Secure Communication

M. NIKHITHA REDDY - 21CSB0B34
PRIYANKA KOTA - 21CSB0B44

Introduction

The assignment addresses the critical challenge of preserving privacy in communication channels, aiming to maintain the confidentiality and integrity of transmitted data. In an era of digital communication dominance, ensuring security is crucial, given the exposure of data to threats like eavesdropping and tampering. There's a pressing need for robust mechanisms to safeguard communication privacy, protecting data integrity and confidentiality.

Importance of the Problem:
Ensuring privacy in communication is vital, considering the reliance on digital platforms for personal, professional, and governmental interactions. Breaches of privacy can lead to financial loss, reputational damage, and compromise national security. Existing solutions focus on encryption and hashing techniques to secure communication channels, but challenges remain in efficiently detecting and thwarting tampering attempts, necessitating innovation in privacy-preserving communication methodologies.

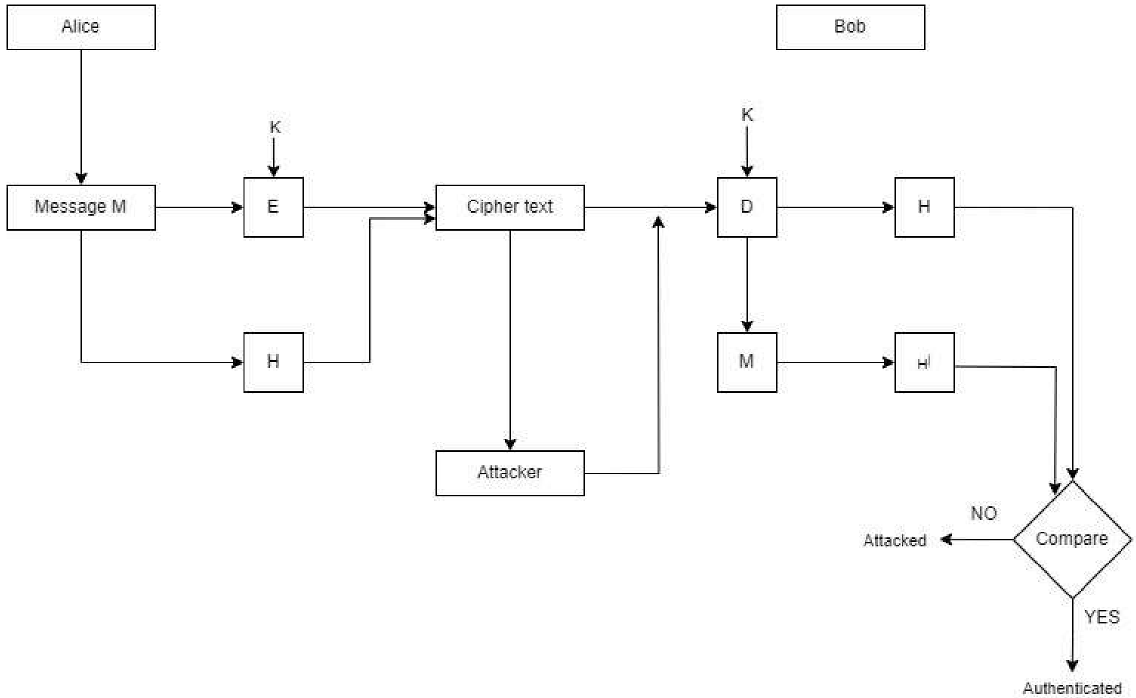
Existing Solutions:
Current approaches to privacy-preserving communication rely on encryption and hashing methods to restrict access to authorized parties and verify data integrity. While effective against unauthorized access and tampering, these methods may not fully address sophisticated attacks targeting confidentiality and integrity.

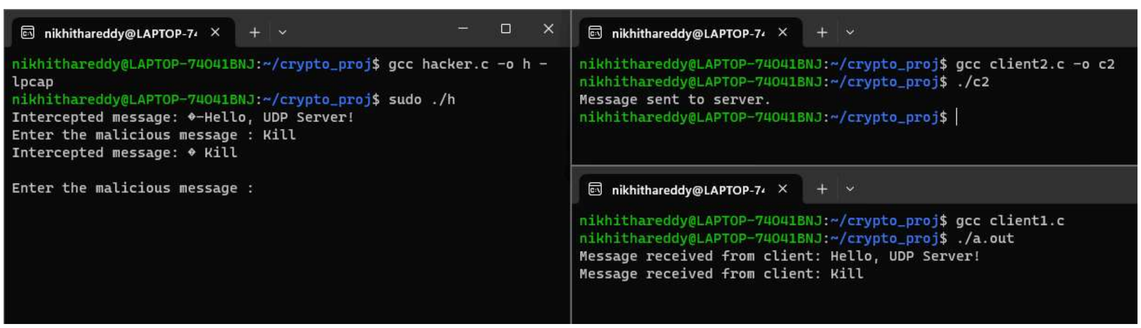
Key Contributions of the Project:
The project aims to enhance privacy-preserving communication by integrating encryption, hashing, and tamper detection mechanisms. By leveraging cryptographic primitives and intelligent detection algorithms, it seeks to detect and mitigate tampering attempts, ensuring data integrity and confidentiality. The project will explore novel approaches to improve the efficiency and scalability of privacy-preserving communication protocols, facilitating seamless integration into existing frameworks.

