\sigma_m}} \, dq_m \, dx \] (13)

where \( H(bins) \) is the entropy of the random variable that determines which bin (\( > q_+ \) or \( < q_- \)) of the quantizer the observation lies in, which in our case equals 1 assuming that the two bins are equally likely\(^6\). The probing rate \( f_s \) is normalized by a factor of \( m \) because a single 'observation' or 'bin' lies in, which in our case equals 1 assuming that the number of s-bits per observation times the number of observations per second, i.e. the probing rate.

The levels \( q_+ \) and \( q_- \) are chosen so as to maintain equal probabilities for the two bins.

\[ \text{Figure 5: Rate in secret bits per second for various values of } m, \text{ against probing rate for a channel with Doppler frequency (a) } f_D = 10 \text{ Hz and (b) } f_D = 100 \text{ Hz} \ (q_\pm = \text{mean } \pm 0.8 \sigma) \]

\[ \text{Figure 6: (a) Secret-bit rate for varying Doppler } f_D \text{ and fixed } f_s \text{ for various values of } m \ (b) \text{ Rate as a function of quantizer levels } q_+ \text{ & } q_- \text{ parametrized by } \alpha. \]

5.3 Randomness of generated bits

Guaranteeing that the generated bits are random is crucial because they are intended for use as a cryptographic key.

Since we have assumed the adversary possesses complete knowledge of our algorithm, any non-random behavior in the bit sequences can be exploited by the adversary to reduce the time-complexity of cracking the key. For example, if the algorithm is known to produce a greater proportion of ‘1’s than ‘0’s, then the effective search space for the adversary would be reduced. Consequently a variety of statistical tests have been devised to test for various defects [29].

In evaluating the randomness of bit sequences generated by our algorithm, we focus on Maurer’s universal statistical test [30], a widely accepted benchmark for testing randomness. The test statistic relates closely to the per-bit entropy of the sequence, and thus measures the actual cryptographic significance of a defect as related to the running time of an adversary’s optimal key-search strategy [30]. It detects a broad class of statistical defects and therefore subsumes a number of other standard tests.

In addition to Maurer’s universal test, we ran a few other basic tests using the public-domain test suite made available by NIST [31]. We refer the interested reader to [32] for a description of these tests and the definitions of \( p - value \) for each test. The results for these are summarized in Table 2. Subsequent runs produced comparable results and thus support the conclusion that our algorithm provides random bits. In particular, Maurer’s test showed the average entropy of our bit-sequences is very close to the value expected for a truly random sequence. This can be possible only if successive bits are almost independent of one another, which in turn requires that they must be separated in time by at least a ‘coherence time’ interval. Since the coherence time of a channel is inversely proportional to the Doppler frequency, extracting bits from a channel at a rate significantly greater than \( f_D \) cannot possibly produce random bits. We observed

\(^6\)The levels \( q_+ \) and \( q_- \) are chosen so as to maintain equal probabilities for the two bins.
Table 2: Results from randomness tests on bit sequences (10^6 bits) produced by our algorithm for \( f_d = 10 \text{ Hz}, f_s = 30 \text{ Hz} \), \( m = 5 \) and \( q_+, q_- = \text{mean} \pm 0.2 \sigma \). In each test, a p-value > 0.01 indicates the sequence is random.

<table>
<thead>
<tr>
<th>Test</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approx. entropy</td>
<td>0.8721</td>
</tr>
<tr>
<td>Monobit frequency</td>
<td>0.9910</td>
</tr>
<tr>
<td>Runs Test</td>
<td>0.1012</td>
</tr>
<tr>
<td>Random excursions</td>
<td>0.9910</td>
</tr>
<tr>
<td>Lempel Ziv</td>
<td>1.0000</td>
</tr>
</tbody>
</table>

in Section 5.2 that the rate at which our algorithm generates secret bits is bounded from above by approximately the maximum Doppler frequency \( f_d \) - therefore our algorithm produces secret bits slow enough for them to be random. Finally, we note that the selection of a random subset of excursions by Alice effectively allows her some control on selecting the final key generated. Thus, even if a particular run happens to produce excursions at Alice containing a statistical defect in the resulting bit sequence, she can fix the defect to some extent by suitably choosing \( L \) from among eligible excursions.

