**CIPHERNET MESSENGER**

**A SECURE P2P MESSENGER UTILIZING A HYBRID RSA- BLOWFISH ENCRYPTION PROTOCOL WITH SHA-256 BASED INTEGRITY CONTROLS**

### Project Report

*Submitted in partial fulfillment of the requirements for the award of credits for*

**Information Security (ITY15)**

*In*

Information Technology

*Of*

Puducherry Technological University

*By*

**ARJUN CHRISTOPHER** [2201112007]

*Under the Guidance of*

**Dr. G. Santhi B.E., M.E., Ph.D.,**

**Professor & Head, Department of Information Technology**

****

**DEPARTMENT OF INFORMATION TECHNOLOGY PUDUCHERRY TECHNOLOGICAL UNIVERSITY PUDUCHERRY – 605 014**

**SEPTEMBER 2025**

**Contents**

1. [Introduction 1](#_TOC_250011)
2. [Problem Statement 1](#_TOC_250010)
3. [Proposed Solution 2](#_TOC_250009)
4. [Methodology 3](#_TOC_250008)
   1. [Algorithms Used 3](#_TOC_250007)
   2. [Security Standards Used 4](#_TOC_250006)
   3. [Protocols Used 4](#_TOC_250005)
5. [Technology Used 5](#_TOC_250004)
6. [Result 6](#_TOC_250003)
7. [Appendices 6](#_TOC_250002)
   1. [Glossary 6](#_TOC_250001)
   2. [References 7](#_TOC_250000)

# Introduction

In an era where digital communication is fundamental, the privacy and security of our conversations have become paramount. The prevailing communication model relies on centralized servers, where platforms act as intermediaries for all messages and files. While convenient, this architecture presents significant security risks, including data breaches on an industrial scale, potential for surveillance, and the creation of single points of failure. User data, and more importantly metadata, becomes a commodity, logged and often analyzed by the service provider.

This project, **CipherNet Messenger**, directly challenges this paradigm by designing and implementing a secure, decentralized, peer-to-peer (P2P) messaging application. The core objective is to create a communication system where user privacy is cryptographically guaranteed, not merely promised by a third party. By leveraging a direct P2P architecture for communication, CipherNet eliminates the central server as a point of vulnerability and surveillance.

The application's security is founded on a robust **hybrid cryptosystem**, combining the strengths of asymmetric (RSA) and symmetric (Blowfish) encryption to provide strong end-to- end encryption (E2EE). Furthermore, all communications are protected by state-of-the-art integrity controls using the SHA-256 standard. This report details the design, methodology, implementation, and results of this project, demonstrating a practical and secure alternative to conventional messaging platforms.

# Problem Statement

The reliance on centralized architectures for digital messaging has given rise to several critical vulnerabilities that compromise user security and privacy.

* **Lack of Confidentiality and Privacy:** In many centralized systems, user data is stored on company servers, sometimes with inadequate encryption. This makes it a prime target for external attackers (data breaches) and accessible to insiders or government agencies. Even with E2EE, the servers still collect vast amounts of sensitive **metadata** (who is communicating with whom, when, and from where), which itself is a major privacy leak.
* **Single Point of Failure and Control:** A centralized server represents a single point of failure. If the server experiences an outage, the entire communication network is disabled. Furthermore, users are subject to the policies and control of the service provider, which can lead to censorship or de-platforming.
* **Message and Data Integrity Risks:** Without robust integrity checks, messages and files sent over a network can be intercepted and altered by a Man-in-the-Middle (MitM) attacker before reaching the intended recipient, leading to the spread of misinformation or malicious content.

The primary objective of this project is to address these issues by engineering a system that provides **provable confidentiality, integrity, and availability** for user communications, free from the constraints and vulnerabilities of a centralized model.

# Proposed Solution

The proposed solution is **CipherNet Messenger**, an application that establishes secure, direct communication channels between users on a local network. Its architecture is designed to be inherently private and resilient.

