Analysis of a MAC Layer Covert Channel in 802.11 Networks

Ricardo Gonçalves
Department of Electrical and Computer Engineering, Naval Postgraduate School
Monterey, California
santana.goncalves@marinha.pt

Murali Tummala
Department of Electrical and Computer Engineering, Naval Postgraduate School
Monterey, California
mtummala@nps.edu

John C. McEachen
Department of Electrical and Computer Engineering, Naval Postgraduate School
Monterey, California
mceachen@nps.edu

***Abstract*—In this work, we present a proof of concept on a new covert channel in IEEE 802.11 networks, making use of the Protocol Version field in the MAC header. This is achieved by forging modified CTS and ACK frames. Forward error correction mechanisms and interleaving were implemented to increase the proposed channel's robustness to error. A laboratory implementation of the proposed channel and the results of tests conducted on the proposed channel, including measurements of channel errors, available data rate for transmission and channel detectability, are presented. The results validate the viability of the proposed covert channel and demonstrate that robustness of the channel to frame errors can be improved by using well known forward error correction and interleaving techniques.**

Keywords - IEEE802.11 MAC frame; frame forging; covert channel; protocol version

I. INTRODUCTION

In this paper, it is our intention to analyze and further extend on the proof of concept presented at the Third International Conference on Emerging Network Intelligence (EMERGING 2011) [1]. Additional tests were conducted, in order to evaluate the usability of the proposed covert channel. At EMERGING 2011, we proposed a covert channel that uses the MAC header of control frames to hide the covert information. This is achieved by forging frames that use the Protocol Version (PV) bits in a way that was not intended by the designers of the IEEE 802.11 standard [2]. Specifically, the PV field and selected control bits in the MAC header field are used to accomplish this. Our work also investigates the error robustness and throughput of the channel, supported by experimental results.

As wireless networks become more ubiquitous, so do our dependencies on them. According to an industry report, in 2012 over a billion devices will be shipped with technology based on this standard onboard and the number is projected to be over two billion in 2014 [3]. Mobility and ease of access of wireless networks are very attractive characteristics to the end users, but along with them come additional security concerns [4], [5].

In order to protect wireless networks from being exploited, we need to constantly evaluate their vulnerabilities and devise techniques to mitigate them. Finding possible covert channels presents an ongoing challenge, and the potential uses for such channels range from well-intentioned authentication mechanisms [6] to malware propagation [7], exfiltration [8] or command and control of botnets [9].

The importance of investigating as many covert channels as possible should be obvious, since each networking standard has its own unique characteristics to exploit. For this reason, it is generally accepted that covert channels cannot be completely eliminated due to numerous variations in their implementation [10], [11].

Many covert channels have been documented over the years and reflect the technological stage of the networks at which they were documented. The idea of network covert channels was documented 25 years ago by Girling [12], although the concept of a system-based covert channel was initially presented by Lampson in 1973 [13]. The vast majority of academic research has focused on documenting covert channels in layer 3 (network layer) or above (transport, session, presentation and application layers) of the OSI model [14]. These types of covert channels based on higher layer protocols span a wider variety of networks, since they are not limited by the physical or medium access mechanisms. The two most explored protocols above layer 2 (data link layer) are IP and TCP [11], [15], [16]. Even higher layer protocols, such as ICMP, HTTP or DNS, have several documented covert channels [17], [18], [19].

Recently, researchers began investigating wireless networks, specifically identifying covert channels in the MAC layer [20], [21], [22], [23]. Frame forging plays a key role in this type of covert channel. Creating fake frames with modified header bits is a recurring theme to implement such channels. MAC header fields such as the sequence number [22], initialization vector [22] or destination address [23], have been used to hide the covert information.

The rest of the paper is organized as follows. Section II presents an overview of the IEEE802.11 MAC frame fields and an analysis of network frame traffic. The proposed covert channel is described in Section III. Section IV presents the results of new experiments. Section V closes this work with a brief conclusion and future work.

II. IEEE802.11 NETWORKS AND FRAME TRAFFIC

IEEE 802.11 based wireless nodes share a common medium for communication. The fundamental building block of the 802.11 architecture is called the Basic Service Set (BSS). One BSS may be connected to other BSSs via a Distribution System (DS). Within this framework, stations

can connect in ad-hoc mode or infrastructure mode. The simpler case is ad-hoc mode, where two stations can connect directly, point to point, without a DS and an Access Point (AP). If we have the stations connecting via an AP and making use of a DS, then we say they are setup in infrastructure mode.

A. 802.11 MAC frame format

A generic MAC format for an 802.11 MAC frame can be seen in Figure 1. The frame consists of the MAC header, the frame body and the Frame Check Sequence (FCS).

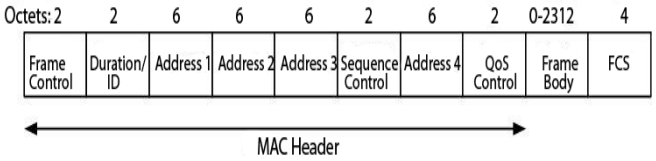

Figure 1. MAC frame format (from [2]).

The first field in the MAC header is the Frame Control (FC), consisting of two octets, and its contents are shown in Figure 2, with the PV field highlighted. This field consists of two bits that represent the version number of the 802.11 protocol being used. As of this writing, PV is expected to be set to zero [2]. This value may change in the future if a newer version of the standard is released.

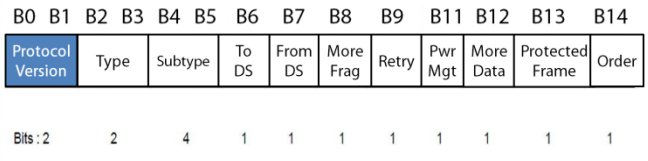

Figure 2. Frame control field (from [2]).

In the proposed covert channel, we utilize the remaining three possible combinations of the PV field to hide the covert information.

B. Frame Types of Interest

Four different types of frames exist in the 802.11 protocol: management, data, reserved and control frames.

Control type frames facilitate the exchange of data frames between stations. Within the existing control subtypes, we are interested in the smaller sized frames, the Acknowledgement (ACK) and the Clear To Send (CTS). These frames also tend to be present in large volume.

The IEEE 802.11 MAC layer makes use of the CSMA/CA scheme, in order to minimize the number of collisions and subsequent frame loss. To address the hidden node problem [24], a RTS/CTS handshake mechanism is used. The CTS is a 14-byte long frame whereas the RTS is 20 bytes long.

