Defence Science Journal, Vol. 63, No. 1, January 2013, pp. 6-14, DOI: 10.14429/dsj.63.3758

Implementation of Physical Layer Key Distribution using Software Defined Radios

S. Kambala, R. Vaidyanathaswami, and A. Thangaraj*

Indian Institute of Technology Madras, Chennai - 600 036, India
**E-mail: andrew@ee.iitm.ac.in*

ABSTRACT

It was well known from Shannon's days that characteristics of the physical channel like attenuation, fading and noise can impair reliable communication. But it was more recently that the beneficial side effects of channel characteristics in ensuring secret communication started getting attention. Studies have been made to quantify the amount of secrecy that can be reaped by combining channel coding with security protocols. The Wiretap channel proposed by Wyner is arguably one of the oldest models of physical layer security protocols. In this paper, we present a brief tutorial introduction to the Wiretap channel, followed by an application of the physical layer model to a class of Key Distribution protocols. We present results from an implementation of key distribution protocols using Software Defined Radio tools along with physical RF hardware peripherals. We believe this approach is much more tangible and informative than computer based simulation studies.

Keywords: Physical layer security, wiretap channel, key exchange protocols, software defined radio

1. INTRODUCTION

The amount of sensitive financial, military and government data traffic over the Internet has increased by orders of magnitude over the past two decades. Another recent trend is the use of mobile channels for such communication. With this explosive growth, data security has never been in need of more urgent attention from the academic and research community.

Encryption standards like RSA, AES, and authentication schemes like MD5 have been the work horses since the beginning of this era. Grouped under 'Computational Security', these time-tested protocols rely on the computational intractability of certain operations like prime factoring. But new breakthroughs in technology are constantly pushing the envelope and the pressure on these protocols is increasing.

An interesting paradigm is physical layer security, which proposes the use of physical properties of communication channels to guarantee data security. Physical layer security has attracted intense recent research attention with several journals dedicating special issues[2-4] and even a recent book summarizing the developments in the area[5].

This tutorial explains a particular security model based on wiretap channels. We show how the physical characteristics of a practical channel can be exploited to provide an additional level of security. Differences in the channel quality between the main channel and the wiretapper's channel can directly lead to measurable improvements to the security model. The exact amount of data security provided by this model can be measured in terms of the equivocation seen by an eavesdropper.

2. THE WIRETAP CHANNEL

Wyner[1] proposed a basic framework for secure communication in 1975 called the Wiretap channel. The model is simple: Alice and Bob are two legitimate users, and Alice sends a block of coded data to Bob over a discrete memoryless channel (DMC) C_1 as shown in Fig. 1. Eve is an Eavesdropper illegally tapping into the conversation through another channel that will be named C_2. An important and vital assumption is that C_1 is in some sense stronger than C_2, and this can be guaranteed by the physical nature of the channel and signals. The use of a weaker channel for the eavesdropper is the most significant difference between conventional cryptography and physical layer security.

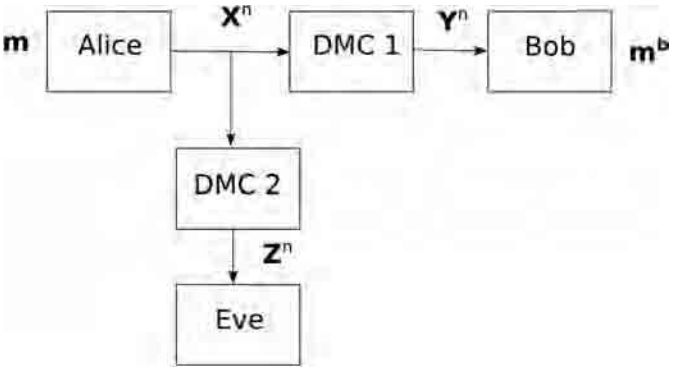

Figure 1. The wiretap channel.

In the wiretap channel, we have twin objectives:

- Reliability: Bob should be able to decode the information

Received 25 October 2012, revised 20 November 2012, online published 24 January 2013

across the channel C_1 with error rate approaching zero as the block length increases.

- Security: Eve's information about the message ***m*** must tend to zero as block length increases.
  In precise information-theoretic terms, Eve's uncertainty about message, namely the equivocation, should be close to the number of message bits. The ideal condition is, of course, $H(\mathbf{m}|Z^n) = H(\mathbf{m})$, where $H(.)$ represents information-theoretic entropy i.e. the uncertainty or entropy of the message ***m*** remains the same in spite of Eve's observations Z^n.

In order to ensure the above two objectives, Alice and Bob have to make some compromise on their data rate. The maximum rate at which the above two objectives can be ensured is called the secrecy capacity of the channel pair C_1 and C_2. If we suppose that C_2 is 'degraded' compared to C_1, the secrecy capacity ends up being the difference of the capacities of these channels i.e. capacity(C_1) - capacity(C_2). For the specific case of C_1 being an ideal error-free channel and C_2 being a binary erasure channel with erasure probability ε, the secrecy capacity (as calculated in Wyner's paper) is ε, which is the fraction of bits that are erased in Eve's observation.

Wyner talks about two wiretap channel models, named wiretap-I and wiretap-II. Under the former model, Eve gets a random subset of μ bits from the total of *n* bits transmitted. In wiretap-II, the adversary is given the freedom to choose the exact subset of bits to be monitored. Obviously, the latter model is significantly more challenging for the system designer.

2.1 Encoding for The Wiretap Channel

It will be informative to compare traditional error correction coding with wiretap encoding. A typical error correcting code takes a *k* bit message and encodes it into *n* bits. The code *C* can be viewed as a subspace of the *n*-dimensional binary space $\{0,1\}^n$. There is a one-to-one correspondence between message words and code words.

This coding scheme is sufficient to ensure reliability. But to address the Security objective, an unusual one-to-many coding scheme is required. The wiretap coding model takes the *k* bit message and maps it to the entire space of *n* bits. So there are $2^{(n-k)}$ code words corresponding to each possible message. One of these code words is chosen at random and transmitted. It is this randomness that is the backbone of the wiretap channel's security.

The most attractive part of these schemes is that there are few secrets to be kept; Eve can have full knowledge of the encoding scheme and its parameters. Also, there are no private keys.

