Using a Novel Blending Method Over Multiple Network Connections for Secure Communication

Jaime C. Acosta
U.S. Army Research Laboratory
White Sands Missile Range, NM 88002–5513
http://www.jaimeacosta.info/

John D. Medrano
U.S. Army Research Laboratory
White Sands Missile Range, NM 88002–5513

***Abstract*—In the field of computer security, covert communication is usually seen as adversarial, but from another perspective, it can be seen as a way to communicate securely by hiding data from a malicious third party, *e.g.*, an inside attacker. In this light, instead of making data unreadable using encryption, it may be possible to hide from an adversary a secure network infrastructure (consisting of several node endpoints) in network traffic.**

Current covert communication techniques, using storage and timing channels, are not suited well for this task. Storage channels typically use properties of a protocol that are ignored, such as unused header fields. In this case, once the vulnerability of the protocol is documented, an attacker may uncover the data and breach the communication. Timing channels work by purposely modifying timing mechanisms on a network such as packet arrival times. In general, timing channels are difficult to detect, but they provide low throughput. In this paper we describe a novel blending technique that is capable of using as carriers the payload fields of multiple connections including audio, video, and voice over IP (VoIP) streams.

To send covert data, the technique executes in three main phases. In the analysis phase the covert sender will analyze traffic in promiscuous mode. In the selection phase, the sender selects locations to place covert data. In order to blend with active network traffic, the sender will select connections with high data rates and a sufficient amount of randomness in the payload. Within these connections, the packets with the highest randomness (considered injection points) are duplicated and slightly modified to include the covert data. Finally, in the sending phase, the modified packets are injected into the network (still containing the original source and destination addresses). By analyzing the same traffic the covert receiver will identify the injection points and extract the covert messages. We implemented the blending covert method (BCM) tool and evaluate it using user datagram protocol (UDP) connections during two network loads. Our results show that our technique works with limited data loss and we analyze the tradeoffs between throughput and detectability.

I. Introduction

Much focus has been placed on security defense mechanisms against outside adversaries, but an insider with malicious intent can overcome these defenses with much less effort. One example of an inside attack is priviledge escalation. In a network environment with promiscuous traffic such as a wireless local area network (WLAN), an inside attacker may eavesdrop on communication to realize a network's infrastructure. Although some of these networks employ encryption, many encryption techniques allow an insider to see all traffic on the network. In this case, an insider could use this information to plan an effective denial of service attack. Similar scenarios are possible in networks with multicast traffic, hubs, or switches (*e.g.*, using a poisoning attack). One way to protect data from unintended receivers is to hide communication by communicating over covert channels within the network.

Traditionally, covert communication is defined as a way for two entities that are not intended to communicate to exchange information [1]. From this perspective, covert communication is adversarial, and is used as a means to leak information to unintended listeners. From a different perspective, covert communication can be seen as a way to protect information from unauthorized recepients. Unlike encryption, where the information is made unreadable, covert communication will allow hiding information including even the node endpoints in a network environment. This will make it difficult for a malicious insider to plan attacks.

Some have used covert communication for legitimate communication, *e.g.*, [2]–[4], but in general, these covert communication methods are tied to a single protocol and the throughput does not scale as the network gets larger. In this paper, we introduce the blending covert method (BCM), which allows nodes in a network to communicate covertly by blending in active network traffic.

In this paper, we make the following contributions:

1) We describe a method that can be used for communicating securely through multiple concurrent covert channels within a network with promiscuous traffic (*e.g.*, WLAN, multicast, hub, switch configured with promiscuous ports). The covert throughput can scale as network congestion increases.
2) We demonstrate the practicality of the method by implementing the BCM tool.
3) We evaluate the performance of the BCM tool under two loads and show that the method works with limited data loss in a hubbed network. We also evalute the tool with different parameter configurations to show the trade-offs between throughput and detectability.