The ACK frame is generated when a station correctly receives a packet, and it is intended to signal the source station that the reception was successful. For such reason, this type of frame also tends to be very common in an operational wireless network. The length of this frame is the same as the CTS, 14 bytes.

Both frames share the same format and they only differ in one bit in the subtype field within the frame control. The ACK frame has the subtype value set to "1101"; the CTS sets it to "1100".

C. Network Analysis

A heavily used 802.11 network on campus is monitored to collect frame traffic on multiple channels. From the MAC frame traffic collected, channel 1 is found to be the one with most traffic volume and number of users. We collected over 22 million packets to analyze the following frame basic characteristics.

Ideally, we want a frame that is short in length, common in occurrence, and still valid if some bits are changed. Additionally, its presence in bursts should not be a rare event. These features are desirable for achieving a reasonable throughput while providing covertness.

The results of our analysis are shown in Figure 3 as a pie chart, which represents the frequency of occurrence of different types of frames. The data frames are dominant, followed by CTS, ACK and beacons. The "others" refers to the sum of all other frames that represent less than 1% individually. From this plot we can clearly see that two types of control frames matching our needs stand out, the ACK and the CTS.

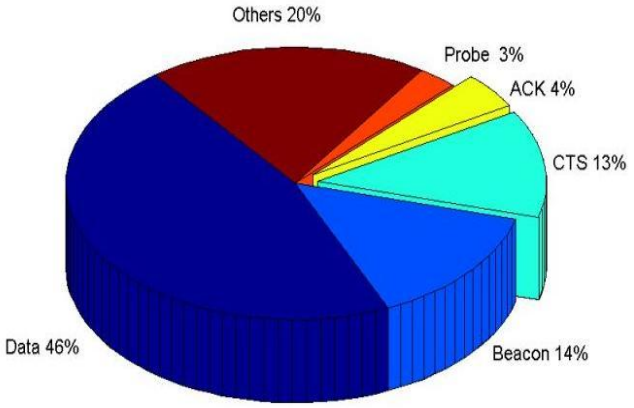

Figure 3. Frequency of occurrence of the monitored frame types.

D. Choosing the Frame Type

In the process of choosing a frame for the covert channel, several frames were considered, such as RTS and ACK. These frames could serve as well as the CTS, but they were found to be less frequent than CTS. In addition, among these three frames, RTS is the longest one with 20 bytes, and the CTS and ACK have only 14 bytes. For this reason, we narrowed the options to ACK and CTS.

From monitoring of frame traffic on the campus wireless network and empirical analysis, we found that the CTSs occur with a frequency two times higher than that of the ACKs. Different traffic scenarios were monitored, ranging from low traffic periods to high levels of utilization of the network. We chose to use CTS for building the proposed covert channel as the CTS traffic volume is large and is of

same frame size as ACK. By choosing CTS, we can minimize the chance of causing a traffic anomaly based on the type and frequency of packets flowing through the network.

Since CTS and ACK have a similar frame structure, it is easier to switch from one to the other, according to our objectives. The main concept of the proposed covert channel applies equally to both frames. It is even possible to have one end of the channel transmitting ACK frames, and the other transmitting CTS frames, without any loss or degradation of performance. Alternating frame types, such as transmitting a forged ACK followed by a forged CTS is also viable. Many other variations are also feasible.

The fact that both CTS and ACK frames do not contain a source address also contributes to a higher level of stealthiness, since it is not possible to immediately identify the source of the transmission.

III. PROPOSED COVERT CHANNEL

This section describes the proposed covert channel and the use of forward error correction and bit interleaving mechanisms to improve its performance.

A. MAC Header Manipulation

In the proposed covert channel we use two bits in the PV field of the MAC header of an 802.11 CTS packet to carry covert information. The proposed covert channel uses the PV bits in a variety of ways to signal the beginning and end of the transmission as well as to carry the information, one bit at a time. A graphical representation of the manipulated bits is shown in Figure 4.

B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B11	B12	B13	B14
Protocol Version		Type		Subtype		To DS	From DS	More Frag	Retry	Pwr Mgt	More Data	Protected Frame	Order
Bits : 2		2		4		1	1	1	1	1	1	1	1

Figure 4. Manipulated bits in Frame Control field (blue squares).

In order to facilitate communication in the proposed covert channel, we divide the transmission into three segments: start message delimiter, message, and end message delimiter. Marking the beginning and end of the covert communication allows us to establish the channel with the covert receiver station, while non-covert receivers will still see the forged frames as non-expected, thus dropping them. The start and end delimiters are realized by transmitting a sequence of five frames with "01" in the PV field. The message bits are transmitted using combinations of "10" as binary "zero" and "11" as binary "one" in the PV field. The message is organized into 8-bit ASCII characters. Figure 5 shows a capture of Wireshark [25] in which we can see the transmission of the ASCII character “A” converted into the binary string "01000001". A total of 18 frames were transmitted. Looking at the first column (Protocol) we see the identification of a valid CTS format, it is only when looking to the second column (Protocol Version) that our frame manipulation becomes noticeable. There we can see the changes in the PV number, from 1 to 3.

B. Forward Error Correction

Since we are operating in a shared media, collisions will eventually occur. This will be interpreted as an error, since a frame carrying covert payload will be lost. To mitigate the effect of frame losses, and thus reduce the number of errors in the covert channel, the use Forward Error Correction (FEC) was considered.

There are several options for implementing FEC: block codes such as Hamming and Reed-Solomon, convolutional codes, turbo codes, or low density parity check codes [26]. In this work, however, a convolutional code was used for error correction.

A convolutional coder takes an m - bit message and encodes it into an n - bit symbol. The ratio m/n is known as the code rate. In our case a code rate of $2/3$ was used, meaning the encoded message will be one and a half times as long as the original message. This will increase the time needed to transmit the same message as before, since a higher number of bits is being sent.