2.2 Low Density Parity Check Codes

Our first task is to choose a suitable code *C*. An error correcting code can be defined in terms of its parity check matrix *H* whose rows are orthogonal to every code word. In symbols,

$$Hx^T = 0, x \in c \tag{1}$$

It has been shown that a particular class of error correcting codes known as low density parity check (LDPC) codes works best for Wiretap encoding. LDPC codes were proposed by Gallager in his path breaking 1960 thesis[7]. These are codes that have a sparse parity check matrix.

Low density parity check codes have several interesting properties. First, they can operate very close to Shannon's channel capacity. Carefully designed LDPC codes are known to reach within 0.0048 dB of the limit in binary Gaussian channels. Secondly, every LDPC code family exhibits a threshold property over most practical channels. In Gaussian channels, if SNR* is the signal to noise ratio (SNR) threshold of an LDPC code, then the code is guaranteed to work with high probability at all SNRs greater than SNR*.

2.3 Coset Encoding Scheme

Let *C* be an LDPC code with generator matrix *G*. Like all linear codes, *C* is a subspace of $\{0, 1\}^n$ with 2^k elements. A set of the form $C+\mathbf{e}$ where ***e*** is any *n*-bit word (that may or may not be in *C*), is called a coset of *C*. By the theory of error-correcting codes, every coset is of size 2^k and there are a total of $2^{(n-k)}$ distinct cosets. The cosets partition the *n*-dimensional space into disjoint subsets as shown in Fig. 2. With this backdrop, we proceed to describe the Wiretap encoding scheme:

Alice starts with an (*n- k)*-bit message ***m***. This is the secret message to be communicated to Bob. She first finds a basis (let us call it *G**) for the complementary subspace of *C* in $\{0, 1\}^n$. *C* and its complement *C'* together form the full *n*-dimensional space of binary numbers: $C \cup C' = \{0,1\}^n$. The next step is crucial: Alice generates a *k*-bit random number ***u***. This is a uniform random number generated at run time, and neither Bob, nor Eve is aware of its value. Alice performs the following calculation with it:

$$x = \begin{bmatrix} m & u \end{bmatrix} \begin{bmatrix} G* \\ G \end{bmatrix} \tag{2}$$

The vector ***x*** above is the code word that is ultimately transmitted over the Wiretap channel. Equation (2) can be expanded as $\mathbf{x} = \mathbf{m}G^* + \mathbf{u}G$. By definition, $\mathbf{u}G$ is a valid code word in *C*. The actual message ***m*** determines a coset of *C* whose size is 2^k. Alice selects an element from the coset at random (recalling that ***u*** is random) and transmits it.

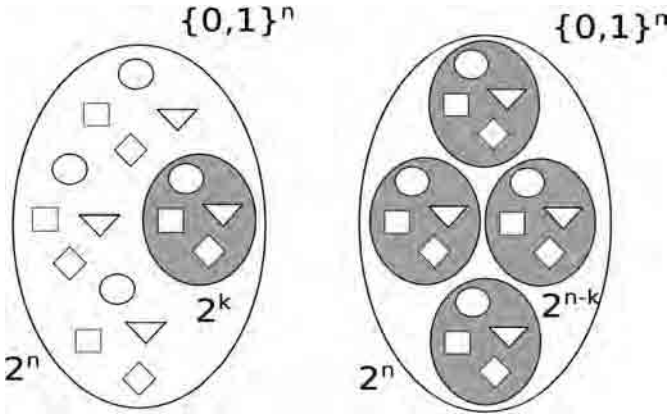

Figure 2. Coset encoding.

2.4 Decoding the Wiretap Channel

As a first approximation, we assume that the main channel is noise-free. So Bob receives the code word x without errors. His goal is to extract the message m, so he performs the computation

$$H\boldsymbol{x}^T = (\boldsymbol{m}G^* + \boldsymbol{u}G)^T = H(\boldsymbol{m}G^*)^T \quad (3)$$

This situation is identical to a code word getting corrupted in transmission. The vector $\boldsymbol{u}G$ is the code word, and $\boldsymbol{m}G^*$ is akin to noise. The product $\boldsymbol{s} = H(\boldsymbol{m}G^*)^T$ is called the syndrome. The syndrome $\boldsymbol{s}$ uniquely determines the coset corresponding to $\boldsymbol{m}G^*$ and thus Bob has recovered the original message.

It is to be noted that Bob never used the actual value of the random vector $\boldsymbol{u}$. This is because the coset is the message; any of the 2^k words in the coset would serve the purpose equally well.

The eavesdropper's aim is also to decode the secret message $\boldsymbol{m}$. But as noted above, Eves' channel is inferior to Bob's and she receives a corrupted version of the codeword $\boldsymbol{x}$. The random nature of the encoding along with a careful design of the code is used to ensure that the corrupted version reveals very little or no information about the message to Eve.

2.5 Security of the Wiretap-I Channel and Column Rank Property

Suppose that Eve's channel is a binary erasure channel, and she receives only r bits of the transmitted codeword $\boldsymbol{x}$. The syndrome $H\boldsymbol{x}^T$ is a linear combination of some columns of H. The columns corresponding to the '1' bits in $\boldsymbol{x}$ are selected for this summation, while the columns multiplied by the '0' bits get nullified. But Eve's copy of $\boldsymbol{x}$ is not complete; her syndrome has several ambiguous positions. These are precisely the places corresponding to the $(n- r)$ erasures.

It is to be noted that the whole idea of Wiretap encoding is based on the assumption that we have a source of perfect random numbers that is cryptographically secure. If the random vector $\boldsymbol{u}$ in Eqn. (2) is revealed to the eavesdropper, then security of the system can be compromised.

We will employ the following useful property of LDPC codes: If an LDPC code C has length n and threshold η over binary erasure channels, then any subset of columns of its parity matrix H, chosen independently with probability η, will have full column rank with high probability. The result is applicable for large values of n, but LDPC codes are usually used with block length running into thousands. As a result, linear combinations of at least $n\eta$ columns of H can produce $2^{n\eta}$ distinct words.

We will use the following theorem due to Ozarow and Wyner:

Theorem 1: C is an $(n, n\text{-}k)$ linear code with generator matrix G. Let the columns of G be denoted by $\{g_1, g_2, \ldots g_n\}$. The bit positions revealed to the eavesdropper are $\{b_1, b_2, \ldots b_r\}$. Then the code will be secure, if the corresponding columns $\{g_{b1}, g_{b2}, \ldots g_{br}\}$ have full rank.

