Sardar Vallbhbhai National Institute of Technology Surat

Introduction to Security & Cryptography – Lab Project

SECURE CHAT APPLICATION – PRIVATEMSG-SECURE-CHAT

Deep Das U23AI052
Yash Ingle U23AI062

Motivation & Problem Statement

- **Motivation**: In today’s digital era, secure communication is essential to protect sensitive information from unauthorized access. Chat applications, in particular, are prime targets for attacks such as eavesdropping and impersonation.
- **Problem**: Conventional chat apps often lack strong end-to-end encryption and proper key management mechanisms.
- **Objective:** To design and implement a client-server chat application with AES-CBC encryption, secure key management, and optional digital signatures.
- **Outcome:** Demonstrate confidentiality, integrity, and authenticity in real-time messaging.

Project Overview & Features

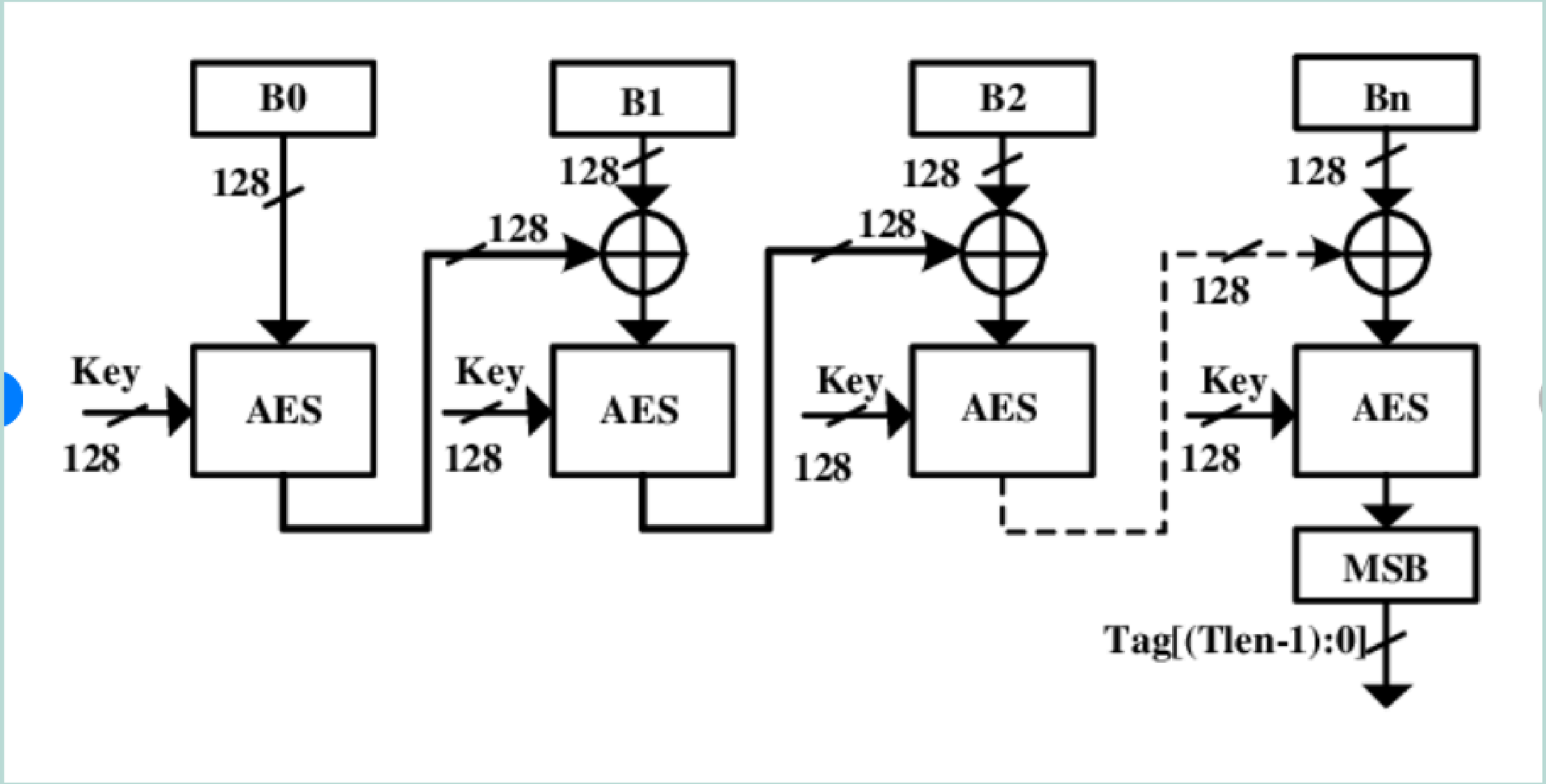
- **Title: Private Messaging Secure App**
- **Purpose**: Educational project showcasing modern cryptographic principles through a secure chat platform.

