Practical Implementation of Physical-Layer Key Generation using Standard WLAN Cards and Performance Evaluation

by

Munder Hamruni

Master Thesis in Electronics Engineering

Ph.D. cand. (ABD) Oana Graur

Prof. Dr.-Ing Werner Henkel

Name and title of the supervisors

Date of Submission: July 31, 2017

Bremen University of Applied Sciences — Faculty of Electrical Engineering and Computer Science

Declaration

I declare that the work in this thesis is carried out in accordance with the Regulations of the Bremen University of Applied Sciences.

I am the sole author of this thesis. The work presented is the result of my own research except as in cited references. I clearly marked all text passage from other sources as references.

Munder Hamruni
Bremen, July 31, 2017

Acknowledgment

First of all, I would like to thank my supervisors, Ph.D. cand. (ABD) Oana Graur and Prof. Dr.-Ing. W. Henkel for their instruction, guidance, and confidence during the time of my thesis. I am also grateful to all colleagues at Jacobs University for offering me their support and help during my work.

I would also like to thank all my professors at Bremen University of Applied Sciences for providing their unlimited help and support in all modules and labs.

Finally, my special thanks to my family and to my friends for providing the continuous encouragement, care, and love through my years of study. This accomplishment would not have been possible without their help. Thank you.

Munder Hamruni
Bremen, July 31, 2017

Abstract

The properties of the physical layer in wireless networks are exploited to generate one-time pads which are subsequently used for secure communication between legitimate users. Previous measurement testbeds relied mostly on RSSI information, since such information could be provided by most off-the-shelf wireless network cards. However, it has been shown that a significant performance gain can be obtained by using more fine-grained measurement samples, such as the channel state information (CSI). In this thesis, a measurement testbed is developed using commercially available Intel 5300 NICs, along with a Linux 802.11n CSI Tool. The platform consists of three laptops, running Ubuntu 14.04 LTS, Kernel 3.2, two acting as the legitimate users, Alice and Bob, while the third is acting as an eavesdropper.

The initial measurement phase of the key generation process is simulated, where the legitimate users take turns in sending signals in consecutive TDD slots, while the other legitimate user and the eavesdropper are listening.

The position of the eavesdropper is varied during the CSI measurements and frequency distributions are obtained. A study of the correlation between the legitimate channels and the eavesdropping channels is presented, along with a performance evaluation of the system.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation and Objectives

The open air nature of wireless communication is very susceptible to various passive attacks like eavesdropping, traffic analysis, and monitoring or to execute active attacks such as jamming, spoofing, and denial-of-service (DoS) attack [4, 5]. Although asymmetric key cryptosystems are fundamental requirements for secure communication to boost confidentiality and authentication services but it is difficult to guarantee availability of a key management in dynamic wireless environments such sensor networks [6]. Then, alternative ways for key agreement between wireless devices are needed [7, 8].

The physical layer characteristics in wireless networks are assumed to achieve information-theoretic security. The physical layer offers solutions to generate common secret keys by the legitimate users without exchanging the secret keys [9, 10]. The Received Signal Strength (RSS) is a wireless channel parameter which is used as a source of secret information between two parties. However, RSS has some problems in stationary environments due to the small-scale variations in the channel measurements, which it can be exploited by the eavesdropper [11]. Another channel parameter called Channel State Information (CSI) describes the current condition of the channel in each subcarrier. CSI is used in IEEE 802.11n networks where the data are modulated on multiple orthogonal frequency-division multiplexing (OFDM) subcarriers simultaneously [12].

The objectives of this thesis to present the implementation of CSI measurements in the key generation process, and to simulate the amplitude and the phase correlation between the legitimate channels and the eavesdropping channel. The goal is to achieve this correlation to extract symmetric keys by the legitimate users by exploiting the channel properties. A commercial wireless network card "*Intel Wi-Fi Link 5300 wireless NIC*" is used to collect the signal amplitude and the phase information for each subcarrier of each transmit-receive pair of the antennas. This will help us gather more information about the channel to use it in the key generation system.

1.2 Thesis Outline

The organization of this thesis is as follows:

Chapter 2 explains a general review on physical layer security. The information-theoretic security of Shannon's perfect secrecy is presented with the information entropy and the one-time pad cryptography. The wiretap is also depicted, where the channel properties are exploited by legitimate users. Multiple antenna channels for enhancing the physical layer security is presented. The different methods of the physical layer to establish a secure communication are provided, where a key-less technique is explained along with key based ones. Chapter 2 focuses more on key generation and its principles such randomness, temporal variation, spatial decorrelation, and channel reciprocity. The channel parameters are presented by including CSI and RSS. At the end of Chapter 2, the key generation operations are briefly discussed such as channel probing, quantization, information reconciliation, and privacy amplification.

Chapter 3 describes the IEEE 802.11 standards, which was relevant in our research. IEEE 802.11 a/b/g/ac standards are also explained with their physical properties. The operating modes of IEEE 802.11n standards are discussed. The frame format of IEEE 802.11n traces are presented including the frame header, frame body, FCS field. The CSI data is depicted with the details of its fields and subfields. The adaptive bit streams generation is presented along with its improvement for based shared key method.

Chapter 4 presents our measurement results. The measurement setup was described including the settings and the configurations of the CSI tool, the wireless network, and the implemented codes. The evaluation also was shown, where the amplitude, the group delay, and the correlation of our CSI measurements in different scenarios were presented.

Chapter 5 concludes the thesis and discusses the future work in the physical layer key generation.

2 Physical Layer Security

Traditionally, information security is handled at the upper layers of the OSI model (see Fig. 1). Examples of protocols above the Physical Layer are Secure Shell (SSH) at the Application layer, Transport Layer Security (TLS/SSL) at the Transport layer, Internet Protocol Security (IPsec) at the Network layer, and Wired Equivalent Privacy (WEP) at the Data-Link layer. Most protocols rely on the generation of asymmetric private and public keys. As such, they are based on mathematical operations like prime number factorization and trap-door functions. Such methods are not secure from an information-theoretic point of view and rely on the assumption that a potential attacker has limited computational power. Due to this drawback they are referred to as as computationally secure methods. Furthermore, recent studies have pointed out several previously unknown weaknesses of key management and distribution schemes, where the attacker can decrypt the cipher text in a few hours [13, 14].

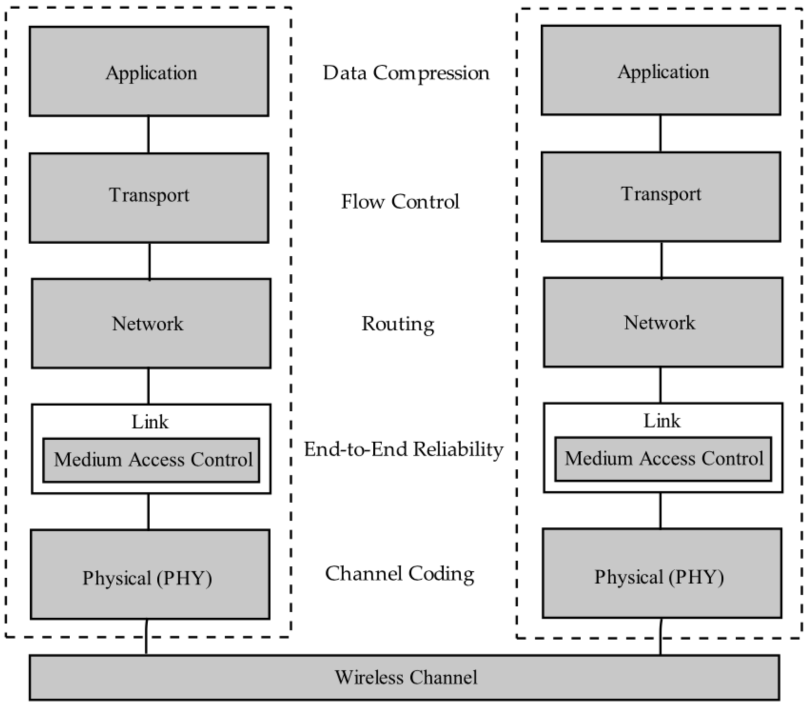

Figure 1: OSI model architecture [1]

Previous results from information theory and signal processing suggest using the imperfection of the physical layer to secure the systems. For instance, the fading in wireless communication could be exploited for symmetric-key generation. Physical Layer key generation is based exactly on this idea, and in contrast to conventional cryptography, it uses a reciprocal radio propagation channel (e.g. Time-Division Duplexing) to generate a shared secret key between two legitimate users, Alice and Bob. With this method, the secret key distribution problem is avoided since, ideally, each legitimate user generates the same key from the channel properties without sacrificing too much data rate [15]. In practice, nonetheless, several aspects have to be accounted for, as we will explain in subsequent chapters.

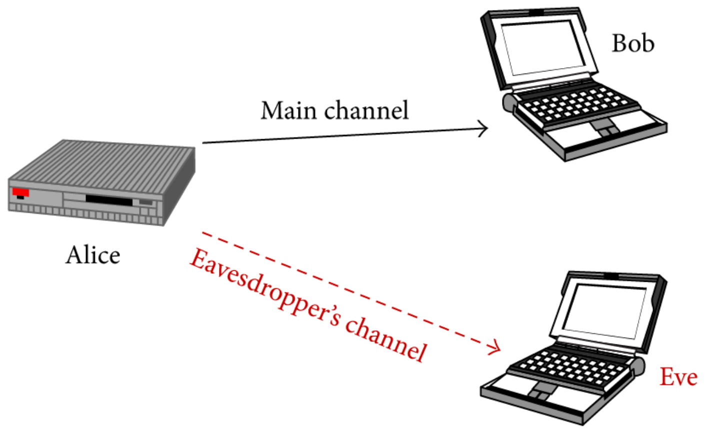

Figure 2: Wireless network model in the presence of an eavesdropper [1]

For instance, the main assumption is that Alice and Bob share a common source of randomness, e.g., the random fluctuations of the channel between them, while a passive eavesdropper (Eve) would experience different channels to the legitimate users (Fig. 2), and, as a result, obtain entirely different measurements, given sufficient spatial distance. These typical assumptions, along with their accuracy are discussed in more detail in Section 2.4.1.

2.1 Information-theoretic Security

2.1.1 Shannon's Perfect Secrecy

In September 1949, Claude Shannon published a paper entitled "*Communication Theory of Secrecy Systems*" [16]. In there, he proposed a cryptosystem with $|K| = |C| = |P|$, where they refer to number of bits in key, cipher text, and plain text respectively. The system is perfectly secure when the following conditions are satisfied:

- The probability of each key is $\frac{1}{|K|}$.
- Each plain text x and cipher text y has a unique key k such that $k(x) = y$.

To prove perfect secrecy, $p(x|y) = p(x)$ for all x and y. In the case of $p(x) = 0$, the keys must be long enough so that any cipher text can be decrypted, that is, $|K| >= |C|$. In the case of $|K| = |C|$, the result will be a unique key for all x and y values [17]. By using Bayes law, and supposing all x values have an equal probability $p(x) = \frac{1}{K}$, we reach the result $p(x|y) = p(x)$.

Two classifications of security are common. *Unconditional security*, also known as information-theoretic security, considers that the cryptographic primitive cannot be broken, even if Eve has infinite computational resources. This is the best security that can be achieved. Under this strict condition, Eve has to try all possible keys through brute force, and Eve has no more chances of guessing the plain text x whether or not she has access to cypher text y, i.e., $p(x|y) = p(x)$.

Computational or cryptographic security, relies on the assumption that an attacker has access to limited computational resources which make it infeasible to decrypt the cypher text within a practical time frame. There is, however, no absolute guarantee on the security of the system [5].

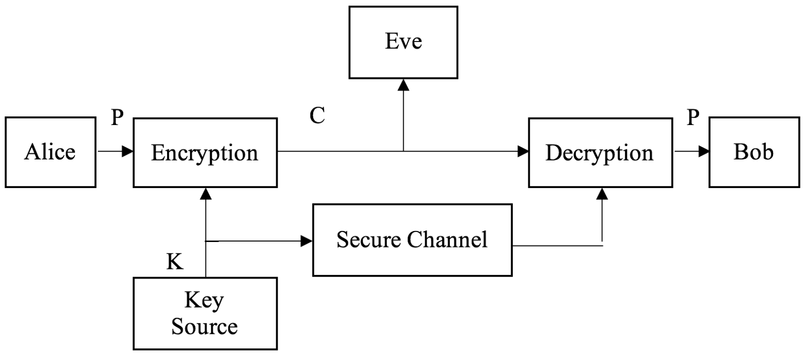

Figure 3: Shannon's model of perfect secrecy for symmetric encryption

Shannon explained the theoretical basis of cryptosystems by using the one-time pad to show perfect secrecy. He proved that as long as the secret key is as long as or larger than the plain text, an eavesdropper has no better chances than randomly guessing, and no information will be revealed, even if Eve obtains the cipher text. Although this aspect has been known since the publication of Shannon's paper, the constraint on the key length, coupled with the difficulty of safely distributing new keys as long as the messages to the legitimate parties, have made Shannon's one-time pad scheme impractical [18].

