CENG 519 - Network Security - Project Phase 2 Report

Kemal Anıl Kekevi
2380608

April 13, 2025

1 Covert Channel Choice

My selection: Modify inter packet delay of packets: IP

2 Covert Channel Design

In this study, I implemented a timing-based covert channel using inter-packet delays. The system consists of a `covert_channel_sender` running in the `sec` container and a `covert_channel_receiver` running in the `insec` container, both implemented in Python.

Sender Design

The sender script accepts the following arguments: `port`, `message`, `zero_bit_delay`, `one_bit_delay`, and `bit_repeat_len`. Here:

- `message` is the string to be transmitted covertly.
- `zero_bit_delay` and `one_bit_delay` are the time intervals (in seconds) the sender sleeps between sending packets representing bits '0' and '1', respectively.
- `bit_repeat_len` specifies how many times each bit is redundantly transmitted to increase robustness against noise.

The sender first initializes the necessary variables and connects to the receiver via a UDP socket. It then iterates through the given `message's` bitstream (uses `create_bitstream_from_message` method to get messages bitstream), and for each bit, it sends the same packet `bit_repeat_len` times, applying the corresponding delay (from a predefined delay mapping) between transmissions.

Receiver Design

The receiver script accepts the following arguments: `port`, `sender_bit_delay`, `given_delay_threshold`, `bit_repeat_len`, and `bitstream_len`. These are defined as follows:

- `sender_bit_delay` is the delay used for bit '0' by the sender.
- `given_delay_threshold` is an offset added to `sender_bit_delay` to form the decision boundary for classifying incoming bits.
- `bit_repeat_len` is the number of packets expected per bit.
- `bitstream_len` is used for logging and evaluation purposes only.

The receiver initializes internal variables and begins listening on the specified UDP port. For each received packet, it computes the inter-packet delay (IPD) and appends it to a list. When the number of recorded IPDs reaches `bit_repeat_len`, the receiver computes the mean IPD for that group and compares it against the threshold value (`sender_bit_delay` + `given_delay_threshold`). If the average delay is less than the threshold, the bit is interpreted as '0'; otherwise, as '1'. The IPD list is then cleared for the next bit. This process repeats indefinitely.

Statistical Analysis Method

To estimate the performance with statistical confidence, the sample mean and the 95% confidence interval were computed using the following formula:

$$\bar{x} = \text{mean of samples}$$
$$\text{SEM} = \frac{s}{\sqrt{n}}$$
$$\text{CI}_{95\%} = \bar{x} \pm t_{(n-1,0.975)} \cdot \text{SEM}$$

Where s is the sample standard deviation, n is the number of samples, and t is the t-distribution critical value.

3 Experimentation Campaign

Experiment 1

Variables:

- Processor Delay Mean: 5e-6
- Iteration Count:3
- Bitstream Length: 100
- 0 Bit Delay:0.1
- 1 Bit Delay: 0.2
- Given Delay Threshold: 0.05

Result:

- BER: 0.0000 (95% CI: [0.0000 – 0.0000])
- Channel capacity(effective throughput — the number of useful bits per second): 2.2032 95% CI: [2.2026 – 2.2038]
- Avg. Transmission time: 45.383 seconds

Experiment 2

Variables:

- Processor Delay Mean: 5e-6
- Iteration Count:2
- Bitstream Length: 10
- 0 Bit Delay:0.1
- 1 Bit Delay: 0.2
- Given Delay Threshold: 0.05

Result:

- BER: 0.28 (95% CI: [0.254, 0.306])
- Channel capacity(effective throughput — the number of useful bits per second): 2.58 (95% CI: [2.42, 2.74])
- Avg. Transmission time: 2.8247 seconds

Experiment 3

Variables:

- Processor Delay Mean: 5e-6
- Iteration Count:10
- Bitstream Length: 10
- 0 Bit Delay:0.1
- 1 Bit Delay: 0.2
- Given Delay Threshold: 0.05

Result:

- BER: 0.0000 (95% CI: [0.0000 – 0.0000])
- Channel capacity (effective throughput — the number of useful bits per second): 0.7081 (95% CI: [0.7075 – 0.7086])
- Avg. Transmission time: 14.1255 seconds

Experiment 4

Variables:

- Processor Delay Mean: 5e-6
- Iteration Count:10
- Bitstream Length: 20
- 0 Bit Delay:0.1
- 1 Bit Delay: 0.2
- Given Delay Threshold: 0.05

Result:

- BER: 0.0000 (95% CI: [0.0000 – 0.0000])
- Channel capacity (effective throughput — the number of useful bits per second): 0.6839 (95% CI: [0.6836 – 0.6842])
- Avg. Transmission time: 29.2458 seconds

Experiment 5

Variables:

- Processor Delay Mean: 5e-6
- Iteration Count:5
- Bitstream Length: 200
- 0 Bit Delay:0.05
- 1 Bit Delay: 0.150
- Given Delay Threshold: 0.05

Result:

- BER: 0.0000 (95% CI: [0.0000 – 0.0000])
- Channel capacity (effective throughput — the number of useful bits per second): 1.9747 (95% CI: [1.9742 – 1.9753])
- Avg. Transmission time: 101.2846 seconds

