ECS 132 Spring 2018: Project

May 4, 2018

1 Introduction

The goal of this part of the project is to learn about how to design and detect timing channels and also how to engineer them. This is a draft write up. Some revisions are likely to arise but this should help you to start to think of the different steps and start working on it.

2 Background

Covert communication is method of exchanging secret messages in which the communication is hidden. A related term is steganography which deals with methods to write/embed hidden messages in such a way that no one, other than the sender and the intended receiver, know the existence of the message. The word steganography is of Greek origin and means ”concealed writing” from the Greek words steganos meaning ”covered or protected”, and graphei meaning ”writing [From Wikipedia]. Steganography and covert channels have a long history and was used in WWII to send secret messages to spies behind enemy lines. In the computer and network security, covert channels and steganography fall in the broad category of security through obscurity.

The advantage of covert channel over cryptography is that messages do not attract attention to themselves. Plainly visible encrypted messages — no matter how unbreakable — will arouse suspicion. The very presence of encrypted messages may be incriminating in countries where encryption is illegal. In such cases the communication channel must itself be hidden and this is achieved using covert channels / steganography. Note that, cryptography protects the contents of a message. Covert communication on the other hand protects both the message and the communicating parties.

Typically, steganography refers to the concealment of information within a document file, image file, or program. Media files are ideal for steganographic transmission because of their large size. As a simple example, a sender might start with an innocuous image file and adjust the color of every 100th pixel to correspond to a letter in the alphabet. The overall change is so small that someone not specifically looking for it is unlikely to notice it. Another method is called the Least Significant Bit (LSB) substitution. In this method, the least significant bit of each pixel in a digital image is modified by the bits of the secret message. Since the LSB contributes very little to the overall (intensity/color/brightness) of each pixel, the change in the image will be imperceptible to the naked eye.

Covert channels are communication channels that are established over some overt medium. For example, we can uses a stream of network packets as the overt carrier for a covert communication channel. As usual, we have our three characters Alice, Bob, and Eve. Alice and Bob are in a repressive country where all communication is monitored and they want to establish a covert channel to exchange secret messages. Eve is a warden who can look at all network packets and try to detect if any covert communication is being used to plan a uprising against the repressive state.

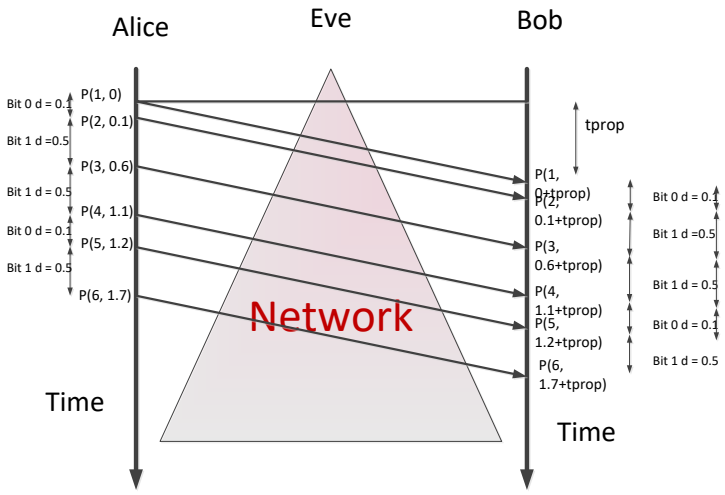

Figure 1: An example of timing channel.

To setup a covert communication channel, Alice and Bob first initiate an overt application let say a (computer to computer) Skype call and they start a regular innocuous conversation. Their interactive conversation produces a stream of IP packets from Alice to Bob and Bob to Alice. For the time being let us only consider covert channel in one direction from Alice to Bob and hence only consider the IP packets stream from Alice to Bob. There are two ways in which Alice can send a secret message. She can replace some unused bits in the protocol header with the bits of the covert message. This is called covert storage channel. These are easily detectable since the protocol header fields that are not used are well known to Eve and she can check bits to detect the covert channel, identify Alice and Bob and the covert message. The other method that Alice can uses function is that she can alter the inter-packet delays of the IP packet using a pre-established to modulate the bits of the secret message. This is called a covert timing channel and will be the focus of our study. Let's looks at an example to make things more concrete.