As a consequence, traditional security protocols use public and private keys, implemented at the upper layers of the protocol stack. Nonetheless, the physical characteristics of the channel can be exploited to at least complement the computational security methods, if not completely replace them.

Entropy

It is important to explain the concept of entropy in information theory, which is a measure of uncertainty or disorder of a random variable [19]. Suppose we have a coin, which is a binary memoryless source. Let's do some experiments with different output probabilities.

- In case $P(\text{head}) = P(\text{tail}) = 0.5$, it is hard to predict the output because they have the same chance of occurrence. Then, the uncertainty is 1 bit, which is the highest entropy.

- The uncertainty is zero if $P(\text{heads}) = 1$ or $P(\text{tails}) = 1$, which is the lowest entropy.

The entropy of a discrete random variable X is defined as:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i) \ , \tag{1}$$

where $P(x_i)$ denotes the probability of X taking value x_i. Among all possible distributions on a discrete set of possible outcomes (Fig. 4).

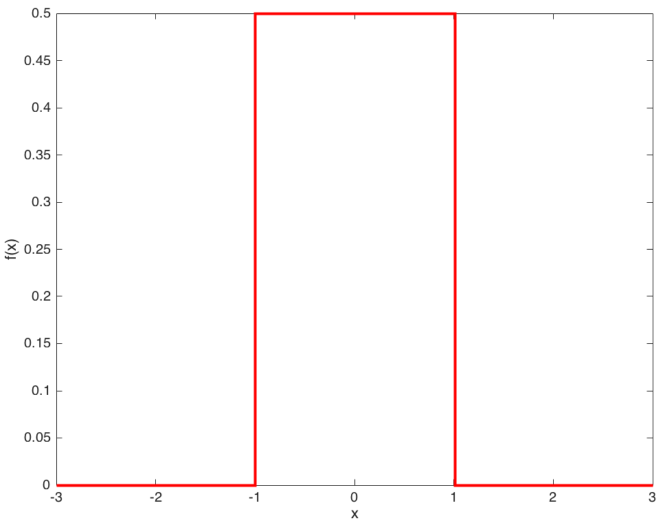

Figure 4: Probability density function of a uniform distribution on $[-1, 1]$

Among unbound probability distributions on $(-\infty, +\infty)$, the maximum entropy is achieved by the Gaussian distribution (Fig. 5).

2.1.2 The Wiretap Channel Model

Shannon's channel model considered noiseless communication. Under this assumption, the communication is only perfectly secure when the cipher text and plain text are mutually independent. Expanding on Shannon's work, Wyner proposed the wiretap channel model [20], in which an eavesdropper obtains a (degraded) version of the signal transmitted over a discrete memoryless channel (DMS), by listening at the output of a wiretap channel, as shown in Fig. 6.

Under Wyner's model, Eve's channel experiences a worse SNR than the Alice-Bob channel, leading to a noisy version of Bob's signal. The wiretap channel can be modeled as a binary symmetric channel (BSC) with error probability p. The goal was to build an encoder-decoder system to maximize the amount of information obtained by Bob, while

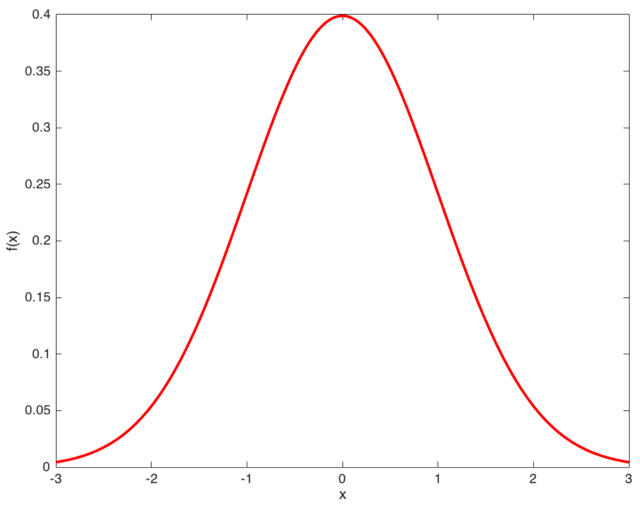

Figure 5: Probability density function of Gaussian distribution

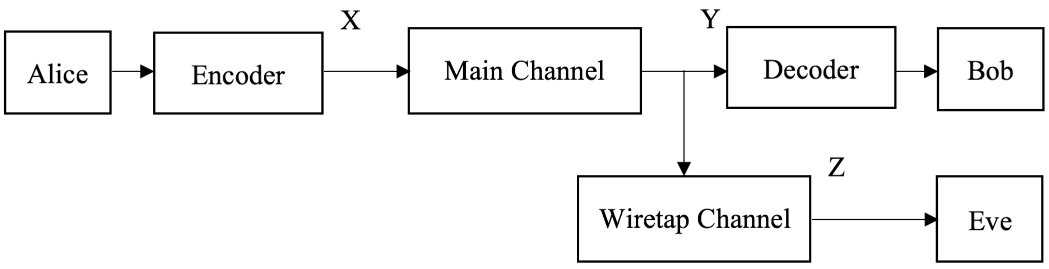

Figure 6: Wiretap channel model

minimizing the amount of useful information obtained by the wiretapper. Wyner further derived a bound [20, 21] on the achievable code rates under these constraints.

Later on, Csiszár and Körner showed that also in the case when Eve has a better channel than the legitimate users, secure communication can still be achieved. Their results generalized Wyner's wiretap channel model for the broadcast channel, where Alice is broadcasting public messages to Bob and Eve, while at the same time also sending a private message to Bob [22].

Previous research efforts have also investigated the achievable secrecy capacity in situations where Eve's channel experiences Rayleigh fading with additive Gaussian noise and the main channel is AWGN [23]. These efforts assumed that the main channel properties are only known at the transmitter side, while the fading of Eve's channel remains unknown to the legitimate users. Under the assumption above, its has been suggested that artificial noise and power bursting can be a solution to achieve secrecy, even if the main channel does not exhibit a better SNR than the eavesdropper's channel [24].

2.2 Multiple-Antenna Systems

The development of multi-antenna systems highly improves the secrecy capacity in wireless communication. Wyner's wiretap channel has been generalized for Multiple-Input Multiple-Output (MIMO) systems by Ekrem and Ulukus [25].

The spectral efficiency in MIMO channels is much higher than for single-antenna channels. In Multiple-Input Multiple-Output (MIMO) systems, instead of using the multi-antenna channel for multiplexing gains, the channel fading can be overcome by increasing the diversity. Each pair of transmit and receive antennas produces a signal path between the transmitter and the receiver. Then, using multiple antennas, same information is transmitted through different paths. This results is multiple independently faded signals that the receiver side can use to enhance the transmission reliability. This is known as diversity gain. The data rate can be increased if the fading path between each transmit and receive antenna pairs is independent where the transmitters and receivers can generate parallel channels.

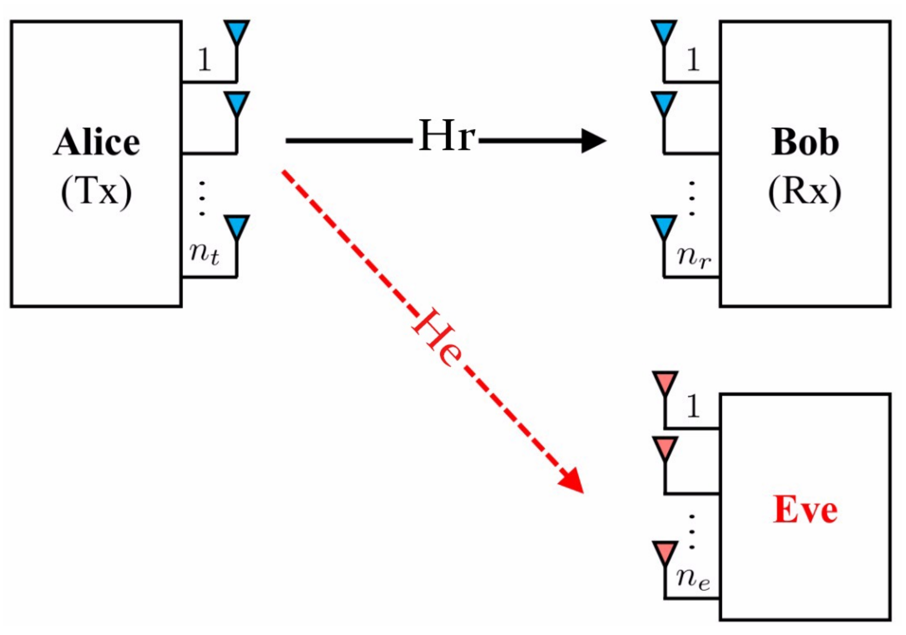

Figure 7: MIMO Gaussian wiretap channel[2]

Sending different portions of data on the different paths is considered by the multiplexing gain [26].

A simple MIMO model can be described within the wiretap channel scenario, as shown in Fig. 7. The users are equipped with n_t, n_r, and n_e antennas, respectively. The received signals can be written as:

$$\begin{aligned} \mathbf{y} &= \mathbf{H}_r\mathbf{x} + \mathbf{w}_r \ , \quad (2) \\ \mathbf{z} &= \mathbf{H}_e\mathbf{x} + \mathbf{w}_e \ , \quad (3) \end{aligned}$$

where $\mathbf{x}$ is the transmit vector, $\mathbf{y}$ and $\mathbf{z}$ represent the receive vectors of the legitimate user and the eavesdropper, respectively. $\mathbf{H}_r$ and $\mathbf{H}_e$ are the MIMO transmission matrices, and $\mathbf{w}_r$ and $\mathbf{w}_e$ are the noise vectors of the main channel, and the eavesdropper's channel, respectively.

The secrecy capacity is introduced as the maximum possible rate that can be achieved between the legitimate users while not disclosing any useful information to a potential eavesdropper.

Typically, the channel coefficients between the transmitter and the eavesdropper are assumed to be fully known at the transmitter, although this might not always be a realistic assumption. When the channel coefficients to Eve are unknown, the secrecy rate optimization could prove challenging. Some previous approaches introduce artificial noise at the transmitter side to increase Eve's uncertainty, while others introduce separate nodes that jam Eve in order to aid a secure communication between Alice and Bob.

2.3 Key-less Physical Layer Security

In Wyner's wiretap channel model, key generation is not performed, and confidential communication is achieved through code design, taking into account the channel properties between Alice and Bob, as well as Eve's channels. As we mentioned above, such key-less strategies set prerequisites on the availability of channel state information for both of the legitimate users and the eavesdropper. Such codes are generically known as wiretap codes. Some examples include coset codes and nested codes, Low-Density Parity Check (LDPC) codes, as well as polar codes. Nonetheless, we have only discussed this here to distinguish both physical layer approaches. The second addresses key generation instead. The key-less physical layer security is outside the scope of this work.

2.4 Key-based Physical Layer Security

In contrast to Wyner's work that relied on code design, Maurer has shown how the random characteristics of the wireless channel between the legitimate users can be used to extract symmetric keys for a secure communication [27]. The keys are generated by Alice and Bob during an initial phase, and there is no further consideration required regarding a key distribution infrastructure. Ideally, this idea could be utilized in a traditional symmetric key cryptosystem such as Shannon's one-time pad. In addition, this method is lightweight and does not need any infrastructure.

Method	**Description**	**Advantages**	**Disadvantages**
Key-less security	Alice and Bob establish a secure communication by using the channel properties and the code design	Without any keys, the communication between legitimate users can be secured in the physical layer	Channel state information of Eve's channel required
Key generation	Alice and Bob generate key according to the common randomness of the channel	No key distribution infrastructure/mechanism required.	Limited by the channel dynamics. Reconciliation scheme required.