5.4 Summarizing insights

The rate in secret bits per second that Alice and Bob derive from a time-varying channel is fundamentally limited by the rate of time-variation in the channel. To maximize rate, we need to probe the channel fast enough. Given the fastest possible rate at which we can probe, the parameters \( m, q_+, q_- \) can be tuned to keep the probability of error within an acceptable bound. Increasing \( m \) or the magnitudes of \( q_+, q_- \) provides a drop in the error probability at the cost of a decrease in the secret-bit rate. Further, increasing the amount of temporal variation in a channel helps increase the secret-bit rate but only up to a point, after which further increase in temporal-variation produces a decrease in the rate, unless it is accompanied by a proportional increase in the channel probing rate. Standard randomness tests indicate that our algorithm is resilient to attacks exploiting randomness defects. Key rates significantly greater than the maximum Doppler frequency cannot result in truly random bits.

6. VALIDATION USING 802.11A

We now describe our experimental efforts to validate our algorithm for typical indoor environments. Our experiments were divided in two parts. In the first study, we delved into the structure of an 802.11 packet to access the preamble of 802.11 packets exchanged between commercially available 802.11a radios, with one legitimate user configured as an access point (AP mode) and the other as a client (station mode), and a third user configured to listen (station mode) on transmissions from both legitimate users.

6.1 CIR method using 802.11a

6.1.1 Experiment setup

Our experimental platform (Figure 7(a)) consisted of an 802.11 development board with commercial 802.11a/b/g modem IP, to which we added custom logic to process CIR information. (b) Timing diagram for collecting CIR information using PROBE packets of 802.11 packets exchanged between commercially available 802.11a radios, with one legitimate user configured as an access point (AP mode) and the other as a client (station mode), and a third user configured to listen (station mode) on transmissions from both legitimate users.

Figure 7: (a) Our experimental platform - a development board for a commercial 802.11a/b/g modem IP, to which we added custom logic to process CIR information. (b) Timing diagram for collecting CIR information using PROBE packets

Figure 8: A layout of the experimental setup for the CIR method (distances in cm)

of 802.11 packets exchanged between commercially available 802.11a radios, with one legitimate user configured as an access point (AP mode) and the other as a client (station mode), and a third user configured to listen (station mode) on transmissions from both legitimate users.

6.1.1 Experiment setup

Our experimental platform (Figure 7(a)) consisted of an 802.11 development board with commercial 802.11a/b/g modem IP, to which we added custom logic to extract the channel impulse response from received packets. This allowed us to pull out received signal information at a level not normally accessible using commodity 802.11 hardware and drivers. Two such boards were set up as Alice and Bob, while a third board was configured to be Eve. Alice was configured to be an access point (AP mode), and Bob was configured to be a client (station mode). The experiment involved Bob sending PROBE request messages to Alice, who then replied with a PROBE response message (Figure 7(b)). Limitations of our development boards allowed us to have Eve listen on either Alice or Bob, but not both. In the results presented here, Eve has been configured to listen in on Alice. In the first experiment, Alice and Eve were placed in a laboratory, while Bob was placed in an office cubicle outside the lab. Figure 8 shows the layout of the three users and the surrounding environment. In the second experiment, Alice and Eve remained in the same positions while Bob circled the cubicle area along the trajectory in Figure 8 in a cart on wheels.

Figure 9 shows a 64-point CIR obtained from a single 802.11a PROBE request packet received at Alice, along with the corresponding CIR computed from the PROBE response packet received by Bob in reply. Also shown is the CIR sequence 

7The Prism header is available on Atheros 802.11 cards using the Madwifi [34] driver and contains RSSI and MAC-layer timestamps among other parameters.
where $u$ selected to vary the quantizer levels. We chose $\alpha$ the average signal power, into the key generation algorithm. Each user locally computes $\text{In other words, the channel in Figure 10 is not stationary.}$

user simply uses this data as input to the level-crossing bit-extraction algorithm, the generated key has long strings of 1s and 0s (see Figure 10). This is because we are attempting to include the effect of $\text{Figure 9: The 64-point CIR from a single 802.11 packet.}$

For our key-extraction algorithm, we use the magnitude of the main peak as the channel parameter of interest.

as computed by Eve, using the overheard PROBE response packet from Alice. For the purpose of our algorithm, we use only the magnitude of the main peak in the CIR.

