**PyTorShare: A Command-Line Peer-to-Peer File Sharing Tool over the Tor Network**

*Project Synopsis Submitted*

*to*

**MANIPAL ACADEMY OF HIGHER EDUCATION**

*For Partial Fulfillment of the Requirement for the*

*Award of the Degree*

*Of*

**Bachelor of Technology**

*in*

**Computer and Communication Engineering**

*by*

Siddharth Warriar, Kabir Panda, Anirudh Anand Reg. No. 230953274, Reg. No. 230953154, Reg. No. 230953146,

*Under the guidance of*

Dr. Divya Rao (Lab Faculty 1) Mr. Sudhina Kumar G K (Lab faculty 2) School of Computer Engineering School of Computer Engineering Manipal Institute of Technology Manipal Institute of Technology MAHE, Manipal, Karnataka, India MAHE, Manipal, Karnataka, India



**September 2025**

**Objective**

The primary objective of this project is to design, develop, and demonstrate a functional prototype of a secure, peer-to-peer file transfer application that operates over the Tor network. The project aims to create a user-friendly tool with a Graphical User Interface (GUI) for the sender, allowing them to easily share a file from their local machine by creating a temporary, anonymous Tor onion service. This ensures the protection of both the sender's and receiver's identities and locations, serving as an effective, practical tool for understanding the principles of anonymity and authentication.

**Scope**

The scope of this project is to create a proof-of-concept application with a defined set of core functionalities, emphasizing ease of use and basic security.

**In Scope:**

* An application built using Python, consisting of a GUI-based sender and a command-line receiver.
* The sender application will feature a simple GUI built with tkinter allowing users to browse for a file, set a username and password, start the sharing service, and stop it.
* Password protection using HTTP Basic Authentication to prevent unauthorized downloads.
* The sender will programmatically create an ephemeral (temporary) V3 onion service by interfacing with a locally running Tor daemon.
* The receiver will be a command-line script that requires the .onion address, username, and password to authenticate and download the file.

**Out of Scope:**

* The application will not support transferring multiple files, directories, or real-time chat.
* The project is an educational prototype. Advanced features like cryptographic key exchange, transfer queues, or rate limiting are not in scope.

**Need for the Application**

In an age of pervasive digital surveillance and increasing internet censorship, the need for private and anonymous communication channels is more critical than ever. Standard file sharing methods (e.g., email, cloud services) rely on centralized third-party servers, which can log IP addresses, inspect data, and become single points of failure.

This project addresses the need for a decentralized, trustless, and anonymous file-sharing mechanism that includes a layer of access control. It provides a solution for individuals like journalists or activists who need to exchange sensitive information only with an intended recipient, without revealing their identity or location.

**Project Description**

Problem Statement: The core problem is the lack of a simple, serverless method for two individuals to directly exchange a file without exposing their IP addresses or relying on a trusted intermediary, while also ensuring that only the intended recipient can access the file.

Project Implementation: This project, "PyTorShare," solves this problem by leveraging the Tor network for anonymity and HTTP Basic Authentication for access control.

* Sender Module (sender.py): The sender uses a GUI window to select a file and set a mandatory username and password. Upon clicking "Start Sharing," the application starts a local HTTP server configured to require these credentials. It then connects to the Tor daemon and programmatically creates an ephemeral hidden service mapped to the local server. The unique .onion address is displayed in the GUI, ready for the user to copy and send to the receiver along with the credentials.
* Receiver Module (receiver.py): The receiver uses a GUI window, providing the .onion address, username, and password. The GUI makes an HTTP request through a local Tor SOCKS proxy. This request includes the authentication credentials, which are sent to the sender's hidden service. If the credentials are correct, the file is downloaded directly. This two-factor approach (knowing the secret address and having the correct password) ensures both anonymity and authorized access.

**Hardware Requirements**

* Computer: A standard desktop or laptop.
* Processor: Any modern dual-core CPU.
* RAM: Minimum 2 GB.
* Storage: 100 MB of free disk space.
* Network: An active internet connection.

**Software Requirements**

* Operating System: Windows 10/11, macOS 10.15+, or a modern Linux distribution (e.g., Ubuntu 20.04+).
* Core Service: Tor Daemon (version 0.4.5+) must be installed and running, with the Control Port (9051) and SOCKS Port (9050) enabled.
* Programming Environment: Python 3.8 or newer (with tkinter support).
* Python Libraries:
  + stem: For programmatic control of the Tor daemon.
  + requests[socks]: For making HTTP requests through the Tor SOCKS proxy.
* Tools: A code editor (e.g., VS Code) and a command-line terminal.

