Journal of Computing and Information Technology - CIT 23, 2015, 4, 283–293
doi:10.2498/cit.1002530

Detecting and Locating Man-in-the-Middle Attacks in Fixed Wireless Networks

Ziqian Dong, Randolph Espejo, Yu Wan and Wenjie Zhuang

School of Engineering and Computing Sciences, New York Institute of Technology, New York, USA

We propose a novel method to detect and locate a Man-in-the-Middle attack in a fixed wireless network by analyzing round-trip time and measured received signal strength from fixed access points. The proposed method was implemented as a client-side application that establishes a baseline for measured round trip time (RTTs) and received signal strength (RSS) under no-threat scenarios and applies statistical measures on the measured RTT and RSS to detect and locate Man-in-the-Middle attacks.

We show empirically that the presence of a Man-in-the-Middle attack incurs a significantly longer delay and larger standard deviation in measured RTT compared to that measured without a Man-in-the-Middle attack.

We evaluated three machine learning algorithms on the measured RSS dataset to estimate the location of a Man-in-the-Middle attacker.

Experimental results show that the proposed method can effectively detect and locate a Man-in-the-Middle attack and achieves a mean location estimation error of 0.8 meters in an indoor densely populated metropolitan environment.

Keywords: Man-in-the-Middle, Wi-Fi, fixed wireless network, location estimation, timing analysis, machine learning

1. Introduction

The rapid adoption of mobile devices and wireless networks has dramatically increased the demand and deployment of Wi-Fi access points across different environments, both indoors and outdoors. These Wi-Fi access points form fixed wireless networks, where clients and servers on the network are connected wirelessly through access points in fixed locations [20]. Since fixed wireless networks can be easily deployed with inexpensive commercial hardware, they offer an advantage over wired networks and are widely adopted in industrial control networks, wireless mesh networks, and sensor networks.

Like any other wireless network, fixed wireless network faces security challenges and can be turned into a hostile environment when a Man-in-the-Middle (MITM) attacker is present within the range of the network. A MITM attack is considered an active eavesdropping attack, often carried out on the media access control (MAC) layer through address resolution protocol (ARP) cache poisoning [17]. The purpose of a MITM attack is to intercept a connection between two legitimate hosts in the network by spoofing the MAC addresses of the legitimate nodes and pretending to be the intended sender or receiver of traffic between two legitimate nodes. Once a connection between the legitimate nodes is intercepted, the attacker can carry out different kinds of attacks such as capturing traffic between the two nodes, rerouting traffic to unintended hosts, injecting malicious code and data to the receiver, or advertising false services to neighboring wireless stations [12].

A MITM attack can be easily carried out using publicly available tools with commodity hardware [19]. The easy access of hacking tools and low cost of attack deployment makes it easy for attackers to target the popular public and enterprise Wi-Fi networks. Detection and defense of such attacks are critical for these fixed wire-

The authors have equal contribution to the paper. Ziqian Dong is the corresponding author. E-mail: ziqian.dong@nyit.edu.

less networks to ensure data integrity and secure connection.

A known quality of a MITM attack is that it introduces delay when a connection is intercepted or hijacked, as the attacker must buffer and replay a signal transmission to the intended receiver. In a fixed wireless network where the access points do not change location, an attacker will provide different signal strength than the node it is masquerading. This is because the attacker may have different computing hardware from the victims', and they may be physically closer or farther from the intended receiver.

In this paper, we demonstrated a testbed setup to conduct a MITM attack through commodity hardware in a fixed network in a densely populated metropolitan area, which poses unique challenges due to the large number of sources of interference. We showed the features of the network measurement data in terms of round-trip time (RTT) and received signal strength (RSS) with legitimate traffic between a client and a server nodes, without a MITM attacker and under the MITM attack. We propose a method of detecting and locating a MITM attack occurring between two nodes in a fixed wireless network by comparing the difference in measured RTT and Wi-Fi received signal strength indicator (RSSI) values with the mean and standard deviations of the RTTs and RSSIs, respectively. We apply three machine learning algorithms: Gaussian Naive Bayes, K Nearest Neighbors, and Support Vector Machine, on the RSS dataset to estimate the location of an attacker and compare their performance on prediction accuracy and time complexity in fitting and predicting phases. We also expose the factors that may impact the location prediction accuracy in fixed wireless networks in densely populated metropolitan areas.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 introduces our experimental testbed setup and our proposed method that uses round-trip time delay measurement and received signal strength to detect and locate MITM attacks. Section 4 presents our experimental results and discussion. Section 5 presents our conclusions.

2. Related Work

It is possible to detect MITM attacks by observing abnormal variation of network measurements with its empirical data such as delay and signal strength [3, 14, 11, 6, 2, 7, 23, 1, 18, 8]. An abnormal network behavior (longer delay and variances in received signal strength) may be caused by disturbances in the physical layer such as collision of packets due to increased transmission requests over the wireless links or due to an intercepted connection. Since the MITM attacks incur time delay for the attacker to spoof the address of the intended sender and receiver to the other party [6, 2], it is possible to detect a MITM attack by carefully sampling and measuring the RTTs while traffic is transmitted between a sender and a receiver.