Table 1: Comparison between key-less and key generation schemes

The strategy behind key generation is based on the idea of common randomness. During an initial phase, Alice and Bob measure the wireless channel in consecutive Time-Division Duplexing (TDD) slots. Ideally, the measurement duration is chosen such that the wireless channel can be considered static during the time Alice and Bob obtain their measurements, while it changes while the next TDD slots, i.e., next sets of measurements. Under this assumption, the channel is known as *quasi-static* or *quasi-fading*.

The key generation techniques are not totally different from key-less techniques as far as secrecy vs. secret key capacity derivation are concerned [14]. Furthermore, they can be implemented in parallel with some existing techniques to further improve security. A comparison between the techniques is illustrated in Table 1.

2.4.1 Principles of Key Generation

2.4.1.1 Channel Reciprocity

The most important and fundamental condition of symmetric secret key generation systems is channel reciprocity. The channel reciprocity property assumes that the multi-path and fading measured by both transmitter and receiver on a link is the same, assuming measurements within the same frequency range. This comes from the channel reciprocity of radio wave propagation.

However, there are some practical aspects that deviate from the theoretical model. For instance, let us assume a TDD system in which Alice and Bob send pilot signals to each other in consecutive time slots in order to measure the channel. During the first slot, Alice is sending her pilot signal, while Bob is listening. This enables Bob to get an estimate of the channel. During the second time slot, Bob is sending a pilot signal, and Alice is listening and collecting her channel estimates. Even if the channel properties have not fluctuated between the first and second time slot, it is very unlikely that they have obtained identical measurements. This is due to the independent noise at their receivers. They will, however, obtain correlated information that will need to be further processed to ensure that identical keys are extracted. The techniques employed to guarantee identical keys fall under the area of *key reconciliation*. The amount of correlation between their measurement sets will depend on the independent noise present at both ends. Furthermore, different hardware at both ends can further impact the symmetry (reciprocity).

Furthermore, the assumed static for the two TDD slots might be violated. Therefore, non-identical yet correlated keys will be generated due to the non-simultaneous measurement process and noise impact. The practical aspects can nonetheless be solved by signal processing techniques and by carefully choosing the duration of the consecutive TDD slots.

2.4.1.2 Temporal Variation

Temporal variation in the measurements is induced by the movement of objects or users in the environment. Any movement will induce variations in the wireless channel characteristics, since such actions will impact the channel paths, e.g., the reflection, refraction, and scattering. Too much movement can result in the channel changing its properties at a faster rate than Alice and Bob obtain their measurements. This, of course, will lead to key discrepancies. Too little temporal variation will result in measurements being mostly static, and the same key symbols being generated over and over again.

Nonetheless, the randomness produced by unpredictable motion can be exploited as a random source for key generation.

2.4.1.3 Randomness

Common randomness is a big challenge for key generation systems in physical layer security [28]. To avoid the risk that the information is compromised, the shared key should be random. This is a direct consequence of the entropy concept described at the beginning of this chapter. For maximum entropy of the key, the key symbols should be uniformly distributed across a finite alphabet, or normally distributed across an infinite alphabet. The eavesdropper will have a harder time in just guessing a key if the key has maximum entropy. If the key, however, has symbols that are more likely than others, then the entropy is reduced. This is equivalent to saying that certain keys/symbols are more likely, thus giving the eavesdropper a better chance of guessing them.

If we enforce the premise that the key symbols have to be uniformly distributed for a final usable key, then any key obtained whose symbols do not have an equal probability of occurrence has to be further processed to remove redundant information. This will result in a shorter usable key and will decrease the amount of usable key bits obtained per unit of time. In other words, the lack of sufficient temporal channel variation will reduce the amount of randomness and negatively impact the key generation rate.

Wireless channels have some features where common randomness is based on. The features such as the channel reciprocity of the radio wave propagation, the temporal variations, and the spatial decorrelation increase the chance to generate identical keys over the main channel without information exchange between the legitimate users. As a consequence, the measurement of the main channel has unpredictable data where the multi-path, the fading, and the movement of objects impact the randomness.

2.4.1.4 Spatial decorrelation

Wireless communication performance is enhanced by using multiple antennas at transmitter and receiver sides. In MIMO system, each path between transmitter and receiver would ideally be independent, where the multiple independent channels have the same characteristics, this refers to spatial correlation. Previous research [28, 29] assumed that the channel between legitimate users is decorrelated to the eavesdropper's channel, in case the eavesdropper is located more than 6 wavelength away from the legitimate users, which is called spatial decorrelation.

Another study [30] indicated that the eavesdropper may obtain largely correlated measurements if he locates within the line of sight (LoS) path of the legitimate users. Spatial decorrelation is essential for key generation, even if it is not valid in all environments.

2.4.2 Channel Parameters

Channel parameters are the measured quantities for key-based generation.

2.4.2.1 Channel State Information (CSI)

Channel state information indicates to the signal propagation between transmitter and receiver, where the common scattering, fading, and power decay are present along the distance. It is important to make an achievable and a reliable communication system with high data rates. The receiver in TDD system responds to the transmitter after it estimates and quantizes the CSI measurements. Therefore, the transmitter and receiver can obtain the same CSI.

The CSI measurements provide more details about the wireless channel where the CSI value is a fine-grained channel parameter. More information about CSI types, Channel Impulse Response (CIR) and Channel Frequency Response (CFR) are discussed in the following subsections.

Channel Impulse Response (CIR)

CIR provides both amplitude and phase information [31]. The phase shift estimation in wide-band systems can be used for the key generation with all channel information. In construct with narrow-band where it can be also used, but the phase is decreased into a single dimension parameter [32]. However, the phase is affected by noise, carrier frequency offset, and asynchronous clocks. The paths with small power are affected more to noise, but as well as the transmitted power is stable, then the amplitude of CIR is the same to the received power.

Channel Frequency Response (CFR)

CFR is mostly implemented in IEEE 802.11 OFDM systems [33, 34]. Where it is appropriate to estimate the wireless channel with the amplitude channel estimation. Practically, the phase estimation is commonly affected by the time and frequency offset. In contrast with CIR, the power in all the frequencies are the same in a decorrelated scattering environment.

2.4.2.2 Received Signal Strength (RSS)

The signal is transmitted over multi-path channel and received with the noise effect, where the instantaneous power of the signal is commonly not reported by NICs. Although the average power level is defined and indicated as RSS. As a result, RSS has a coarse grained channel information where each packet provides only one RSS value.

Practically, RSS is currently implemented in key generation. It is available and applied in many systems such as IEEE 802.11 systems [35, 36]. RSS might be impacted, in the case different devices and hardware resources are used at transmitter and receiver sides.

2.4.3 Key Generation Operations

2.4.3.1 Channel Probing

Channel probing is used to measure the randomness and other parameters over the channel. The process consists of two legitimate users and a common channel. Alice sends a probing signal to Bob, then, Bob will obtain and store some channel parameters according to the received signal. After that, Bob will send probing signal to Alice, and Alice will obtain and store corresponding channel parameters. The time difference between receiving probing signals should be smaller than the channel coherence time to ensure that the channel remains constant between the two measurements.

2.4.3.2 Quantization

Quantization in the key-based physical layer security is not that different from the quantization in Analog-to-Digital Converters (ADC). It is a process to map the analog measurements into binary words. After quantization, we have a number of key bits from each measurement which refers to a quantization region. The quantization grid (Voronoi region) is adjusted according to the signal-to-noise ratio (SNR) of the channel. The key conflict is reduced by using some methods [5], e.g., multi-bit quantization and Gray coding.

2.4.3.3 Information Reconciliation

There might be the key disagreement between the legitimate users after quantization. Reconciliation methods are implemented to correct the measurements. The information reconciliation can be realized with some protocols such as Cascade or with channel coding procedure such as Low-Density Parity-Check (LDPC), and Reed-Solomon codes along with a Cyclic Redundancy Check (CRC) may be used to confirm the key agreement along with other tools [5].

2.4.3.4 Privacy Amplification

During the information reconciliation process, some measurements are publicly transmitted, which can be exploited by an eavesdropper. This can affect the security of the communication system, specifically in the key sequence. Privacy amplification is implemented to overcome the detected measurements from the key sequence at legitimate users. This can be done by different types of the hashing functions [37].

Information reconciliation and privacy amplification are always considered jointly.

3 802.11n traces over Wireless Channel

Wireless Local Area Networks (WLANs) are based on standards of the Institute of Electrical and Electronic Engineers, specifically, the IEEE 802.11 standards [38]. Modern wireless network cards have advantages such a large and growing range of physical layer configuration to gain a better performance.

In this chapter, we focus more on the IEEE 802.11n standard that incorporates multiple antennas. IEEE 802.11n increases the size of the search area by adding additional factors in order to become more effective to channels changes [39]. We used a special CSI tool [40] to collect CSI measurements along with received IEEE 802.11n packet traces. The CSI measurements include channel information for each pair between transmitter and receiver at the level of subcarriers. It is different from RSS values, which depend on the total received power.

In our work, a commercial wireless network card "*Intel Wi-Fi Link 5300 wireless NIC*" was used, which allows for three antennas [40] to log the signal amplitude and the phase information for each subcarrier of each transmit-receive pair of the antennas. The Intel Wi-Fi Link 5300 wireless NIC only works with a Linux operating systems. MATLAB scripts were used to analyze the CSI measurements. This will help us gather more information about the channel to use it in the key generation.

3.1 IEEE 802.11 Series Protocols

IEEE 802.11 is a group of standards for wireless area networks (WLANs), which is implemented in the industrial, scientific, and medical band. The goal of IEEE 802.11 is to provide fixed, portable, and moving stations in wireless network connection with Medium Access Control (MAC) and physical layer (PHY) specifications [39].

3.1.1 IEEE 802.11a

IEEE 802.11a standard was ratified in 1999 [41], it operates at 5 GHz with high data rate that reaches to 54 Mbps. IEEE 802.11a has problems since the attenuation is very high at 5 GHz compared to 2.4 GHz. On the one hand, it is easily affected by obstacles and high noise whereby the signal loss is increased. On the other hand, at 5 GHz, the signal has less interference than at 2.4 GHz. However, the connection is maintained by automatically reducing the data rate to 48, 36, 24, 12, 9, 6 Mbps. IEEE 802.11a uses Orthogonal Frequency Division Multiplexing (OFDM) modulation on up to 12 channels [39]. IEEE 802.11a provides specifications for wireless outdoor systems and it is used in access points and routers.

3.1.2 IEEE 802.11b

A not-for-profit organization called the Wireless Ethernet Compatibility Alliance (WECA) was established in 1999 by industry leaders to test IEEE 802.11b products and guarantee the compatibility of different vendors' products [39, 41]. The IEEE 802.11b standard operates at 2.4 GHz, which provides a data rate of up to 11 Mbps.

IEEE 802.11b uses Direct Sequence Spread Spectrum (DSSS) at the physical layer, where the original data signal is multiplied with a continuous string of a pseudo noise sequence. The output can be shared by multiple transmitters with more protection and privacy. IEEE 802.11b requires a smaller number of access points than IEEE 802.11a to cover a specific area [39].

3.1.3 IEEE 802.11g

The IEEE 802.11g standard was released in 2003 to extend the data rate in IEEE 802.11b [41]. IEEE 802.11g is the third modulation standard for WLANs operating at 2.4 GHz such as IEEE 802.11b. However, IEEE 802.11g uses the same physical layer technology in IEEE 802.11a, which provides data rate of up to 54 Mbps. The compatibility with IEEE 802.11b products is still required to protect investments [39].

The 802.11g standard offers 14 channels at 2.4 GHz, but only 11 channels are available to users according to regulations of the respective country. Practically, Channel 1 at 2.412 GHz, Channel 6 at 2.437GHz, and Channel 11 at 2.462GHz are the only three non-overlapping channels.

3.1.4 IEEE 802.11n

The IEEE 802.11n standard is a modified standard after IEEE 802.11a/b/g with enhancements in WLAN reliability, range, and data rate. High Throughput (HT) enhancements can increase the data rate from 54 Mbps to 600 Mbps. IEEE 802.11n products operate at 2.4 GHz and 5 GHz bands whereas the 5 GHz band is optional [42].

As mentioned in Subsection 2.2, MIMO has advantages, IEEE 802.11n specifies antenna configurations of up to 4×4. By using MIMO more information can be transmitted and the data rate can be increased. The multiplexing gain is limited by the minimum number of the antennas at both sides.