- **Key Features:**
- Single-Process CLI Application supporting multiple users.
- AES-CBC based End-to-End Encryption with password-derived keys.
- Argon2 key derivation for secure password hashing.
- Ed25519 Digital Signatures for authentication and non-repudiation.
- Interactive Command-Line Interface (CLI) with persistent storage.
- Technologies Used: C++, libsodium, OpenSSL, JSON file-based persistence.

Architecture Overview:

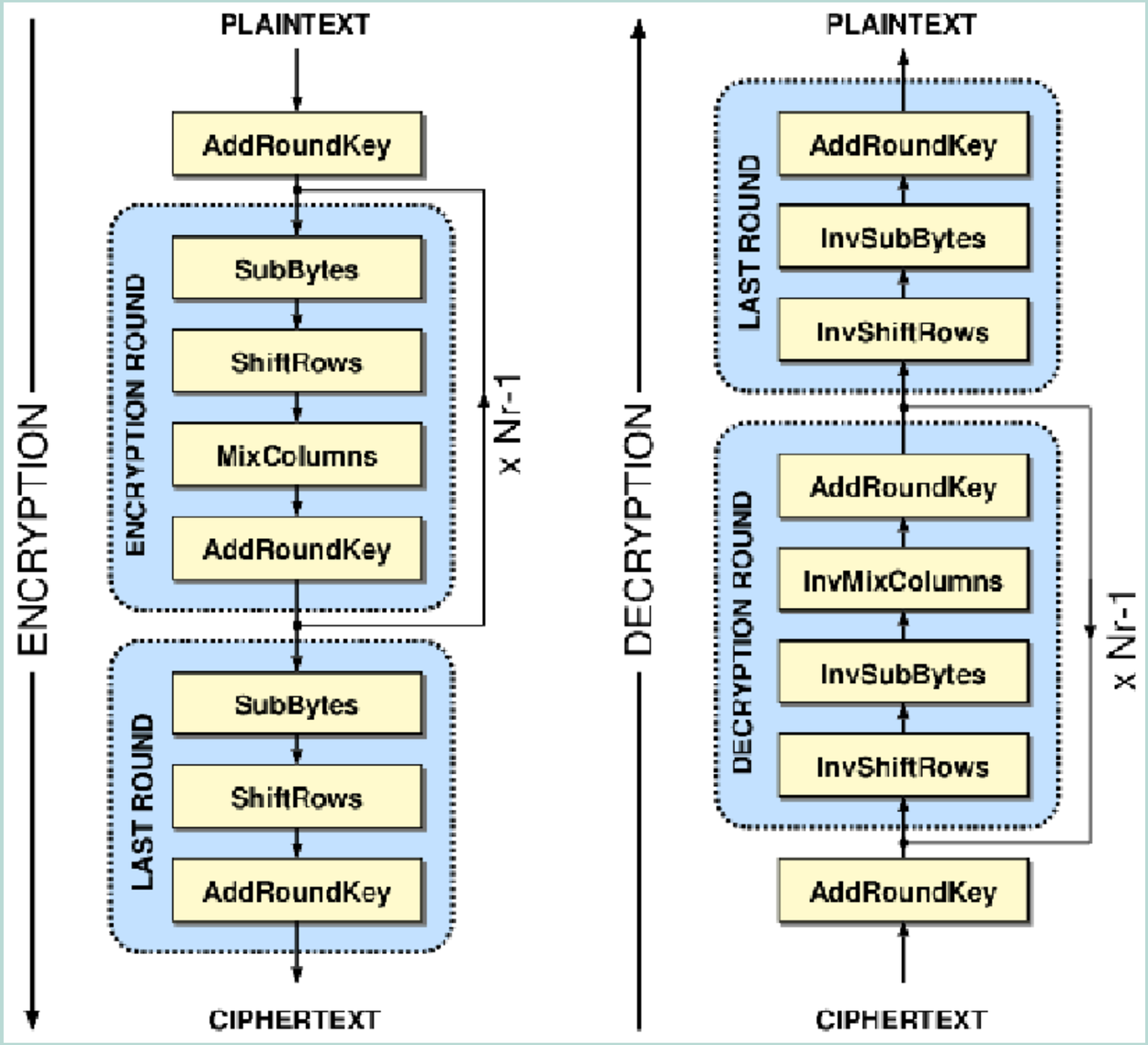
- **Local Multi-User Model:** Single application manages multiple user accounts with persistent storage.
- **Application Role:** Handles user registration, AES-CBC encryption/decryption, and secure message routing between local users.
- **User Role:** Registers with password-based authentication, sends AES-encrypted messages, and receives messages in personal inbox.
- **Storage:** JSON-based persistent storage with encrypted private keys and message inboxes.

