Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Assignment 1

Blockchain Technology

Submitted by

Bijay Sapkota(43)

Submitted to

Suresh Gautam

Department of Computer Science and Engineering

Submission Date: 14 Mar 2025

Chapter 1: Introduction

In this assignment, a secure communication system between a client and a server using AES and RSA encryption is implemented. The primary objective was to ensure the confidentiality and integrity of messages exchanged between the two parties. The implementation securely encrypts messages before transmission and decrypts them upon receipt.

Chapter 2: Implementation Details

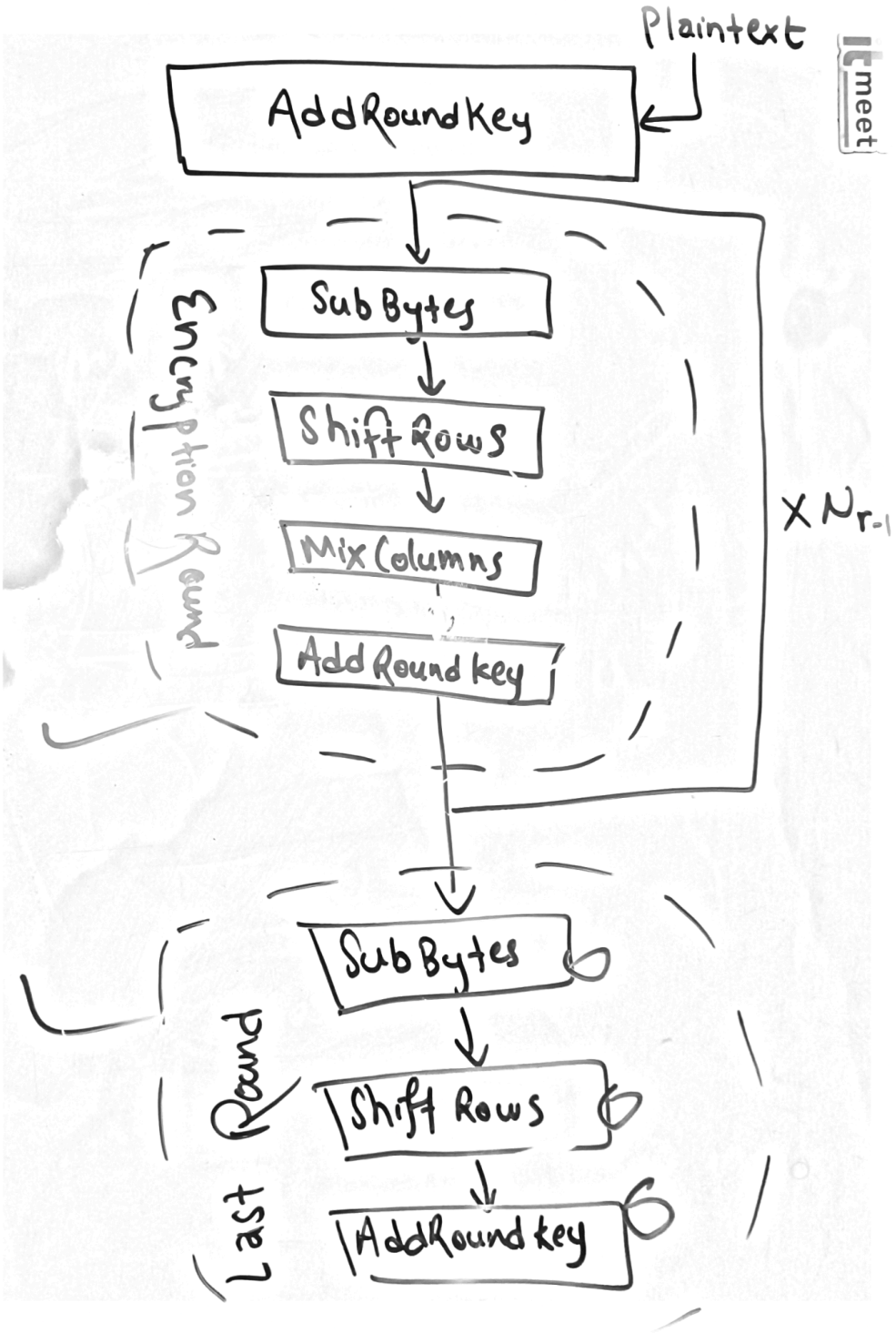
AES (Advanced Encryption Standard): Used for encrypting the messages exchanged between the client and the server.

Below is an Implementation Overview of AES Implementation. The details of each subsection are provided on a notebook “test.ipynb” on github repository.

- Firsty a given message is broken in 16 byte blocks and AES is applied in each blocks.
- For hexadecimal representation of Each blocks, a non linear A S-Box substitution is applied.
- Then the message is converted into 4*4 matrix, on which rows are circularly shifted.
- It is followed by Column Mixing Operation and a round key is X-OR ed with the state of matrix after above operations.

Above Operations are performed N-1 times and for last round, three operations (without Mix Column Operations are performed).

Which encrypts given message and inverse of this process using same key can decrypt the message as well.