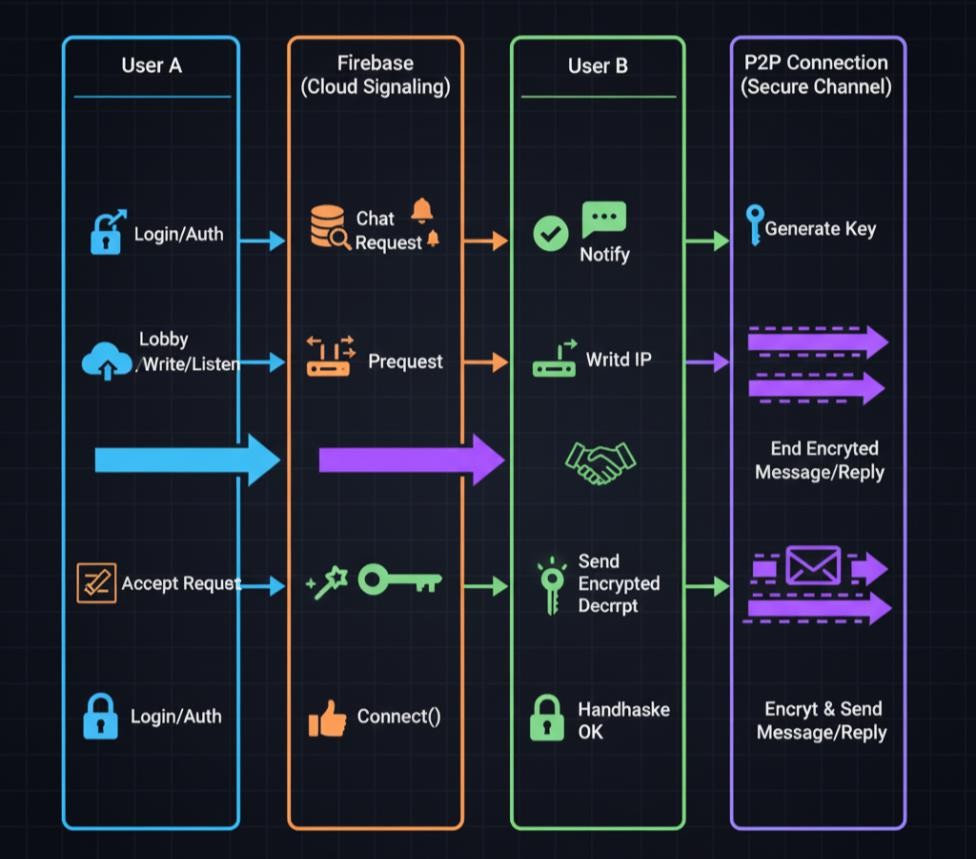
### Architectural Design:

CipherNet employs a hybrid model that uses a cloud service (Google Firebase) strictly as a lightweight Authentication and Signaling Server. Its roles are limited to:

1. Securely authenticating users via email and password.
2. Serving as a temporary, private "address book" for users to discover each other and securely exchange initial connection details (IP addresses).

Crucially, **no messages, files, or sensitive metadata ever pass through or are stored on the Firebase server.** All communication is handled directly peer-to-peer over an encrypted TCP socket.

### Process Flow:

****

### Key Features:

* + **Decentralized Communication:** By routing all messages directly between peers, the project eliminates the single point of failure and prevents any central entity from logging communication patterns.
  + **Strong End-to-End Encryption:** A hybrid encryption protocol is implemented. An **RSA-2048** key pair is used to securely and automatically negotiate a unique, one-time **Blowfish** session key. This session key is then used for fast, efficient encryption of all messages and files.
  + **Private Peer Discovery:** To avoid exposing user IP addresses publicly, CipherNet uses a "private relay" pattern. A chat request is sent through a private channel in Firebase, and only upon acceptance is the recipient's IP address shared back through another private, temporary channel, visible only to the two consenting parties.
  + **Verifiable Data Integrity:** All text messages are authenticated using **HMAC-SHA256**, while all files are verified using a full-file **SHA-256** hash. This guarantees that received data is identical to the sent data and has not been tampered with.
  + **Secure File Handling:** The solution includes a robust file-sharing mechanism that not only encrypts the file in transit but also validates the file type and sanitizes images upon receipt to protect against malicious payloads.

# Methodology

The security of CipherNet Messenger is built upon a combination of established cryptographic algorithms, standards, and custom-designed communication protocols.

## Algorithms Used

* + - **RSA (Rivest-Shamir-Adleman):** An asymmetric encryption algorithm used for the secure exchange of the session key. Each user possesses a 2048-bit RSA key pair (e,N) and (d,N). To initiate a chat, the sender encrypts the newly generated Blowfish session key (K\_session) with the recipient's public key (e\_R,N\_R). Only the recipient can decrypt this using their private key (d\_R,N\_R).
    - **Blowfish:** A fast and secure symmetric block cipher. It is used with a variable-length key (up to 448 bits) and operates on 64-bit blocks. Its role is to provide high-speed confidentiality for the bulk of the data transfer, including all text messages and file chunks, using the securely exchanged K\_session.
    - **SHA-256 (Secure Hash Algorithm 256-bit):** A cryptographic hash function that produces a fixed-size 256-bit hash value. Its properties of pre-image resistance and collision resistance are used for two purposes:
      1. As the underlying hash function for HMAC to verify message integrity.
      2. To generate a unique fingerprint of an entire file to verify its integrity after transfer.

## Security Standards Used

* + - **HMAC-SHA256 (Hash-based Message Authentication Code):** A standard for simultaneously verifying the **integrity** and **authenticity** of a message. It is a keyed hash function, meaning it combines the message content with a secret key (K\_session) to produce a tag. An attacker cannot forge a valid tag without knowing the secret key. The formula is represented as: HMAC(K,m)=H((K′⊕opad)∥H((K′⊕ipad)∥m)).
    - **PKCS#1 OAEP Padding:** A padding scheme used with RSA encryption. OAEP (Optimal Asymmetric Encryption Padding) adds randomness to the plaintext before encryption, which prevents several types of attacks and ensures that encrypting the same session key multiple times will result in different ciphertexts.