The rest of the paper is organized as follows. First, we review related covert communication methods. Next, we describe the covert communication method and each phase (monitor, select, and insert/extract) in detail along with the configurable parameters. We then describe the experimental setup and report the results from the experiments. Finally, we present our conclusions and future work.

II. Related Work

There are two main types of channels that are used for covert communication [5]. The first are timing channels, which work by purposely modifying timing mechanisms on a network such as packet arrival times. In general, timing channels are difficult to detect, but they provide low throughput. The second type of channels are storage channels, which insert covert data inside header or footer fields in specific protocols, *e.g.*, [6], [7] and within payload fields of invalid messages. For example, HICCUPS [8] works by hiding data within the payload of messages with bad checksums in the datalink layer. This method works for wireless networks and requires specialized hardware that has the capability to modify data link layer checksums. In general, once these channels are documented, an adversary may know where to find these.

Recent methods use a combination of timing and storage channels. The Lost Audio PaCKets Steganography (LACK) method [9] works by having a legitimate voice over IP (VoIP) stream between nodes on a network. The sender will then hide data in purposely delayed packets. In a VoIP stream messages older than a certain period are usually ignored because each message generally carries only a few milliseconds of audio. It would not make sense for a receiver to process older messages, especially in streams with high message rates. Only a covert receiver captures and extracts the covert data within these messages. From a defensive perspective, just as a covert receiver monitors the old messages an adversary could potentially do the same and therefore breach the covert communication.

One possible solution for this problem is to blend covert communication with normal traffic. In the field of system vulnerability analysis, one method that uses blending to bypass intrusion detection systems (IDSs) that identify anomalies in payload data is the polymorphic blending attack (PBA) [10]. PBA works by monitoring traffic and identifying byte frequencies. In order to send malicious packets without triggering an anomaly detector, PBA pads the malicious data with bytes that make the entire contents match the known byte frequencies.

Applying this blending idea to covert communication requires additional considerations. First, covert communication requires a sender and a receiver, so the data must be coded in a way that a receiver can identify and extract the messages. In addition, assuming that invalid messages are not ignored, a covert communicator should modify only portions of messages that will not be noticed. Yarochkin *et al.* [11] demonstrate a method that creates connections that blend with network traffic (including protocols and services used). These connections are used to communicate covertly and they are created and removed dynamically as the network changes. The method relies on having a network structure where new connection attempts are common and failed connection attempts are blocked and ignored, such as in Internet traffic. This is not the case in tactical networks where all nodes and connections among nodes follow some specifications. Huang *et al.* [12] present another method that blends covert communication with existing traffic by hiding information in VoIP streams. In this work, the authors use the least significant bits (LSBs) in VoIP streams to hide data. The problem with this is that the LSBs are used regardless of the data that is being sent. One problem that can occur is if the LSBs are unchanging, when inserting covert data the covert communication may become obvious. An extension of this work [13], improved the covert insertion process by selecting the LSB most similar to a given covert message to improve the blending (or transparency). The latter two methods use a single protocol and use fixed fields in the payload for covert communication. The method described in this paper chooses insertion points dynamically and embeds covert data over multiple connections.

III. Blending Covert Method

The blending covert method works in three main phases. During each phase, there are several parameters that a covert communicator may set in order to balance between throughput and detectability, *e.g.*, one may risk higher detectability to achieve higher throughput in a network with some level of congestion depending on the protocols and services used.

The three phases and the parameters used are shown in Figure 1.