The IEEE 802.11 a/b/g standards use channels that have a bandwidth of 20 MHz. The IEEE 802.11n standard added another feature, where a 40 MHz channel bandwidth can be used as well. The 40 MHz channel bandwidth provides twice the data rate [39].

3.1.5 IEEE 802.11ac

IEEE 802.11ac is an evolution of IEEE 802.11n, the aim is to achieve Very High Throughput (VHT) at the 5 GHz band by having up to 8×8 MIMO system at a 160 MHz channel bandwidth, multi-user MIMO, and high-density modulation of up to 256-QAM. The data rate will increase at least to 1 Gbps.

In case IEEE 802.11ac products operate at 80 MHz channel bandwidth, single spatial stream, and 64-QAM. The data rate will reach 293 Mbps. Other products use optional 160 MHz channel bandwidth, eight spatial streams and 256-QAM. The data rate will reach to 7 Gbps [39].

3.2 Operating Modes of IEEE 802.11n

IEEE 802.11n wireless modes specify the communication for the wireless devices. There are different operating modes, which are used for different purposes as we will explain in the following sections.

3.2.1 Infrastructure Mode

Most wireless devices use infrastructure mode, where one station acts as an access point and the other stations are clients. The communication between the clients is done through the access point only, there is no direct connection between the clients themselves. The access points provide network name SSID and other network configurations to the clients in order to establish the connection between them.In example of an access point is the home router, which operates in the infrastructure mode.

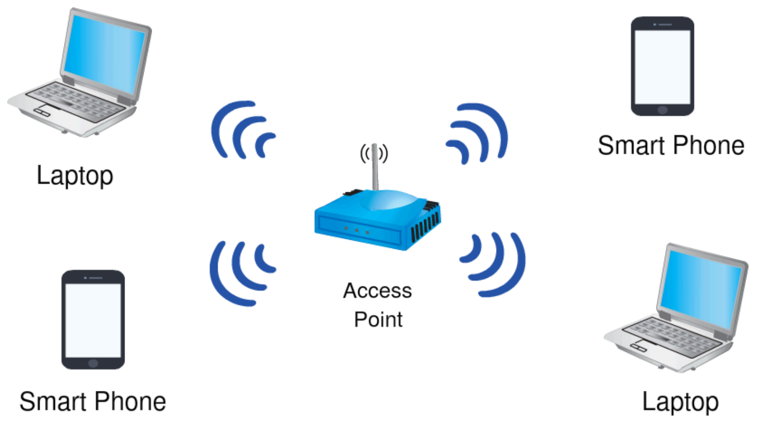

Figure 8: Infrastructure mode

In case an Internet connection is required, the access point is connected via a cable to the wired network to allow all other clients to access the Internet. Other access points could be joined to the wireless network to increase the wireless range and support more clients.

3.2.2 Ad-hoc Mode

The ad-hoc mode is to connect devices in a peer-to-peer fashion, so the access points are not required where only the wireless devices are used for the connection. Ad-hoc networks are commonly used to share files between two stations for a short time.

Smart Phone
Laptop

Figure 9: Ad-hoc mode

Some wireless devices do not support the ad-hoc mode due to hardware restrictions. However, the advantage is that just a few stations are required to communicate with one another without an access point.

3.2.3 Monitor and Injection Modes

Some special wireless network cards such Intel 5300 adapter operates in monitor mode along with modified firmware. The adapter can monitor and sniff the IEEE 802.11 channel between the stations without a network connection.

The monitor mode adaptor can operate also in injection mode, where it injects custom frames to other stations. These frames are structured in a way to be transmitted as IEEE 802.11 data frames. This is much simpler for some experiments if the goal is to collect measurements between stations without the need of IEEE 802.11 network or exchanging IP traffic.

3.3 IEEE 802.11n Frame Format including CSI Data

The IEEE 802.11 frame format includes a set of fields that contain octets to represent a specific data information [41]. The IEEE 802.11 frame format has three basic components:

- frame header includes frame control, duration/ID, MAC addresses, sequence control, QoS Control, and HT Control fields,

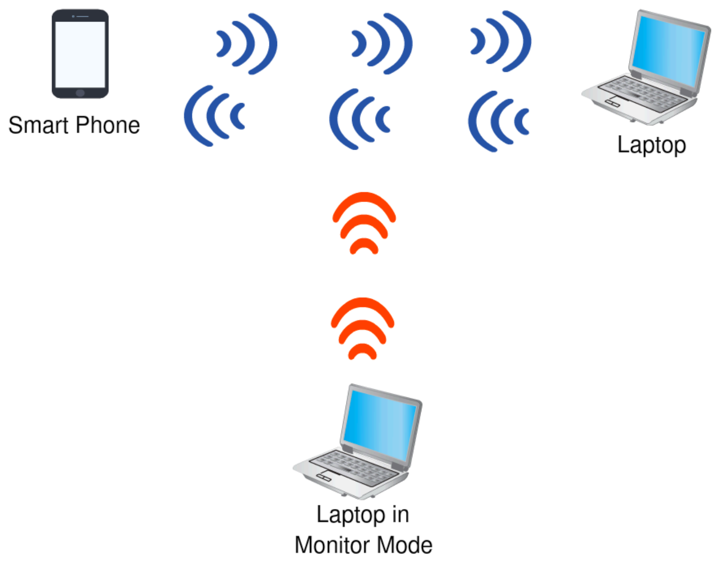

Figure 10: Monitor mode

- a variable length frame body that holds information about the frame type and the original data,
- the Frame Check Sequence (FCS),

The components and subfields have various octet lengths according to their function as seen in Fig. 11.

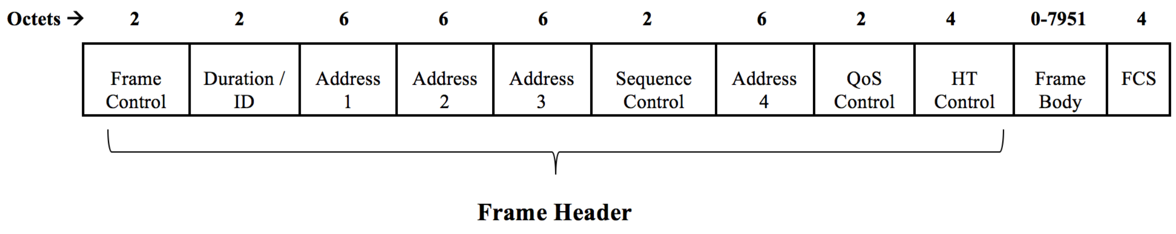

Figure 11: IEEE 802.11 frame format

Other fields can exist in some certain frames. In the next subsections, we will see the fields that are related to the CSI measurements, these fields will help us calculate the CSI values in a proper way.

3.3.1 Frame Header

3.3.1.1 Frame Control

The Frame Control field contains 16 bits. These bits are distributed to the following subfields: Protocol Version (2 bits), Type (2 bits), Subtype (4 bits), To DS (1 bit), From DS (1 bit), More Fragments (1 bit), Retry (1 bit), Power Management (1 bit), More Data (1 bit), Protected Frame (1 bit), and Order (1 bit) [41].

3.3.1.2 Duration/ID

The duration value in microseconds defines the updates of the network allocation vector for each service set transition. The duration value is used in the network allocation vector calculations to prevent collisions. The duration value decreases every time by a microsecond and the service set transition cannot access the medium when the network allocation vector is bigger than zero [41].

3.3.1.3 Address Fields

The address fields represent the physical or Medium Access Control (MAC) addresses of the following stations: Address 1 contains the Basic Service Set Identifier (BSSID), which represents the access point in infrastructure mode, Address 2 is the source address, Address 3 is the destination address, and Address 4 field contains of the station address, which transmits or receives the radio frequency frames. Each address field has six octets that are commonly written as 48 bits in hexadecimal form. Some address fields are not required in certain frame types [41].

3.3.1.4 Sequence Control

This field includes two subfields of in total 16 bits, where the fragment number subfield is 4-bit in length and the sequence number consists of 12 bits. The sequence control field is very important to recognize the packets or the frames that belong to the same session. Figure 12 depicts the Sequence Control field.

Figure 12: Sequence control field

Fragment Number

The fragment number subfield represents each fragment number of a frame, which consists of four bits. This number is set to zero in the first frame and is increased by one for the next fragment of that frame. This subfield keeps having the same number in case of a retransmission of the fragment [41].

Sequence Number

The sequence number subfield consists of 12 bits, where it declares the frame sequence number. The sequence number starts from zero and is incremented by one, where the range is from 0 to 4097. All fragments of the high-level packet have the same sequence number. In a case of frame retransmission, the subfield remains constant [41].

3.3.1.5 QoS and HT Control Fields

The QoS Control field is present only in QoS frames and identifies the Quality of Service (QoS). This field consists of 16 bits, which are divided in five to eigth subfields [41]. The HT Control field is included into a control wrapper frame and in QoS data and management frames. The High Throughput (HT) frames do only exist in 802.11n frames [41].

3.3.2 CSI Data

The CSI data is formed immediately after the frame header, which contains the following fields:

- **Timestamp Field**

  Timestamp is a time when a machine records an event that does not represent the event time itself. However, the timestamp is recorded by the computers and it should be close to the real event time, sometimes the difference is negligible. In CSI data, the time stamp field has 32 bits and it is used when the CSI measurements are written into the log file [3]. The timestamp is used in many other applications such as VoIP technology where the voice packets have timestamps to overcome the delivery of the non-ordered packets.

- **Beamforming Counter Field**

  It counts the total number of beamforming measurements, where the driver simply records the measurements and sends them to userspace. The abbreviation is "*Bfee count*" and it contains 16 bits. so these can be used to detect measurements that were dropped in this pipe [3].

- **N_{tx} and N_{rx} Fields**

  N_{tx} and N_{rx} represent the number of the transmit and receive antennas, respectively. For example, let us consider N_{tx}=1 and N_{rx}=3. Then, a single stream was transmitted by a single transmit antenna and three antennas are used to receive the signals. Each field of N_{tx} and N_{rx} is represented by eight bits in CSI data [3].

- **RSS, Noise, and AGC Fields**

  The Received Signal Strength (RSS) exists in CSI data for each antenna as RSS A, RSS B, and RSS C where each of them is 8 bits in length. RSS is measured by the receiving network card at the input of each antenna port [3]. These measurements are performed during the packet preamble. The RSS value is shown in dB without considering the noise and the Automatic Gain Control (AGC). RSS is combined with AGC values to obtain RSS in dBm. In case we consider the noise, it is represented by the Signal-to-Noise Ratio (SNR) [3]. The noise and AGC are represented by 16 bits in CSI data, where each one of them uses eight bits.

- **Perm Field**

  It has a length of 8 bits and provides information on how the network card permuted the signals from the three receive antennas into the three RF chains [3]. For example, the value of [213] explains that Antenna B was sent to RF Chain A, Antenna A to Chain B, and Antenna C to Chain C. The antenna selection module of the network card is responsible for this operation [3].

- **Rate Field**

  It is represented by 16 bits in CSI data, it describes at which data rate the packet was sent. The antenna bits are omitted, as there is no way for the receiver to know which transmit antennas were used [3].

- **CSI Matrix**

  It contains CSI values as a 3D matrix tensor where the dimensions are "$N_{tx} \times N_{rx}$ *x 30*". The third dimension is across 30 subcarriers in the Orthogonal Frequency Division Multiplexing (OFDM) channel [3]. In case of a 20 MHz bandwidth, it is about the half of the OFDM subcarriers and in a 40 MHz bandwidth is about the quarter of the OFDM subcarriers. Each entry in the matrix is a complex number with a length of 8 bits. This entry defines the amplitude and the phase of a signal path between transmitter and receiver.

3.3.3 Generation of Adaptive Bit Streams

As mentioned in paragraph 2.4.3.1, Alice and Bob are sending probes to measure the changes that happen in the wireless channel over time to establish a shared secret key.

The problem with wireless networks is the half duplex communication, where Alice and Bob cannot exchange data simultaneously. The solution to this problem is to send the message in one direction at a specific time, where the time difference between the two-directional channel measurements shall be smaller than the coherence time.

We use *ping* messages to achieve the small time difference between the two directional channel measurements. The receiving time difference on both sides should be small enough to generate symmetric keys.

Fig. 13 depicts the timing diagram for the ping messages between Alice and Bob.