Another important parameter in convolutional coding is the constraint length. This parameter, *k*, represents the number of bits in the encoder memory that affect the

Protocol	Prot Version	Dest Addrr	#	Length	Flags
Clear-to-send, Flags=.....	1	11:0c:f1:0b:7e:1e (RA)	1	14	0x00
Clear-to-send, Flags=.....	1	11:0c:f1:0b:7e:1e (RA)	2	14	0x00
Clear-to-send, Flags=.....	1	11:0c:f1:0b:7e:1e (RA)	3	14	0x00
Clear-to-send, Flags=.....	1	11:0c:f1:0b:7e:1e (RA)	4	14	0x00
Clear-to-send, Flags=.....	1	11:0c:f1:0b:7e:1e (RA)	5	14	0x00
Clear-to-send, Flags=.....	2	11:0c:f1:0b:7e:1e (RA)	6	14	0x00
Clear-to-send, Flags=.....	3	11:0c:f1:0b:7e:1e (RA)	7	14	0x00
Clear-to-send, Flags=.....	2	11:0c:f1:0b:7e:1e (RA)	8	14	0x00
Clear-to-send, Flags=.....	2	11:0c:f1:0b:7e:1e (RA)	9	14	0x00
Clear-to-send, Flags=.....	2	11:0c:f1:0b:7e:1e (RA)	10	14	0x00
Clear-to-send, Flags=.....	2	11:0c:f1:0b:7e:1e (RA)	11	14	0x00
Clear-to-send, Flags=.....	2	11:0c:f1:0b:7e:1e (RA)	12	14	0x00
Clear-to-send, Flags=.....	3	11:0c:f1:0b:7e:1e (RA)	13	14	0x00
Clear-to-send, Flags=.....	1	11:0c:f1:0b:7e:1e (RA)	14	14	0x00
Clear-to-send, Flags=.....	1	11:0c:f1:0b:7e:1e (RA)	15	14	0x00
Clear-to-send, Flags=.....	1	11:0c:f1:0b:7e:1e (RA)	16	14	0x00
Clear-to-send, Flags=.....	1	11:0c:f1:0b:7e:1e (RA)	17	14	0x00
Clear-to-send, Flags=.....	1	11:0c:f1:0b:7e:1e (RA)	18	14	0x00

Figure 5. Wireshark capture of an “A” being transmitted using the proposed covert channel.

generation of the n output bits [26]. A constraint length of 4 is used in our experiments.

Forward error correction is typically applied to the transmission of a stream of bits sent and received sequentially. In our case, however, the bits are embedded into independent frames, which are prone to loss. As a result, when a frame is lost, the receiver has no indication that a bit is missing. Consequently, we now need to know exactly which frames were lost in order to apply the FEC correctly.

One option is to use the eight flag bits in the frame control field of the MAC header to index a longer sequence number, which makes determining the location of lost frames an easier task. These flag bits will not carry any covert information but serve only the error correction function. However, it is important to state that applying this use of the flag bits will increase the probability of detection of the covert channel, since unexpected flag attributions will be present. In this situation, we move from a minimum deviation, from a legitimate CTS, of two bits (as in Figure 4) to a maximum of 10 bits (as in Figure 6). This presents a tradeoff between detectability and error performance, and the user must exercise the option to choose one over the other as dictated by the application. In order not to use the flag bits one could use the type and subtype fields of the MAC header. The IEEE802.11 standard defines some bit combinations of the subtype field as "Reserved" [2]. Exploring these combinations could be an option, although we did not test it.

Figure 6 is a representation of how we accommodated the information and sequence bits within the MAC header.

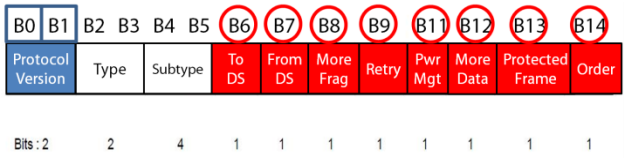

Figure 6. Representation of the frame structure using the flag bits for sequencing (red circles).

The blue squares represent our covert channel bits. These bits are used in the same way as before: the first bit (B0) signals the presence of the channel and the second is payload (B1). The red circles refer to the sequence bits, which are placed in the flag bits of the frame control field.

Given that we have eight flags, this gives us a total of 2^8 possible sequence numbers. This alone provides a reasonable amount of protection against a long burst of frame losses, when compared to the previous approach.

C. Forward Error Correction and Interleaving

We now consider sending more than one bit of information per forged frame.

Since each frame now carries more than one information bit, the loss of one or more frames has a larger impact on the total number of errors in the channel. In order to mitigate this effect, we interleave the bit string resulting from the convolutional coder. This consists of breaking the coded message in blocks of 8 bits, building a matrix with each block in a different row. By reading the matrix out by column, from top to bottom, we generate a new string of bits, effectively interleaving all the 8 bit blocks [27], [28]. The number of rows depends on the length of the message we are transmitting.

At the output of the convolutional coder we interleave the bits in groups of 8 bits. This results in a new string of zeros and ones, which goes into the covert channel processing block. Here the string is separated in groups of n bits, and each group will become the payload of the forged frames. Figure 7 is a schematic representation of this idea.

Notice that only information bits are encoded and interleaved; in this implementation the convolutional coder is applied after we gather the complete message we want to transmit.

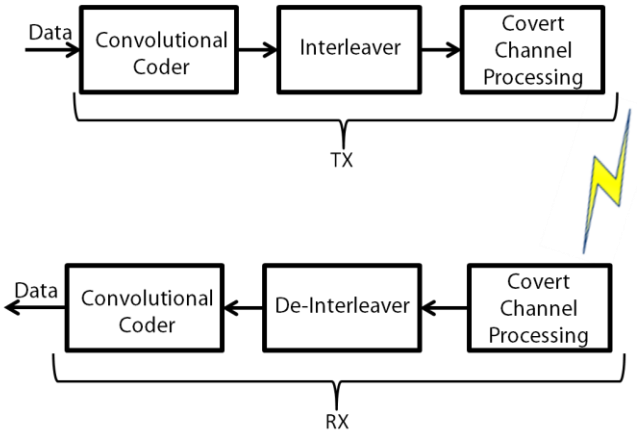

Figure 7. FEC and interleaving block diagram.

One possible implementation is to use six bits for payload. The frame is forged as follows: six information bits are placed in the selected flag bits, three other bits are used for sequence numbers, and the first PV bit is set to one, indicating the use of the covert channel. Figure 8 illustrates the proposed structure. The blue squares indicate payload bits, and the red circles are sequence numbers. The green diamond (B0) indicates the presence of the covert channel. Bits B1, B8 and B9 form the sequence number yielding a sequence length of eight. Bits B10-B15 form the payload, using six bits to carry the message.

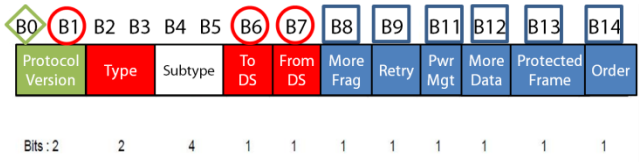

Figure 8. Representation of the frame structure using three bits for sequencing (red squares) and six bits for payload (blue squares).