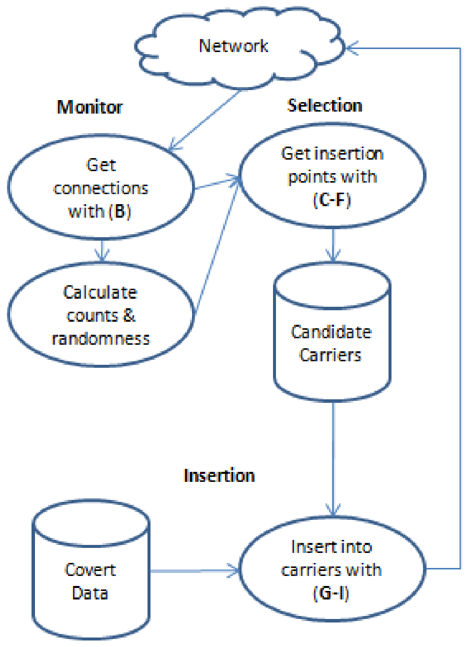
Network
Monitor
Selection
Get connections with (B)
Get insertion points with (C-F)
Calculate counts & randomness
Candidate Carriers
Insertion
Covert Data
Insert into carriers with (G-I)

Fig. 1. Blending covert method dataflow diagram. Letters in parantheses correspond with the parameters in Table I.

A. Monitoring

In the monitor phase, all network traffic is analyzed in promiscuous mode. A connection consists of the messsages sent between two nodes that share the same source and destination media access control (MAC) addresses, Internet protocol (IP) addresses, and port numbers. In addition, messages in

TABLE I
BLENDING COVERT METHOD PARAMETERS

Type	Parameter Name	Description	Value Range
Monitor	(A) Window Size	Determines how often the packet counts, byte counts, and randomness values are computed. Also defines the smallest wait time before sending a covert message for a connection. For example, if this value is set to one second, for each connection, only one covert message can be sent each second.	[100ms,5000ms]
Monitor	(B) Protocol To Use	Indicates which network protocol(s) must be used by a connection to be a candidate for covert message insertion.	[UDP,any]
Selection	(C) Rate Threshold	The minimum rate (packets per window) that a connection must send data to be a candidate for covert data insertion. A higher value will decrease detectability, but will also reduce throughput.	[1,50]
Selection	(D) Connection Randomness Threshold	This value is calculated by summing the randomness of all bytes in a connection. This value indicates the minimum randomness that a connection must exhibit to be a candidate for covert data insertion.	[1,1024]
Selection	(E) Byte Randomness Threshold	Byte randomness is calculated using the technique described in Section III-A. This value indicates the minimum randomness that a byte in a connection must exhibit to be a candidate for covert data insertion.	[0,1]
Selection	(F) Contiguous Random Bytes	Indicates the number of contiguous bytes that must satisfy the *Byte Randomness Threshold* in order for a connection to be a candidate for covert data insertion.	[*Sync Bytes* + *Checksum Bytes* + 1,1024]
Insertion/Extraction	(G) Sync Bytes	Indicates the number of bytes to use for identifying the start of a covert message within a connection's payload. A higher sync byte count will result in fewer false positives, but may also result in higher detectability and less throughput.	[0,5]
Insertion/Extraction	(H) Checksum Bytes	Indicates the number of bytes to use for calculating the message checksum used to validate covert messages. A higher value will result in fewer false positives, but less throughput.	[0,3]
Insertion/Extraction	(I) Rate To Use	Indicates the percentage of the connection's rate that will be used for covert data insertion.	[0,1]

a connection have equal protocol types and packet lengths. Therefore, for each connection, only the bytes in the payload fields change.

In this phase, connections that communicate over the protocol specified in the *Protocol to Use* parameter are stored. For each of these connections, several statistics are calculated. First, the rate is calculated by averaging the number of packets received during the last two windows of duration given by *Window Size*. Next, the randomness of the each byte in the connection and the total over the entire connection are calcuated. Randomness is calculated as follows.

Using the packets collected for a given connection, the randomness of each byte in the buffer is calculated using a histogram. Given that each byte in the buffer can take one of 256 possible values, a histogram is generated using the number of occurances of values or range of values for each byte. Figure 2 shows sample histograms for three cases. In these cases, the value ranges for the bytes are stored in eight size bins (x-axis). Each time a new packet is received, the bin corresponding to the byte value is incremented (y-axis). The leftmost histogram is for a byte that exhibits a predominate value with some occurances of surrounding values. The middle histogram shows a byte value that is mostly evenly distributed (which is most favored for covert data placement), while the rightmost graph shows a byte value that has three discrete value ranges.