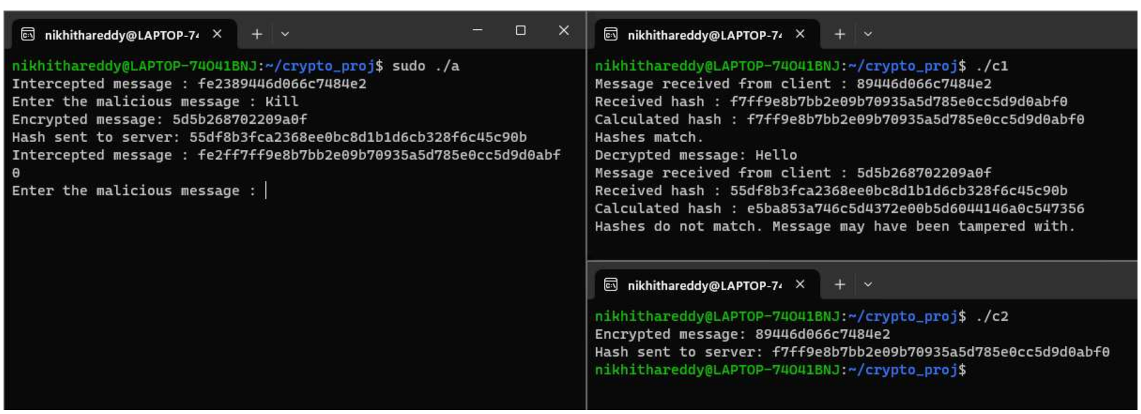
Objectives

- ❖ Develop an innovative privacy-preserving communication protocol combining encryption, hashing, and tamper detection.
- ❖ Evaluate the protocol's effectiveness and efficiency through rigorous testing and analysis, aiming for real-world applicability.

Implementation and Results analysis:







Conclusion

In conclusion, the project underscores the importance of privacy-preserving communication in safeguarding data integrity and confidentiality. By integrating encryption, hashing, and tamper detection mechanisms, the proposed protocol offers robust protection against unauthorized access and tampering attempts. Through rigorous testing and analysis, the project aims to validate the efficacy and viability of the protocol for real-world applications, addressing critical challenges in modern communication security.

Learning outcomes

- ➢ Enhanced understanding of cryptographic techniques for privacy preservation.
- ➢ Proficiency in designing and implementing secure communication protocols.
- ➢ Insight into the complexities of balancing security and efficiency in real-world applications.

Source code:

Receiver side:

```c
    unsigned char key[8] = "12345678";
    unsigned char decrypted_message[MAXLINE] = {0};
    decryptDES(buffer, key, decrypted_message);


    // Calculate hash of received message
    unsigned char hash[SHA_DIGEST_LENGTH];
    calculateHash(decrypted_message, hash);
    printf("Calculated hash : ");
    for (int i = 0; i < SHA_DIGEST_LENGTH; i++)
        printf("%02x", hash[i]);
        printf("\n");

    // Compare received hash with calculated hash
    if (memcmp(hash, received_hash, SHA_DIGEST_LENGTH) == 0) {
        printf("Hashes match.\n");
        // Print decrypted message
        printf("Decrypted message: %s\n", decrypted_message);
    } else {
        printf("Hashes do not match. Message may have been tampered with.\n");
    }
```

Sender side:

```c
unsigned char key[8] = "12345678";
unsigned char encrypted_message[MAXLINE] = {0};
encryptDES((unsigned char *)message, key, encrypted_message);

// Print the encrypted message
printf("Encrypted message: ");
for (int i = 0; i < strlen((char *)encrypted_message); i++)
    printf("%02x", encrypted_message[i]);
    printf("\n");

// Calculate hash of encrypted message
unsigned char hash[SHA_DIGEST_LENGTH];
calculateHash(message, hash);

// Send encrypted message to server
sendto(sockfd, encrypted_message, strlen((char *)encrypted_message), 0,
(struct sockaddr *)&servaddr, sizeof(servaddr));

// Send hash to server
sendto(sockfd, hash, SHA_DIGEST_LENGTH, 0, (struct sockaddr *)&servaddr,
sizeof(servaddr));
```

Attacker side:

```c
void packet_handler(u_char *args, const struct pcap_pkthdr *header, const u_char *packet) {

    const u_char *payload;
    int payload_len;

    // Ethernet header is 14 bytes, IP header is 20 bytes, UDP header is 8 bytes
    payload = packet + SIZE_ETHERNET + 20 + 8; // Adjust the offset to skip headers
    payload_len = header->len - (SIZE_ETHERNET + 20 + 8); // Calculate payload length

    printf("Intercepted message : ");
    for (int i = 0; i < payload_len; i++)
        printf("%02x", payload[i]);
        printf("\n");

    char mal_message[100];
    printf("Enter the malicious message : ");
    fgets(mal_message,sizeof(mal_message),stdin);
    if(mal_message[0]=='#') return;

    sendMaliciousMessage(mal_message);
}
```