IV. Experiments and Results

In order to implement the proposed covert channel, we developed the necessary code to forge, transmit, and receive frames. Python [29] was the chosen programming language,

due to its simplicity, available libraries and extension modules that facilitated our task. The elected OS was Linux, for being more flexible, open source and GNU licensed.

The code is divided into three threads running simultaneously, as presented in Figure 9. One thread runs as the receiver, counting the initialing sequence that marks the opening of the covert channel, buffering the received message and identifying the closing sequence. At that point, it starts the recovery part of the process, corresponding to the right half of Figure 7. Thread2 acts as the transmitter, executing all the tasks shown on the left half of Figure 7.

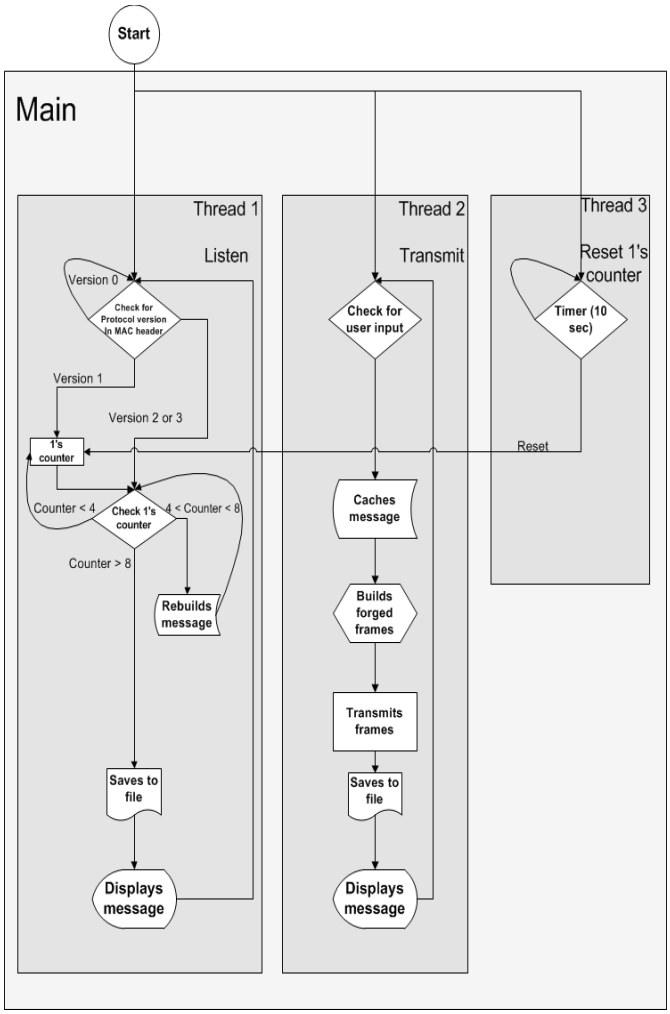

Figure 9. Flow chart of the covert channel code implementation.

Finally, the third thread contains a control mechanism to filter possible discrepancies in the identification of the beginning and end of the covert communication. During our monitoring of real working wireless networks, other version 1 frames (with bad checksums) were found circulating in the network, thus becoming noise to our version 1 frames. Thread3 is responsible for filtering out these unwanted frames.

A. Test bed

Tests were conducted in a laboratorial environment, with controlled levels of interference. All measurements were made using the same stations in the same relative positions inside a closed room. The levels of interference ranged from 0 to 1000 frames per second. ARP frames were used as the interfering frames, at a fixed transmission rate, due to its simplicity to generate with common exploit tools, such as *aireplay-ng* [30]. In order to have a considerable level of interference, all ARP frames were made to be 72 bytes long.

The tested scenario consisted of transmitting the same covert message, while varying the number of payload bits and applying different levels of interfering traffic in each trial. The standard sentence used in our tests has an original length of 1408 bits, which translates to a total of 2112 bits after applying FEC and interleaving, with the intention of improving the error robustness of the channel. In order to collect a statistically relevant sample, each sentence was repeatedly transmitted between 500 to 1500 times for each payload size (from 1 to 6 payload bits) and interference level (from 0 to 1000 in steps of 125 ARP frames per second). This brings the number of analyzed bits close to 60 million for each trial. The different number of sentence transmissions is due to the need of balancing the number of transmitted frames in order to make a fare comparison between trials, since different payload sizes impact on the total number of frames to be forged in order to send the entire message.

Three laptops with the same hardware configuration were used, utilizing a PCM 3COM 3CRPAG175 with an Atheros chip AR5212 as the wireless network adapter. One laptop was setup as transmitter (Station A), another as noise source (Station B), and the third one as passive monitor (Station C). Stations A and B were running Backtrack4 as OS, and Station C ran Windows XP SP2. The monitoring program used was Airopeek NX, version 3.0.1 [31].

It is important to notice that stations A (source of messages) and B (source of interference) are operating in ad-hoc mode, outside any infrastructure wireless network. The stations transmit without coordination from access points. Our intention is to cause collisions, and thus frame losses, which are interpreted as errors for analysis purposes.

Experimental data is then collected by station, C, set to collect all frames in promiscuous mode.

All active communications during the tests are unidirectional, being broadcasted from one unique station. This setting rules out any tests to the receiving part of the covert channel code, since station C is collecting all the traffic for later analysis, not decoding the message in real time.

The expected error performance is displayed in Figure 10, showing the number of errors, in a log scale on the Y-axis, versus the interference level on the X-axis. We expect to see an increase in the number of errors as the interference level rises. At some point, the use of FEC will introduce more errors than the ones originally received, becoming disadvantageous. The blue solid line represents the number of errors after the use of FEC, where the red dotted line

represents the number of error at the reception, before going through FEC. We assume this last condition to be equivalent to having the exact same message without FEC, since we do not process that algorithm, and measure errors in raw as they are received.