A clock synchronization protocol was proposed that uses affine clocks to synchronize data transmission on any channel and proves the possibility of detecting a MITM attack by identifying the attacker-delayed packets [3]. This protocol securely synchronizes clocks over any link by passing timestamped messages, which ensures that any connection can only introduce constant delay. This protocol also enables that any attacker imposing a non-constant delay can be detected by the legitimate receiving nodes.

A modification of media access control (MAC) layer protocol is presented to detect the MITM attack, which is based on the fact that a successful attack must ensure acknowledgement (ACK) is received within the ACK Timeout period [8]. This detection strategy uses the fact that sender and receiver secretly agree that certain frames must not be acknowledged on their first transmission. This way the MITM attacker will be exposed when the frame was sent and acknowledged by the attacker while the legitimate nodes remain silent.

This method first adds a test to detect unauthentic acknowledgements in the send procedure, then the receive procedure is modified by adding a test to the corresponding frame and suppresses the initial ACK, and finally an extra procedure to implement the suppress-ACK function that coordinates the sender and the receiver to agree on the frames whose initial ACK should be suppressed. Even though this method introduced a very high detection rate, the implementation of

modified MAC protocol is difficult. Especially in some equipment, the MAC protocol cannot be re-programed because of the implementation in hardware or interface firmware.

Physical-layer information, such as RSS, is hard to forge arbitrarily and can be used to detect MITM attack. In a fixed wireless network, the wireless access points often do not change location and the users are usually stationary when they use the service. The RSS of the Wi-Fi access points sensed at the intended receivers would be different from that of an unintended transmitter located at a different location. An attacker will have an unexpected WiFi RSS different than the node it is masquerading. This is because the attacker has different hardware, and may be physically located closer to or farther from the receiver than the node measuring the expected signal strength.

A method was proposed to detect MITM attacks using RSSI values measured at the receiver node [11]. The effect of changing a few parameters over the RSSI values of the data packets received at the receiver node in order to find that the variation of RSSI values could identify the location changing of the sender nodes. It would be interesting to see if these results can be reproduced using different computing platforms and wireless hardware.

Another approach builds RSS profiles and detects spoofing based on Gaussian mixture models (GMM) [23]. This method shows that antenna diversity is the root cause of the multimodal RSS distributions. The difference in mean RSSI by using two antennas for either transmitting or receiving can be high enough to impact the detection accuracy of existing algorithms and the multi-modal RSS patterns can be defined in GMM. However, the driver for Wi-Fi device could reduce the signal to noise ratio and further reduce the detection accuracy.

In this paper, we propose a simple detection and localization framework to counter MITM attack in fixed wireless network in densely populated metropolitan areas where interference and multipath may have higher impact in the physical layer properties.

3. Proposed MITM Attack Detection Method

In this section, we present the attacker model and the proposed method to detect and locate a MITM attack occurring between two nodes in a fixed wireless network using a client-side application that uses the measured RTT and Wi-Fi RSS. We deployed a testbed to simulate a MITM attack through ARP cache poisoning using commodity hardware in a suburban area. We also evaluated the accuracy of locating the node using RSS dataset collected in a densely populated metropolitan building through three machine learning techniques.

3.1. MITM Attack through ARP Cache Poisoning

MITM attacks are often carried out by an attacker through ARP cache poisoning which fakes the network identity of the attacker for the other nodes in the network to believe it to be a designated server or a network node. Figure 1 shows the timing diagram of an ARP cache poisoning attack. Nodes *A* and *B* are the legitimate nodes in the network. The MITM attacker broadcasts its ARP requests to all nodes in the network to learn the MAC addresses of the server and the client. Once it has the MAC addresses of Nodes *A* and *B*, it sends an ARP reply to the client, *B* to make Node *B* associate its MAC address to the server, Node *A*. Since ARP is trusting, it will accept the ARP reply and believe the MITM attacker is the server in the

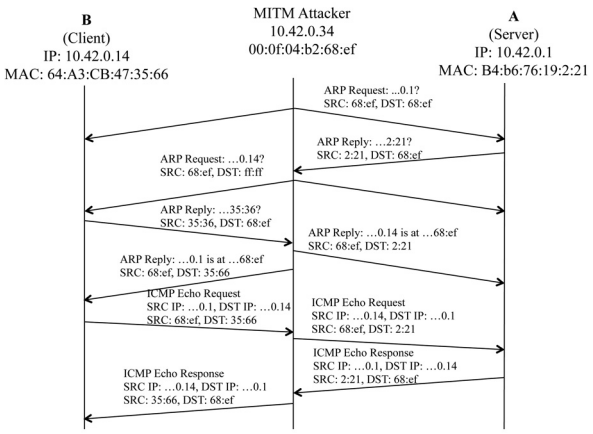

Figure 1. Timing diagram for case study of ARP cache poisoning attack.

network. Hence, the name ARP cache poisoning. So when Node *A* sends packets to Node *B*, all the packets are sent to the MITM attacker and the MITM attacker can simply eavesdrop what is being communicated or alter the message and forward it to Node *A*.