Protocols Used: AES-256-CBC for message encryption, Argon2 for password hashing, Ed25519 for digital signatures.



AES-CBC encryption process showing block chaining and key reuse for secure message encryption.

ABOUT AES



SubBytes = SBox

ShiftRows = cyclic shift, n-rows= n-byte shift

MixColumns = Each column is treated as a polynomial over GF(2^8) and multiplied (modulo an irreducible polynomial) by a fixed matrix.

AddRoundKey = Each byte of the state is XORed with the corresponding byte of the round key.

10 rounds → AES-128
12 rounds → AES-192
14 rounds → AES-256

Key Management & Secure Exchange

- **Private Key Protection:** All sensitive keys encrypted at rest using password-derived encryption.
- **Key Derivation:** Argon2 algorithm derives encryption keys from user passwords with random salts.
- **Dual Encryption Approach:**
  - AES-256-CBC keys for message confidentiality (shared per user)
  - Ed25519 keypairs for digital signatures (public/private per user)
  - Password-based encryption protects private keys when stored
- **Storage Security:**
  - Private signing keys encrypted with user password before JSON storage
  - AES encryption keys encrypted with user password before JSON storage
  - Only plaintext data stored: usernames, password hashes, public keys, encrypted messages

User Experience: Seamless key management - users only need to remember their login password.

Authentication & Integrity (Digital Signatures)

- **Authentication:** Ed25519 digital signatures ensure message sender legitimacy and non-repudiation.
- **Integrity:** SHA-256 message hashing ensures content has not been tampered with during storage.
- **Digital Signature Mechanism:**
  a. Sender signs the original plaintext message using Ed25519 private key.
  b. Signature is stored alongside the encrypted message in recipient's inbox.
  c. Receiver verifies signature using sender's Ed25519 public key after decryption.

Implementation & Demonstration

- **Development Stack:** C++, OpenSSL, Sockets.

- **Libraries:**
- **libsodium:** Key derivation (Argon2), digital signatures (Ed25519), SHA-256 hashing
- **OpenSSL**: AES-CBC encryption/decryption
- **nlohmann::json:** JSON file handling (user data, inbox)
- **Environment:** CLI (Command Line Interface)

Conclusion & Future Enhancements

Conclusion:

- Successfully implemented a secure client-server chat system with AES-CBC encryption.
- Demonstrated understanding of symmetric encryption, key exchange, and authentication mechanisms.
- Reinforced key principles of cryptography: Confidentiality, Integrity, and Authenticity.

Future Work:

- Implement AES-GCM for authenticated encryption.
- Integrate Diffie-Hellman for perfect forward secrecy.
- Add Graphical Interface (GUI).
- Include logging & intrusion detection modules.
- Extend to multi-server federated architecture.

Thank You