In this project, I developed and simulated a secure two-party communication system using hybrid encryption (RSA for key exchange and AES for message encryption). As part of the system, I introduced three key attack simulations to understand real-world vulnerabilities in messaging programs: eavesdropping, integrity attacks, and denial-of-service (DoS) attacks. Each represents a unique threat to secure communications.

Eavesdropping

Eavesdropping simulates a passive attacker listening in on the communication channel to view messages being exchanged between User A and User B. This attacker can see encrypted content but does not alter or redirect it. If messages were sent in plaintext, an eavesdropper could read their contents. Even with encryption, poor key management or weak encryption schemes can make messages vulnerable to brute-force or cryptanalysis attacks.

Mitigation Strategies

- Use strong, modern encryption standards (e.g., AES-256, RSA-2048+).

- Regularly rotate symmetric keys (key refresh).

- Implement secure key exchange protocols (e.g., Diffie-Hellman with authentication or TLS).

- Monitor traffic for anomalies or unauthorized access patterns.

Integrity Attack

An attacker injects fake or tampered messages into the communication stream. This simulates a scenario where an adversary tries to alter the content or impersonate a party. Without strong authentication or message integrity checks, systems may accept forged messages as legitimate, leading to misinformation or unauthorized actions. In my program if the eavesdrop attack is active when the users exchange their symmetric keys the attacker will grab both and be able to send encrypted messages to both users as either user.

Mitigation Strategies

- Use authenticated encryption modes like AES-GCM or AES-EAX, which validate data integrity via cryptographic tags.

- Verify the authenticity of senders using digital signatures.

- Log suspicious behavior or failed decryption attempts as potential signs of attack.

Denial-of-Service (DoS) Attack

The attacker floods the message queue with meaningless or malicious messages, overwhelming the system and potentially preventing legitimate messages from being processed. Queue flooding can consume resources (CPU/memory) and delay or block valid communication.

Mitigation Strategies

- Implement rate limiting or CAPTCHA-style challenges before message acceptance.

- Set maximum queue size and discard excess traffic or deprioritize untrusted sources.

- Use thread-safe and resilient queue management with timeouts and backpressure mechanisms.

Conclusion

Through these simulations, my project demonstrates critical vulnerabilities that can arise even in encrypted systems. Proper cryptographic design is only effective when paired with sound implementation practices, secure key management, and vigilant monitoring. By understanding and addressing these common attack vectors, we can build more robust and trustworthy secure communication systems.