3.2. Attack Model

In this study, the MITM attack involves an attacker, a legitimate access point (AP) whose MAC address is spoofed by the attacker, and a victim who is deceived to believe the attacker is the intended AP. We assume both the attacker and the victim use off-the-shelf devices that use hardware and firmware that support standard 802.11 protocols, the wireless channels between the communicating parties are half-duplex. We also assume the attacker has the capability of manipulating the fields of the 802.11 frames to change the source and destination MAC addresses, BSSID, ESSID, sequence number, etc. to intercept an intended connection between the victim and the legitimate AP, however, with no access to change the physical layer information such as timestamp and signal strength. We do not make any assumption of the type of antennas used in these devices. A detailed attack procedure is presented in Subsection 3.3.

3.3. Testbed Setup

3.3.1. Testbed Environment

We created our testbed in a residential room in a suburban environment where Wi-Fi traffic is moderate for RTT measurement approach to detect MITM attack and in a high-rise building in a densely populated metropolitan environment for estimation of location of the network node. For detection of MITM by the RTT measurement approach, we carried out the experiments in a confined suburban testbed to perform MITM attacks. Figure 2 depicts the test environment. The dimension of the test environment was 5 m$\times$4 m. The test environment was divided into grids using a cartesian coordinate system with each point 1 meter apart from its neighboring points. The client node was placed at $(0, 0)$ and the server node was placed at $(5,3)$. The client and the server were 6.4 meters apart.

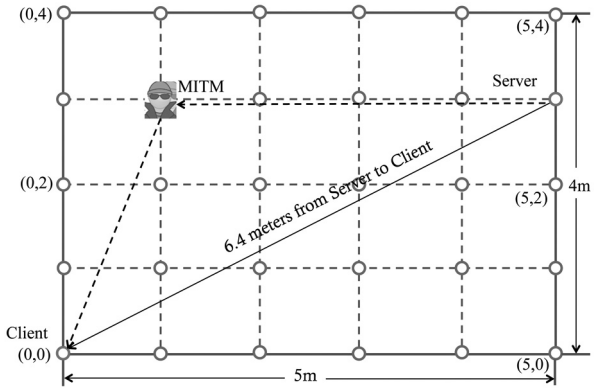

Figure 2. Floor plan of the experiment setup.

The client and server nodes were two laptops running Ubuntu 13.10. The applications were implemented in Python. The client and server nodes had external Alfa AWUS 036H Network Interface Card utilizing the RTL8187 chipset that supports 802.11 b/g. This adapter was chosen for its flexibility as it supports all Wi-Fi communication modes. Both the client and server utilized a 9Bi antenna for a stable connection between them.

For a baseline reading, the client application took RTT and RSSI readings of the server with 1000 iterations. We use the expected values of RTT and RSSI measured without the presence of an attacker as baselines for our experiments.

3.3.2. MITM Attack Deployment

The MITM attacker was simulated using a laptop running Ubuntu Linux 13.10. We used Ettercap [5] and Arpspoof [4] to carry out MITM attacks in our experiments. Ettercap is a comprehensive suite for MITM attacks. It provides more functions in tuning the parameters for an ARP cache poisoning attack. The user can calibrate the parameters to do an ARP cache poisoning attack between the sender and the receiver node. It is also imperative that the attacker was running as a Half-Duplex attacker. This will introduce the most delay when relaying traffic between the sender and receiver.

Another tool we used to deploy the MITM attack is Arpspoof, which is available in the dsniff package [4]. Arpspoof is a lightweight MITM attack tool that can be initiated from the Linux command line environment. Arpspoof works

by sending out ARP requests to the client and server machines, in turn learning their MAC addresses when it receives their ARP replies. It then sends out ARP replies to the client and server, rewriting their ARP tables so they believe the attacker running the Arpspoof is the intended receiver. The server believes the attacker is the client, and the client will believe the attacker is the server. Any traffic between the client and server will be relayed through the MITM attacker. In our experiment, the ARP replies were sent every 2 seconds, ensuring the client and server ARP tables are constantly rewritten.

3.4. Proposed Man-in-the-Middle Detection Method

The proposed MITM detection method utilizes and detects the delay variance between a normal transmission and an intercepted transmission between two nodes. Figure 3 shows the timing diagram of packet transmission between two network nodes A and B when the packets are intercepted and relayed by a MITM attacker. The RTT of the transmission between nodes A and B increased by $2 \times mBA$, where mBA was the processing time at the MITM machine for packet buffering and forwarding.

To measure RTT from the client to the server, we developed a client application using Python. The server host IP address, port number, and timeout are defined in the initial set up. A socket object is created and connected with the server application. When a successful connection is established, a message is sent to the server. The client then receives the acknowledgement message back from the server to confirm the receipt of the packet. The RTT is measured by taking the difference between the message send time and message receive time.

We modified the client application based on *Ping* to get statistical measurements. The client application sends ICMP packets to the server, awaits ICMP echos from the server, and calculates the RTT. We take the measurements of the average, minimum, maximum, and standard deviation of RTT and the percentage of lost packets.