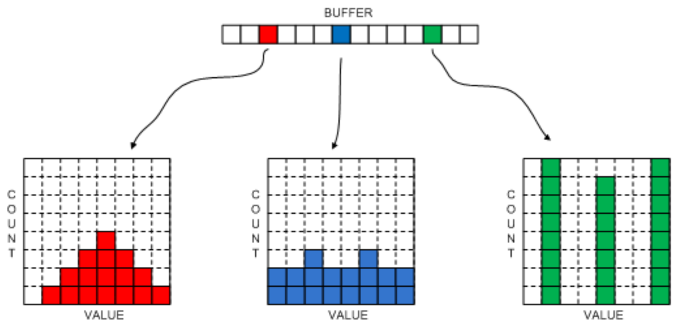

Fig. 2. Sample histograms for different randomness values

For this work, we used histograms with equal size bins with width size of 32 (8 bins) in order to produce a good estimate of the probability density function (PDF) [14]. This was verified with informal testing. A histogram with evenly distributed value counts produces a uniform PDF.

Given that the BCM runs in real time, we calculated a single value representation of randomness for each byte using a method that requires minimal resources. We used the following equation.

$$R_b = Min/Max$$

Min and Max represent the minimum and maximum count values of the histogram. If the values are well distributed then the maximum and minimum values will be close in value and the randomness will be close to one. Higher counts in discrete bins will result in decreased randomness.

For a given connection, the total randomness is given by the following.

$$\sum_{b=1}^{n} R_b$$

where n is the number of bytes in a given connection.

B. Selection

During the selection phase, locations for placing covert data are identified and stored. Based on the counts from the first phase, in order to blend with active network traffic, connections that satisfy the *Rate Threshold*, and the *Connection Randomness Threshold* are selected. Within these connections, the sequences of contiguous bytes satisfying the parameters *Contiguous Randomness Bytes* and *Byte Randomness Threshold*) are identified. Next the start and end position of the longest sequence, along with a copy of the packet is stored as a candidate carrier.

C. Insertion and Extraction

The insertion phase runs on a separate thread that continually pulls covert data from a buffer. This buffer is filled by reading the contents of a file. If candidate carriers exist, then covert data are distributed among the candidate carriers into the insertion points. Along with the covert data, sync bytes and checksum bytes are also included in the sent data. The number of sync and checksum bytes used are taken from the *Sync Bytes* and *Checksum Bytes* parameters. In the case that candidate carriers do not exist in the current window, then the data are held in the buffer. When a message is sent, the candidate carriers that are used will not be considered again for a specific number of windows, given by the *Rate to Use* parameter.

One important note is that due to the header fields exhibiting zero randomness (because they never change for a connection), these fields are not considered as insertion points. Therefore, the covert data are sent in packets without modification to the source, destination and protocol fields.

During the extraction phase, a covert receiver extracts covert data by monitoring the network traffic and looking for sync bytes and valid checksums. Assuming a covert sender uses insertion points that exhibit uniform PDFs, then an estimate of the chance of a false positive is given by the following.

$$P_{FalsePositive} \approx (1/256)^n$$

One way to reduce the number of packets needed to be analyzed by a receiver is to match the parameters used by the sender, however, this may also result in more packet loss.

IV. Evaluation

In order to test the performance of the BCM, we implemented a tool and tested it in real network environment. The tool is written in the Java programming language and uses the open source JNetPcap library for network operations.

TABLE II
NETWORK TRAFFIC LOADS DURING THE TWO EXPERIMENTS

	Experiment 1	Experiment 2
Overt Nodes	6	12
Packets/sec	80–100	5200–5500
Bytes/sec	95,000–115,000	2,700,000–3,500,000
Active Connections	15–20	40–50

A. Experimental Setup

We evaluted our covert method by measuring to throughput, reliability, and detectability under different network loads. We evaluate reliability by measuring packet loss and then we evalute tradeoffs between throughput and detectability by testing the method using different parameters values.