## Protocols Used

1. **Hybrid Encryption Protocol:** This is the core cryptographic protocol.
   * User A generates a random 128-bit session key, K\_session.
   * User A encrypts K\_session using User B's public RSA key: C=RSA\_encrypt(K\_session,PubKey\_B).
   * User A sends C to User B.
   * User B decrypts C using their private RSA key: K\_session=RSA\_decrypt(C,PrivKey\_B).
   * All further messages, M, are encrypted using Blowfish: Ciphertext=Blowfish\_encrypt(M,K\_session).

### Secure Signaling & Peer Discovery Protocol:

* + Upon login, users publish their public key and username to a public /lobby/ on Firebase.
  + To initiate a chat, User A writes a request to a private path only User B can read:

/requests/{UserID\_B}/.

* + If User B accepts, they create a new private path /chats/{ChatID}/ and write their local IP address to it.
  + User A is notified and reads the IP from this private path. The IP is never publicly visible.

### Secure File Transfer Protocol:

* + The sender calculates the file's SHA-256 hash and sends a control message with the filename, filesize, and hash.
  + The receiver acknowledges and prepares for the transfer.
  + The sender reads the file in 4096-byte chunks, encrypts each with Blowfish, and sends them.
  + The receiver decrypts and reassembles the file. Finally, it calculates the hash of the received file and compares it with the sender's original hash to ensure integrity.

# Technology Used

* **Programming Language:** Python 3.9+

### Graphical User Interface (GUI):

* + **CustomTkinter:** A modern Python UI library based on Tkinter, used for creating the application's visual components.

### Cryptography:

* + **pycryptodome:** A comprehensive cryptographic library providing implementations of RSA, Blowfish, SHA-256, and HMAC.

### Backend as a Service (BaaS):

* + **Google Firebase:** Used for both the Authentication service (for secure user management) and the Realtime Database (as the signaling server).

### Networking:

* + **socket:** Standard Python library for low-level network communication (TCP sockets).
  + **threading:** Standard Python library to manage concurrent operations, ensuring the UI remains responsive while handling network traffic.

### Notifications:

* + **desktop-notifier:** A cross-platform library for sending native desktop push notifications.

### File Handling:

* + **Pillow (PIL Fork):** Used for robust image file validation and sanitization.

# Result

The CipherNet Messenger project was successfully completed, resulting in a functional prototype that meets all the specified objectives. The application serves as a robust proof-of- concept for secure, decentralized communication on local networks.

### Key Achievements:

* A fully functional P2P messaging application with a clean, intuitive graphical user interface was developed.
* The hybrid RSA-Blowfish cryptosystem was successfully implemented, providing strong end-to-end encryption for all user communications.
* The secure signaling protocol effectively facilitates peer discovery and connection without exposing user IP addresses in a public lobby, thus protecting user metadata.
* Message and file integrity is guaranteed through the correct implementation of HMAC- SHA256 and full-file hashing, respectively.
* The application demonstrates high performance with negligible latency in message delivery, attributable to the software efficiency of the Blowfish algorithm and the direct nature of P2P connections.

The final application is a testament to the feasibility of building highly secure, private, and user-centric communication tools that do not rely on a centralized trust model.

# Appendices

## Glossary

* + - **Asymmetric Encryption:** A cryptographic system that uses a pair of keys: a public key (which may be disseminated widely) and a private key (which is known only to the owner). RSA is an example.
    - **Blowfish:** A fast, symmetric block cipher that operates on 64-bit blocks of data and can have a key length of up to 448 bits.
    - **End-to-End Encryption (E2EE):** A system of communication where only the communicating users can read the messages.
    - **HMAC:** A mechanism for calculating a message authentication code involving a cryptographic hash function in combination with a secret key.
    - **Hybrid Cryptosystem:** A system that combines the convenience of asymmetric (public-key) cryptography with the efficiency of symmetric cryptography.
    - **Peer-to-Peer (P2P):** A decentralized network architecture in which participants make a portion of their resources directly available to other network participants, without the need for central coordination instances.
    - **RSA:** A public-key cryptosystem that is widely used for secure data transmission.
    - **Signaling:** The process of exchanging information between peers to coordinate communication and establish a connection.
    - **Symmetric Encryption:** A cryptographic system that uses the same key for both encryption of plaintext and decryption of ciphertext. Blowfish is an example.

## References

1. Blowfish Algorithm: Schneier, B. (1993). Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). *Fast Software Encryption, Cambridge Security Workshop Proceedings*. Link - <https://www.schneier.com/academic/blowfish/>
2. RSA Cryptosystem: Rivest, R., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*. Link - <https://people.csail.mit.edu/rivest/Rsapaper.pdf>
3. HMAC Standard: RFC 2104 - HMAC: Keyed-Hashing for Message Authentication. Link - <https://tools.ietf.org/html/rfc2104>
4. pycryptodome Library Documentation. Link - <https://www.pycryptodome.org/en/latest/>
5. Google Firebase Documentation. Link - <https://firebase.google.com/docs>
6. CustomTkinter Library Documentation. Link - <https://github.com/TomSchimansky/CustomTkinter>