For simplicity, we will assume that Alice has buffered a large number of the IP packets that she has generated. This is obviously not realistic [why?] but in this study we will not worry about the quality of the Skype call. Each packet has two attributes 1) a sequence number and 2) the time when the packet was generated. Thus, P(n, tn) denotes packet n which was generated at time tn. We will assume that the first packet is numbered 1 and is generated at time 0, i.e., t1 = 0. Note that the time field gives the cumulative time. To obtain the inter-packet delay, we can take the time difference between the consecutive packets. This packet stream is the unmodified overt traffic. Alice and Bob have a priori decided that an inter-packet delay of 0.5 will be used to code 1 and an inter-packet delay of 0.1 will be used to code a 0. So if Alice wants to send the alphabet "h" (which is 68 Hex or 01101000) she will generate the following sequence of packets P(1, 0), P(2,0.1), P(3, 0.6), P(3, 1.1), P(4, 1.2), P(5, 1.7), P(6, 1.8), P(7, 1.9), P(8, 2.0). A part of this is shown in Figure 1.

If the timing between the packets are not altered by the network or by Eve, then Bob can observe the inter-packet delays, translate them to binary bits and then determine the corresponding character. In this assignment, we will try to design a method of modulating the bits into inter-packet delays such that Eve is not able to discover the channel. We will assume that the network or Eve will not modify the inter-packet delays.

3 Design

Here are the details

1. The secret message is "this is a secret message" The characters are encoded using 8 bits. You will be given a table that gives the mapping of the characters to 8 bit codes.

2. There is an attached excel sheet that gives the packet stream (packet number and the when it was generated). You do not need to use all the samples.

You will answer the following question.

Question 1: Alice and Bob decide to use the following modulation scheme to map the bits to the inter-packet delay. A delay of 0.25 is used to encode a bit 0 and delay of 0.75 is used to encode a bit 1. Write a short R code that will generate the modified packet stream that contains the secret message.

Question 2: Plot the histogram of the inter-packet delays of the overt packet stream. Plot the histogram of the covert packet stream. Will Eve be suspicious?

Question 3: Alice and Bob decide to use the following modulation scheme. Let m, min, and max denote the median, min, and max of the inter-packet delay of the overt packet stream. If Alice needs to send a 0 she randomly generates a delay between min and m. If she want to send a 1 she randomly generates a delay between m and max. First, compute m, min, and max. Next, modify the code in Question 1, to generate the packet stream that contains the secret message.

Question 4: Plot the histogram of the inter-packet delays of the overt packet stream and that of the new covert packet stream. Do you think Eve will be suspicious?

Question 5: Answer the following questions briefly (in 1 or 2 sentences)

1. How can you improve upon the method in Question 3?
2. We assumed the Alice will buffer up the packets and we mentioned that it was unrealistic. Why?
3. We have assumed that the network does not alter the inter-packet delays. What would be the problem if it did? Can you suggest methods to mitigate the effect of the changes of the inter-packet delay (noise)?

4 Detection

In this section we learn about how to detect the timing channel. Recaping Part 1 of the project, Alice and Bob are secretly trying to embed a secret message in the inter-packet delays (IPDs) of packets generated by their innocuous Skype call. They do so by following a well-known statistical model of the Skype. This type of covert timing channels are called Model-based Covert Timing Channels (MBCTCs). Eve is the warden and she wants to detect the timing channel. Eve knows the model of the IPDs of the (unmodified) Skype traffic. She observes the covert traffic and uses `qqplot` which can be used to see if two data sets come from a population with a common distribution.

