1. Objective

To establish a covert communication channel between two isolated processes (sender and receiver) using only shared memory and CPU cache behavior (Flush+Reload technique), without using any traditional communication mechanisms like sockets, pipes, signals, or direct reads/writes.

---

2. Design and Implementation

Shared Memory:

We used `mmap()` with `MAP_SHARED | MAP_ANONYMOUS` to allocate a shared memory page accessible by both processes.

Cache-based Signaling:

- **Sender** transmits bits by accessing or not accessing a shared memory address within a defined timing interval.
    - Bit `1`: repeated memory accesses → cached
    - Bit `0`: no access → flushed
- **Receiver** uses `rdtsc` to measure access time to that memory:
    - Fast (< threshold) = bit `1`
    - Slow = bit `0`

Synchronization:

A **start byte pattern** (`0b10101010`) is sent before transmission to help receiver synchronize with the sender's stream.

Timing:

- Each bit is transmitted for a fixed interval (`INTERVAL = 400000` CPU cycles)
- Delay cycles between bits ensure alignment and reduce noise

---

3. System Setup

- OS: Ubuntu 22.04 LTS
- Compiler: GCC
- Architecture: x86_64
- Execution:
    - `taskset -c 0 ./receiver > output.txt &`
    - `taskset -c 1 ./sender`

---

4. Performance Metrics

Time taken to send the message: 0.585141
Message size: 381
Bits per second: 5209.000907
Accuracy ~99.73%

Demo Video

https://youtu.be/xbTRP-DNFko