As an example, let $\boldsymbol{x}$ be a 7-bit word, and let us suppose that Eve received the first 3 bits correctly. The next 4 bits have been erased in the channel. We denote Eve's received word as

$\boldsymbol{x}$ = { 1 1 0 * * * * }

From this partial information Eve has to guess which coset the message belongs to. If every coset has at least one word starting with 1 1 0..., then Eve's uncertainty is total. As there are $2^{(n-k)}$ cosets, each of them has probability of $1/2^{(n-k)}$ of being right and the equivocation seen by the attacker is $(n\text{-}k)$ bits.

This can be practically verified through simulation. Figure 3 shows an LDPC code belonging to the EGLDPC code family. It is a (4095, 3367) code. A 728 bit secret message is encoded into 4095 bits and transmitted. The graph plots the equivocation of Eve against the number of bits revealed to her. As long as less than 3367 bits are revealed, the equivocation remains at its maximum value of 728 bits (complete security). After that point, the equivocation drops linearly.

Though we have described secrecy in wiretap channels using the example of binary erasure channels, these ideas can be extended to other channels including the most practical case of Gaussian channels.

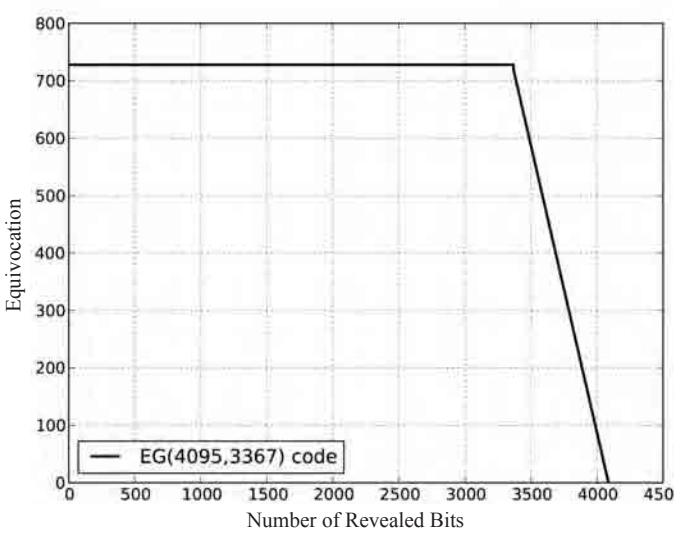

Figure 3. Security of EG (4095, 3367) code.

3. KEY EXCHANGE PROTOCOLS

Shannon[6], showed that one time padding (OTP) is an effective way to encrypt a message for unconditional security. But OTP schemes suffer from the equally intractable problem of key distribution: if an n-bit message needs an n-bit key to be securely distributed, then encryption offers no additional advantage. So all practical schemes use a much smaller key of k bits, $k << n$, and use some form of pseudo random number generator (PRNG) to expand the key to the necessary block length.

For example, a block cipher like 3DES or AES can be used in chained mode with a counting nonce. This can generate a sequence of random number blocks using a small seed as the initial key. Protocols like Fortuna accumulate entropy from various sources for the seed, and keep updating the seed periodically. Another approach is to use linear feedback shift registers (LFSR) with a true random seed as the initial state of the register.

Many of these algorithms are vulnerable to timing attacks and correlation attacks, and there is ample literature on countermeasures. This paper will focus mainly on the generation and sharing of the initial key between two legitimate

parties Alice and Bob, while keeping an eavesdropper Eve completely ignorant of it.

3.1 Key Generation and Reconciliation

Diffie and Hellman showed in 1976 that it is possible for the legitimate parties to exchange a pair of keys through an insecure channel and use it for public key cryptography. But we look at a variant of this technique, where no agreed-upon public or private keys are available to begin with. Instead, the secret key is distilled from a larger string of binary digits exchanged over a noisy channel. Alice and Bob also have access to an error-free and authenticated public channel over which they exchange a series of messages about the key bits. This process, called reconciliation, results in distilling mutually agreed upon bits that have sufficient entropy to provide an advantage over the eavesdropper. We now describe the above steps in greater detail for the Gaussian Wiretap channel.

3.2 Key Exchange for a Gaussian Wiretap Channel

Figure 4 shows a Gaussian wiretap channel. Alice and Bob are separated by a Gaussian channel with noise variance Σ^2, while Eve's channel is also Gaussian with noise variance σ^2.

We will suppose that $\Sigma < \sigma$, which makes Eve's channel degraded in comparison with Bob's channel. This advantage is exploited in the key exchange protocol. Further, we suppose that Alice has a transmit power limitation of P. For the purposes of key exchange, Alice generates *n i.i.d* realisations of a Gaussian random variable $X_i \sim N(0, P)$, $i = 1, 2,\ldots n$ and transmits them to Bob through the main channel. The channel adds noise Z_i that is again *i.i.d* Gaussian: $Z_i \sim N(0, \Sigma^2)$.

Alice also quantises X_i to *r* bits of precision and gets an *r*-bit vector $[X_{i1}\, X_{i2} \ldots X_{ir}]$. Each of these *r* bits are said to form a 'level'. All bits in a particular level are grouped to form *n*-bit vectors $Q_j = [X_{1j}\, X_{2j} \ldots X_{nj}]$, $j=1,2,\ldots r$. Using these *r* level vectors, Alice computes *r* syndromes S_j using a series of parity check matrices H_j, $j = 1, 2, \ldots r$:

$$S_j = H_j Q_j^T$$

The rates of the codes represented by H_j and the quantisation intervals used are carefully chosen in order to optimise the mutual information between the received soft values and the quantised bits, as explained later.

Bob receives $Y_i = X_i + Z_i$, $i = 1,2,\ldots n$ through the Wiretap channel and the syndromes S_j, $j = 1, 2, \ldots r$, without any errors, through an authenticated public channel. Note that Eve receives $Y_i' = X_i + Z_i'$ where $Z_i' \sim N(0, \sigma^2)$ are *i.i.d* Gaussian with $\sigma > \Sigma$. Since the syndromes are sent on a public channel, Eve also obtains S_j, $j=1,2,\ldots r$ without any errors.

The goal of the communication process is to ensure that Bob can recover the vectors Q_j, which are *r*-bit quantised versions of X_i, without any error. The syndromes are sent to enable Bob to perform error correction. Since there are *r* different levels that are intertwined in each received value, we will employ a method known as multi-level decoding, which is described next.