1. See the following URL `http://www.itl.nist.gov/div898/handbook/eda/section3/qqplot.htm` to understand `qqplot`.

2. Using the R help function for `qqplot` learn about the usage of `qqplot`.

3. While `qqplot` works with data sets of different sizes, in this project we will consider data sets that are of equal size.

4. Generating values that follow a specific distribution can be easily done in R. For example, the function `rnorm` can be used to generate values that follow the normal distribution with mean μ and standard deviation σ. Similarly, `runif` for data sets following the Uniform distribution, and `rexp` for data sets following the exponential distribution. Use the R help function to learn their usage.

4.1 Steps

Do the following steps.

1. Generate two different samples of the standard Normal distribution (i.e., mean $\mu = 0$ and $\sigma = 1$) of size $n = 30$ using `rnorm`. Do a `qqplot` of the two data sets.

2. Repeat the above step for $n = 100, 1000$. Give a summary of you observations.

3. Generate two data sets each of size $n = 100$, one for standard Normal and the other for Normal with $\mu = 5$ and $\sigma = 3$. Draw the `qqplot` and record you observation.

4. Do step 2 for Exponential distribution with $\lambda = 1$.

5. Using `qqplot` compare the standard Normal with Exponential with $\lambda = 1$ for sample size $n = 100, 500$.

6. Now consider the naive embedding scheme (Question 1/Step 1 of Part 1). Using `qqplot` compare the distribution of the IPDs of the overt and covert traffic.

7. Now compare the distribution of the IPD of the overt and the covert traffic generated by improved scheme in Question 3 in Part 1 of the project.

8. Finally, the Compare the IPD of the overt and covert traffic for the embedding scheme that you developed for Part 1 of Question 5.

5 Implemenation

In this section, we will work through the implementation issues. The overall sender-side system is shown in