The application records the RSSI from the server as well as the average, minimum, maximum, and standard deviation of the RSS in dBm. In the case of a single Wi-Fi access point, the reference points may be limited to get accurate location estimations. In a densely populated metropolitan environment we use IT infrastructure as our testbed to estimate the location where multiple Wi-Fi access points can be heard by the receiver at the same time. We conducted our experiments on the 6th floor of the Edward Guiliano building at New York Institute of Technology's Manhattan Campus. The 6th floor contains three Wi-Fi Access Points (AP) spaced evenly throughout the inverted "U" shaped hallway. As seen in Figure 4, we selected 26 reference points, labeled A through Z, and marked the AP locations. The distance from each reference point, A through Z, is approximately 1.5 meters.

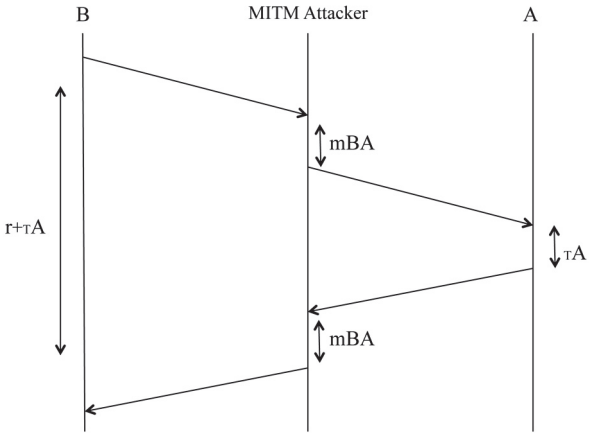

Figure 3. Timing diagram of an ARP cache poisoning attack.

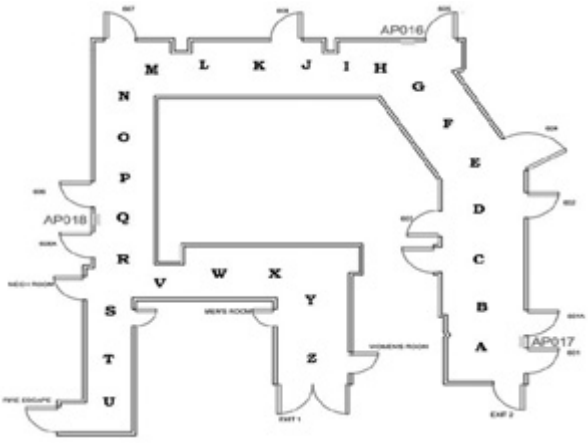

Figure 4. Floor plan of the 6th floor of the Edward Guiliano Global Center Building at NYIT.

Wi-Fi signals are electromagnetic waves that propagate through transmission media. The signal loses its strength as it's transmitted over a distance. Free-space path loss is defined as the loss in signal strength in free space (air). The indoor path loss equation estimates the path loss inside a room or closed-in area inside a building that is surrounded by walls [21]. Eqn. 1 is a standard indoor path loss function, which requires a number of variables in order to determine the path loss [21].

$$L = 20\log_{10}(f) + N\log_{10}(d) + L_f(n) - 28 \quad (1)$$

Here, L denotes the indoor path loss and is measured in decibels (dB). N denotes the distance power loss coefficient (dB) which can be obtained from the table given below:

Residential	Office	Commercial
28	30	22

Our experiment environment is matched to the "Office" type from the table as this building has neither a "Residential" or "Commercial" layout. f represents frequency of the transmitted signal. The Wi-Fi signals we are measuring in our experiments use 802.11n standard and have a frequency of 2.4 GHz. d represents the distance between nodes in meters, and is the distance between our reference points and access points. L_f and n are parts of the floor penetration factor where n is the number of floors. In this experiment we are only using one floor, therefore $L_f(n)$ is 0. Eqn. 1 is reduced to:

$$L = 39.6 + 30\log_{10}(d) \quad (2)$$

This equation is used to solve for distance given a signal strength.

$$d = 10^{\frac{L-39.6}{30}} \quad (3)$$

The indoor path loss equation is a measurement of signal loss and not the signal strength. In order to find signal strength, we must use another equation called the Link Budget to calculate received signal strength at the receiver (P_{RX}) [9]. This accounts for transmit power (P_{RX}), transmit antenna gain (G_{TX}), receive antenna gain (G_{RX}), along with path loss.

$$P_{RX} = P_{TX} + G_{TX} + G_{RX} - L \quad (4)$$

We assume that the power of our transmitter (dBm), power of our receiver (dBm), and gain of our transmitter and receiver (dB) are constant with respect to our hardware. This means that only path loss relies on the distance. Therefore, we make the assumption that path loss is equal to received signal strength in our experiments. We proceed to measure the distance from each access point to the reference point and then use the measured Wi-Fi signal strength at each reference point to determine the distance by using the indoor path loss equation.

4. Results and Discussion

4.1. MITM Detection through RTT Measurement

In the MITM detection experiment, we measured the RTTs at every grid point in the suburban testbed. Based on 1000 measurements at each point, we observed that in a clean connection between the client and the server without MITM attacks, the average RTT was consistently below 3.4 ms, with a standard deviation of about 2 ms. Figure 5 shows the results of RTT measurements at points $(0,3)$ and $(3,2)$ with a normal connection and a connection under MITM attack.

A clean connection experienced zero ICMP packet loss. Based on the hardware of our server machine, the client reads the RSSI of the server consistently at -58 dBm, with a standard deviation of 2.62 dBm. The average RTT under MITM attack for all points was found to be 2.4 times the baseline average of a clean connection.