Figure 10 shows the traces of the CIR’s main peak’s magnitude at Alice and Bob for our first experiment. While our experiment ran for $\sim 22$ minutes, in the interest of space and clarity we show 1000 CIRs collected over a duration of $\sim 110$ seconds. The traces show significant changes in average signal power, ostensibly due to time-variations in the wireless environment between Alice and Bob (see Figure 8). If each user simply uses this data as input to the level-crossing bit-extraction algorithm, the generated key has long strings of 1s and 0s (see Figure 10). This is because we are attempting to include the effect of shadow fading [22] (also called large-scale fading) that produces large but slow swings in the average signal power, into the key generation algorithm.

In other words, the channel in Figure 10 is not stationary. Each user locally computes $q_+$ and $q_-$ as:

$$q_+ = \text{mean}(\hat{h}_u) + \alpha \cdot \sigma(\hat{h}_u)$$

$$q_- = \text{mean}(\hat{h}_u) - \alpha \cdot \sigma(\hat{h}_u),$$

where $u$ can be Alice or Bob, $\hat{h}_u$ is the set of magnitudes of the CIR’s main peak collected by user $u$, and $\sigma(\hat{h}_u)$ represents the standard deviation of $\hat{h}_u$. The factor $\alpha$ can be selected to vary the quantizer levels. We chose $\alpha = 4$ for the CIR-method. The effect of the underlying shadow fading contained in the collected data can be removed by subtracting a moving average of each trace from the original trace. This leaves only the small scale fading that we wish to use in our algorithm. The result is shown in Figure 11. In this way, not only do we do away with the problem of long strings of 1s and 0s, we also prevent the average signal power from affecting our key generation process. Using the small scale fading traces, our algorithm generates $N = 125$ s-bits in 110 seconds ($m = 4$), yielding a key rate of about 1.13 s-bits/sec.

6.1.2 Contrasting Eve’s attempts

Figures 10 shows a trace of Eve’s CIR peak as overheard from Bob along with Alice’s and Bob’s traces. Figure 11 shows the bits that Eve would generate if she carried through with the key-generation procedure. The mutual information [9] (M.I.) between Eve’s data and Bob’s data is a useful measure of the information learned by Eve about Bob’s measurements $\hat{h}_b$ and can be compared to the mutual information between Alice’s and Bob’s estimates $\hat{h}_a$ and $\hat{h}_b$. Table 3 gives these mutual information values computed using the method in [35]. As a consequence of the data processing inequality [9], any processing of the received signal by Eve would only reduce her information about the Alice-Bob channel, and therefore the M.I. values in Table 3 provide upper bounds on the information about the Alice-Bob channel leaked out to Eve. The results from our second experiment with a moving Bob are very similar to the ones shown for the first experiment, although with fewer bits produced. Due to space limits, we do not present plots for the mobile experiment but instead summarize our results in Table 3. It is notable that in the static case the M.I. between Eve and Bob is orders of magnitude smaller than that between Alice and Bob and very close to zero, indicating that Eve is unable to derive any significant information about the Alice-Bob channel. Further, the M.I. between Eve and Bob is lower in the mobile case compared to the static case, indicating that mobility actually helps strengthen the secrecy of generated keys.

6.2 Coarse measurements using RSSI

6.2.1 Experiment setup
The setup consisted of three off-the-shelf 802.11 radios. Alice was configured in AP mode along with a virtual monitor interface to capture received packets. Bob was a client, consisting of a laptop with a 802.11a card configured in station mode, along with virtual monitor mode for capturing received packets. Eve was a third 802.11a node, identical in configuration to Bob, but capable of receiving packets from both Alice and Bob. In our experiment, Alice was stationary, while Bob and Eve moved along fixed trajectories. Atheros [36] WiFi cards based on the 5212 chipset were used at each end along with the Madwifi driver [34] for Linux. The experiments were done in the 5.26 GHz channel. The AP-station configuration ensured that MAC-layer clocks at the two nodes were synchronized. Figure 12 (b) shows the layout of the office building along with the location of the fixed AP and path followed by the mobile client. ICMP PING packets were sent from the AP to the client at a rate of 20 packets per second. Each PING request packet received at the client generates a MAC-layer acknowledgment packet sent back to the AP, followed by a PING response packet. Upon receiving the PING response packet, the AP similarly replies with a MAC-layer ACK packet.

