CIS4212 Trustworthy Infrastructures

Undergrad Group 17

Henrique Gamonal De Castro & Carlos Pineda

January 29, 2026

Assignment 1 Extra Credit

A. Potential Vulnerabilities

When I analyzed the scenario where Alice's computer crashes and restarts, I realized that the main issue is not that the encryption stops working, but that the program forgets where it was. In class, we learned that a Pseudo Random Number Generator is a determinstic function. This means that if the program starts over with the exact same seed file and no memory of where it stopped, it will produce the exact same sequence of random bits that it used before the crash. This creates a dangerous situation called a two time pad vulnerability because the same keystream is being used to encrypt two different sets of data. If an attacker captures both the old ciphertext and the new one, they can combine them to cancel out the key entirely. This leaves them with just the combination of the two original messages. Since real messages usually have predictable words or headers, the attacker can use this result to seperate and read the original plain texts without ever knowing the key.

B. Potential Remedies

To fix this, I thought about how we need to make sure the keystream never repeats, even if the power goes out. One way to solve this locally is to use a state file that updates constantly. The lectures mentioned that a generator must be supported with a state to ensure it does not produce the same randomness for the next message. Instead of just reading the static seed, the program could also read and write a counter number to the hard drive every time it encrypts a message. If the computer crashes and turns back on, the program would check this file to see where it left off or use that number to change the seed slightly so the random numbers are fresh. This prevents the generator from starting at the beginning again.

Another way to handle this, especially if we have an internet connection, is to rely on a different computer to help us coordinate. The lectures mentioned that using unique numbers called nonces is critical for security. We could have the program connect to a central server to ask for a unique session ID or token before it starts encrypting. This server would make sure that even if my specific machine crashes or if I try to run the code from a different laptop, I never get the same starting parameters twice. This prevents the accidental reuse of the keystream that leads to the vulnerability I described earlier.