With the presence of a MITM attacker halfway between the client and the server, the RTT could jump to 51 ms, with an observed 25 percent packet loss. Because the MITM attacker was running on a low-power netbook, the observed RSS had an average of -45 dBm with a standard deviation of 26 dBm.

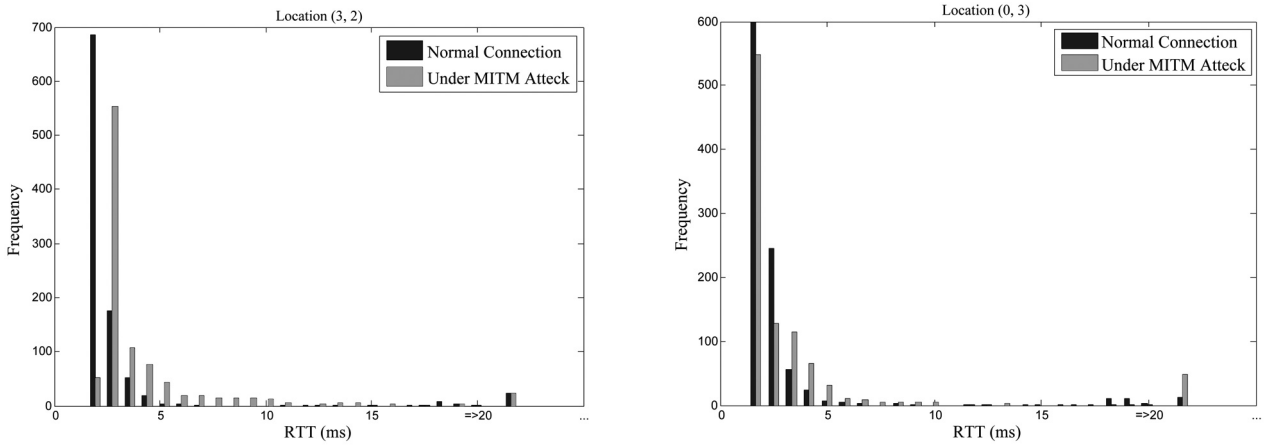

Figure 5. Histogram of measured RTT at locations $(3,2)$ under normal connection and under MITM attack.

4.2. Locating a MITM Attacker through RSS Measurement

To estimate the location of a MITM attacker, we used the dataset collected at the NYIT 6th floor testbed with recorded RSS data measured on a laptop at measurement points A through Z. The dataset included approximately $500-1000$ RSS records from each access point measured at each location during July 2013 [10]. We evaluated three machine learning algorithms to predict the location of the node: Gaussian Naive Bayes (GNB), K Nearest Neighbors (KNN), and Support Vector Machines (SVM). We used half of the dataset as training set and the other half of the dataset as test set. We used the scikit-learn (sklearn) python machine learning library in our experiments [22].

$$P(y|x_1,\cdots x_n,) = \frac{P(x_1,\cdots x_n|y)P(y)}{P(x_1,\cdots x_n)} \qquad (5)$$

Naive Bayes theorem is based on the Bayes' theorem as shown in (5) and gets its name from the strong assumption that each feature is independent of any other feature. This assumption is often false in real world applications. However, despite this assumption, Naive Bayes has been proven to perform well in certain cases, especially where the number of training samples is small [13]. In an effort to achieve high accuracy whilst reducing pre-deployment efforts, we select this method of computing the probabilities of the locations' given measurements. In our study, the RSS measurements are assumed to follow Gaussian distribution which follows the classification rule in (6).

$$\hat{y} = \text{argmax}_y P(y)\prod_{i=1}^{n} P(x_i|y) \qquad (6)$$

K Nearest Neighbor algorithm is a simple classification algorithm that finds the K nearest neighbors of an element and categorizes the element to the class most of its neighbors belong to. In our experiment, we use $K = 5$ and Minkowski distance in calculating the nearest neighbors to classify an RSS measurement. Minkowski distance is defined as $\left(\sum_{i=1}^{k}(|x_i - y_i|^q)\right)^{\frac{1}{q}}$, where q is the power parameter. When $q = 1$, it is equivalent to Manhattan distance, $q = 2$, it is equivalent to Euclidean distance. Here, we use $q = 2$ in our experiments.

Several algorithms used to detect nearest neighbors are provided by sklearn, including brute force, ball tree and KD tree. With the size of dataset N, the time complexity of the brute force query is $O(N)$ and tree-based query is $O(logN)$. However, for tree-based nearest neighbor algorithm, the complexity of constructing a tree in the fitting phase is $O(NlogN)$. For smaller samples, with size N less than 30, brute force algorithm is faster due to its small overhead. When sample size is larger than 30, tree-based algorithm has an advantage because of its logarithmic growth in the query phase. In our experiment, we use the default option where sklearn will automatically choose the most efficient nearest neighbors based on the sample size.

Sklearn provides several SVM classifiers, including SVC, NuSVC and LinearSVC. Here, we choose support vector classification (SVC), which is a C-Support Vector Classification (C-SVC) implementation based on libsvm [16]. SVC in sklearn supports multi-class classification using the "one-against-one" approach [15]. In our experiment, we use radial basis function (rbf) kernel with penalty parameter C of C-SVC set to 1.0.