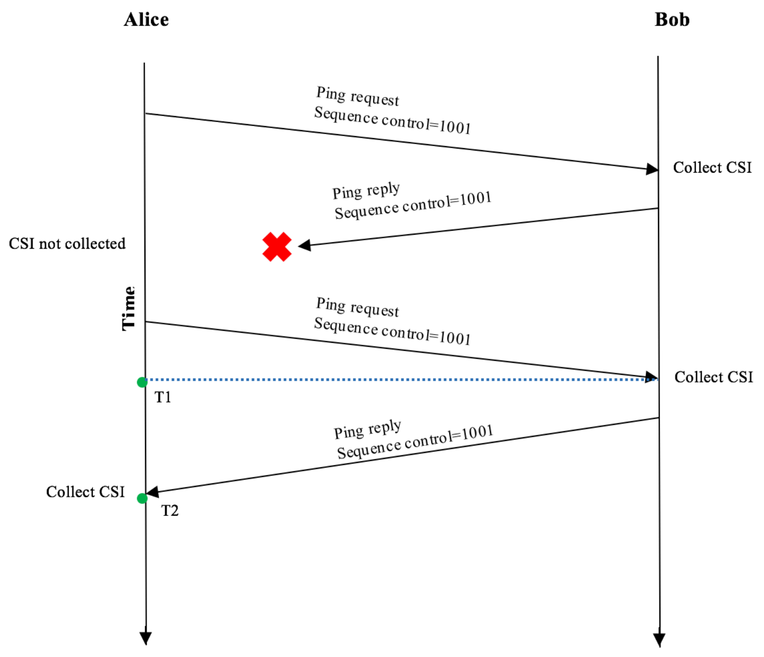

Figure 13: Timing diagram for ping request and reply

The sequence control field is a key factor to distinguish between the receiving packets of different ping messages, whereby the request and reply should have the same sequence control number if they belong to the same message. We can see in Fig. 13 that the acknowledgment or the reply packet was not received by Alice at the beginning of the session. This means, Alice has to repeat the request packet until she obtains the acknowledgment with the same control number from Bob. CSI measurements will be only collected when the receiver obtains the transmitted packet.

As seen in Fig. 13, T1 and T2 represent Bob and Alice receiving time, respectively. The intervals between the receiving times at Alice and Bob have to be smaller than the coherence time to guarantee that there are no changes in the channel.

4 Measurement Results

4.1 Measurement Setup

Our experiment required three laptops that have Linux 14.04 LTS as an operating system with kernel version 3.2 installed. Two laptops were acting as the legitimate users, Alice and Bob, where the third one was acting as an eavesdropper (Eve). We have used a tool called the CSI tool [3], which was designed for Intel Wi-Fi Wireless Link 5300 NIC, using a custom firmware and Linux wireless drivers. With using the IWL5300 NIC, we could collect 802.11n channel state information in a format that provides line sequence of 30 subcarriers over time. Each subcarrier channel transfer characteristic is represented by complex values, where we could specify the amplitude and phase of the signal path between the transmitter and the receiver.

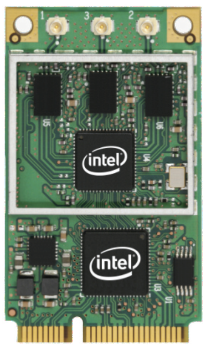

Figure 14: IWL5300 NIC [3]

One of the laptops was used as an access point to provide a wireless connection, using Hostapd software [43]. In order to maintain control over the wireless interface during our experiment, we used the command-line utility "*iw*" [44] and disabled the *NetworkManager* from controlling the wireless card. We used infrastructure mode as a wireless operating mode at a frequency of 2.5 GHz, and we configured the Hostapd software to operate at the wireless channel number 11. Fruthermore, we enabled the IEEE 802.11n standard to collect CSI measurements. For this, we used only a single antenna (SISO transmission). In order to establish a full communication between laptops, a static IP address in Class C was set for each wireless network cards.

After the described settings and configurations, we had to write a C code in order to obtain the frame headers of the collected data along with CSI measurements. After the start of the Hostapd software and having connected the laptops, we sent request messages from one laptop, and this laptop received the reply messages from its destination. For this, we used a data rate of 100 packets per second to achieve the scenario that Alice and Bob are sending very short packets, where the time difference between the receiving messages on both sides should be smaller than the coherence time of the wireless channel. A Ping command-line was run in the above scenario to transmit and receive the data.

We started logging the CSI measurements on each laptop by using *log-to-file* of the CSI tool, which is a command-line tool to write the CSI obtained via the NIC to a file. It also printed a message for each received packet to the terminal. We used MATLAB to read the collected data. Some packets were duplicated because their acknowledgment messages were lost, and other packets were not in the right order due to the time delay of the round trip. Then, we had to write MATLAB scripts to order and filter the packets according to their sequence control number in the frame header and the timestamp field in CSI data. We chose only those packets that had the same sequence control numbers at the transmitter and the receiver, and we selected the packets that had smaller timestamp difference between the duplicated packets as mentioned in Section 3.3.3.

4.2 Evaluation

We conducted our experiments under several scenarios in different environment settings, as illustrated in Table 2. We measured the amplitude and the group delay at Alice, Bob, and Eve in all scenarios. The amplitude values at a specific subcarrier index vs. time were plotted. We also depicted the group delay, which represents the negative derivative (slope) of the phase response vs. frequency, whereby we could observe the relative delay at different subcarriers.

The correlation factors between Alice/Bob and Alice/Eve were calculated and plotted vs. frequency. The correlation value is equal $+1$ in case of a positive linear relationship where the data are correlated, -1 in the case of a negative linear relationship. It approaches zero when the correlation is weak.

Scenario	**Status**	**Environment**	**Distance between Alice and Eve**
A	Static	Stationary	2 m
B	Static	Non-stationary	5 m.
C	Mobile	Quasi-stationary	Bob walks along the corridor (6 m - 12 m)

Table 2: The settings of the scenarios

- Scenario *A*

  The laptops of Alice, Bob, and Eve were located inside the office. Alice and Bob are kept at a distance of 2 m, like with Alice and Eve, without any object moving in between.

  Figure 15 shows the amplitude variations of the CSI measurements collected inside the office. Although we can note that Alice and Bob have different amplitude values, their variation trends look similar. We can see also that Eve's amplitude values look totally different from the amplitude values of Alice and Bob.

Figure 16 illustrates the group delay of the CSI measurements in scenario *A*. On the one hand, Alice and Bob measure a somewhat similar shape, but the group delay is still not symmetric in both directions. On the other hand, Eve experiences significantly different group delay values than Alice and Bob.

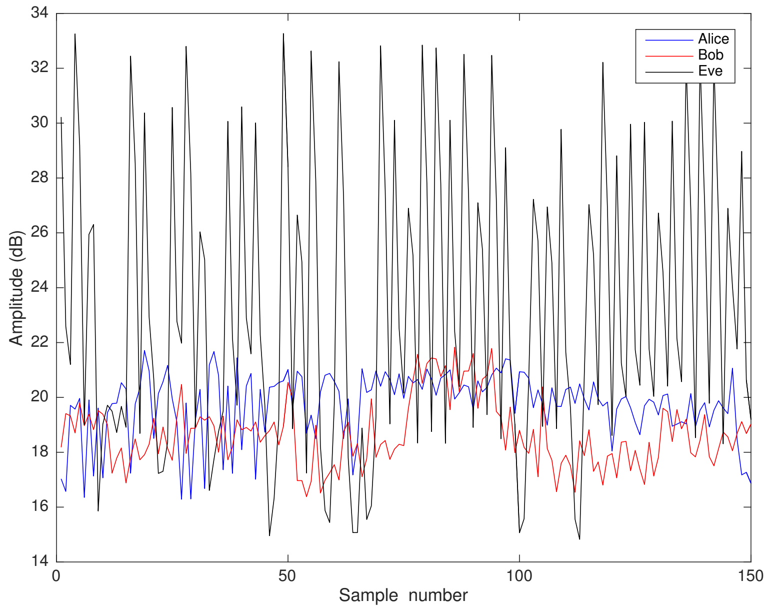

Figure 15: Amplitude in Scenario A

Figure 17 shows the correlation between Alice and Bob, and Alice and Eve. We note that all values are between 0.4 and 0.8 between Alice and Bob, and the correlation values between Alice and Eve lie between -0.4 and 0.2 for the 30 subcarriers. The results show that Eve's key definitely was dependent on the measurement of the legitimate users. Although Eve and Bob are only 2 m away from Alice and Bob is still more correlated than Eve.

These results are probably considered by the changes in the wireless channel due to environment influences and hardware differences in the laptops, which are non-reciprocal. The results also show the non-reciprocal channel of Eve.

- Scenario *B*

  Same as *scenario A* with moving objects in between. In figures 18 and 19, we observe that the amplitude and the group delay curves for Alice and Bob do not follow each other. This indicates that CSI measurements are very sensitive to environmental changes. However, the CSI measurements of Alice and Bob show a better correlation than Eve's measurements as seen in Figure 20.

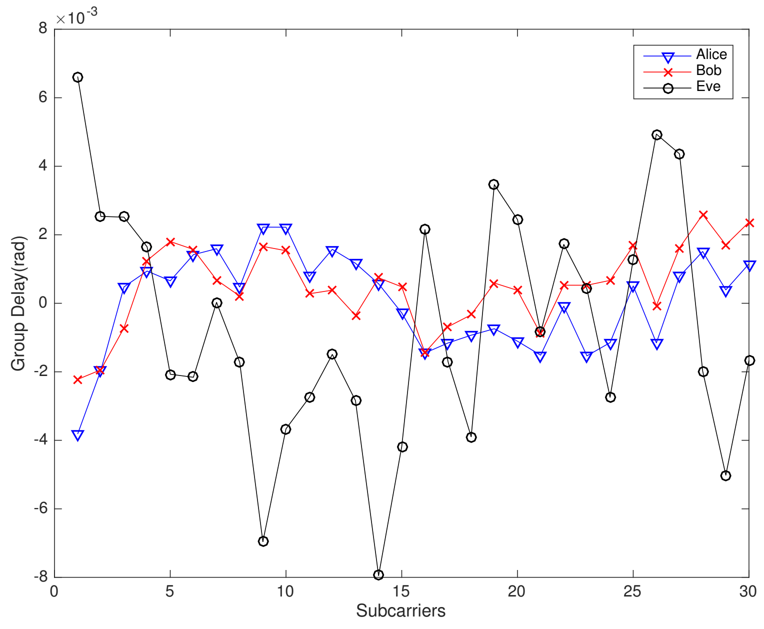

Figure 16: Group delay in Scenario A

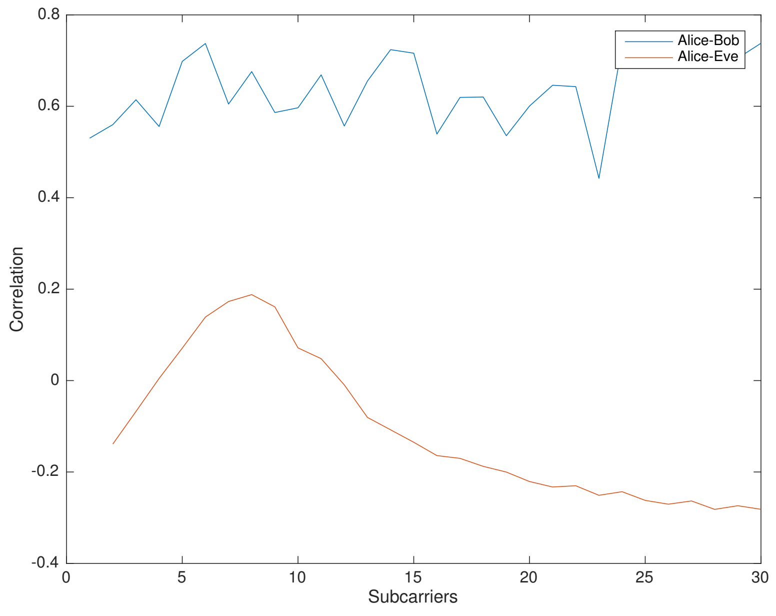

Figure 17: Correlation with Scenario A

- Scenario *C*

  Alice and Eve were 2 m apart inside the office. Bob walked slowly along the corridor without any objects moving in between.

  In Figure 21, the amplitude values of Alice and Bob vary widely ($5 - 25$ dB).