3.3 Multi Level Decoding with LDPC Codes

The decoder for two levels ($r = 2$) is shown in Fig. 5. The received values Y_i, $i = 1,2,\ldots n$ and the syndromes $S_j = HQ_j^T$, $j=1,2,\ldots r$, are processed at the decoder by levels. As shown in Fig. 5, Level 1 is decoded first using the received values and S_1. The result of Decoder 1 and S_2 are used in Decoder 2 along with the received values. If there are more levels, we continue in a similar fashion. For the first stage we assume that *a priori* 1 and 0 are equally probable for X_{i1}. For the subsequent stages, the extrinsic information from stages 1 through *(j-1)* form the *a priori* input LLR for stage *j* and helps the decoder converge with better accuracy.

In general, the vector $Q_j = [X_{1j}\, X_{2j} \ldots X_{nj}]$ is decoded at Decoder j using the syndrome $S_j = H_j Q_j^T$ (sent on the public channel), the received values $Y = [Y_1\, Y_2 \ldots Y_n]$ and the previously decoded vectors $Q_1, Q_2, \ldots Q_{j-1}$.

To describe the decoders in more detail, we will suppose that LDPC codes are being used (i.e. H_j are LDPC matrices) and the decoders are soft-in, soft-out (SISO). Therefore, Decoder *j* will output log likelihood ratios (LLRs) L_{ij} for the bits X_{ij} given the entire received vector $Y = [Y_1\, Y_2 \ldots Y_n]$. That is, we have

$$L_{ij} = \frac{\Pr(X_{ij} = 0 \mid Y)}{\Pr(X_{ij} = 1 \mid Y)} \tag{4}$$

So, if $L_{ij} > 0$, we decide that $\hat{X}_{ij} = 0$, and, if $L_{ij} < 0$, we decide $\hat{X}_{ij} = 1$. The input for Decoder *j* is the LLR l_{ij} for each bit X_{ij} given only *Y* and the outputs of previous decoders L_{1j}, L_{2j},

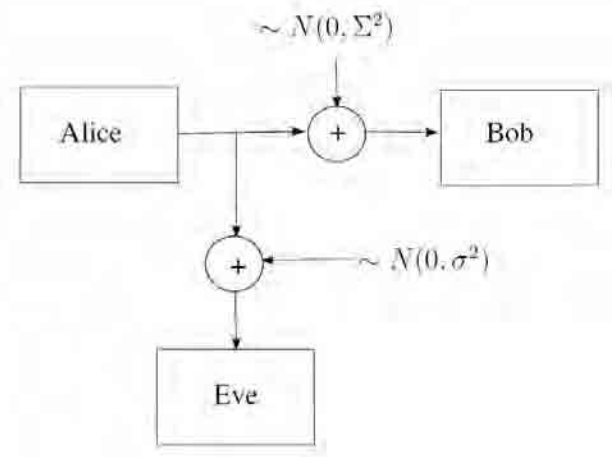

Figure 4. Gaussian wiretap channel.

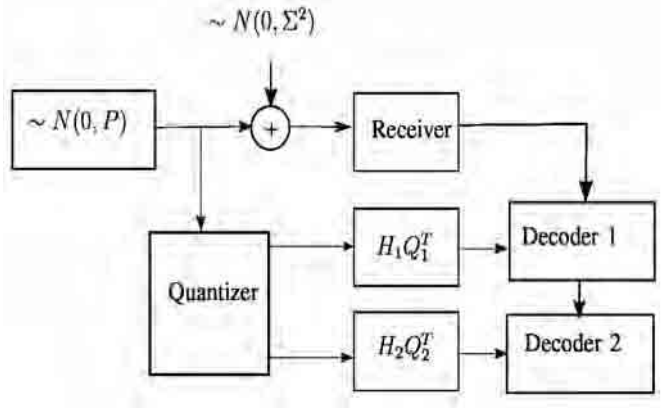

Figure 5. Two level decoder.

… $L_{i,j-1}$. The input LLR l_{ij} in the j-th stage is computed as follows:

$$l_{ij} = \log \frac{\sum_{q0,q1,..qj-1} f(y \mid q_j = 0, q_0, q_1, ...q_{j-1}) \Pr(q_0, q_1, ...q_{j-1})}{\sum_{q0,q1,..qj-1} f(y \mid q_j = 1, q_0, q_1, ...q_{j-1}) \Pr(q_0, q_1, ...q_{j-1})} \quad (5)$$

where $f(.)$ represents a suitable probability density function (PDF) and $\Pr(q_0, q_1 \ldots q_{j-1})$ is computed using the LLRs L_{i1}, L_{i2}, …, $L_{i,j-1}$.

3.4 Quantisation and Code Rates

The quantisation levels used in quantising X_i are chosen to maximise the mutual information

$I(X_{i1}, X_{i2}, \ldots X_{ir}; Y_i)$.

For $r = 2$, the mutual informations $I(X_{i1}; Y_i)$ and $I(X_{i2}; Y_{ij}|X_{i1})$ are plotted against normalised SNR (dB) in Fig 6 for a two level decoder. As expected, the total mutual information approaches 2 bits for sufficiently high values of channel SNR.

Each parity-check matrix H_j, $j = 1, 2, \ldots r$ is chosen such that its rate R_j falls within the mutual information between level X_{ij} and Y_i given $X_{i1}, X_{i2}, \ldots, X_{i,j-1}$ at the chosen SNR of operation. Given such rates R_j, a suitable parity-check matrix H_j of rate R_j is formed using a standard LDPC code generator.

We discuss one specific design that is used in our experiments. For a two-level decoder, with P = 1 and $\Sigma^2 = 1/10$ resulting in an SNR of 10 dB, we see that the mutual information of level 1 and level 2 are 0.67 and 0.51 using Fig 6. So, we choose $R_1 = 0.625$ and $R_2 = 0.4$ to be within the capacity bounds. Then, we choose a regular LDPC degree distribution of (3, 8) for H_1 and (3, 5) for H_2. These LDPC matrices were constructed for a block length of n =1000 and the entire decoder was simulated. The resulting frame error rate (FER) performance of the multi-stage decoder is plotted in Fig. 7.

We see that, as per our design, the FER falls after the target SNR of 10 dB. This plot also confirms the threshold property of LDPC codes. If the block length is increased further, the drop in FER after the threshold will be more steep.