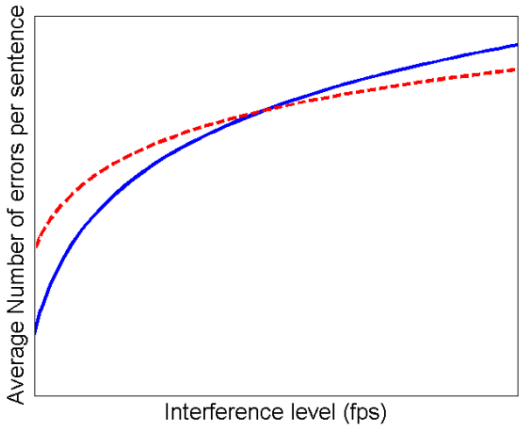

Figure 10. Expected evolution of the number of errors without FEC (red dashed line) and using FEC (blue solid line).

B. Results

1) Fixed payload size of 2 bits

For a payload size of 2 bits and 7 bits for sequencing, the collected data is displayed in Figure 11. In this case we are able to resist some bursts of errors, since the addressing space is still fairly large (2^7), enabling us to pinpoint the missing bits. We can see all the errors being corrected for an interference level up to 125 frames per second.

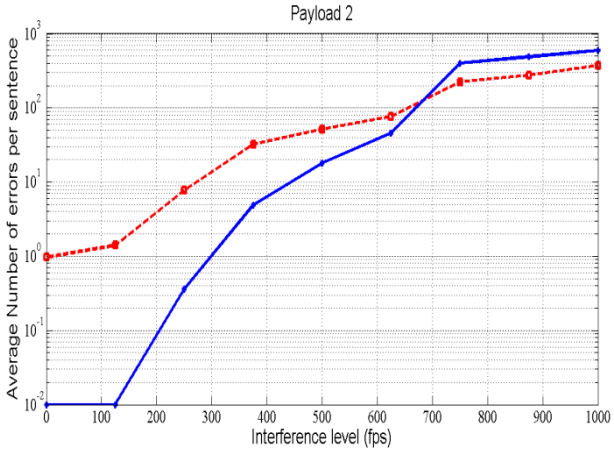

Figure 11. Number of errors for a fixed payload size of 2 bits at different interference levels. Blue solid line is the result after FEC, red dashed line before FEC.

Above that interference level, the number of errors after FEC tends to increase and at approximately 675 frames per second of interference it crosses over and surpasses the number of errors when no FEC is in place. With such payload per frame, it was necessary to generate 1056 frames for the complete message, plus the 10 marking frames. This adds to a total of 119392 bits for each sequence. In this trial the sequence was transmitted 500 times, so the total number of transmitted bits approaches the $60x10^6$.

The benefits of this configuration are present in low to medium levels of interference. This result is consistent with our expectations.

2) Fixed payload size 4 bits

For a payload size of 4 bits and 5 bits for sequencing, the collected data is displayed in Figure 12. Here we can see how the number of errors increases at low interference levels, when compared to the previous results, confirming an expected degradation in the channel quality. The crossover point happens at around 475 frames per second.

By increasing the payload we reduced the number of forged frames to send the covert message. In this case, the number of repetitions of each sentence was 1000, as opposed to the 500 of the previous setting.

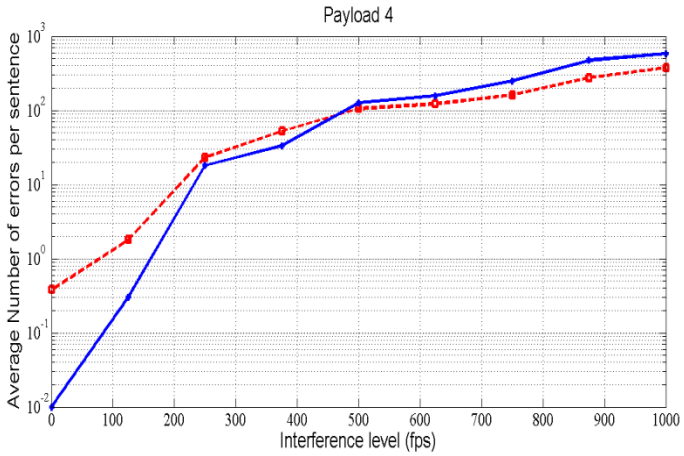

Figure 12. Number of errors for a fixed payload size of 4 bits at different interference levels. Blue solid line is the result after FEC, red dashed line before FEC.

3) Fixed payload size 6 bits

For a payload size of 6 bits and only 3 bits for sequencing, the collected data is displayed in Figure 13.

With these settings, the level of interference needed to induce a large number of errors does not have to be very high. The crossover point happens at a lower level of interference, around 200 frames per second.

For this payload size we had to increase the number of repetitions up to 1500 for each trial.

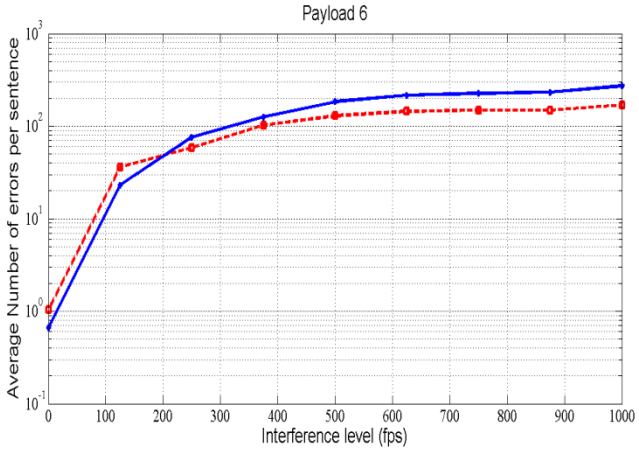

Figure 13. Number of errors for a fixed payload size of 6 bits at different interference levels. Blue solid line is the result after FEC, red dashed line before FEC.

4) Fixed interference rate

Taking a different approach to the collected data, we analyzed the effect of a fixed interference level (500 frames per second) given different payload sizes, ranging from 1 to 6 payload bits. Here we can observe how the crossover point sits between a 3 and 4 payload size. Before that point FEC is an important technique to enhance the channel resilience. From that point on, we can clearly see the effect FEC has in generating more errors than the ones received.

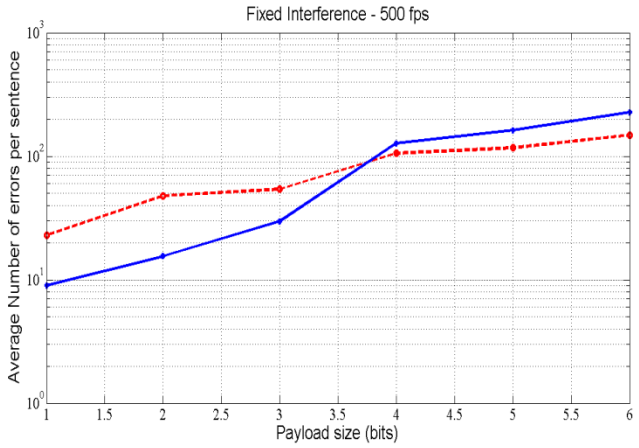

Figure 14. Percentage of errors for a fixed interference rate and payloads between 1 and 6. Blue solidline is the result after FEC, red dashed line before FEC.