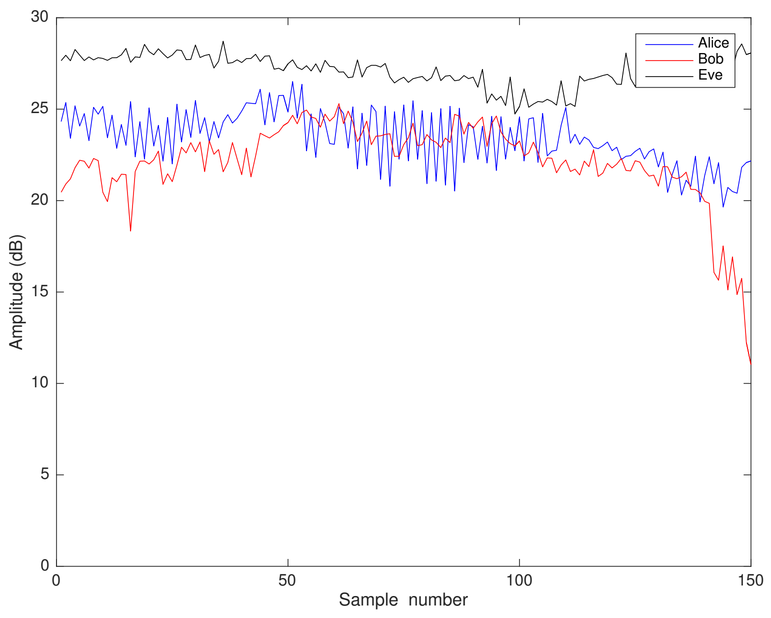

Figure 18: Amplitude in Scenario B

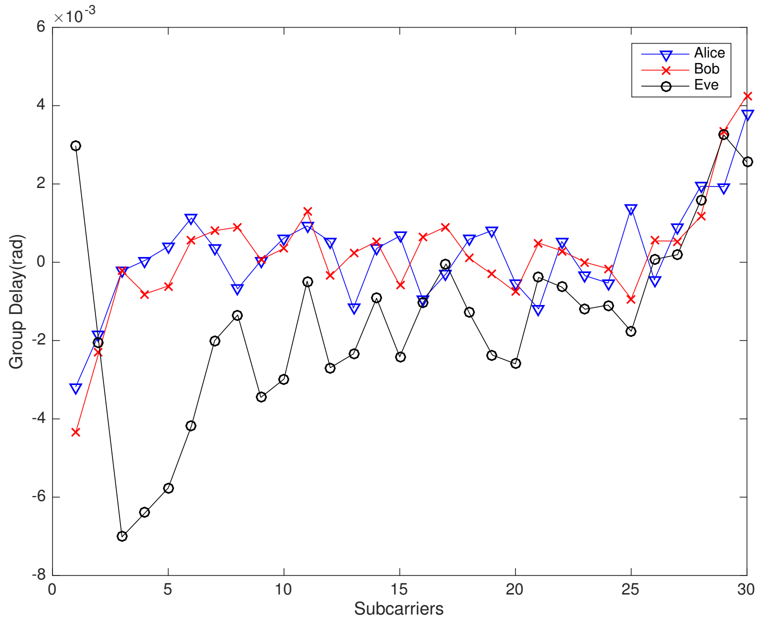

Figure 19: Group delay in Scenario B

Figure 22 also shows that the group delay values of $AlicetoBob$ and $BobtoAlice$ are not similar. This may happen because of the variations of different environmental factors.

It is interesting to note that Eve's observation is quite similar to Alice's and Bob's as

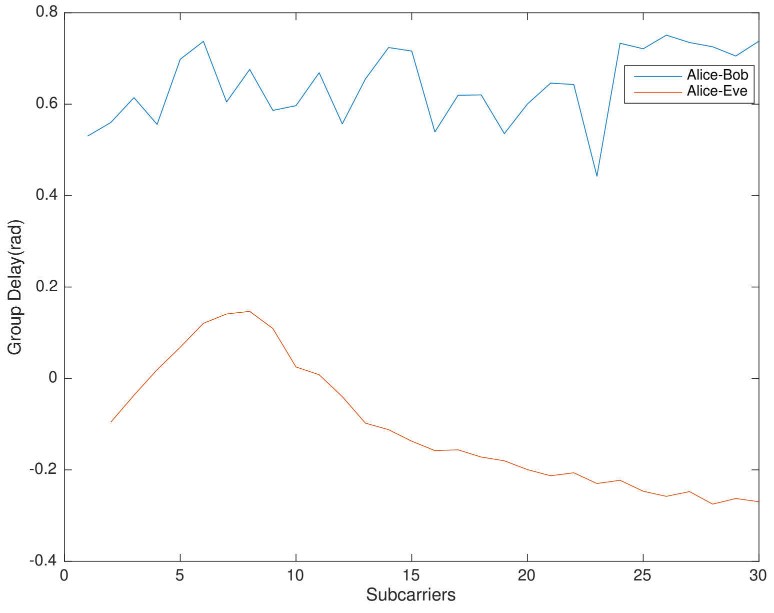

Figure 20: Correlation with Scenario B

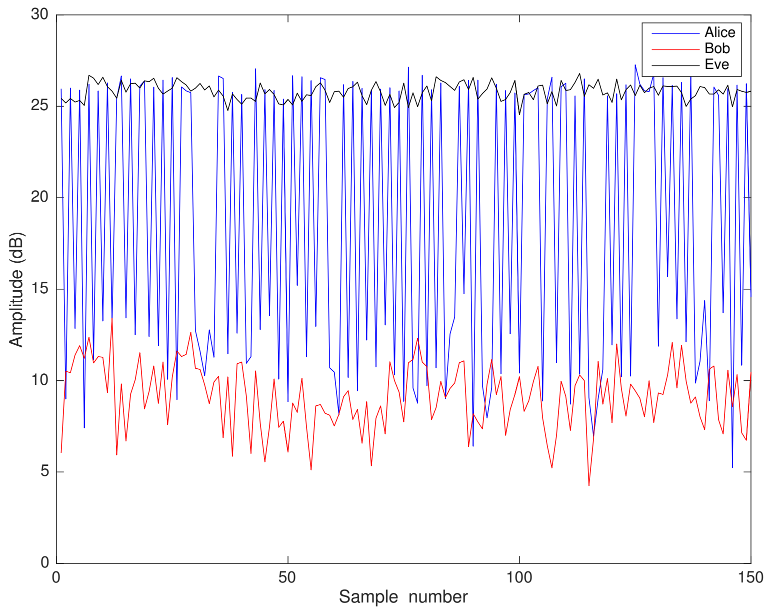

Figure 21: Amplitude Scenario C

far as the group delay is conserved. This is in line with the low correlation (Fig. 23) of larger distances.

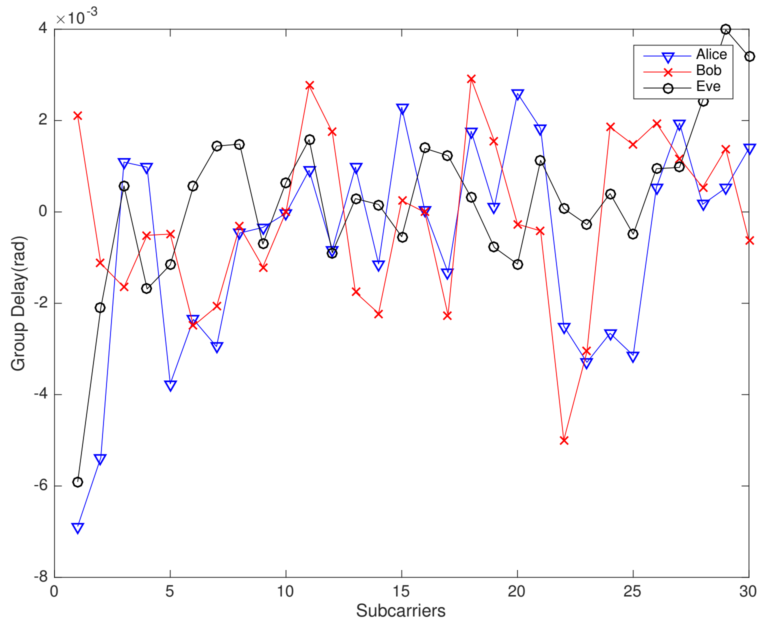

Figure 22: Group delay Scenario C

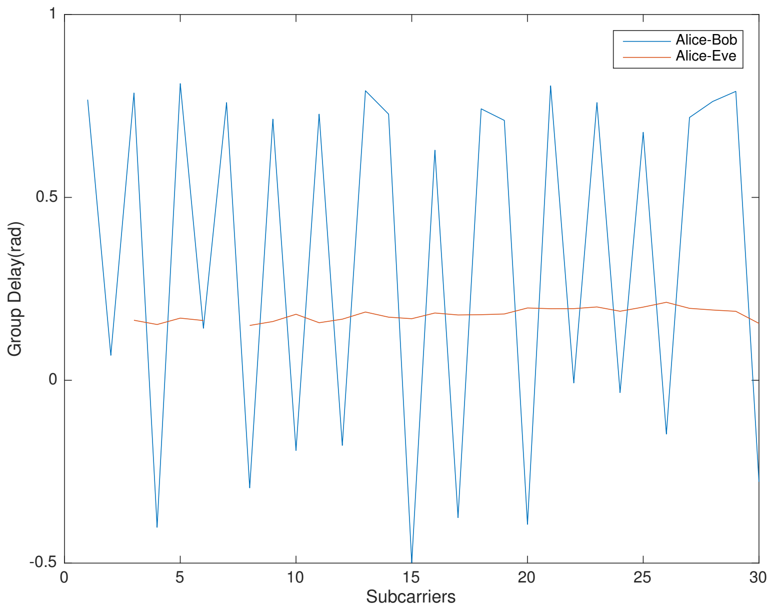

Figure 23: Correlation with Scenario C

5 Conclusions

In this thesis, we tried to practically implement physical layer key generation by gathering IEEE 802.11n traces. We used infrastructure mode and ping command to send signals between the communicating parties. We evaluated CSI measurements experimentally with the *Intel Wi-Fi Link 5300 wireless NIC*. We implemented different scenarios to collect CSI measurements using the so-called CSI tool by exploiting the properties of the wireless channel.

In our experiments, we noticed that it is difficult to generate a common key using the CSI tool due to some limitations the noise interference, manufacturing variations, and hardware limitations. In our scenarios, we analyzed the amplitude and the group delay for Alice, Bob, and Eve. We observed that Alice and Bob have different CSI values, but their variation trends are similar. We calculated the correlation coefficients between Alice and Bob, and Alice and Eve. The results showed that the eavesdropper experiences low correlation to Alice or Bob compared to the legitimate links between Alice and Bob.

Practically, CSI measurements are very sensitive to environmental changes. We found that the CSI tool is instable in infrastructure mode, where it is necessary to send and receive fast packets to ensure the channel reciprocity. It needed quite some effort to understand the CSI tool and adapt the programs to our needs. Finally, we found out that the relation between the two-sided channel measurements was too weak and it was not possible to explain the differences. However, online, other colleagues meanwhile reported similar problems.

The original goal to really do key extraction and key reconciliation using the chosen WLAN cards could not be met, but this was due to deficit of the WLAN cards and the computer environment, which could not be fixed.

A Installation of Linux 802.11n CSI Tool

- **System requirements**

  A tool so-called CSI Tool that works only with Linux operating systems and a commercial wireless network card *Intel Wi-Fi Wireless Link 5300 802.11n* collects CSI measurements along with CSI Tool.

- **Install the custom Linux Kernel**

  1. Install the required packages

  We need a package called *git-core* package.

  ```
  sudo apt-get -y install git-core kernel-package fakeroot build-essential
  ncurses-dev
  ```

  We grab the latest source code using the git command, the fakeroot tool is useful to build *.deb files.* Then, *build-essential* for building kernel and *ncurses-dev* by menuconfig.