For the first experiment, the network setup contains eight nodes connected on a 100mbps network hub. One node is the covert sender, one node is the covert receiver and the other six nodes are overt communicators that exchange mp3 audio using the real time service protocol (RTSP) enclosed in UDP. In total there are three pairs of connections among the six nodes. Each node acts as a server and client to a neighboring node (*i.e.* node A serves node B and node B also serves node A, etc.). The publicly available live555 media server software is used to serve the audio and the video lan client (VLC) tool is used to play the audio. Each node serves a different mp3 file (a total of six different mp3 files are used). The audio files are looped so the data exchange is continuous. The mp3 audio files are recorded conversations taken from a dialog corpus [15].

In order to observe the effects of the network load on throughput (specifically packet loss), we conducted a second experiment with a higher amount of network congestion. For the second experiment, the network setup consisted of the same eight nodes, plus six additional nodes exchanging tranmission control protocol (TCP) packets. We used a traffic generator to generate the TCP traffic, which was mainly audio and video data.

The following table shows the network load conditions during both experiments.

In both experiments, only the UDP connections were used as carriers for covert communication. The reason UDP was used is because the protocol is connectionless and there is no control mechanism as there is in TCP (with sequence numbers). During the send phase, only payload data are modified, not header fields. Duplicating TCP messages could alert an adversary to the covert communication. One way that TCP could be used is if a covert sender is also a legitimate sender that embeds covert data in the payload of select messages.

As the control variables, the following parameter values were used. We used two sync bytes and two checksum bytes to virtually eliminate false positives. The other parameters were chosen favoring detectability over throughput.

- Window Size = 1000ms
- Sync Bytes = 2
- Checksum Bytes = 2
- Rate to Use = 0.1
- Protocol to Use = UDP
- Rate Threshold = 10
- Connection Randomness Threshold = 10
- Contiguous Random Byte = (1 + 3) + 1 = 4

We ran each experiment with varying values for the *Byte Randomness Threshold* parameter and we measured the throughput and reliability during each run.

For each run, we first started a covert sender, which would automatically begin sending covert packets when valid threshold values were observed. Each packet included a sequence number (0–255) as the first byte of data. The sequence numbers were used to count packet loss. On the receiver missing or out-of-order sequences indicated missing packets.

For the sender, a large file was loaded into the covert data buffer. As a result, the buffer always contained enough data to fill all candidate carriers during each window. This was done to measure maximum throughput.

The receiver was started fifteen seconds after the covert sender. Packet loss was observed only after the first packet was received. The receiver was run for five minutes, during which all incoming covert messages were logged. After the experiment, we analyzed the logs and measured throughput and packet loss.

V. Results

The results are presented in Figures 3 to 5.

Figure 3 shows misses graphed against the throughput available given the *Byte Randomness Threshold* value. The data points are total number of misses averaged over the five minute runs. Even using UDP connections, which are inherently unreliable, the number of misses is very small compared to the available throughput received under both network loads. The variation in the packet loss during the higher network load may be attributed to the bursty nature of some of the TCP communication that was present. Since the number of misses were small, it may be feasible for a receiver to request resubmission of lost packets. Also, this shows that it may be favorable to communicate covertly over congested networks to decrease detectability because the communication is still reasonably reliable.

Figure 4 shows a graph of received throughput with different *Byte Randomness Threshold* values. As expected, the amount of throughput is consistent under both network loads (because the same UDP traffic was present for both). Also, as the threshold value is increased, the throughput decreases, but this also means that the data are placed in areas with more randomness, which would be more difficult for an adversary to find. It should be noted that when using a value of 0.9 for randomness, no covert data were was sent. Although there were some individual bytes that exhibitied this high randomness, there were not enough contiguous bytes to fit the covert data.