A relevant aspect taken from these experiments is the motion of the cross over point, as we change the interference level. This can be used as a reference for changing the payload size as the level of interference changes. To find the level of interference, and since we intend to operate inside a structured network, we can use the data rate information embedded in each packet. This would allow an adaptive covert channel to be created, responding to different levels of interference with different payload sizes.

C. Throughput Analysis

In order to evaluate the throughput offered by the proposed channel, the rate at which the frames are transmitted is measured. Being a proof of concept, code efficiency was not a major concern, and the results are presented for analysis purpose only, meaning significant improvements may be easily achieved. This was done using Airopeek [31] and by averaging the rate of the forged frames on a per second basis. Depending on the network usage at the time, the frame rate varies significantly. Another factor responsible for this variation is the continuous adjustment of the maximum data rate of the network as dictated by the channel conditions. For IEEE 802.11b networks the maximum network data rate possible values are 1, 2, 5.5, and 11 Mbps [2].

To obtain a benchmark for performance comparison, we first determine the maximum data rate possible for the covert channel under optimal conditions. The following conditions are assumed:

(i) The channel is ideal with no errors;
(ii) there is only one station with frames to transmit;
(iii) we use a data rate of 2 Mbps, the highest possible for 802.11b control frames (basic rate set) [2].

The medium access scheme has to obey some predetermined timing constraints, set by the standard. Figure 15 is a graphical representation of the timing requirements for transmitting a frame.

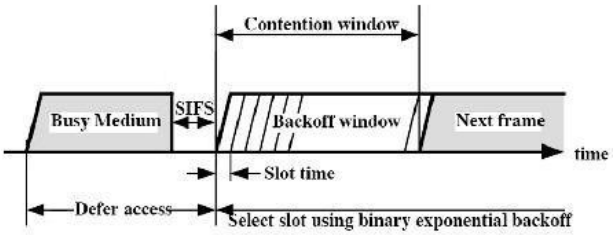

Figure 15. Timing constraints in an 802.11 frame transmission [After 32].

Applying the work of Xiao and Rosdhal [33] and Jun et al. [34] to the proposed covert channel, the minimum amount of time necessary to transmit a forged CTS is $t_{min} = 376$ µs, corresponding to a maximum of 2659 forged frames per second. At one bit per frame the maximum bit rate is 2659 bps; at six bits per frame we get 15.954 kbps. The measured throughput values, however, will be significantly smaller.

When we transmit one bit of information in each forged frame, we have an overhead of the start and end delimiters for a total of 10 signaling frames. The measured average frame rate is 61 frames per second. Since each frame represents a bit, and considering our message payload of 1408 bits, we transmit a total of 1418 bits. At 61 frames per second this corresponds to a total transmission time of 23.25 sec, and a useful bit rate or throughput of 60.5 bps.

On the other hand, when we transmit 6 bits per forged frame and introduce the use of interleaving, the measured average transmission rate is 32 frames per second. By transmitting a total of 2122 bits, we obtain a transmission

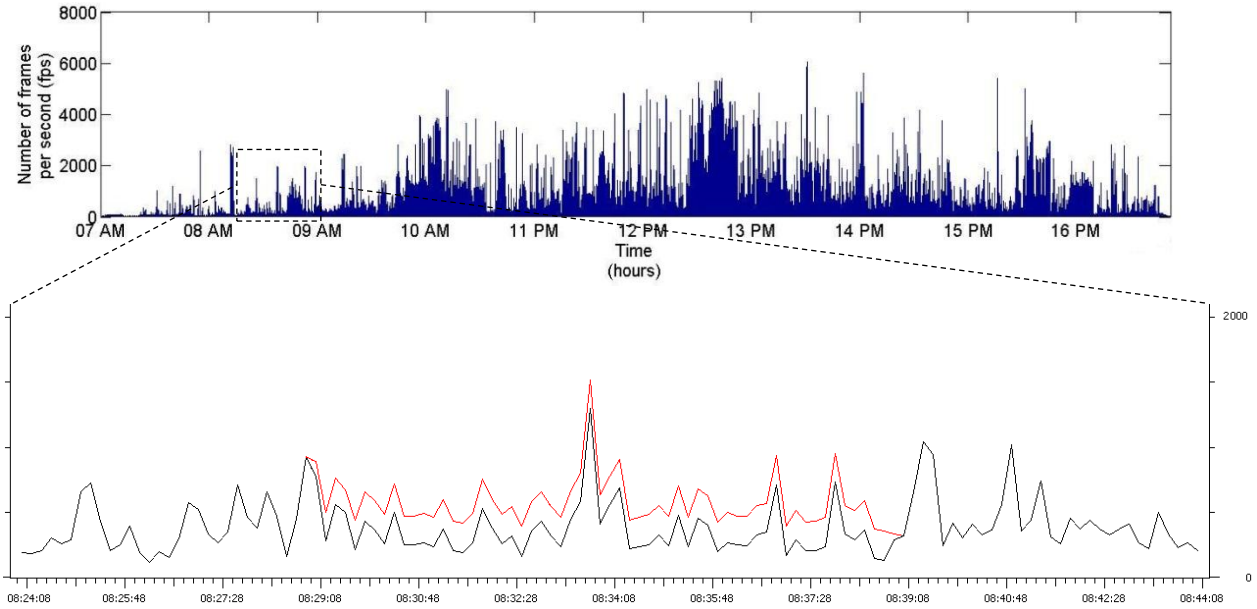

Figure 16. Traffic profile of a low usage WiFi channel on campus

time of 11 seconds. The resulting throughput value is 127.4 bps, considerable improvement over the previous case.

D. Covertness Analysis

By evaluating the impact our proposed channel has in an operating wireless network, we can work to reduce its detectability by making use of traffic profile analysis. Figure 16 is a partial magnification of the traffic profile of a busy channel during a normal working day at campus, where the black (lower) line represents the normal network traffic, and the red (upper) line shows the normal traffic plus the traffic due to covert (forged) frames. As we can see in Figure 16, the difference between the red line and the black line corresponds to the amount of traffic added by the use of the covert channel. Since the network traffic is fairly heavy, the presence of the covert channel is not obvious; our traffic just blends in with the overall traffic.

