The 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016)

Evaluating Hamming Distance as a CRC-Based Side-Channel Detection Measure in Wi-Fi Networks

Visal Chea, Miguel Vargas Martin, Ramiro Liscano

University of Ontario Institute of Technology, 2000 Simcoe St. N., Oshawa L1H 7K4, Canada

Abstract

Wireless technology has become a main player in communication through its desirable mobility characteristic. However, like many technologies, there are ways that it can be exploited. One of these ways is through side-channel communication, whereby secret messages are passed along by the purposeful corruption of frames. These side channels can be established by intentionally corrupting the Frame Check Sequence (FCS) field by using a Cyclic Redundancy Check (CRC) polynomial that is different from the standard CRC polynomial. Malicious nodes can exploit the fact that normal unsuspecting nodes will drop these frames since they appear as naturally corrupted frames. This paper presents a CRC Hamming distance metric as a feature for the detection of this type of side-channel communication. We previously proposed the use of Hamming distance as a metric to compare CRC values that are generated by different CRC polynomials. The hypothesis is that the mean Hamming distance between two CRC values generated by two different CRC polynomials would be significantly far apart than the mean Hamming distance of a CRC value of a frame that was naturally corrupted but was generated by the same CRC polynomial. Previously, to test that hypothesis, we used F-Scores on real data experiments under varying noisy conditions and side-channel throughput to show that there is a consistent and significant difference between the mean Hamming values of naturally corrupted frames to those that use the Koopman polynomial to calculate the CRC for side-channel communications. In the present work we evaluate the Hamming distance using Perceptron Learning and the Pocket Algorithm to classify packets as side-channel or otherwise.

1. Introduction

While wireless solutions allow for maximum portability in communication, they are vulnerable to many security risks that are aimed at exploiting this medium. The risks become more imperative when they relate to the mobile communication amongst a platoon of wireless nodes, where the information relayed amongst the nodes could be confidential or life critical.

© 2016 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Keywords: Side-channels; CRC polynomial; machine learning
One possible form of exploitation is done through manipulating the error detection mechanism put in place to guard against the unreliable nature of wireless communication. Errors in the transmission of frames can occur from signal fading, collisions and shadowing effects; the errors can be attributed to many factors such as terrain, distance of nodes and interference. To mitigate these unreliability issues, a Frame Check Sequence (FCS) field is added to the end of the frame which stores a Cyclic Redundancy Check (CRC) value calculated based on the contents of the frame. This value is calculated using a predetermined CRC polynomial before it is sent and checked at the receiver for consistency. One may exploit this function to intentionally corrupt the FCS by means of choosing a different CRC polynomial in order to establish a covert communication channel for the exchange of information between malicious nodes. This type of covert communication creates a virtual side-channel. Other malicious nodes can now communicate through this side-channel by interpreting these intentionally corrupted frames using the same CRC polynomial. The surrounding nodes that are unaware of the use of a different CRC polynomial will interpret any frames from the malicious nodes as corrupted and will see them as added noise.

The amount of noise in a wireless network is never constant which poses the first challenge in detection. If it were constant then it would be easy to detect the presence of side-channel when there is an abnormal increase in corrupted frames. This is not the case because a baseline noise value cannot be determined for comparison due to the unpredictable nature of wireless communications. This is evident when a nearby microwave is turned on and all of a sudden the transmission rates between nodes almost come to a crawling stop due to a significant corruption of frames.

The next challenge of detection, which stems from the type of network infrastructure that these side channels can coexist in, is that these infrastructures usually lack a central monitor, which increases the likelihood of errors when the communication medium is shared amongst its nodes. One infrastructure that possesses these characteristics and is of main interest is the Wireless Ad Hoc Network. In this Wireless Ad Hoc Network detection becomes very hard when the only metric used is the frame error rate caused by noisy environments. These vulnerabilities can also be extended to mobile environments such as Mobile Ad Hoc Networks (MANETs). In both environments the underlying challenge is the same which is distinguishing legitimate from illegitimate noise. This calls on the need to find another metric that can be used to identify an intentionally corrupted frame from a naturally corrupted frame. The metric evaluated in this paper is the Hamming distance measure of the received frame’s CRC from the standard CRC. In order to be able to confirm that this is a good feature to use for the detection of side-channel communication, it must be thoroughly validated.