Figure 12 (a) shows the sequence in which these packets are sent. A tcpdump [37] application running on both the AP and the client side recorded and time-stamped all packets received on the monitor interface of each user. The experiment consisted of sending 8,000 packets from Alice to Bob. The tcpdump traces at each end were filtered using the MAC address field to keep only the four types of packets described above. Further, RSSI and MAC-timestamps were pulled out of each packet to generate a trace of (timestamp, RSSI) pairs.

6.2.2 Modification to handle timestamps

We note that since the precise time instants at which the PING response and PING request messages are received Alice and Bob respectively cannot be controlled, there was no way to guarantee that successive PING request messages received by Bob were separated in time by exactly one PING response received in between by Alice. Therefore MAC-layer timestamps were essential to time-align RSSI information at Alice & Bob since we did not have index numbers with which to reference RSSI values. This required a slight variation in our algorithm to handle MAC-timestamps instead of indexes in the messages exchanged between Alice and Bob. The modifications may be summarized as:

1. Instead of sending index numbers to Bob, Alice now sends MAC-timestamps in the message L (see Algorithm 1 in Section 4).
2. For each MAC-timestamp sent by Alice, Bob finds the MAC-timestamp in his own trace that is closest in time to the value of the timestamp sent by Alice.
3. Bob uses the packet determined in Step 2 above as if it were the index sent by Alice. He checks for the presence of excursions above $q_+$ or below $q_-$ centered at this packet as in Algorithm 1.

The RSSI field in the Prism header of received 802.11 packets reports RSSI as integers, thereby providing only coarse channel information. Moreover, the 802.11 cards at Alice and Bob may not be relatively calibrated and thus may report different values of RSSI. We found in our experiments that although lacking calibration, the temporal variations in RSSI are matched in Alice’s and Bob’s traces. This problem was solved by subtracting out a moving average of the trace to remove the effects of slowly varying average signal power, as in the CIR method. Figure 13 shows the raw RSSI traces...
collected by Alice and Bob plotted against their received MAC-timestamps. As in the CIR-method, the traces exhibit strong variations in average signal power. We average out the large-scale variations and keep only the small scale fading effect. The result is shown in Figure 14. Our algorithm produces secret bits at a rate of almost 1.3 s-bits/sec using $m = 4$, where $q_u$ and $q_v$ were computed independently by each user as in (14)-(15) with $\alpha = \frac{1}{2}$.

### 6.2.3 Contrasting Eve’s attempts

We plot the RSSI traces captured by Eve for both Alice’s and Bob’s signal in Figure 13. The traces from Alice and Bob after considering only variations about a moving average, are shown in Figure 14 along with the key generated by Alice and Bob. Even with coarse RSSI measurements that represent the average received signal power per-packet over the entire 802.11 channel bandwidth, Alice and Bob can exploit reciprocity of their channel to successfully generate secret bits at a fairly good rate. We compute the pair-wise M.I. between the traces of Eve, Alice and Bob in Table 3. As in the CIR-method, we find that Eve gets almost no information about the Alice-Bob channel.

### 7. DISCUSSION AND CONCLUSIONS

The properties of the wireless medium can support security objectives for wireless systems by making it easier to establish cryptographic keys. In this paper, we proposed a protocol that exploits the reciprocity of the transfer function of the wireless multipath channel to establish a common cryptographic key between two communicating entities. Our protocol obtains a security advantage from the fact that the channel response decorrelates rapidly with distance from either communicator, implying that there is strong protection against a passive eavesdropper as well as an active adversary attempting a spoofing attack. The performance of our scheme was evaluated and important insights relating the probing rate, quantizer parameters and the resulting secret key generation rate were provided.