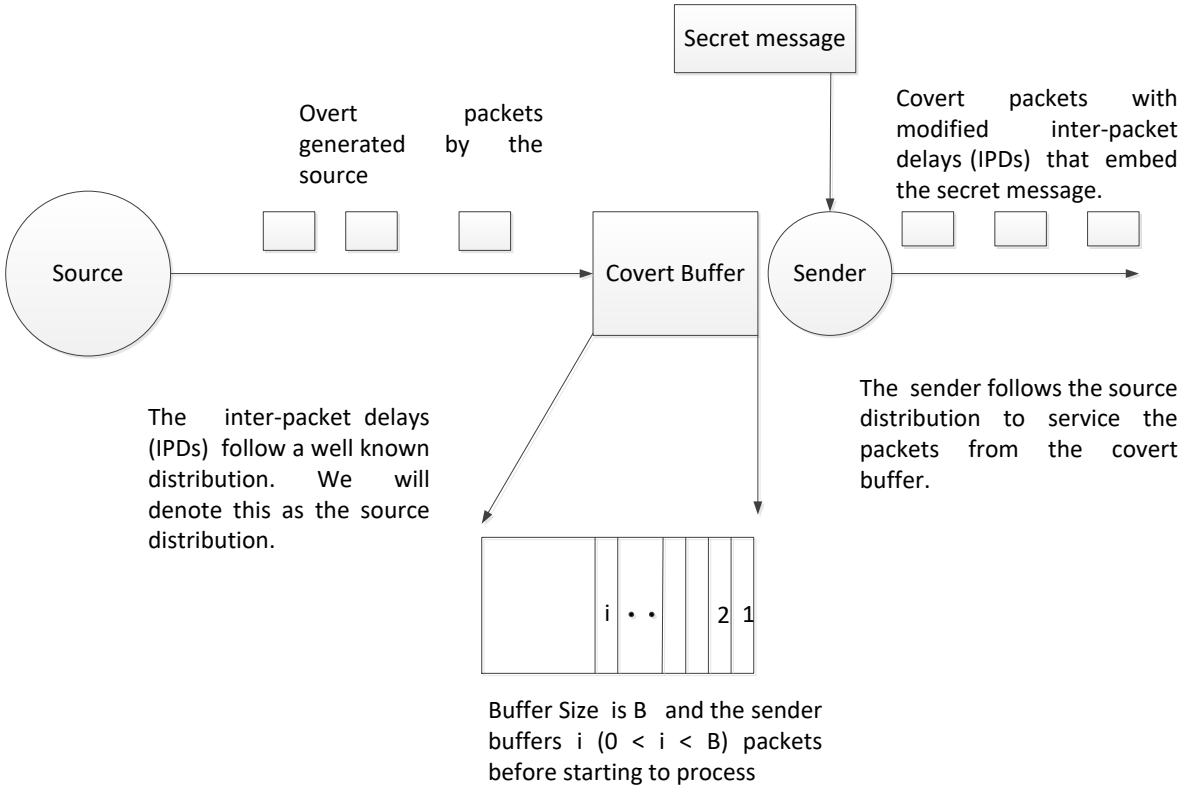

Figure 2: The overall system diagram of the source and the covert sender.

We will make the following assumptions

1. We will consider that the source generates packet following a well-known IPD distribution. Specifically, we will consider two cases a) Exponential and b) Uniform. The sender also knows this distribution and follows it to inject the delay between the packets to embed the secret message. It is important to note that the source and the sender are independent. Hence, even though they follow the same distribution, the sequence of delays generated by the source will be different from the sequence of delays generated by the sender.

2. To embed a 0, the sender generates a delay between the minimum value (min) and the median. To embed a 1 the sender generates a delay between the median value and the maximum value (max).

3. The secret message is a randomly generated sequence of 1s and 0s of size m bits and is given. We will consider two values $m = 16, 32$.

4. The sender has a buffer of size B and initially the sender buffers i packets before starting to send the secret message.

5. Buffer overflow: when a packet arrives at the sender and there are already B packets in the buffer.

6. Buffer underflow: when the sender needs to send a packet but there are no packets in the buffer.

In this project complete the following steps.

1. For buffer size $B = 20$ we want to find out the probability of overflow and underflow, when the IPD follows the Exponential with $\lambda = 1$ and $i = 2, 6, 10, 14, 18$. Use message size $m = 16, 32$ bits.

2. For buffer size $B = 20$ we want to find out the probability of overflow and underflow, when the IPD follows the Uniform distribution in the range (0,1) and $i = 2, 6, 10, 14, 18$. Use message size $m = 16, 32$ bits.

3. Using the Gambler's Ruin problem determine bounds on overflow and underflow probabilities.

4. Propose methods to deal with buffer overflow and underflow.

5.1 Some Notes on Simulating the Implementation

For steps 1 and 2, since the source and the sender are independent processes, a proper way to simulate would be using a discrete event simulation module such as `simmer` in R. However, we can simplify and use basic R. To do this, we can pre-generate the times when the source generates packets and store it in a list. Then we can write the code to simulate the buffer, the encoding scheme, and the sender. This can be done in a single "process." Based on this, following is a very **rough** set of steps to simulate the system.

For each experiment we can break it down to the following steps

1. Generate the random bit pattern of 1s and 0s of size m which is the secret message.

2. Generate a sequence of times when the source will generate the packets. This is based inter-packet delay (IPD) distribution of the packets generate by the source. You can intuit what is the worst case number of packets that you need.

3. For the buffer you need to keep some variables such as B: buffer size, i: the initial buffer size to start sending he secret message bits and CB: current buffer size.

4. For the sender you need to maintain some variables such as the time when the next packet will be sent.

5. For each secret message bit:

   (a) Generate a delay following the encoding scheme and hence determine when the next packet will be transmitted.

   (b) Update the state of the buffer depending on the number of arrivals during that time.

   (c) At appropriate places check for buffer underflow and overflow and break out if it the case.

   (d) Appropriately update the current time.

6. Do the experiment multiple times to calculate the different probabilities.

6 Submission Guidelines

Here are the submission guidelines

1. You are required to submit the project online as Rmarkdown file.

2. You must work groups of size 2.

3. For schedule regarding submission see Canvas.