In this paper we validate the Hamming distance metric under a number of realistic scenarios using Perceptron Learning and the Pocket Algorithm (PLPA). The remaining of the paper is organized as follows. Section 2 summarizes the related work; the Hamming distance metric and the hybrid testing environment are outlined in Sections 3 and 4. We present the validation of the metric in Section 5 and close with discussion and conclusions in Sections 6 and 7.

2. Related Work

Steganography is a Greek word meaning covered writing [1]. This is different from encryption because steganography deals with hiding secret messages from untrained eyes and encryption deals with masking or converting the secret message in a way that the message is known to be there but unreadable. Steganography or information hiding dates back to 440 B.C. where the Greeks used to tattoo the messengers body with wax with the secret message [1]. To the untrained eye the messenger is just seen as an unexpected visitor. The main goal is to hide that there is any communication going on between the two parties. This is one of the key characteristics of the side-channel presented in this paper.

Unlike existing intrusion detection techniques, detection of this type of side-channel cannot base its features or metrics on abnormal usage behaviour patterns or profiles [3]. For example, in the attempt of creating an intrusion detection system model, Denning [3] has listed some types of intrusions that could be alerted by abnormal usage patterns or profile. These include attempted break-in, masquerading/successful break-in, penetration by legitimate user, leakage by legitimate user, inference by legitimate user, Trojan horse, virus, and denial-of-service. According to Denning, in all these intrusion examples, the usage pattern can be compared to normal usage or legitimate users or processes. Furthermore, for example, denial-of-service can be detected by very high activity or access to resources by a particular user when all other activity by other users is low.
The detection of the side-channel anomaly does not have the same luxury of having a behavioural norm to compare with. As described, these side-channel frames can hide amongst legitimate corrupted frames and becomes difficult to distinguish when the wireless environment is unpredictable and normally is prone to errors. This confirms that existing intrusion detection features or metrics cannot be directly transferred in the use for detection of the side-channel and shows the need for new ones to be discovered.

Li et al.\textsuperscript{4} proposed the use of the frame error rate (FER) as a metric for detection of side-channel. The idea was to measure the FER within a time window and if there is an abnormal increase in errors within that window then this may signal the presence of the side-channel. The challenge remains that side-channel frames cannot be confidently distinguished from legitimate errors frames.

Work done by Madtha et al.\textsuperscript{5} proposed measuring Request to Send (RTS) and Clear to Send (CTS) control frames as a way of detecting side-channel communication. The assumption made was that before every data frame was sent a RTS and CTS combination would be sent and received in order to avoid collision. If there were no error in transmission the number of data frames would match the number of RTS/CTS frames counted. Conversely, if there were error in the transmission the number of data frames would be less than the number of RTS/CTS frames counted because the data frames would be seen as errors and dropped. The hypothesis was that abnormally high RTS frames with low data frames could signal the presence of side-channel communication because the side-channel frames would be seen as errors. The challenge with this metric is that it is prone to false positives due to natural frame errors. In addition, the RTS/CTS mechanism is an option for manufacturers to implement it, not a standard requirement. Both the FER and the RTS/CTS count can be useful metrics so long as they are combined with others to reduce the overall false positive rate of side-channel detection. One such complementary metric is the Hamming distance we proposed in\textsuperscript{2} and evaluated using F-Scores and confidence values to find optimal threshold values under different noisy conditions and side-channel rates. The present work complements our previous evaluation of the Hamming distance by means of machine learning using Perceptron Learning and the Pocket Algorithm.

Moore et al.\textsuperscript{6} tested the Hamming distance in a MANET resembling a platoon-like formation, and evaluated it using F-Scores, confidence levels, and a ROC curve. Moore et al. found the optimal threshold (i.e. the one with highest true positives but lowest false negatives) for their scenario to be 11 or 12, which falls in between the optimal threshold range of 10 to 12 that we found in\textsuperscript{2} for our own static (i.e., non-mobile) scenarios.

