***SAFE P2P***

*Eitan Coronel 315468132*

*Elyashiv Newman 206387177*

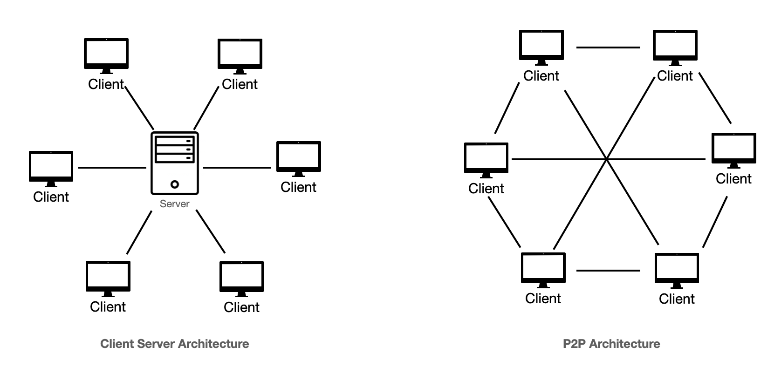
*Note:*

*The code includes a GUI showcasing the full capabilities of our P2P system. For a detailed overview of the GUI, please refer to the README [1].*

1. **Explain the problem space**:

Our project addresses privacy concerns in peer-to-peer (P2P) file-sharing networks.

In traditional P2P systems, users share and download file segments across a decentralized network where each peer acts as both a client and a server. Files are split into chunks, and peers download these chunks from multiple sources simultaneously, often with the help of a Distributed Hash Table (DHT) that tracks file availability.



As we can see in this picture [2], the P2P network (right) differs significantly from a traditional client-server network (left). In a client-server model, all clients connect to a centralized server to download or upload data. This means that the server is the sole distributor of files, creating a potential bottleneck and single point of failure. In contrast, in a P2P network, all peers communicate and exchange data directly with each other without relying on a central server. Each peer can download file chunks from multiple other peers, increasing resilience.

While this decentralized approach makes file sharing more robust, it also exposes users to privacy risks, as adversaries can observe network traffic and track which files or file segments are being requested and downloaded by specific peers.

1. **Motivate the importance of the problem you’re targeting:**

The need for privacy in digital communication and file-sharing is critical in today's data-driven world. P2P networks are widely used for various applications, sharing large files via platforms like BitTorrent, and even use to power streaming services such as Spotify (which previously used P2P technology to distribute music).  
A decentralized communication system is vital in today's world, where centralized communication channels are susceptible to monitoring or censorship by third parties such as governments, hackers, or observers [3]. As well as by the platform hosting the communication, which can share or censor the data it collects. Nevertheless, without robust privacy safeguards, users in decentralized networks remain vulnerable to surveillance and tracking by malicious entities [4].  
This exposure allows adversaries to monitor and analyse which files users are downloading, which could lead to potential risks, especially in regions where internet usage is closely monitored, or for users accessing politically sensitive content. The ability to observe user behaviour in P2P networks can result in privacy breaches, data mining, or even legal consequences for individuals who unintentionally expose their downloading activity to the public or adversaries.

1. **Describe your solution and how it compares to the state of the art:**
   1. **What is the attacker model? What assumptions does it make?**

In our project, the attacker model assumes that adversaries can monitor network traffic within the peer-to-peer (P2P) network. These adversaries could be other peers in the network, a network observer, or a malicious entity with access to the P2P communications. The goal of the attacker is to track and infer which files a particular user is downloading. The attacker is considered to have the capability to observe all communications between peers, including the frequency, size, and timing of the data packets exchanged. This allows the attacker to identify patterns and potentially deduce which files or file segments are being requested, even if the content itself is encrypted.

The model assumes that while the attacker cannot decrypt encrypted data, they can still see the encrypted packets being exchanged, giving them visibility into communication patterns. Additionally, the attacker could be strategically positioned within the network to correlate requests over time, making it easier to identify consistent downloaders of files or content.

Their overall aim is to use this gathered information to profile users, determine their download behaviours, or even attempt to uncover sensitive information based on their observed activity.