Figure 6 shows the percentage of accurate predictions for each measurement location using the GNB, KNN, and SVM machine learning algorithms. The results show that for majority of the locations, the prediction accuracy is high (over 80% of the prediction is accurate). Some locations have very poor prediction accuracy, e.g. locations *S*, *V*, and *W*. The reason for this poor estimation is the fact that these points are located next to the mechanical rooms where magnetic interference is very high and the RSS measurements could be skewed by the interference.

In this paper, we use Euclidean distance to calculate the estimation error. Figure 7 shows the cumulative distribution of the estimation error of the three machine learning algorithms on the RSS dataset. The mean estimation errors for the GNB, KNN, and SVM are 0.9034 meters, 0.8325 meters, and 1.3242 meters, respectively. The standard deviation for the GNB, KNN, and SVM are 2.1603 meters, 2.1158 meters, and 3.117 meters, respectively. This shows that KNN and GNB have very close estimation errors and they both outperform SVM for this dataset. KNN and GNB have 90% of the estimation errors below 2 meters, while SVM can only achieve estimation errors below 4.5 meters at the 90 percentile.

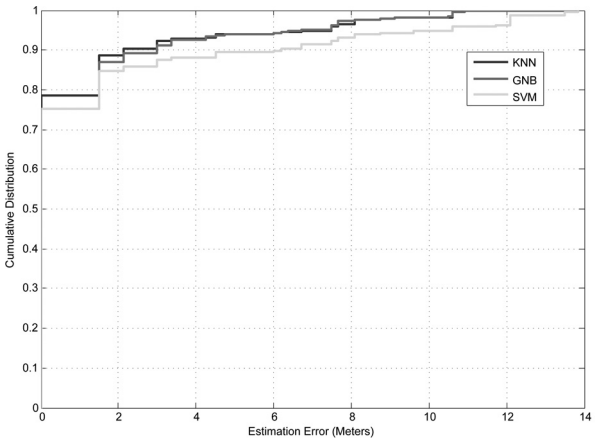

Figure 7. Location prediction accuracy comparison of Gaussian Naive Bayes, K Nearest Neighbors, and Support Vector Machines.

Table 1 compares the prediction accuracy and complexity of the three machine learning algorithms for location estimation. KNN and GNB have better prediction accuracy than SVM, as shown in Figure 7. The two phases of machine learning include fitting (training) phase and prediction phase. For our dataset, GNB and KNN have similar performance in time complexity. Since tree-based query is adopted for KNN, the fitting phase includes tree building which has complexity of $O(KNlogN)$. This could make it slower than GNB in the fitting phase. However, fitting is usually done offline and does not impact the prediction speed. SVM is more complex to implement (the complexity of the fitting phase could be $O(n^2)$, and the complexity of the prediction phase is dependent on the number of support vectors generated) and therefore,

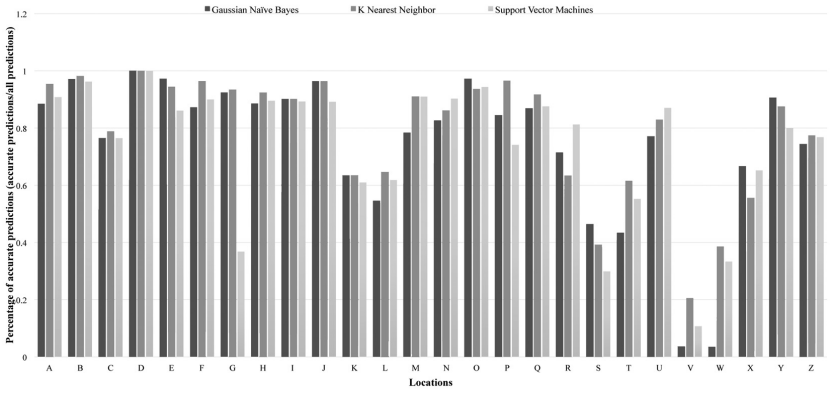

Figure 6. Location prediction accuracy comparison of Gaussian Naive Bayes, K Nearest Neighbors, and Support Vector Machines.

Algorithm	Prediction accuracy	Complexity
GNB	High	Low
KNN	High	Low
SVM	Medium	Medium

Table 1. Performance comparison of the applied machine learning algorithms.

has the highest complexity among the three algorithms. For our dataset, SVM underperforms KNN and GNB in both prediction accuracy and complexity.

To preserve energy and computing resources, the proposed method implements the client application that measures the average RTT through modified *ping* program and detects if it exceeds a threshold (in our experiment, the threshold is set to 2.4 times the baseline RTT average). If an abnormal RTT is detected, the system will call the prediction scheme using KNN or GNB on the gathered RSS values from the clients and servers, to further confirm the presence and estimate the location of the MITM attacker. The key indicator in our results was the increased RTT and variation in RSS measurements compared to the baseline values.