3. The Hamming Distance Metric

The Hamming distance is a common metric used in code theory to compare two different bit strings.\textsuperscript{11} The Hamming distance measures the number of bits that are different between any bit strings. What is interesting about the Hamming distance is that it is commonly used for error correction. A method called the “nearest neighbor decoding” uses the Hamming distance to figure out what possible code word was sent if the received code word was in error. This method states that the code word that was sent but was in error can be identified by the smallest Hamming distance value between the error code word and all its possible code words. This is under the assumption that the channel is not that noisy. This assumption is a valid one to make as there would be no point to communicate if the channel is so noisy that mostly all the frames are lost.

Even for an adversary a noisy channel is not ideal when trying to move data around. This assumption that the number of bits flipped are minimal during transmission paired with the knowledge that CRC polynomials are hash functions designed to avoid as many collisions as possible, sparked the idea of using Hamming distance as a possible metric for detection of side-channel frames. We propose to detect these intentionally corrupted side-channel frames by comparing the Hamming Distance (HD) difference between the received CRC value and the calculated value and if the Hamming distance is significant this would trigger an alert. The hypothesis is that the HD value of CRCs (calculated and received in FCS) may be significantly low compared to that of an intentionally corrupted frame. This is under the following assumptions:

1. The CRC polynomial used is one that is known and used in a real implementation such as the default, Koopman or Castagnoli.
2. The adversary will not use encryption or change the integrity of the frame’s data, and the only portion that is different is the FCS containing the CRC value generated by a different CRC polynomial.
4. Hybrid Testing Environment

The most challenging part about setting up a testing environment for this domain-specific research area is the lack of existing real-life implementation or simulation tools that support the delivery of intentionally corrupted frames to form a side-channel. This limitation does not prevent the manufacture of custom built chipsets that do not follow the standards and would allow for alterations of the CRC polynomial for the FCS calculation and to create a side-channel communication, but it does introduce a challenge for a low-cost experimental test-bed.

In some earlier works, Najafizadeh et al.7 discovered that it was not possible to calculate the CRC polynomial in software on current existing network cards that are based on the 802.11 protocol. This was due to the fact that the FCS calculation found in the media access control (MAC) layer is in the firmware of the communication chipset. Ironically certain older discontinued network cards, like the model DWL-AG530 produced by D-Link that uses the Atheros AR5212 chipset support modifications to its MAC Layer, but they are challenging to acquire and use with today’s advanced 802.11 protocols. Software-defined radio kits also support the ability to define custom CRC polynomials but they are expensive and challenging to set up.

In a survey of simulators, Odor et al.8 looked into simulators such as NS-2, QualNet and MATLAB/Simulink for finding a suitable environment that facilitated side-channel implementation as a starting point. The authors found that both NS-2 and QualNet have the same stochastic model on handling frames that were found to be in error and determined that it was very challenging to modify the frame error handling in NS-2 due to the difficulty of accessing the contents in the frame. Recent work by us in QualNet also has confirmed the effort it takes to do this. After spending several months working with the QualNet stack it was not possible to modify the contents of the frame in QualNet without achieving inconsistent CRC values.

Najafizadeh et al.7 were able to successfully add a FCS field and implement a FCS generator function in the Sinalgo simulator, however the downfall of the Sinalgo simulator is that it does not come with a full network stack implementation. In particular, the network stack lacks the 802.11 CSMA/CA MAC protocol.

In order to encompass all the real-life elements of wireless networking into the MATLAB/Simulink, a new hybrid approach had to be taken. This hybrid approach enlists the help of the AirPcap Network adapter9 and Wireshark12 to capture real network data and frames as shown in Fig. 1. As it can be seen from this figure the testing environment is comprised of the “Wireshark & Java” and the “MATLAB/Simulink” main sections. The MATLAB/Simulink section is split further into three subsections identified by letters B, C, and D in the same figure. Section A makes up the real portion of the testing environment. It is responsible for the capturing of real wireless frames using Wireshark and performs pre-processing using a custom Java application developed to format the data for input into the MATLAB/Simulink section. Section B is the starting point in the MATLAB/Simulink simulation which is responsible for simulating the side-channel communication. There are only two CRC polynomials used here in the “General CRC
Generator”, one is termed as “Default” and the other as “Koopman”. The Default CRC polynomial will be used to tag normal communication while Koopman will be used to tag the side-channel communication frames. The CRC generator function is designed to take a frame, calculate the CRC value based on the specified polynomial and append it to the end of the frame.