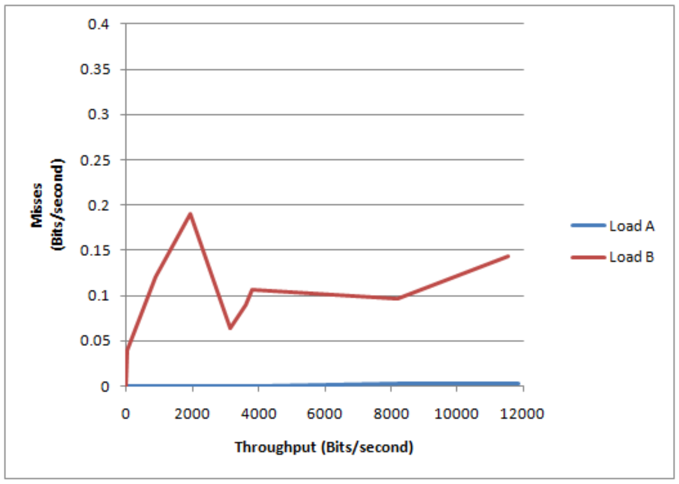

Fig. 3. Misses graphed against throughput

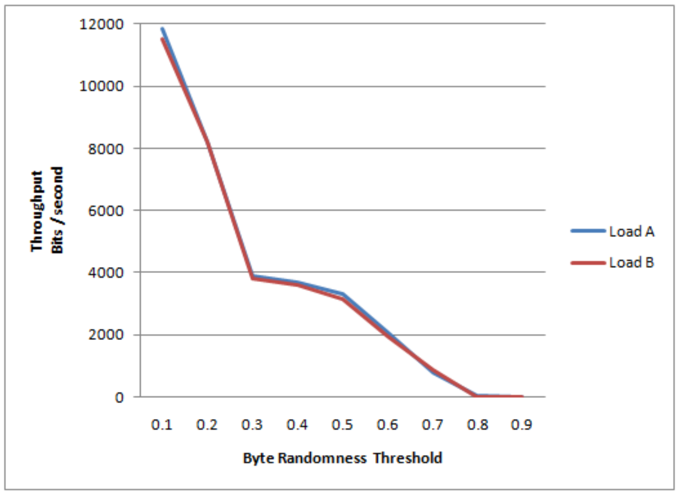

Fig. 4. Throughput graphed against Byte Rate Thresholds

Figure 5 shows the number of packets that were sent during both network loads. As mentioned earlier, the window size used during the experiments was one second. Also a maximum of six connections were considered as candidate carriers (only the six satisfied the threshold parameters). Although there were other UDP connections, used for controlling the media sessions, only the connections with audio data realized suitable randomness values. As a result, the maximum number of packets that were sent each window was six. This is observed in the graph. A high degree of randomness is expected because the the packets contain speech audio. When the threshold parameter values are higher, the number of packets used decreases. This is a good indication that the audio data differed. Some connections had higher levels of randomness due to the fact that in some dialogs, the speakers were more verbal. If the speakers talked less, there was more silence, which translated to less randomness. Although the same connections were used as covert carriers during both network loads, there are slight differences in the values. This is because the sender and receiver were started while background traffic was already running. The randomness threshold is reached at different times depending on the network traffic present during the instantiation of the BCM tool.