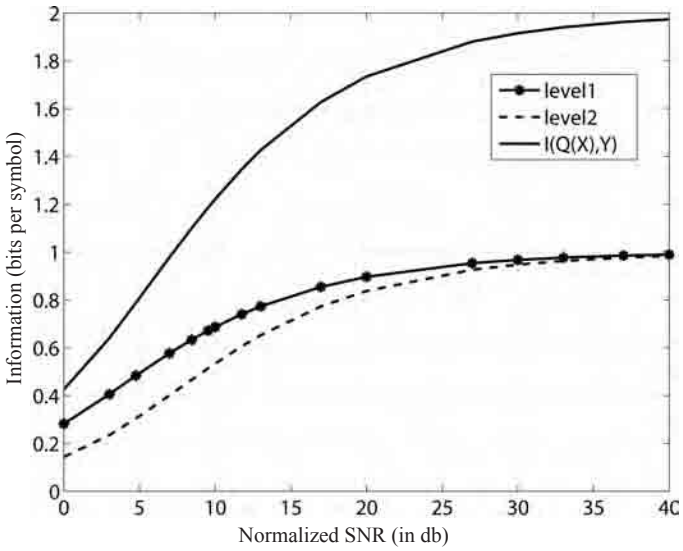

Figure 6. Mutual information of two level decoder.

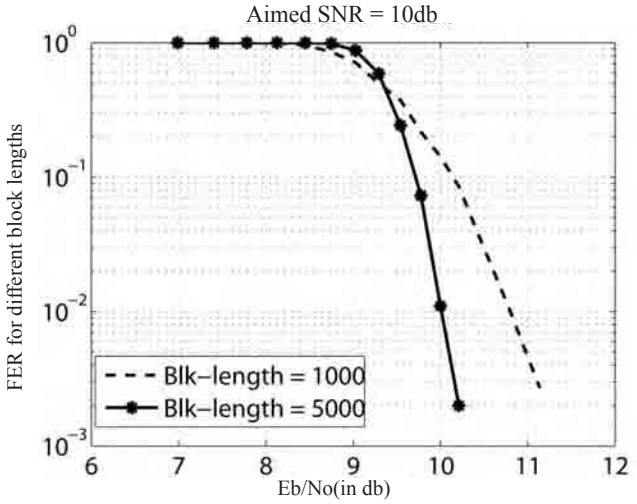

Figure 7. Frame error rate of two level decoder.

After multilevel decoding, assuming the decoder converged, the receiver Bob will have correctly decoded the r quantised vectors $Q(X) = \{Q_1, Q_2, \ldots Q_r\}$. Notice that now both Alice and Bob have access to this common nr-bit string $Q(X)$.

3.5 Privacy Amplification

The common string $Q(X)$ shared by Alice and Bob is not entirely secure. This is because Alice had sent the syndromes $S_j = H_j Q_j^T$ over the public channel and this was received by Eve as well.

In the final stage of key distribution known as privacy amplification, Alice and Bob agree upon a common substring from the larger binary string $Q(X)$. This forms the ultimate secret key that they use for subsequent PRNG seed. Universal Hash functions can be used to arrive at the final key in the privacy amplification phase.

The design of key distribution over a Gaussian wiretap channel, described in this section, was largely based on[10,11]. Next, we present the details and results of an experimental implementation of this key distribution protocol.

4. SOFTWARE DEFINED RADIO

Software defined radio is an approach to implement radio communication modules like mixers, filters, modulator/demodulators, etc on general purpose or embedded computers through DSP software. They usually employ a minimalistic hardware front end with the RF stages and ADC/DAC modules and an optional ASIC for high speed computations. The IF and baseband processing is delegated to a general purpose computer for maximum flexibility and control. The spectrum band, transmit power, antenna directionality, etc. are adaptively controlled by software to suit dynamically changing channel conditions. This is also the corner stone of cognitive radio that enables agile use of the spectrum through environment sensing and opportunistic transmission.

Universal software radio peripheral (USRP) denotes a family of flexible RF hardware devices that are meant for software radio applications. It is widely used with the open source GNU Radio project[13]. Most popular versions of USRP are manufactured by Ettus Research and its principal National

Instruments[14]. The USRP has since been deployed in diverse fields like telecom, radio astronomy and medical imaging.

USRP1 type devices along with RFX 2400 Transceiver boards were used in a series of experiments with key exchange protocols. The rest of the paper describes some of the results.

4.1 USRP Fundamentals

A basic block diagram of a USRP device is shown in Fig. 8. The main board consists of an FPGA that handles most of the operations, along with analog-to-digital and digital-to-analog converters that operate at RF frequencies. The RF stages are on a set of 4 daughter boards mounted on the motherboard. The USRP1 has slots for 2 TX boards and 2 RX boards.

Advanced models like N210 have a Gigabit Ethernet interface for high bandwidth applications with high speed ADCs and DACs. The current study used the original USRP1 device. It has four 12 bit ADCs operating at 64 MS/s and four 14 bit DACs at 128 MS/s. The FPGA is an Altera EP1C12 device. Its has a highly pipelined architecture with a clock frequency of 64 MHz. The software is implemented in Verilog using QuartusII. Communication with the host computer is through a USB2 port. The power supply is 6V/2A.

The RFX2400 is an RF Transceiver board operating in the 2.3-2.9 GHz range capable of sourcing 100 mW output. TriBand rubber duck antennas that have a range of 2.4 GHz to 5.8 GHz were used. Though the USRP can operate in full duplex mode, the experiment used two USRPs in simplex mode. The FPGA has 4 down converters that can handle any of the four ADC inputs through a flexible routing mechanism. A decimating low pass filter provides I and Q outputs. The channels are interleaved and sent to the host through the USB bus. On the transmit path, up conversion is handled by four AD9862 codec chips, with the FPGA providing only interpolation support.

The ADCs have an analog input bandwidth of 200 MHz, which is the nominal upper limit for the IF frequency. However, upto 500 MHz IF can be handled if the loss can be tolerated. To give this flexibility to the designer, the USRP does not have an anti-aliasing filter.

4.2 Software Interface

The natural platform for the GNU Radio tool chain is Linux, though it has been ported to other environments like Mac OS and MS Windows. The open source graphical user interface GNU radio companion was extensively used for this study. The preferred native language of the open source community is Python, but it is easy to write tight subroutines in C/C++ and plug in using the *SWIG* interface. The current study implemented all the multilevel decoding routines in C and the decoding was performed offline.