Following Fig. 1 there are two types of traffic captured in order to simulate side-channel and normal traffic. Commencing at Point 1, if the frame is identified as FTP then the CRC is calculated using the Koopman CRC polynomial to simulate side-channel traffic. If the frame is identified as HTTP then the CRC is calculated using the default CRC polynomial simulating normal traffic. At Point 2 there is a second CRC generator that uses the Default polynomial to generate the uncorrupted version of the same frames for comparison later in Section D where the Hamming distance is calculated. Section C computes the lossy additive white Gaussian noise (AWGN) channel condition and implements the errors in both normal and side-channel frames though bit flipping. At Point 3 both HTTP and FTP frames are given a chance to be in error based on the probability input to the “Frame Error Probability Decider” function. At this same point, a percentage of FTP frames can be specified to be simulated as side-channel frames in the simulation. That way the ratio of side-channel frames to normal frames can be varied. At the decision gate at Point 4, if the frame is determined not be in error then it is sent to the next Section D of the simulation. Note that when the frame is determined to be in error it is passed through the “AWGN” channel for corruption and bit flipping based on the dynamic SNR values of the actual real data captured for each corresponding frame. Section D has two responsibilities, the first one is to compare the two different frames received from Point 2 and the frames that are passed through the simulation from Point 1 by calculating the Hamming distance at Point 6. The second responsibility of Section D is that after the Hamming distance is calculated, the CRC value of the frame originating from Point 1 and its resulting Hamming distance is sent for output at Point 7.

5. Validating the Hamming Distance Metric

In order to confirm our theory that the CRC Hamming distance value of a naturally corrupted frame is significantly lower compared to an intentionally corrupted frame using a different CRC polynomial, real data is passed through the simulator and the corresponding CRC values are generated for each frame using both the default and the Koopman CRC32 polynomial. Finally, the total mean Hamming distance is determined for both normal and side-channel frames, and then fed to a Perceptron Learning and Pocket Algorithm (PLPA) classifier.

We used the same data described in 2. Here we reproduce the description of the data collection for convenience. The setup consisted of 5 nodes communicating and one agent node in promiscuous mode responsible for capturing data equipped with the AirPcap adapter and Wireshark. Each node in the setup was given a static IP that started from 192.168.1.2 to 192.168.1.6. For the purpose of referencing, each node is identified by the far right least significant digit in its IP address.

Referring to the experimental setup depicted in Fig. 2, Node 2 acts as a central server that runs both the HTTP and FTP server applications. The HTTP stream rate is approximately 23.4kB/s while FTP transfer rate is set to 16kB/s. The HTTP source streams a 22.5MB mp3 music file while the FTP source transfers a 5MB zip file for all three varied rates. Node 3 acts as a malicious node that participates in the FTP transfer. As it was described in the previous test environment section, the FTP traffic is simulated as the side-channel while the HTTP is simulated as the normal traffic. Along with the side-channel FTP traffic, nodes 2 and 3 also participate in normal traffic by streaming HTTP traffic. Nodes 4, 5, and 6 only participate by streaming normal HTTP traffic.

The Perceptron Learning Algorithm (PLA)10 is a supervised learning algorithm that uses hyperplanes to separate the different clusters of data points. The main idea is that it uses a simple line to separate the different clusters of data points; each line that separates correctly the two different cluster is called a perceptron. The equation of the line can be written as \( w_1 x + w_2 y + w_3 b = 0 \) where \( b \) is equal to 1 and coefficients that make up the weight vector are \( w_1, w_2, w_3 \). PLA uses 0 and 1 to identify different classes or clusters. The learning phase starts by drawing a line with a random slope and intercept which are specified by randomly picking the weights. The next step is to evaluate the position of the line from the data points and how well the line separated the classes. Based on this PLA will readjust its weight vector and redraw the line. It will repeat these weight adjustments until a perceptron is found (if the data points were linearly separable). The Pocket Algorithm (PA)10 is incorporated into the PLA to handle non-linearly separable data by changing two things. Firstly, PA requires that a predetermined maximum iteration value must be set. Secondly, PA
will keep track of the last weight vector values obtained after each iteration and compare them with the next. Based on the calculation for that iteration, the best weight vector is kept and the others are discarded.