5. Conclusions

We propose a novel method to identify a Man-in-the-Middle attack between two nodes in a fixed wireless network by analyzing the measured round-trip time and received signal strength from Wi-Fi access points. To evaluate the effectiveness of the proposed method, we use a fixed wireless network testbed comprising a client, a server, and an attacker machine in a room designated as the test environment. The proposed method evaluates round trip time and received signal strength at the client, and calculates the mean and the standard deviations of the measured RTTs and RSS. These values are used as baselines to detect Man-in-the-Middle attacks. We show that the presence of a Man-in-the-Middle attack incurs significantly longer delay and larger standard deviation in measured RTT compared to that measured without a Man-in-the-Middle attack. The measured RSS also provides indication of the difference in distance where the designated server and the Man-in-the-Middle attacker is located. To locate the Man-in-the-Middle attacker, we applied and compared K Nearest Neighbors, Gaussian Naive Bayes, and Support Vector Machine machine learning algorithms on the collected RSS dataset from a densely populated metropolitan building. Results show that K Nearest Neighbors and Gaussian Naive Bayes provide less than 2-meter estimation error 90% of the times with a mean estimation error of 0.8 and 0.9 meters, respectively. The proposed method integrates the delay and signal strength variables to detect and locate a Man-in-the-Middle attacker with high estimation accuracy and low complexity.

Acknowledgment

This research is funded by the National Science Foundation Research Experience for Undergraduates Grant number CNS-1263283 and by the New York Institute of Technology.

References

[1] M. ALICHERRY, A. D. KEROMYTIS, Doublecheck: Multi-path verification against man-in-the-middle attacks. In *Computers and Communications, 2009. ISCC 2009. IEEE Symposium on*, (2009) pp. 557–563.

[2] K. BENTON, T. BROSS, Timing Analysis of SSL/TLS Man in the Middle Attacks. (2013). *arXiv preprint arXiv:1308.3559*

[3] J. T. CHIANG, J. J. HAAS, Y-C. HU, P. KUMAR, J. CHOI, Fundamental limits on secure clock synchronization and man-in-the-middle detection in fixed wireless networks. In *INFOCOM 2009, IEEE*, (2009) pp. 1962–1970.

[4] *dsniff-arpspoof*. (2014). Retrieved from: `http://www.monkey.org/ dugsong/dsniff`

[5] *ettercap*. (2014). Retrieved from: `http://ettercap.github.io/ettercap`

[6] N. M. FRERIS, P. KUMAR, Fundamental limits on synchronization of affine clocks in networks. In *Decision and Control, 2007 46th IEEE Conference on*, (2007) pp. 921–926.

[7] R. S. GILL, J. SMITH, M. H. LOOI, A. J. CLARK, Passive techniques for detecting session hijacking attacks in IEEE 802.11 wireless networks. In *Information Technology Security Conference*, (2005), University of Queensland.

[8] S. M. GLASS, V. MUTHUKKUMARASAMY, M. PORTMANN, Detecting man-in-the-middle and wormhole attacks in wireless mesh networks. In *Advanced Information Networking and Applications, 2009. AINA'09. International Conference on*, (2009) pp. 530–538.

[9] J. GRIFFIN, G. DURGIN, Complete Link Budgets for Backscatter-Radio and RFID Systems. *IEEE Antennas and Propagation Magazine*, **51** (2009).

[10] N. GUTIERREZ, C. BELMONTE, J. HANVEY, R. ESPEJO, Z. DONG, Indoor localization for mobile devices. In *Networking, Sensing and Control (ICNSC), 2014 IEEE 11th International Conference on*, (2014) pp. 173–178.

[11] S. HUSSAIN, M. S. RAHMAN, Using received signal strength indicator to detect node replacement and replication attacks in wireless sensor networks. In *SPIE Defense, Security, and Sensing*, (2009) pp. 73440G–73440G.

[12] H. HWANG, G. JUNG, K. SOHN, S. PARK, A study on MITM (Man in the Middle) vulnerability in wireless network using 802.1 X and EAP. In *Information Science and Security, 2008. ICISS. International Conference on*, (2008) pp. 164–170.

[13] A. JORDAN, On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, **14** (2002), pp. 841.

[14] A. KARNIK, A. IYER, C. ROSENBERG, Throughput-optimal configuration of fixed wireless networks. *IEEE/ACM Transactions on Networking (TON)*, **16**(5) (2008), 1161–1174.

[15] S. KNERR, L. PERSONNAZ, G. DREYFUS, Single-layer learning revisited: a stepwise procedure for building and training a neural network. In *Neurocomputing*, (1990) pp. 41–50. Springer.

[16] *libsvm*. (2014). Retrieved from: `http://www.csie.ntu.edu.tw/cjlin/papers/libsvm.pdf`

[17] C. NACHREINER, Anatomy of an ARP Poisoning Attack. *Retrieved July*, **4** (2003), pp. 2005.

[18] S. Y. NAM, D. KIM, J. KIM, Enhanced ARP: preventing ARP poisoning-based man-in-the-middle attacks. *Communications Letters, IEEE*, **14**(2), (2010) pp. 187–189.