  Some libraries are needed to compile the userland which interacts with the kernel.

  ```
  sudo apt-get -y install libnl-dev libssl-dev
  ```

  For configuring Linux wireless devices and enable some modes, *iw* tool is used.

  ```
  sudo apt-get -y install iw
  ```

  2. Configure and install a custom kernel

  ```
  git clone -b csitool-stable git://github.com/dhalperi/linux-80211n-csitool.git
  ```

  We download the supplemental files which contain of C codes, MATLAB scripts and a firmware.

  ```
  git clone git://github.com/dhalperi/linux-80211n-csitool-supplementary.git
  ```

  ```
  cd linux-80211n-csitool
  ```

  To use the optimized kernel configuration.

  ```
  make oldconfig
  ```

  Enable system-specific hardware.

  ```
  make menuconfig
  ```

  After the previous settings, the kernel modules have to be compiled and installed as the following:

```
make -j3 bzImage modules
sudi make
sudo make install modules_install
sudo mkinitramfs -o /boot/initrd.img-‘cat include/config/kernel.release‘
čat include/config/kernel.release‘
sudo update-grub
```

The Linux kernel headers will be installed in order to export the header files to use from a userspace

```
make headers_install
sudo mkdir /usr/src/linux-headers-‘cat include/config/kernel.release‘
sudo cp -rf usr/include /usr/src/linux-headers-‘cat include/config/
kernel.release‘/include
```

Now, we restart our the laptop by using:

```
sudo update-grub
sudo reboot
```

Grub will discover the new kernel and reboot will automatically run it.

We check the version of our own new kernel by using:

```
uname -r
```

3. Install the custom firmware

The custom firmware is located under *linux-80211n-csitool-supplementary* file.

```
cd ∼/linux-80211n-csitool-supplementary/firmware
sudo cp iwlwifi-5000-2.ucode.sigcomm2010 /lib/firmware/
sudo cp iwlwifi-5000-2.ucode.sigcomm2010 /lib/firmware/iwlwifi-5000-2.ucode
```

In last commands, we copied the firmware to */lib/firmware/*.

4. Install the userspace logging utility

Now, we will install userspace tool that is located under *linux-80211n-csitool supplementary /netlink/* file. Userspace refers to the libraries and programs that interact with our kernel.

```
cd ∼/linux-80211n-csitool-supplementary/netlink/
sudo make
```

5. Download and configure Hostapd software

We need a laptop that is acting as an access point in order to provide wireless connection, so we chose *Hostapd software*

```
cd sudo git clone git://w1.fi/srv/git/hostap-07.git
cd hostap-07/hostapd
sudo cp ~/linux-80211n-csitool-supplementary/hostap-config-files/hostap-dot
config ~/hostap-07/hostapd/hostapd.config sudo make
sudo cp ~/linux-80211n-csitool-supplementary/hostap-config-files/hostapd.conf-real
~/hostap-07/hostapd/hostapd.conf
```

There is already basic configurations inside *hostapd.conf*, which can be modified.

```
channel=country_code=DE
driver=nl80211
hw_mode=g
interface=wlan0
ssid=YOURSSID
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_passphrase=YOURPASSPHASE
```

6. Load *iwlwifi* module

We use in the next step *modprobe*, which is a Linux program to add or remove loadable kernel modules.

```
sudo modprobe iwlwifi
sudo modprobe -r iwlwifi mac80211
sudo modprobe iwlwifi connector_log=0x5
```

The *iwlwifi* module is reinserted to load and set the module of beamforming and frame header.

To collect only CSI measurements

```
sudo modprobe iwlwifi connector_log=0x1
```

for CSI and frame Headers

```
sudo modprobe iwlwifi connector_log=0x5
```

Hence, the network manager should not control the wireless settings. Therefore, we have to be sure that *managed=false* in *NetworkManager.conf*. We can access this file through:

```
sudo gedit /etc/NetworkManager/NetworkManager.conf
```

Our wireless network card *Intel Wi-Fi Wireless Link 5300 802.11n* has three antennas. In case of Single-Input Single-Output (SISO) is needed, we have to terminate the other two antennas and keep one to transmit and receive the signal.

```
echo 0x2 | sudo tee /sys/kernel/debug/ieee80211/phy0/iwlwifi/
iwldvm/debug/rx_chains_msk
```

7. Network Configuration

Now, we set the network configurations by providing the wireless network card in each laptop an IP address to establish the wireless network.

```
sudo gedit /etc/network/interfaces
```

We chose static IP addresses instead of automatic IP addresses (from DCHP server). This an example of the settings in one laptop with an IP address, a subnet mask, and the gate way address.

```
# The wifi network interface
auto wlan0
iface wlan0 inet static
address 192.168.3.1
netmask 255.255.255.0
network 192.168.3.0
```

To verify the IP address on the laptop use:

```
ifconfig wlan0
```

8. Establish the Connection and Collect CSI

We run the Hostapd software on the laptop, which had been installed under:

```
sudo ~/hostap-07/hostapd/hostapd  /hostap-07/hostapd/hostapd.conf
```

To connect the wireless network using the Terminal:

```
sudo iwconfig wlan0 essid YOURSSID
```

To log the CSI measurements, we have to use `log_to_file` as the following:

```
cd ~/linux-80211n-csitool-supplementary/netlink
sudo ./log_to_file NAME.dat
```

The *NAME.dat* file will be read by the MATLAB.

Ping messages are needed to send and receive some packets between two communicating laptops

```
ping 192.168.3.1
```

To make sure that we obtain reply messages during the transmission. A messages will be printed on the Terminal under `./log_to_file` NAME.dat. Then, we can realize that we have started recording our CSI data.

B MATLAB Scripts and C Code

B.1 MATLAB Scripts

```matlab
%   READ_BF_FILE Reads in a file of beamforming feedback logs.
%   This version uses the *C* version of read_bfee, compiled with
%   MATLAB's MEX utility.
%
%   We edited the code in order to be enabled to read the header
%   of the CSI packets.
%
%    Created by Daniel Halperin <dhalperi@cs.washington.edu>
%    Edited by Munder Hamruni <monder.hamruni@yahoo.com>

function [ret, ret2] = read_bf_file(filename)
%% Input check
error(nargchk(1,1,nargin));

%% Open file
f = fopen(filename, 'rb');
if (f < 0)
    error('Couldn''t open file %s', filename);
    return;
end

status = fseek(f, 0, 'eof');
if status ~= 0
    [msg, errno] = ferror(f);
    error('Error %d seeking: %s', errno, msg);
    fclose(f);
    return;
end
len = ftell(f);

status = fseek(f, 0, 'bof');
if status ~= 0
    [msg, errno] = ferror(f);
    error('Error %d seeking: %s', errno, msg);
    fclose(f);
    return;
end

%% Initialize variables
ret2={};
```

```matlab
ret={};
cur = 0;
count = 0;
broken_perm = 0;
triangle = [1 3 6];
b = 0;

%% Process all entries in file
% Need 3 bytes -- 2 byte size field and 1 byte code
while cur < (len - 3)
  % Read size and code
%   disp(['----']);
  field_len = fread(f, 1, 'uint16', 0, 'ieee-be');
  code = fread(f,1);
  cur = cur+3;

  % If unhandled code, skip (seek over) the record and continue
  if (code==193)
      mpdu= fread(f, field_len-1, 'uint8=>uint8');
      cur = cur + field_len - 1;

      if (length(mpdu) ~= field_len-1)
          fclose(f);
          return;
      end

      field_len = fread(f, 1, 'uint16', 0, 'ieee-be');
      code = fread(f,1);
      %display('code is-----')
      cur=cur+3;

      if (code == 187) % get beamforming or phy data
          count = count + 1;
          ret2{count} = read_mpdu2(mpdu);
          bytes = fread(f, field_len-1, 'uint8=>uint8');
          cur = cur + field_len - 1;

          if (length(bytes) ~= field_len-1)
              fclose(f);
              return;
          else
              ret{count} = read_bfee(bytes);
              perm = ret{count}.perm;
              Nrx = ret{count}.Nrx;
              if Nrx == 1 % No permuting needed for only 1 antenna
             % continue;
              end
              if sum(perm) ~= triangle(Nrx) % matrix does not contain default values
                  if broken_perm == 0
                      broken_perm = 1;
```

```
                        fprintf('WARN ONCE: Found CSI (%s) with Nrx=%d and
    invalid perm=[%s]\n', filename, Nrx, int2str(perm));
                    end
                else
                    ret{count}.csi(:,perm(1:Nrx),:) = ret{count}.csi(:,1:Nrx
    ,:);
                end
            end
        else

            fseek(f, -3, 'cof');
            cur = cur-3;

        end

    else
        fseek(f, field_len-1, 'cof');
        cur = cur +field_len -1;
    end

end
ret = ret(1:count);

%% Close file
fclose(f);
end
```

```

%   Function to calculate the group delay

function t = GroupDelay(f,phase)
elements = size(phase);
k = elements(2);

for n = 2:k-1
    t(n) = (-1/720) * (((phase(n) - phase(n - 1)) / (f(n) - f(n - 1)))...
        + ((phase(n + 1) - phase(n)) / (f(n + 1) - f(n))));
end
t(1) = (-1/360) * (((phase(2) - phase(1))/(f(2) - f(1))));
t(k) = (-1/360) * (((phase(k) - phase(k - 1))/(f(k) - f(k - 1))));
```

```
%   This script is to compare the sequence numbers and the timestamp
%   of the received packets at the both sides.
%
%   Munder Hamruni <monder.hamruni@yahoo.com>

  a=csi_trace_first5;
  b=csi_trace_first6;
  lenA=length(a);
  lenB=length(b);
     j=1;
```

```
for i=1:lenA
  seqNrA(i)=payloads5{i}.sequence_control;
     % timeStampA(i)=csi_trace_first5{i}.timestamp_low;
end
for i=1:lenB
  seqNrB(i)=payloads6{i}.sequence_control;
      % timeStampB(i)=csi_trace_first6{i}.timestamp_low;
end
[minLen, AB]=min([lenA lenB]);

indicesVect=[];
toKeepA=[];
toKeepB=[];

for i=1:minLen
  if AB==1
    clear indices;
    currentSeq=seqNrA(i);
    [vals indices]=find(seqNrB==currentSeq,1);
    if ~isempty(indices)
      indicesVect=[indicesVect indices];
      toKeepA=[toKeepA i];
      end
  else
    clear indices;
    currentSeq=seqNrB(i);
    [vals indices]=find(seqNrA==currentSeq,1);
    if ~isempty(indices)
      indicesVect=[indicesVect indices];
      toKeepB=[toKeepB i];
    end
  end
end

if AB==1
  for i=1:length(indicesVect)
    bTrim(i)=b{indicesVect(i)};
    bSequenceTrim(i)=payloads6{indicesVect(i)};
    aTrim(i)=a{toKeepA(i)};
    aSequenceTrim(i)=payloads5{toKeepA(i)};
          timeStampB(i)=bTrim(i).timestamp_low;
          timeStampA(i)=aTrim(i).timestamp_low;
          diffTime(i)=abs(timeStampA(i)-timeStampB(i));

          if diffTime(i) <= 80 % This difference has to be assumed
              NewB(j)=bTrim(i); % because the tool read it every time in
              NewA(j)=aTrim(i); % different way.
              j=j+1;
          end
  end
else
```

```
    for  i=1:length(indicesVect)
      aTrim(i)=a{indicesVect(i)};
      aSequenceTrim(i)=payloads5{indicesVect(i)};
      bTrim(i)=b{toKeepB(i)};
      bSequenceTrim(i)=payloads6{toKeepB(i)};
            timeStampB(i)=bTrim(i).timestamp_low;
            timeStampA(i)=aTrim(i).timestamp_low;
            diffTime(i)=abs(timeStampA(i)-timeStampB(i));

            if  diffTime(i) <= 80
                NewB(j)=bTrim(i);
                NewA(j)=aTrim(i);
                j=j+1;
            end
      end
  end

for  n=1:length(aTrim)
    for  i=1:30
            data1{n}=get_scaled_csi(NewA(n));
            data2{n}=get_scaled_csi(NewB(n));
            RE{n}=real(data1{n}(:,:,15));
            IM{n}=imag(data1{n}(:,:,15));
            RE2{n}=real(data2{n}(:,:,15));
            IM2{n}=imag(data2{n}(:,:,15));
            CSI1_A{n}=db(abs((data1{n}(:,:,15))));
            CSI2_A{n}=db(abs((data2{n}(:,:,15))));
            CSI1_P{n,i}=angle((data1{n}(:,:,i)));
            CSI2_P{n,i}=angle((data2{n}(:,:,i)));

            n=n+1;
    end
end

% This loop is to plot the group delay as a moive

for  j=1:length(NewA)

    CSI1_p1=CSI1(j,:);
    CSI2_p1=CSI2(j,:);
    %CSI3_p1=CSI3(j,:);
    %CSI4_p1=CSI4(j,:);

    Un1=unwrap(CSI1_p1);
    Un2=unwrap(CSI2_p1);
    %Un3=unwrap(CSI3_p1);
    %Un4=unwrap(CSI4_p1);
    y=1:30;

    t1= GroupDelay(y,Un1);
    t2= GroupDelay(y,Un2);
    %t3 = GroupDelay(y,Un3);
    %t4 = GroupDelay(y,Un4);

    plot(y,t1(y),'b-v',y,t2(y),'r-o');
    ylabel('Group Delay(rad)');
    xlabel('Subcarriers');
    legend('Alice', 'Bob');

    hold off;
    pause(1);

end
```

```
%   This script calculates and plots the correlation factors for set of CSI
%   measurements between the Alice and Bob and Alice and Eve over frequency.
%
%   Munder Hamruni <monder.hamruni@yahoo.com>

for n=1:30
Sig1 {n}     = cov(CSI1(:,n),CSI2(:,n));
rho1 (n)     = Sig1{n}(1,2)/sqrt(Sig1{n}(1,1)*Sig1{n}(2,2));

Sig2 {n}     = cov(CSI1(n,:),CSI3(n,:));
rho2 (n)     = Sig2{n}(1,2)/sqrt(Sig2{n}(1,1)*Sig2{n}(2,2));
end

y=1:30;
plot(y,rho2(y),y,rho1(y))
ylabel('Correlation');
xlabel('Subcarriers');
legend('Alice-Bob', 'Alice-Eve');
```