The features fed into the PLPA algorithm are given by Hamming distance counts. The experiment runs as follows, at time 0 seconds all 4 nodes (3, 4, 5, and 6) start the HTTP music streaming from Node 2. The agent node starts capturing traffic using Wireshark. When the time hits approximately 120 seconds the FTP transfer is started from Node 3. At time approximately 440 seconds (approx. 7 minutes) the FTP transfer is complete. The streaming continues on all 4 nodes until the full 480 seconds (8 minutes) duration is completed, at which time the agent node’s Wireshark capture is stopped and saved for preprocessing to be incorporated in the simulator as described in Section 4. The data flow through the simulation as described earlier with the FE% varied from 0%-100% with 5% step increases. This yields a total of 21 different results. Once the simulation is finished the summary of the resulting frame counts are provided along with the Hamming distance output results. For each of the 21 results the Hamming distance values and both the naturally corrupted and intentionally corrupted (total counts listed below) are consolidated into one data file output. This output data is now used for input into the PLPA algorithm for training (see Fig. 3). The testing data was obtained from running the same 5-node scenario with the same parameters except that it was ran only for approximately 5 minutes 38 seconds and the file sent over FTP was started approximately after 1 second to ensure that the whole file was transmitted. The total counts for the testing data are also listed below. Both the training and the testing data were formatted to form a scatter plot where each Hamming distance was set as both x and y coordinates. Fig. 4 depicts the results of the PLPA classification. Both figures 3 and 4 depict the Hamming distance values classified into the two classes 0 and 1. Class 0 represents the Hamming distance values obtained from non-side channel frames and Class 1 are the values from the side-channel frames. The perceptron found during training, shown in Fig. 3, was then used on the testing data as depicted in Fig. 4.

The resulting frame counts from the simulation used for training and testing were as depicted in Table 1.

6. Discussion

The data depicted in Fig. 4 showed that PLPA was able to correctly classify side channel from non-side channel frames approximately 99% of the time using the Hamming distance metric. However, it was only able to correctly
classify side-channel frames approximately 28% of the time. The lower 28% classification rate of side-channel frames could be attributed to a higher amount of naturally corrupted frames versus the intentionally corrupted frames which was 20,476 and 210 respectively. This means that this lower 28% classification rate could be attributed to the higher volume of errors and may yield a better result if a windowing analysis technique was applied in sampling only a small portion of frames suspected of being side-channel. This is left for future work that would explore the best window size for an effective analysis. The resulting threshold obtained from the PLPA algorithm is approximately 18 which is high compared to the F-Score approach results in \(^2\) which yielded a Hamming distance threshold of 10-12. From Fig. 4, it can be seen that a perceptron line intercepting a Hamming distance threshold of around 12 would be a better threshold to reduce the number of misclassified side-channel frames as non side-channel. This could indicate that this machine learning algorithm is sensitive to high volumes of error and not as accurate as the F-Score approach.

Overall the results suggest that the Hamming distance is a valid metric for distinguishing side-channel frames from non-side channel frames. And while the threshold of 10-12 obtained with F-Scores and confidence levels in \(^2\) showed promise for the scenarios tested, a universal detection mechanism with an adaptive threshold remains to be found. It would be interesting to find out the accuracy of machine learning when a windowing approach is taken.
7. Conclusion

We used a hybrid testing mechanism to generate data that allowed us to apply Perceptron Learning and the Pocket Algorithm to classify network frames into side-channel and non-side channel. The CRC32 polynomial explored for the side-channel communications was the Koopman CRC. The default CRC polynomial was designated as the normal communication as per the 802.11 standard. The CRC Hamming distance metric was introduced before as a detection measure for side-channel communication and we validated it further as a promising metric due to its consistency across different side-channel throughput and frame error rates.

The results show that the use of Hamming distance and machine learning as a metric for detection of side-channel communication is very promising. These results could be used as the basis for future work in mobile scenarios. This would entail further exploration in mobility patterns and parameters that could change the effectiveness of the Hamming distance metric in detection of side-channel communication. Another future work is to test the scalability of the Hamming distance metric in scenarios with more nodes present. This might call for the modification of an existing simulator to facilitate side-channel communication or by custom building actual network cards that do not conform to the existing 802.11 standards.

Acknowledgements

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and Defence Research and Development Canada (DRDC).

References