[19] OCCUPYTHEWEB, (2014). *Hack like a pro: how to conduct a simple man-in-the-middle attack*, Retrieved from: `http://null-byte.wonderhowto.com/how-to/hack-like-pro-conduct-simple-man-middle-attack-0147291/`

[20] *Outdoor Wi-Fi Market [(Municipality Networks; Outdoor Hotspots; Private Networks); by Products (Access Points; WLAN Controllers; Wireless gateways)]: Global Advancements, Business Models, Worldwide Market Forecasts and Analysis (2013 – 2018)*, (2014). Retrieved from: `http://www.marketsandmarkets.com/Market-Reports/outdoor-wi-fi-market-945.html`

[21] C. PEREZ-VEGA, J. M. L. HIGUERA, ET AL., A simple and efficient model for indoor path-loss prediction. *Measurement Science and Technology*, **8**(10), (1997) pp. 1166.

[22] *scikit-learn, Machine Learning in Python*, (2014). Retrieved from: `http://scikit-learn.org/stable/`

[23] Y. SHENG, K. TAN, G. CHEN, D. KOTZ, A. CAMPBELL, Detecting 802.11 MAC layer spoofing using received signal strength. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, (2008).

Received: October, 2014
Revised: May, 2015
Accepted: August, 2015

Contact addresses:

Ziqian Dong
School of Engineering and Computing Sciences
New York Institute of Technology
1855 Broadway
New York
NY 10023
USA
e-mail: `ziqian.dong@nyit.edu`

Randolph Espejo
School of Engineering and Computing Sciences
New York Institute of Technology
1855 Broadway
New York
NY 10023
USA
e-mail: `respejo@nyit.edu`

Yu Wan
School of Engineering and Computing Sciences
New York Institute of Technology
1855 Broadway
New York
NY 10023
USA
e-mail: `ywan04@nyit.edu`

Wenjie Zhuang
School of Engineering and Computing Sciences
New York Institute of Technology
1855 Broadway
New York
NY 10023
USA
e-mail: `wzhuang@nyit.edu`

ZIQIAN DONG is an Assistant Professor of electrical and computer engineering at the New York Institute of Technology. She received her B.S. degree in electrical engineering from Beihang University (formerly Beijing University of Aeronautics and Astronautics), Beijing, China. She received her M.S. and Ph.D. degrees in electrical engineering from the New Jersey Institute of Technology, Newark, NJ in 2002 and 2008, respectively. She was awarded the Hashimoto Prize for the best Ph.D. dissertation in electrical engineering at NJIT. She is the recipient of 2006 and 2007 Hashimoto Fellowship for outstanding scholarship and the recipient of the New Jersey Inventors Hall of Fame Graduate Student Award for her inventions in network switches. Her research interests include architecture design and analysis of practical high-performance packet switches, network security and forensics, and wireless sensor networks. She was associated with Networking Research Laboratory at the New Jersey Institute of Technology and MySYNC Laboratory at the Stevens Institute of Technology for her postdoctoral research. She is the Principal Investigator and a faculty mentor for the Research Experience for Undergraduates (REU) Program on smartphone and wireless network security at NYIT funded by the U.S. National Science Foundation.

She is a member of the IEEE Communications Society, IEEE Women in Engineering, the American Society for Engineering Education (ASEE), ACM, and the Environmental Sensing, Networking and Decision-Making (ESND) technical committee. She has served in technical program committee of IEEE HPSR, IEEE Sarnoff, IEEE GreenCom and ChinaCom, and as a reviewer for IEEE journals, conferences and NSF panels.

RANDOLPH ESPEJO received his B.S. and M.S. degrees in computer science from the New York Institute of Technology in 2013 and 2014, respectively. Randy served in the US Army as a chemical, biological, radiological and nuclear specialist and as a human resource specialist from 2002 – 2011. He was a graduate research assistant at NYIT researching indoor localization of mobile devices. He served as a graduate mentor and the social coordinator for the NYIT Research Experience for Undergraduates program. He participated in the Niksun Cybersecurity Tournament in 2013 and was ranked fourth in the final competition.

YU WAN received her B.S. degree in electrical engineering from the Dalian Jiaotong University, China and M.S. degree in electrical and computer engineering from the New York Institute of Technology, USA. She is a graduate research assistant and has mentored undergraduate researchers at the NYIT Research Experience for Undergraduates program. Her research interests include data analysis, machine learning, specifically in dimension reduction and indoor localization tracking and mapping. She has experience in kinematics data collection, processing and analysis, motion detection and position tracking of Parkinson's disease patients for gait and posture analysis. She has experience in mobile application development and data interface design with wearable devices.

WENJIE ZHUANG received his M.S. degree in computer science from the New York Institute of Technology (NYIT). He attended a "$3+1$" international joint program and received his B.S. degree in computer science from both the Nanjing University of Posts and Telecommunications (NUPT) and the NYIT. In his senior year at NYIT, Wenjie received Provost's recognition, Dean's recognition and won the third prize in the Motorola Golden-idea Contest. Wenjie is a versatile programmer who enjoys programming in different programming languages using different tools. His research interests lie in programming languages and machine learning. Wenjie has experience in implementing simulators for research experiments, programming mobile apps, implementing, deploying and maintaining servers, developing applications for wearable devices, implementing game logic for 3D games, and collaborating with engineering students in building automated systems. In his spare time, Wenjie enjoys playing and watching basketball, playing video games and watching anime.