B.2 C Code

```
/* This code is to read and ordere the bits of the frame header of our
 * CSI data in Bytes form.
 *
 * Munder Hamruni, Jacobs University
 */
#include "mex.h"

/* The computational routine */
void read_mpdu(unsigned char *inBytes, mxArray *outCell)
{
    unsigned short frame_control = inBytes[0] + (inBytes[1] << 8);
  unsigned int duration_ID = inBytes[2] + (inBytes[3] << 8);
  unsigned int addr1= (inBytes[5]);
    unsigned int addr2= (inBytes[6]);
    unsigned int addr3= (inBytes[7]);
  unsigned int sequence_control = inBytes[22] + (inBytes[23] <<8);

```

```c

    unsigned short frame_control = inBytes[20] + (inBytes[21] << 8);
  unsigned int duration_ID = inBytes[22] + (inBytes[23] << 8);

    mxDestroyArray(mxGetField(outCell, 0, "frame_control"));
  mxDestroyArray(mxGetField(outCell, 0, "duration_ID"));
  mxDestroyArray(mxGetField(outCell, 0, "addr1"));
    mxDestroyArray(mxGetField(outCell, 0, "sequence_control"));


    mxSetField(outCell, 0, "frame_control", mxCreateDoubleScalar((double)frame_control));
  mxSetField(outCell, 0, "duration_ID", mxCreateDoubleScalar((double)duration_ID));
  mxSetField(outCell, 0, "addr1", mxCreateDoubleScalar((unsigned int)addr1));
    mxSetField(outCell, 0, "addr2", mxCreateDoubleScalar((unsigned int)addr2));
    mxSetField(outCell, 0, "addr3", mxCreateDoubleScalar((unsigned int)addr3));
    mxSetField(outCell, 0, "sequence_control", mxCreateDoubleScalar((unsigned int)sequence_control));

}

/* The gateway function */
void mexFunction(int nlhs, mxArray *plhs[],
           int nrhs, const mxArray *prhs[])
{
  const char* fieldnames[] = {
        "frame_control",
    "duration_ID",
    "addr1",
        "addr2",
        "addr3",
        "sequence_control"
    };
  unsigned char *inBytes;   /* A beamforming matrix */
  mxArray *outCell;   /* The cellular output */

  /* check for proper number of arguments */
  if(nrhs!=1) {
    mexErrMsgIdAndTxt("MIMOToolbox:read_bfee_new:nrhs","One input required.");
  }
  if(nlhs!=1) {
    mexErrMsgIdAndTxt("MIMOToolbox:read_bfee_new:nlhs","One output required.");
  }
  /* make sure the input argument is a char array */
  if (!mxIsClass(prhs[0], "uint8")) {
    mexErrMsgIdAndTxt("MIMOToolbox:read_bfee_new:notBytes","Input must be a
   char array");
  }

  /* create a pointer to the real data in the input matrix */
  inBytes = mxGetData(prhs[0]);

  /* create the output matrix */
  outCell = mxCreateStructMatrix(1, 1, 6, fieldnames);

  /* call the computational routine */
  read_mpdu(inBytes,outCell);

  /* */
  plhs[0] = outCell;
}
```

References

[1] G. Bagheri-Karam, "Physical-layer security in wireless communication systems," 2010.

[2] M. Vaezi, W. Shin, and H. V. Poor, "Optimal Beamforming for Gaussian MIMO Wiretap Channels with Two Transmit Antennas," *IEEE Transactions on Wireless Communications*, pp. 1–10, 2017.

[3] https://dhalperi.github.io/linux-80211n csitool/.

[4] L. Zhou and Z. J. Haas, "Securing ad hoc networks," *IEEE Network*, vol. 13, no. 6, pp. 24–30, Nov 1999.

[5] J. Zhang, T. Duong, A. Marshall, and R. Woods, "Key Generation from Wireless Channels: A Review," *IEEE Access*, vol. 4, pp. 1–1, 2016.

[6] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, ser. SP '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 197–.

[7] 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.

[8] P. P. C. Lee, J. C. S. Lui, and D. K. Y. Yau, "Distributed collaborative key agreement and authentication protocols for dynamic peer groups," *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 263–276, April 2006.

[9] Y.-S. Shiu, S. Chang, H.-C. Wu, S. Huang, and H.-H. Chen, "Physical layer security in wireless networks: a tutorial," *IEEE Wireless Communications*, vol. 18, no. 2, pp. 66–74, 2011.

[10] E. a. Jorswieck, A. Wolf, and S. Gerbracht, "Secrecy on the Physical Layer in Wireless Networks," *Trends in Telecommunications Technologies*, pp. 413–436, 2010. [Online]. Available: http://www.intechopen.com/books/trends-in-telecommunications-technologies/secrecy-on-the-physical-layer-in-wireless-networks

[11] S. N. Premnath, S. Jana, J. Croft, P. L. Gowda, M. Clark, S. K. Kasera, N. Patwari, and S. V. Krishnamurthy, "Secret key extraction from wireless signal strength in real environments," *IEEE Transactions on Mobile Computing*, vol. 12, no. 5, pp. 917–930, May 2013.

[12] J. Zhao, W. Xi, J. Han, S. Tang, X. Li, Y. Liu, Y. Gong, and Z. Zhou, "Efficient and secure key extraction using CSI without chasing down errors," *CoRR*, vol. abs/1208.0688, 2012.

[13] X. Zhou, L. Song, and Y. Zhang, *Physical Layer Security in Wireless Communications.* Boca Raton, FL, USA: CRC Press, Inc., 2013.

[14] M. Bloch and J. Barros, *Physical-Layer Security: From Information Theory to Security Engineering.* Cambridge University Press, 2011.

[15] T. Mazloum and A. Sibille, "Analysis of Secret Key Randomness Exploiting the Radio Channel Variability," *International Journal of Antennas and Propagation*, 2015.

[16] C. Shannon, "Communication Theory of Secrecy Systems," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, 1949.

[17] D. Stinson, "Cryptography: Theory and Practice," 2002.

[18] K. Liu, “On Enhancements of Physical Layer Secret Key Generation and Its Application in Wireless Communication Systems,” Ph.D. dissertation, The University of Western Ontario, 2015.

[19] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* New York, NY, USA: Wiley-Interscience, 1991.

[20] A. D. Wyner, “The Wire-Tap Channel,” *Bell System Technical Journal*, vol. 54, no. 8, pp. 1355–1387, 1975.

[21] L. H. Ozarow and A. D. Wyner, “Wire-Tap Channel II,” *Bell System Technical Journal*, vol. 63, no. 10, pp. 2135–2157, 1984.

[22] I. Csiszar and J. Korner, “Broadcast Channels with Confidential Messages,” *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 339–348, 1978.

[23] Z. Li, R. Yates, and W. Trappe, “Secret communication with a fading eavesdropper channel,” in *2007 IEEE International Symposium on Information Theory*, June 2007, pp. 1296–1300.

[24] M. Bloch, J. Barros, M. R. D. Rodrigues, and S. W. McLaughlin, “Wireless information-theoretic security,” *IEEE Transactions on Information Theory*, vol. 54, no. 6, pp. 2515–2534, June 2008.

[25] E. Ekrem and S. Ulukus, “The Secrecy Capacity Region of the Gaussian MIMO Multi-receiver Wiretap Channel,” *IEEE Transactions on Information Theory*, vol. 57, no. 4, pp. 2083–2114, 2011.

[26] G. Foschini, “Layered Space-Time Architecture for Wireless Communication in a Fading Environment when Using Multi-Element Antennas,” *Bell Labs Technical Journal*, vol. 1, no. 2, pp. 41–59, 1996.

[27] U. Maurer and S. Wolf, “Secret-Key Agreement over Unauthenticated Public Channels - Part II: The Simulatability Condition,” *IEEE Transactions on Information Theory*, vol. 49, no. 4, pp. 832–838, 2003.

[28] S. Mathur, W. Trappe, N. Mandayam, C. Ye, and A. Reznik, “Radio-Telepathy: Extracting a Secret Key from an Unauthenticated Wireless Channel,” in *14th ACM international Conference on Mobile Computing and Networking (MobiCom ’08)*, 2008, p. 128.

[29] S. Jana, S. N. Premnath, M. Clark, S. Kasera, N. Patwari, and S. Krishnamurthy, “On the Effectiveness of Secret Key Extraction from Wireless Signal Strength in Real Environments,” in *Mobile Communications (MobiCom09)*, Beijing, 2009, pp. 1–2.

[30] X. He, H. Dai, W. Shen, P. Ning, and R. Dutta, “Toward proper guard zones for link signature,” *IEEE Transactions on Wireless Communications*, vol. 15, no. 3, pp. 2104–2117, March 2016.

[31] Y. Liu, S. C. Draper, and A. M. Sayeed, “Exploiting Channel Diversity in Secret Key Generation from Multipath Fading Randomness,” *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 5, pp. 1484–1497, 2012.

[32] J. Wang, T. Courtade, H. Shankar, and R. D. Wesel, “Soft information for LDPC decoding in flash: Mutual-information optimized quantization,” *GLOBECOM - IEEE Global Telecommunications Conference*, 2011.

[33] W. Xi, X. Y. Li, C. Qian, J. Han, S. Tang, J. Zhao, and K. Zhao, "KEEP: Fast secret key extraction protocol for D2D communication," *IEEE International Workshop on Quality of Service, IWQoS*, pp. 350–359, 2014.

[34] H. Liu, Y. Wang, J. Yang, and Y. Chen, "Fast and Practical Secret Key Extraction by Exploiting Channel Response," *Proceedings - IEEE INFOCOM*, pp. 3048–3056, 2013.

[35] K. Zeng, D. Wu, A. Chan, and P. Mohapatra, "Exploiting Multiple-Antenna Diversity for Shared Secret Key Generation in Wireless Networks," *Proceedings - IEEE INFOCOM*, 2010.

[36] S. N. Premnath, S. Jana, J. Croft, P. L. Gowda, M. Clark, S. K. Kasera, N. Patwari, and S. V. Krishnamurthy, "Secret Key Extraction from Wireless Signal Strength in Real Environments," *IEEE Transactions on Mobile Computing*, vol. 12, no. 5, pp. 917–930, 2013.

[37] C. H. Bennett, G. Brassard, C. Crepeau, and U. M. Maurer, "Generalized Privacy Amplification," *IEEE Transactions on Information Theory*, vol. 41, no. 6, pp. 1915–1923, 1995.

[38] "IEEE standard for information technology- telecommunications and information exchange between systems- local and metropolitan area networks specific requirements part 11: Wireless lan medium access control MAC and physical layer PHY specifications," *ANSI/IEEE Std 802.11, 1999 Edition (R2003)*, pp. i–513, 2003.

[39] Q. Zeng and D. Agrawal, *Introduction to Wireless and Mobile Systems.* Brooks/Cole, 2015.

[40] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Tool release: Gathering 802.11n traces with channel state information," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, pp. 53–53, Jan. 2011.

[41] "Official IEEE 802.11 working group project timelines," Jan. 2017.

[42] D. Halperin, "Simplifying the configuration of 802.11 wireless networks with effective SNR," *CoRR*, vol. abs/1301.6644, 2013.

[43] https://w1.fi/hostapd/.

[44] https://wireless.wiki.kernel.org/en/users/documentation/iw.