Experiment 6

Variables:

- Processor Delay Mean: 5e-6
- Iteration Count:5
- Bitstream Length: 200
- 0 Bit Delay:0.005
- 1 Bit Delay: 0.015
- Given Delay Threshold: 0.002

Result:

- BER: 0.0140 (95% CI: [0.0097 – 0.0183])
- Channel capacity (effective throughput — the number of useful bits per second): 17.7032 (95% CI: [17.583 – 17.824])
- Avg. Transmission time: 11.1626 seconds

Experiment 7

Variables:

- Processor Delay Mean: 0.05
- Iteration Count:5
- Bitstream Length: 200
- 0 Bit Delay:0.05
- 1 Bit Delay: 0.15
- Given Delay Threshold: 0.05

Result:

- BER: 0.138 (95% CI: [0.115 – 0.160])
- Channel capacity (effective throughput — the number of useful bits per second): 1.7017 (95% CI: [1.669 – 1.734])
- Avg. Transmission time: 101.38 seconds

Experiment 8

Variables:

- Processor Delay Mean: 0.05
- Iteration Count:5
- Bitstream Length: 200
- 0 Bit Delay:0.005
- 1 Bit Delay: 0.015
- Given Delay Threshold: 0.005

Result:

- BER: 0.4995 (95% CI: [0.496 – 0.503])
- Channel capacity (effective throughput — the number of useful bits per second): 1.8593 (95% CI: [1.823 – 1.896])
- Avg. Transmission time: 53.7152 seconds

Experiment 9

Variables:

- Processor Delay Mean: 5e-6
- Iteration Count:5
- Bitstream Length: 200
- 0 Bit Delay:0.005
- 1 Bit Delay: 0.015
- Given Delay Threshold: 0.009 (Total Threshold 0.14)

Result:

- BER: 0.0685 (95% CI: [0.058 – 0.079]) (Since external delays added threshold less then 1 Bit Delay does not create significant erros)
- Channel capacity (effective throughput — the number of useful bits per second): 16.5481 (95% CI: [16.337 – 16.759])
- Avg. Transmission time: 11.2453 seconds

Experiment 10

Variables:

- Processor Delay Mean: 5e-6
- Iteration Count:5
- Bitstream Length: 200
- 0 Bit Delay:0.010
- 1 Bit Delay: 0.015
- Given Delay Threshold: 0.005

Result:

- BER: 0.106 (95% CI: [0.089 – 0.123]) (Nearly bit delays causes bigger BER)
- Channel capacity (effective throughput — the number of useful bits per second): 13.074 (95% CI: [12.87 – 13.28])
- Avg. Transmission time: 13.656 seconds

4 Optimization Options

To enhance the performance and increase the capacity of the covert timing channel, several optimization strategies can be considered:

- **Multi-bit Symbol Encoding:** Instead of encoding a single bit per delay interval, we can use multiple discrete delays to represent multi-bit symbols (e.g., 2-bit or 3-bit symbols). This significantly increases the channel capacity, provided that the delay values remain distinguishable under channel noise.
- **Parameter Sweeping:** A systematic set of experiments can be conducted to explore various combinations of 0-bit and 1-bit delays. The goal is to identify the pair that maximizes capacity while keeping the Bit Error Rate (BER) acceptably low. This can be further automated through grid search or optimization algorithms.
- **Error Correction Coding (ECC):** Implementing lightweight error correction schemes (such as Hamming codes or repetition coding with majority voting) can help reduce the impact of noise without drastically lowering throughput. This allows for a more aggressive capacity configuration while maintaining reliability.

5 Conclusion

In this assignment, I implemented a covert timing channel based on inter-packet delays (IPD) to transmit binary data between a sender and receiver operating across an insecure network. The sender encoded each bit by introducing a predefined delay between UDP packet transmissions, while the receiver decoded the message by measuring inter-arrival times and applying a fixed delay threshold.

To evaluate system performance, I conducted a series of controlled experiments by varying parameters such as 0-bit and 1-bit delay values, delay thresholds, and bit repetition lengths. For each configuration, I measured the Bit Error Rate (BER), effective channel capacity (in bits per second), and average transmission time. I also computed 95% confidence intervals to quantify variability across trials.

The results reveal a clear trade-off between reliability and capacity. Configurations with larger delay gaps and higher bit repetition achieved near-zero BER but incurred significantly higher transmission times and lower throughput. In contrast, configurations with minimal delay gaps or limited bit repetition produced higher capacities but were prone to BERs near 0.5, making them unreliable without additional correction. These issues were especially pronounced due to the random nature of the test bitstreams, which often contained balanced numbers of 0s and 1s. Threshold tuning was also critical; the optimal decision threshold was typically close to the midpoint between the 0-bit and 1-bit delays but varied depending on noise introduced by the processor.

Overall, the experiments demonstrate that with careful delay tuning and the possible integration of error correction mechanisms, a covert timing channel based on IPD can achieve a practical balance between stealth, reliability, and throughput under realistic operating conditions.

All of the experiments are made in sec environment's phase2_experiment_sender.py file.

6 GitHub Repository

Project code and this report are available at:
`https://github.com/ANILKE/middlebox/tree/phase2`