Selected portion of network traffic profile showing the additional traffic generated by the use of the proposed covert channel (red top line).

Applying the same reasoning to a low traffic channel, in the same campus area, during a normal working day, we can notice a substantial difference. Figure 17 shows the traffic profile of an available but modestly used channel, and without any covert traffic.

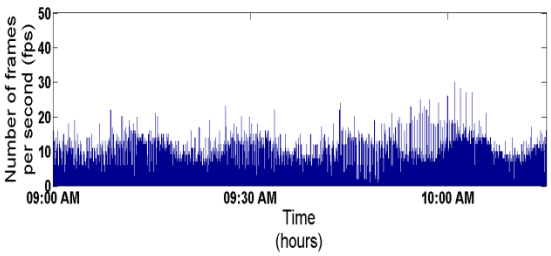

Figure 17. Traffic profile of a low usage WiFi channel on campus.

Figure 18 clearly shows the impact our covert transmissions have in the traffic profile of such channel. The traffic peaks are the result of injecting forged frames. In this case, our covert message being transmitted was a random sequence of 3500 bits. Zooming in and splitting the two types of traffic, as seen in Figure18, reveals the presence of an abnormal amount of traffic. In this case, there is a significant difference between the two types of traffic, with or without covert transmissions.

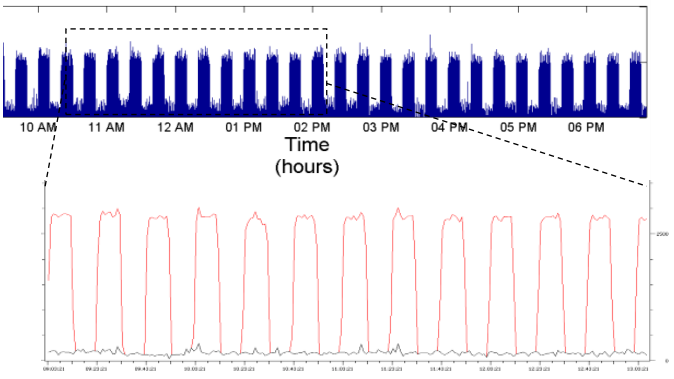

Figure 18. Selected portion of network traffic profile showing the additional traffic generated by the use of the proposed covert channel (red top line).

Figure 19 displays the traffic profile when using different rates of transmission and their impact. The stealthiness of the channel can be improved by spacing the transmission of forged frames. How the covert traffic can be made less visible by introducing spacing between frames is illustrated below. One of the down sides of this manipulation is the obvious reduction of throughput. Segment (a) in Figure 19 corresponds to normal frame transmission with no additional

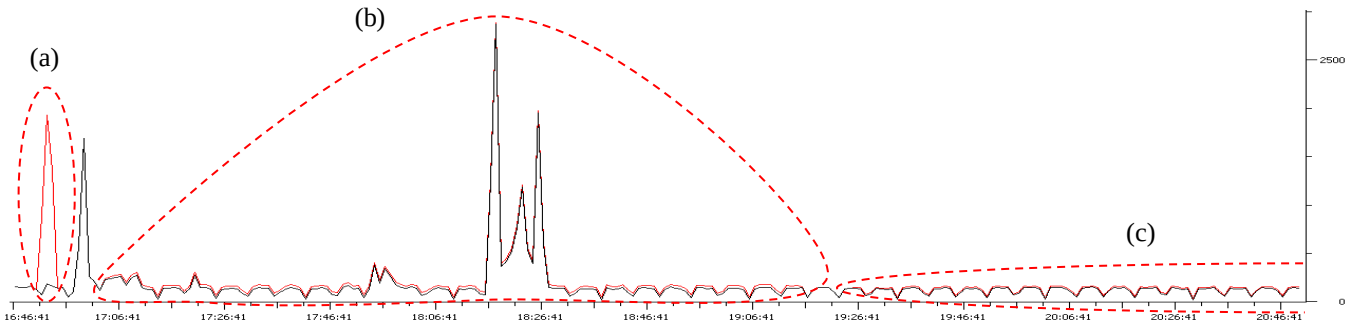

Figure 19. Selected portion of network traffic profile showing the effect of different transmition rates.

spacing between the frames, i.e., no delay was introduced between transmitted frames. For this segment, the total transmission time is approximately two minutes at a measured average of 30 frames per second.

In segment (b), frames are sent once every two seconds, resulting in a total transmission time of 2 hours and 12 minutes. Finally, segment (c) is shown only partially; we sent one frame every four seconds for a total transmission time of 4.5 hours.

The important aspect is that the difference between the legitimate and covert traffic becomes smaller and smaller as the spacing increases; at some point, it is possible to make it almost invisible as we extend the spacing. On the other hand, the throughput is degrading proportionally.

Another technique to camouflage our use of the covert channel is to space the forged frames transmission in a non-uniform way instead of sending the frames at regular time intervals. Although considered, this variation was not tested.

V. CONCLUSIONS

This work presented, implemented, and tested a MAC layer 802.11 covert channel. We used the PV field in the MAC header to hide and transfer the covert information. Within the MAC header we used the PV field, as well as the flag bits, to hide our message.

The proposed covert channel was implemented by developing the necessary code in Python. A GUI chat console is used for message transmission. In our approach, users only have to run a single Python script in order to access the chat console. The test bed used for experiments operated in a Linux environment.

Robustness to errors in the covert channel is improved by the use of forward error correction and bit interleaving. Results indicate significant improvement in the error performance of the channel for low interference rates.

Detectability of the use of the proposed covert channel was also investigated, demonstrating a method of minimizing our exposure to such type of analysis. Throughput is severely affected once we try to reduce our traffic profile footprint.

The achieved throughput of the covert channel was measured under two different scenarios, in which we changed the size of the payload. The maximum channel data rate was also determined. The case of 6-bit payload along with convolutional coding and interleaving yielded the highest measured throughput.

This proof of concept can benefit considerably from future work. This may include code optimization in order to increase the frame rate, testing other error correction algorithms, and embedding this covert channel into legitimate traffic, directly at a firmware level, instead of relying on the injection of forged frames.