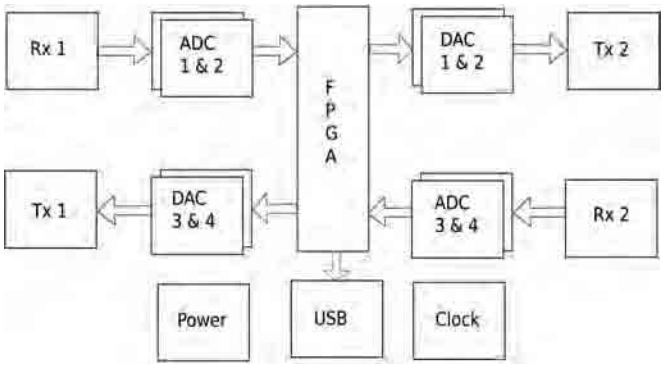

Figure 8. USRP block diagram.

As noted, most of the operations of the USRP are handled in software. This includes programmable gain amplifiers for the mother board input (upto 20dB gain) as well as the DSP routines for modulation/ demodulation. Many software projects including GSM, CDMA 2000, IEEE 802.11 (Wi-Fi), IEEE 802.16 (Wi-Max) and LTE have been implemented in software Defined radio. FM and DVB receivers, Software GPS and RFID detectors can be easily assembled around the USRP. Matlab and Simulink also support the USRP.

5. KEY DISTRIBUTION WITH THE USRP

In this section we describe the experiments performed in the IIT Madras lab on key distribution protocols using software radio peripherals and present the main results. The layout of the experimental setup was optimised for easy measurements. The communication lab was rectangular with size 11 m x 5.75 m and a brick wall partition in the middle (Fig. 9). Standard wooden partitions, office furniture and PCs were present. The room had 4 large grilled windows facing a corridor outside.

The transmitter was centrally kept at a fixed place and measurements were made at multiple points in the rooms and the corridor.

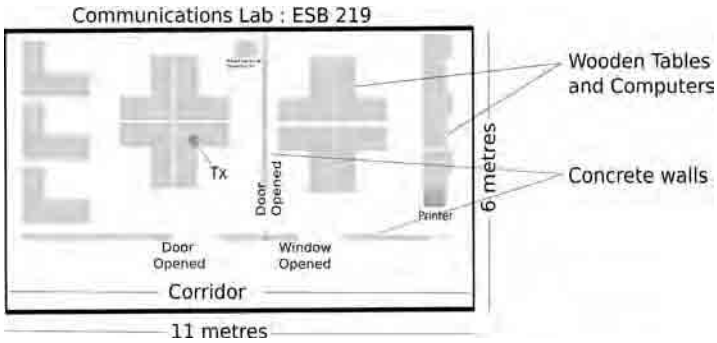

Figure 9. Experimental setup at IIT-Madras lab.

5.1 Experiment Design

The base station was fixed at one location and the receiver was moved along fixed distance contours. The sender transmitted a series of Gaussian distributed random data blocks, which were generated in MATLAB offline and transferred to the USRP board for transmission. The symbol rate used was 12,500 symbols per second. The receive USRP board collected samples of the received baseband waveform with an oversampling of 20. These samples were transferred to a desktop computer, on which the entire receiver processing was done offline. Measurements were repeated at different locations, which corresponded to different values of estimated channel SNR. The Transmitter and Receiver have a programmable gain of -20 to 0 dB and 0 to 70 dB respectively, which gives the total dynamic range 90 dB. The receiver used a training sequence to perform rudimentary channel estimation and to calibrate itself.

5.2 Frame Configuration

Every transmitted frame starts with a preamble to assist the receiver to lock on to the symbols. As shown in Fig 10. This consists of a block of 100 all-1 symbols followed by a block of 400 BPSK modulated known symbols. The preamble serves four purposes:

- As a training sequence to help the receiver synchronise with the transmitter
- To estimate the channel SNR
- To estimate the attenuation, and
- Derive a scaling factor for the symbols.

The payload is a block of 1000 symbols from a Gaussian distributed random real number sequence.

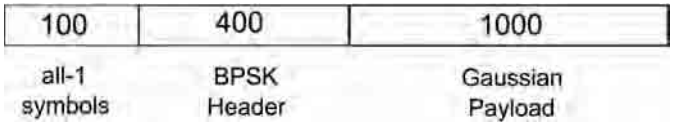

Figure 10. Transmission frame configuration.

5.3 Transmitter

The experimental setup consisted of a pair of USRP1 software radio peripherals from Ettus research. Each consisted of an RFX 2400 model daughter board operating in the band 2.3 GHz - 2.9 GHz. The original symbol rate was 1/T = 12.5 K samples/sec implying a Nyquist band width of 1/2T as the minimum cut off. But since this places impractical demands on the receiver filter design, we over-sample the symbols by a factor of 20 and then filter it at 35% extra bandwidth. Our implementation had 401 filter taps spread over the duration of 20 samples (±10T). Since the symbol rate R_s = 12500 and the filter roll off factor $\beta = 0.35$, we see that the bandwidth required is 0.5 $R_s(1+\beta)$ = 8.438 Hz.

Pulse shaping is done by a matched pair of root raised cosine (RRC) filters at the transmitting and receiving ends. The RRC filter was chosen for its simplicity and the protection it offers against inter symbol interference (ISI). The final output rate was 12.5 K symbols x 20 samples = 250 K samples/s, which is safely within the operating range of the digital to analog converter (DAC) in the USRP.

In our experiment we kept our transmitter gain at its maximum 0 dB and transmitter power is varied by changing a multiplying constant (Fig. 11) for convenience. The RF carrier frequency was set to be 2.4752 GHz.

5.4 Receiver

A schematic of the receiver is shown in Fig 12. Input data to the USRP in the receiver in Fig. 12 arrives at a sampling rate of 64 M samples/s. Even though the transmitted symbols were purely real, carrier phase offset introduces a spurious $e^{j\theta}$ factor, resulting in a phase rotation. For this reason the receiver operates in the complex domain with a pair of analog to digital converters (ADCs) within the USRP operating at 128 M samples/s. After RF processing, down conversion and decimation in the FPGA in the USRP, signal samples of the two I and Q streams are interleaved and passed to the PC through a high speed USB 2.0 cable at the original 250 K samples/s. This data is transferred to a general purpose computer for further signal processing. The PC used for the software radio was an off-the-shelf Dell Vostro laptop running Ubuntu Linux 10.04. The PC hardware consisted of an Intel Core 2 Duo processor at 2.1 GHz and 2 GB DDR2 RAM.