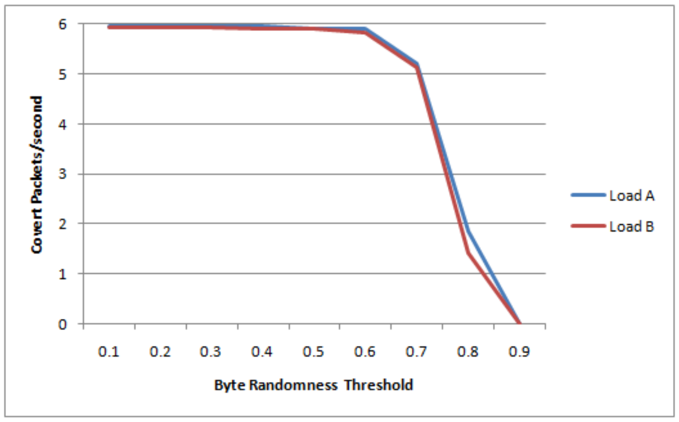

Fig. 5. Packets received graphed against Byte Rate Thresholds

VI. Conclusions and Future Work

We have developed a novel method for secure communication that communicates through multiple concurrent covert channels and can scale as network congestion increases. Evaluation of the method shows that the communication is reasonably reliable. We also demonstrate tradeoffs between throughput and detectability.

For future work we will determine the effects of higher throughput on quality of service. We will also investigate whether it is more beneficial to hide covert data based on byte similarity as in [13]. We will also evaluate the performance of the method using different network configurations such as WLAN, multicast, and switches configured with promiscuous ports. Lastly we will look into automatic methods for tuning the parameters in real time depending on the network.

Acknowledgments

The authors would like to thank Brenda Medina, Juan Ulloa, and Dan Landin for their valuable comments and suggestions.

References

[1] B. Lampson, "A Note on the Confinement Problem," *Communications of the ACM*, vol. 16, no. 10, pp. 613–615, 1973.
[2] A. Baliga and J. Kilian, "On covert collaboration," in *Proceedings of the 9th workshop on Multimedia & security*. ACM, 2007, pp. 25–34.
[3] W. Mazurczyk and Z. Kotulski, "Covert channel for improving VoIP security," *Advances in Information Processing and Protection*, pp. 271–280, 2008.
[4] T. Calhoun, R. Newman, and R. Beyah, "Authentication in 802.11 LANs using a covert side channel," in *Communications, 2009. ICC'09. IEEE International Conference on*. IEEE, 2009, pp. 1–6.
[5] S. Zander, G. Armitage, and P. Branch, "A Survey of Covert Channels and Countermeasures in Computer Network Protocols," *Communications Surveys & Tutorials, IEEE*, vol. 9, no. 3, pp. 44–57, 2007.
[6] N. Lucena, G. Lewandowski, and S. Chapin, "Covert channels in ipv6," in *Privacy Enhancing Technologies*. Springer, 2006, pp. 147–166.
[7] K. Szczypiorski and W. Mazurczyk, "Steganography in IEEE 802.11 OFDM symbols," *Security and Communication Networks*, 2011.
[8] K. Szczypiorski, "A performance analysis of HICCUPSa steganographic system for WLAN," *Telecommunication Systems*, pp. 1–5, 2009.
[9] W. Mazurczyk and J. Lubacz, "LACKa VoIP steganographic method," *Telecommunication Systems*, pp. 1–11, 2010.
[10] P. Fogla, M. Sharif, R. Perdisci, O. Kolesnikov, and W. Lee, "Polymorphic blending attacks," in *Proceedings of the 15th USENIX Security Symposium*, 2006, pp. 241–256.
[11] F. Yarochkin, S. Dai, C. Lin, Y. Huang, and S. Kuo, "Towards Adaptive Covert Communication System," in *2008 14th IEEE Pacific Rim International Symposium on Dependable Computing*. IEEE, 2008, pp. 153–159.
[12] Y. Huang, B. Xiao, and H. Xiao, "Implementation of covert communication based on steganography," in *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. IEEE, 2008, pp. 1512–1515.
[13] H. Tian, K. Zhou, H. Jiang, Y. Huang, J. Liu, and D. Feng, "An adaptive steganography scheme for voice over IP," in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 2922–2925.
[14] J. Rice, *Mathematical statistics and data analysis*. Thomson Learning, 2006.
[15] J. Acosta and N. Ward, "Achieving rapport with turn-by-turn, user-responsive emotional coloring," *Speech Communication*, 2010.