* 1. **Motivate your design given the attacker model:**

Given the attacker model and the project's focus on download privacy within the network (assuming upload privacy is either handled by a third party or is irrelevant because the file has already been uploaded and the main objective is to retrieve it securely), we will implement a solution using a single server with Computational Private Information Retrieval (CPIR) [4].

CPIR (Computational Private Information Retrieval) is a technique that enables data retrieval without revealing which specific data is being accessed, ensuring privacy without the inefficiency of downloading the entire database. Unlike the naive approach—where downloading the entire database would theoretically preserve privacy but is highly inefficient—CPIR allows users to query data selectively, achieving similar privacy with far greater efficiency.

In this project, our core strategy is to segment each file into multiple "small" files (subfiles) which are distributed across a network of peers. Each subfile is approximately 1MB in size. To enhance data resilience, we apply Reed-Solomon error correction codes[5], allowing TODO i dont know how to explain this clearly. This ensures TODO.

1. **Evaluate your solution**

*An important note:*

*In a distributed system the overall system performance is determined by the slowest component: the “bottleneck”.*

To evaluate our solution, we will analyze it across three distinct dimensions:  
a) Computational Time Complexity  
b) Storage Complexity  
c) Communication Complexity

**a) Computational Time Complexity**

Our solution utilizes Reed-Solomon (RS) encoding (via the zfec library) to split a file into smaller parts for redundancy and reconstruct it later. The encoding process transforms a file into n parts, each of size original\_size/k. This encoding results in a computational complexity of O(n\*k), due to the matrix operations inherent in zfec.

Similarly, during reconstruction, combining k parts yields a decoding complexity of O(k^2). Our solution also incorporates asymmetric cryptography for encryption and decryption operations, which have a complexity of O(n^2) per data point, where n represents the modulus size in bits (e.g., 2048 bits). This encryption complexity can be significant depending on file size and required security level. During Private Information Retrieval (PIR), a dot product is computed between vectors, with a complexity of O(n), where n is the vector length.

**b) Storage Complexity**

The ratio of n/k (total parts / required parts) determines the redundancy factor, which directly impacts storage overhead. For instance, if k = 1000 and n = 2000, the redundancy factor is 2, meaning storage requirements double. This storage overhead must be carefully managed to avoid excessive duplication, especially when files are divided into many smaller parts. Furthermore, if the same file is uploaded by multiple peers, storage overhead increases proportionally. To mitigate unnecessary duplication, deduplication checks are implemented at the node level to ensure efficient use of space.

**c) Communication Complexity**

Communication in our system occurs through socket programming for operations such as uploading, downloading, and managing file parts. The connection setup has a constant complexity of O(1), while message passing with size m has a complexity of O(m). File upload and download operations traverse distributed nodes in a linear fashion, resulting in a worst-case complexity of O(n), where n is the number of nodes. Additionally, downloading a file requires a linear search in the Distributed Hash Table (DHT), resulting in a combined complexity of O(n + m), where m represents the number of messages exchanged during retrieval.

**Bibliography:**

*[1]* [*https://github.com/ElyashivN/Safe\_P2P/blob/main/README.md*](https://github.com/ElyashivN/Safe_P2P/blob/main/README.md)

*[2]* [*https://systemdesignschool.io/blog/peer-to-peer-architecture*](https://systemdesignschool.io/blog/peer-to-peer-architecture)

*[3] growing trend of censorship in the world:* [*https://freedomhouse.org/article/new-report-global-battle-over-internet-regulation-has-major-implications-human-rights*](https://freedomhouse.org/article/new-report-global-battle-over-internet-regulation-has-major-implications-human-rights)

*[4] Millions of Cameras Using P2P Protocols Are Now Exposed to Hackers* [*https://central.bitdefender.com/blog/hotforsecurity/millions-cameras-using-p2p-protocols-now-exposed-hackers/*](https://central.bitdefender.com/blog/hotforsecurity/millions-cameras-using-p2p-protocols-now-exposed-hackers/)

*[5] Reed-Solomon:* [*https://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed\_solomon\_codes.html*](https://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed_solomon_codes.html)