In the PC, the following processing is performed. The throttle is a buffer used for congestion control, ensuring average rate does not exceed sampling rate. This is followed by a matched filter (MF) module. The MF filter correlates the received symbols with the transmitted pulse shape. The output of the MF continues to be at 250 K samples/s, which is 20 times oversampled. This is to facilitate timing and carrier phase recovery, as described later.

After recovering the clock, the in-phase component is sent to a file sink for storage and mutli-stage decoding to recover the quantised values. The multi-stage decoder is run on a general purpose computer, along with the syndrome information, and error correction is carried out as described earlier.

5.5 Timing Recovery

The most challenging part of the implementation was symbol synchronization. The ADC in the receiver operates on a free running clock, which is non-coherent with the transmitter clock. The offset between the two clocks needs to be estimated and corrected for, in order to decide the exact sampling instant. This will help us get the best possible SNR and minimise ISI. Timing recovery is handled using the modified Mueller and Muller (MM) method[13]. Our implementation used an interpolation filter at 20 samples/symbol. The step size for updating successive phase estimates was 0.001.

Since M & M algorithm is sensitive to carrier phase offset, we used a Costas loop stage to first recover the carrier phase[13]. It is a pair of phase locked loops (PLLs) operating in inphase and quadrature modes. The Costas loop approach was preferred over feed-forward techniques, because of its inherent track-and-hold ability. This helps in self-correcting the phase and frequency of the received carrier. The BPSK preamble in

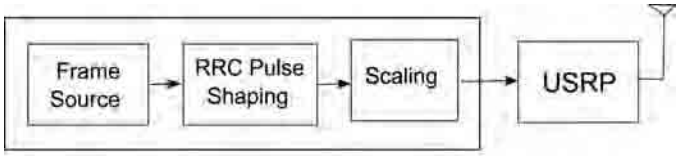

Figure 11. Transmitter schematic.

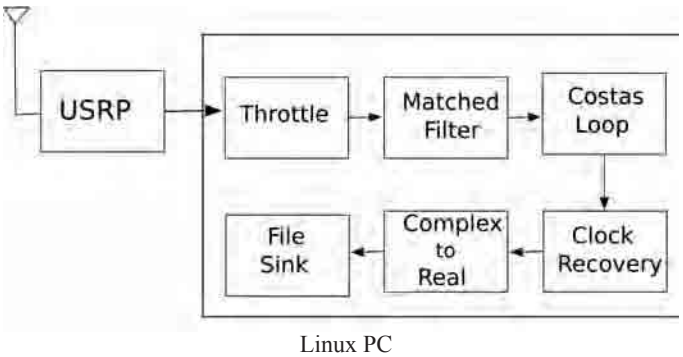

Figure 12. Receiver schematic.

the frame was used as the training signal to lock the loop. The Costas loop bandwidth was adjustable from $\pi/100$ to $2\pi/100$. In practice the built-in Oscillator in USRP1 has frequency accuracy upto 3 ppm. So when we operate in 2.4 GHz range we get a frequency offset of around 10 kHz.

The receiver gain was fixed, and the transmitter gain was kept at its maximum, but transmitted power is varied by varying the multiplying constant and the measurements were repeated at different locations. The receiver power level was calibrated by sending a known symbol sequence at constant Tx power and bench marked. A pilot signal consisting of all 1's was used to sync the start of transmission. The sample rate was 20 samples/symbol. After calibrating the SNR reference point, we send the Gaussian symbol sequence. The receiver uses the pilot to perform rudimentary channel estimation and uses the calculated SNR value for decoding. The Carrier frequency at which the experiments were conducted is 2.4752 GHz.

Input bit error probabilities were calculated by comparing hard-decoded values at the receiver with the transmitted symbols. The experiment was repeated and the resulting error surface is shown in Figs. 13-16 by varying transmit power. Transmitter position is showed in the plots by 'Tx'. The code rates are 0.625 and 0.4 and the block size of 1000 was used. Each plot shows the probability of successful recovery of the key against the physical placement of the receiver. As

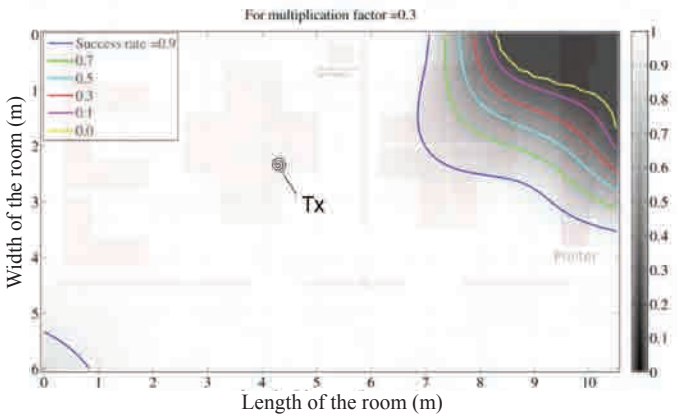

Figure 13. Success rate representation of frames decoded when multiplication factor = 0.3.

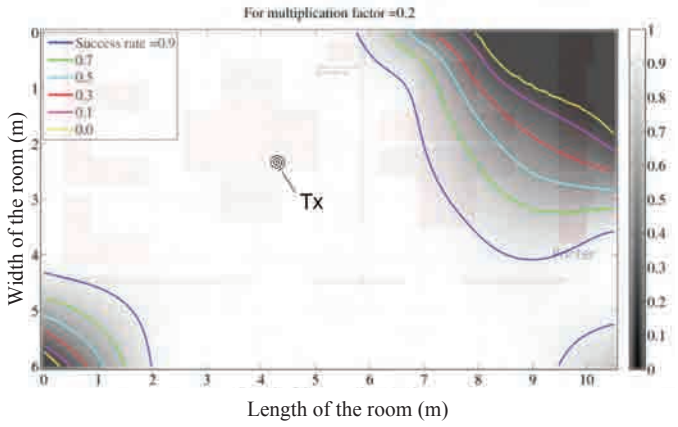

Figure 14. Success rate representation of frames decoded when multiplication factor = 0.2.

may be expected, there are equiprobable contours around the transmitter with the central region representing high probability of successful key recovery. For instance, in Fig. 16, we see that the success probability for key distribution falls to very low values even at locations inside the same room as the transmitter. Such a scheme can be used for key distribution, which is secure from eavesdroppers outside of the room.

