# Mission 3: Occupancy-based Covert Channel Attack

**Top Secret Briefing**

Imagine trying to pass a message from the Boys Hostel to the Girls Hostel, no phones, no texts, just creating chaos in the common mess. If it's crowded, that's a '1'. If it's calm, that's a '0'. That's exactly what you'll do here, create CPU traffic so the other side can "feel" the delay and decode your message.
No shared memory. Just strategic overload. Welcome to the world of **Occupancy-based Attacks**.

## 1 Mission Objective

Establish a covert channel between two separate processes, a **sender** and a **receiver**, running on different CPU cores, using an **occupancy-based covert channel**. The sender must transmit the secret message stored in `msg.txt`.
**Important:** No shared memory is allowed between the sender and receiver processes.

## 2 What Is an Occupancy-based Attack?

In this type of covert channel, communication is achieved not through shared memory, but through intentional **LLC contention**. One process thrashes LLC, while the other detects and decodes the usage via timing measurements.

- Logical '1': High cache usage to cause LLC contention
- Logical '0': Low or no LLC usage
- The receiver decodes information based on variations in its own access latencies

Think of it as sending Morse code through LLC congestion, each bit encoded in how busy the cache feels.

## 3 Instructions

- Use the same **starter template files** provided for Mission 2.
- Modify `sender.c` to implement the occupancy-based encoding mechanism.
- Modify `receiver.c` to measure execution latency and decode the message.
- Ensure the message in `msg.txt` is transmitted accurately and efficiently.

## 4 Deadline

**All agents must complete this mission by: Saturday, 6th July.** Submissions received beyond this deadline will be considered intercepted by enemy forces and compromised.

## End of Transmission

This message will self-destruct in 5 clock cycles… unless the scheduler gets there first.