REFERENCES

[1] R. Gonçalves, M. Tummala, and J. McEachen, "A MAC Layer Covert Channel in 802.11 Networks", The Third International Conference on Emerging Network Intelligence EMERGING 2011, November 20-25, 2011 - Lisbon, Portugal

[2] Institute of Electrical and Electronics Engineers, 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (accessed January 17, 2011).
http://ieeexplore.ieee.org

[3] D. McGrath, "WLAN chip set shipments projected to double," in EE Times, 2/17/2011. (accessed March 17, 2011)
http://www.eetimes.com/electronics-news/4213260/WLAN-chip-set-shipments-projected-to-double

[4] H. Yang, F. Ricciato, S. Lu, and L. Zhang, "Securing a wireless world," in Proceedings of the IEEE, vo1.94, Issue 2, pp. 442-454, February 2006.

[5] Y. Xiao, C. Bandela, and Y. Pan, "Vulnerabilities and security enhancements for the IEEE 802.11 WLANs," in Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM) 2005, pp. 1655-1659, 2005.

[6] T.E. Calhoun, R. Newman, and R. Beyah, "Authentication in 802.11 LANs Using a Covert Side Channel," in Communications, 2009. ICC '09. IEEE International Conference, pp. 1-6, 14-18 June 2009.

[7] E. Couture, "Covert Channels," SANS Institute InfoSec Reading Room (accessed January 17, 2011).
http://www.sans.org/reading_room/whitepapers/detection/covert-channels_33413

[8] A. Giani, V. H. Berk, and G. V. Cybenko, "Data Exfiltration and Covert Channels," Process Query Systems, Thayer School of Engineering at Dartmouth (accessed February 02, 2011).
http://www.pqsnet.net/~vince/papers/SPIE06_exfil.ps.gz

[9] D.T. Ha, G. Yan, S. Eidenbenz, and H.Q. Ngo, "On the effectiveness of structural detection and defense against P2P-based botnets," in Dependable Systems & Networks, 2009. DSN '09. IEEE/IFIP International Conference, pp. 297-306, June 29 2009-July 2 2009.

[10] S. Hammouda, L. Maalej, and Z. Trabelsi, "Towards Optimized TCP/IP Covert Channels Detection, IDS and Firewall Integration," in New Technologies, Mobility and Security, 2008. NTMS '08., pp.1-5, 5-7 Nov. 2008.

[11] C. H. Rowland, "Covert channels in the TCP/IP protocol suite," in Tech. Rep. 5, First Monday, Peer Reviewed Journal on the Internet, July 1997.

[12] C.G. Girling, "Covert Channels in LAN's," in Software Engineering, IEEE Transactions, vol. SE-13, no. 2, pp. 292-296, Feb. 1987.

[13] B. Lampson, "A note on the confinement problem," in Communications of the ACM, vol. 16, pp. 613-615, October 1973.

[14] U.S. Department of Defense, Trusted Computer System Evaluation Criteria, pp. 80, DoD 5200.28-STD, July 1985.

[15] S. Cabuk, C.E. Brodley, and C. Shields, "IP Covert Timing Channels: Design and Detection," in Proc. 11th ACM Conf. Computer and Communications Security (CCS), pp. 178–87, October 25–29 2004.

[16] S. J. Murdoch and S. Lewis, "Embedding Covert Channels into TCP/IP," in Proc. 7th Information Hiding Workshop, June 2005.

[17] M. Bauer, "New Covert Channels in HTTP: Adding Unwitting Web Browsers to Anonymity Sets," in Proceedings of the 2003 ACM Workshop on Privacy in Electronic Society, pp. 72–78, October 2003.

[18] A. Galatenko, A. Grusho, A. Kniazev, and E. Timonina, "Statistical Covert Channels Through PROXY Server," Proceedings 3rd International Workshop - Mathematical Methods, Models, and Architectures for Computer Network Security, pp. 424–29, September 2005.

[19] M. Smeets and M. Koot, "Research report: covert channels," Master's thesis, University of Amsterdam, February 2006.

[20] S. Li and A. Ephremides, "A network layer covert channel in ad-hoc wireless networks," Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference, pp. 88-96, 4-7 October 2004.

[21] T. Calhoun, X. Cao, Y. Li, and R. Beyah, "An 802.11 MAC layer covert channel," in Wireless Communications and Mobile Computing, Wiley InterScience (accessed January 2011).
http://onlinelibrary.wiley.com/doi/10.1002/wcm.969/pdf

[22] L. Frikha, Z. Trabelsi, and W. El-Hajj, "Implementation of a Covert Channel in the 802.11 Header," in Wireless Communications and Mobile Computing Conference, 2008. IWCMC '08., pp. 594-599, 6-8 Aug. 2008.

[23] L. Butti, Raw Covert (accessed September 2010) http://rfakeap.tuxfamily.org/#Raw_Covert

[24] F.A. Tobagi and L. Kleinrock, "Packet switching in radio channels: the hidden node problem in carrier sense multiple access modes and the busy tone solution," in IEEE Trans. Commun., vol. 23, pp. 1417-1433, 1975.

[25] Wireshark v.1.4.0 http://www.wireshark.org (accessed October 2010).

[26] S. Lin and D. J. Costello., Error Control Coding: Fundamentals and Applications, Pearson Prentice Hall, New Jersey, 1983.

[27] J. Proakis and M. Salehi, Digital Communications, Fifth edition, McGraw Hill, New York, 2008.

[28] L. Chen,T. Sun, M. Y. Sanadidi, and M. Gerla, "Improving wireless link throughput via interleaved FEC," in Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on , vol.1, no., pp. 539- 544 Vol.1, 28 June-1 July 2004.

[29] PythonTM Programming Language (accessed July 2010) http://www.python.org/

[30] Aircrack-ng / Aireplay-ng (accessed March 2012) http://www.aircrack-ng.org/doku.php?id=aireplay-ng

[31] Airopeek NX v.3.0.1 (accessed December 2010) http://www.wildpackets.com/

[32] W. Stallings, Wireless Communications and Networks, Second edition, Pearson Prentice Hall, New Jersey, 2005.

[33] Y. Xiao and J. Rosdahl, "Throughput and delay limits of IEEE 802.11," in Communications Letters, IEEE , vol.6, no.8, pp. 355- 357, Aug 2002.

[34] J. Jun, P. Peddabachagari, and M. Sichitiu, "Theoretical maximum throughput of IEEE 802.11 and its applications," in Network Computing and Applications, 2003. NCA 2003. Second IEEE International Symposium on, pp. 249-256, 16-18 April 2003.