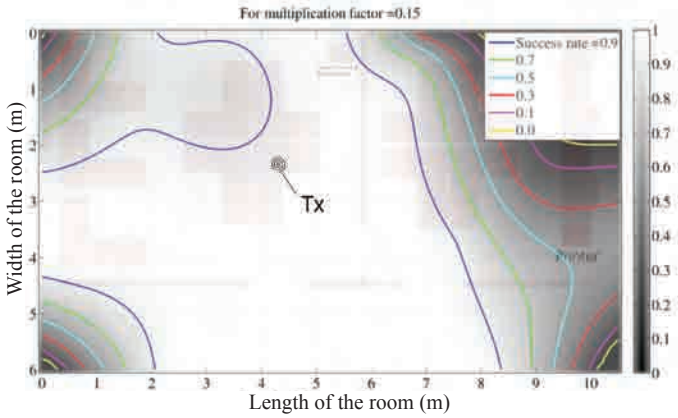

Figure 15. Success rate representation of frames decoded when multiplication factor = 0.15.

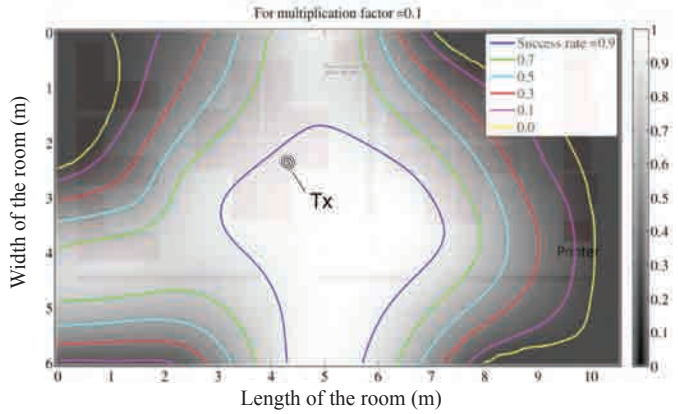

Figure 16. Success rate representation of frames decoded when multiplication factor = 0.1.

6. CONCLUSION

We looked at the physical layer security model for key exchange protocols. Studies indicate that the difference in channel qualities of the legitimate parties and the eavesdropper can be used to exchange a new random key constructed at run time. Use of a reliable public channel for key negotiation enables us to arrive at a secret key about which the eavesdropper has negligible knowledge. We made detailed measurements using the USRP device at the physical layer and key exchange protocols at the application layer. The results confirm that the effect of the physical channel on the secrecy of communication is indeed very pronounced. This knowledge can be used in designing communication systems with verifiable security specifications. One possible future direction of work would be to use this set up for a study of ad-hoc wireless networks to design energy efficient and secure coding schemes.

REFERENCES

1. Wyner, A.D. The wire-tap Channel. *Bell Syst. Tech. J.*, 1975, **54**(8), 1355-1387.
2. Bloch, M.; Debbah, M.; Liang, Y.; Oohama, Y. & Thangaraj, A. Special issue on physical-layer security. *J. Comm. Networks*, 2012. **14**(4), 349-51.
3. Trappe, W.; Poor, V.; Iwai, H.; Yener, A.; Prucnal, P. & Barros, J. Guest editorial special issue on using the physical layer for securing the next generation of communication systems. *IEEE Trans. Infor. Forensics Security*, 2011, **6**,(3), 521-522.
4. Imai, H.; Hanaoka, G.; Maurer, U. & Zheng, Y. Introduction to the special issue on information theoretic security. *IEEE Trans. Infor. Theory*, 2008, **54**(6), 2405-2407.
5. Bloch, M. & Barros, J. Physical layer security: From information theory to security engineering. Cambridge University Press, Nov. 2011.
6. Shannon, C.E. Communication theory of secrecy systems. *Bell Syst. Tech. J.*, 1949, **2**8, 656-715.
7. Gallager, R.G. Low-density parity-check codes. *IRE Trans. Infor. Theory*, 1962, IT-8, 21-28.
8. Ozarow, L.H. & Wyner, A.D. Wire-tap channel II. B.S.T.J., 1984, **63**(10), 2135-157.
9. Maurer, U.M. Secret key agreement by public discussion from common information. *IEEE Trans. Inform. Theory*, 1993, **39**(3), 733-742.
10. Bloch, M.; Thangaraj, A.; McLaughlin, S.W. & Merolla, J.-M. LDPC-based secret key agreement over the Gaussian wiretap channel. *In* IEEE International Symposium on Information Theory, Seattle, Washington, United States. July 2006. pp. 1179-1183,
11. Bloch, M. Physical layer security. Georgia Institute of Technology, August 2008. PhD Thesis
12. Schneier, B. & Ferguson, N. Practical cryptography. Wiley Eastern, 2003.
13. GNU Radio. http://gnuradio.org/redmine/projects/gnuradio/wiki, [accessed on 5 December 2012].
14. Ettus Research. http://www.ettus.com/, [accessed on 5 December 2012]
15. Winlab Orbit. http://www.winlab.rutgers.edu/docs/focus/ORBIT.html, [accessed on 5 December 2012]
16. GNU Radio Forum. http://www.ruby-forum.com/forum/gnuradio, [accessed on 5 December 2012]

Contributors

Dr Andrew Thangaraj received his BTech (Electrical Engg) from Indian Institute of Technology (IIT), Madras, India in 1998 and a PhD (Electrical Engg) from the Georgia Institute of Technology, Atlanta, USA in 2003. He was a post-doctoral researcher at the GTL-CNRS Telecom lab at Georgia Tech Lorraine, Metz, France during 2003-2004. Currently working as an Associate Professor in the Department of Electrical Engineering, IIT-Madras.

Mr Sreenath Kambala received his BTech (Electronics and Computer Engg) from Srinidhi Instistute of Science and Technology, Hyderabad, Andhra Pradesh in 2009. Currently he is pursuing his MTech (Communication Engineering) from IIT-Madras.

Mr Rajaraman Vaidyanathaswami received his MTech (Communication Engg) from Indian Institute of Science, Bangalore. He is currently working on coding schemes for physical layer security for a Doctoral program at the Indian Institute of Technology, Madras, Chennai.