We also presented the results of a thorough effort to experimentally validate the utility of the wireless channel for secret key generation. We validated our algorithm through two complementary efforts. First, we constructed a system to extract channel impulse response information on a customized 802.11 development platform, where we utilized the 802.11a preamble to compute channel impulse responses on a per-packet basis. Second, we used off-the-shelf 802.11a cards for collecting coarse RSSI measurements. In both cases, our algorithm worked well, generating secret bits at a useful rate without any errors. We showed that an eavesdropper shares minuscule mutual information with legitimate communicators, thereby supporting security against eavesdroppers. Our work demonstrates that the multipath information that is inherent in any wireless system (and is normally discarded after physical layer processing), can successfully support key establishment. More importantly, we showed that although this capability is possible with custom architectures, it can actually be achieved using off-the-shelf radio platforms, and thus could lead to an immediate impact on the security of commodity wireless systems.

The astute reader might inquire about whether varying levels of interference at different locations in the environment would affect the secret key generation process. Our work has provided fundamental tradeoffs relating signal-to-interference levels to quantizer parameter selection for an isotropic noise background. By conservatively selecting protocol parameters (e.g., selecting a larger value of $m$ (see Figure 4)), it is possible to improve robustness in the key generation process, though at the cost of lowering the rate.

Beyond our fundamental observations and feasibility studies, there are many avenues for advancing the proposed protocol that could make our scheme more powerful. In particular, we have used a simple two-level scalar quantizer, and we note that the quantizer could be optimized through vector quantization, which would allow for improved trade-off between key generation rate and probability of error. This is a non-trivial optimization problem though and is part of our ongoing effort. Finally, we note that many emerging wireless systems are moving toward adopting MIMO or OFDM as a means to enhance communication rates. Our algorithm would naturally benefit from both of these, as multiple uncorrelated channels between two users would lead to a proportional increase in the secret-bit extraction rate.

### 8. REFERENCES


<table>
<thead>
<tr>
<th>CIR-based method</th>
<th>Value of $m$ used</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choice of $q_u$, $q_v$</td>
<td>mean $\pm 1.126$</td>
<td></td>
</tr>
<tr>
<td>Duration of experiments</td>
<td>$1526$ sec ($\sim 22$ min.)</td>
<td></td>
</tr>
<tr>
<td>Inter-probe duration</td>
<td>$110$ msec</td>
<td></td>
</tr>
<tr>
<td>Static case:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average secret-bit rate</td>
<td>$1.28$ s-bits/sec</td>
<td></td>
</tr>
<tr>
<td>$I(Alice; Bob)$</td>
<td>$3.294$ bits</td>
<td></td>
</tr>
<tr>
<td>$I(Bob; Eve)$</td>
<td>$0.0462$ bits</td>
<td></td>
</tr>
<tr>
<td>Mobile case:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average secret-bit rate</td>
<td>$1.17$ s-bits/sec</td>
<td></td>
</tr>
<tr>
<td>$I(Alice; Bob)$</td>
<td>$2.18$ bits</td>
<td></td>
</tr>
<tr>
<td>$I(Bob; Eve)$</td>
<td>$0.108$ bits</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RSSI-based method</th>
<th>Value of $m$ used</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choice of $q_u$, $q_v$</td>
<td>mean $\pm 0.58$</td>
<td></td>
</tr>
<tr>
<td>Average secret-bit rate</td>
<td>$1.3$ s-bits/sec</td>
<td></td>
</tr>
<tr>
<td>Inter-probe duration</td>
<td>$59$ msec</td>
<td></td>
</tr>
<tr>
<td>Duration of experiment</td>
<td>$400$ sec</td>
<td></td>
</tr>
<tr>
<td>$I(Alice; Bob)$</td>
<td>$0.18$ bits</td>
<td></td>
</tr>
<tr>
<td>$I(Alice; Eve)$</td>
<td>$0.60$ bits</td>
<td></td>
</tr>
<tr>
<td>$I(Bob; Eve)$</td>
<td>$0.07$ bits</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Summary of experimental results. $I(u_1; u_2)$ denotes the mutual information (M.I.) between the measurements of users $u_1$ and $u_2.$


[33] “IEEE standard 802.11a: Part 11 wireless LAN medium access control (MAC) and physical layer (PHY) specifications: High-speed physical layer in the 5 GHz band.”

[34] “http://www.madwifi.org/.”