Plaintext
AddRoundKey
Encryption Round
SubBytes
ShiftRows
MixColumns
AddRoundkey
X N_{r-1}
Last Round
SubBytes
ShiftRows
AddRoundkey

RSA (Rivest-Shamir-Adleman): Used for secure key exchange between the client and the server.

Below is an implementation overview of the RSA algorithm.

GCD Function (`gcd(a, b)`)

- Finds the greatest common divisor of two numbers.
- Helps ensure that the encryption key is valid.

Modular Inverse (`mod_inverse(a, m)`)

- Finds a number that, when multiplied by another, gives 1 after division by a given value.
- Used to compute the private key from the public key.

Key Generation (`generate_keypair(p, q)`)

- Multiplies two prime numbers to create a key foundation.
- Calculates a value needed for encryption key selection.
- Chooses a random encryption key that works with the calculated value.
- Computes the corresponding decryption key.
- Returns both encryption (public) and decryption (private) keys.

Encryption (`encrypt_key(public_key, plaintext)`)

- Converts each character of a message into a number.
- Uses the public key to transform the numbers into encrypted values.
- Outputs the encrypted message as a list of numbers.

Decryption (`decrypt_key(private_key, ciphertext)`)

- Takes the encrypted numbers and converts them back into their original values using the private key.
- Transforms the numbers back into readable characters.
- Returns the original message.

Key Generation and Exchange

- AES keys are randomly generated by generating pseudo random 16 byte hex keys.
- The AES key is encrypted using the RSA public key and securely sent to the recipient.
- The recipient decrypts the AES key using their RSA private key and subsequently uses it for symmetric encryption and decryption.

Chapter 3: Project Structure

- **KeyExchange/**
  - `RSA.py`: Handles RSA key pair generation, encryption, and decryption for secure key exchange.
- **MessageEncryption/**
  - `AES_Implementation.py`: Implements AES encryption and decryption.
  - `AESHelper.py`: Provides helper functions for key handling and padding.
  - `Encrypt.py`: Encrypts messages using AES.
  - `Decrypt.py`: Decrypts received messages.
- **Messaging/**
  - `Client.py`: Implements the client-side logic for sending encrypted messages.
  - `Server.py`: Implements the server-side logic for receiving and decrypting messages.
- **TestCases/**
  - `Test_AES.py`: Unit tests for AES encryption and decryption.
  - `Test_RSA.py`: Unit tests for RSA encryption and decryption.
  - `test_server_client.py`: Integration tests for secure communication between client and server.

Chapter 4: Design Choices and Trade-offs

- **Choice of AES for Message Encryption:**
  - AES is a symmetric encryption algorithm, making it efficient for encrypting large amounts of data.
  - A trade-off is that both parties need to share the key securely, which is handled by RSA encryption.
- **Choice of RSA for Key Exchange:**
  - RSA enables secure transmission of AES keys without prior shared secrets.
  - RSA encryption is computationally expensive compared to AES but is used only for small key exchanges, keeping performance optimal.
- **Implementation in Python 3:**
  - Python provides robust cryptographic libraries, making the implementation easier and secure.
  - The choice of Python allows for cross-platform compatibility.

Chapter 5: Testing Strategy:

- **Unit Testing:** AES and RSA implementations were tested separately to ensure correctness.
- **Integration Testing:** The client-server communication was tested to verify encryption, decryption, and key exchange processes.

Chapter 6: Appendix

```
(venv) venvbijaysapkota@Bijays-Air new % python3 client.py
Enter your name: Bijay
Enter your message: Hello World
Server Response: {'response': 'Server received: 9e9e080d58f7a5906f62bbd3ea0ea63f
from 829ad1e1c544dc18ac9946e86b231876'}
```

Fig: standalone AES Encryption

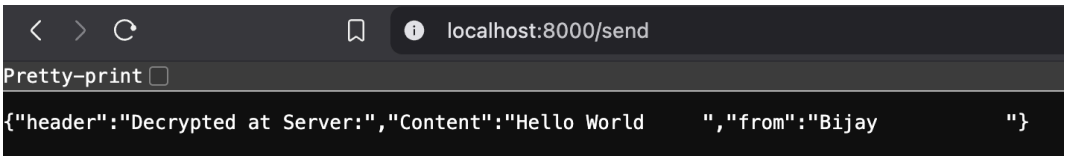


Fig: Decryption at Server Side

```
(venv) venvbijaysapkota@Bijays-MacBook-Air new % python3
client.py
AES Key Before Encryption:  db00ebff2bd30ff99aed0d35515d345e
Response Status Code: 200
Response JSON: {'public_key': [3079, 3233]}
Encrypted AES Key (Before Sending): [1936, 2778, 1146, 1146, 1729, 2778, 2077, 20
77, 3000, 2778, 1936, 2457, 1146, 2077, 2077, 2269, 2269, 1361, 1729, 1936, 1146,
 1936, 2457, 2173, 2173, 858, 2173, 1936, 2457, 1430, 2173, 1729]
Server Response: {'response': 'Server received and stored AES Key successfully.'}
Enter your name: Bijay
Enter your message: Hello
Server Response: {'response': 'Server received: b8aaf3b266f625328f9e85fab21a5265
from 3eeb8e79d5b49badd6ced2f75fe56265'}
(venv) venvbijaysapkota@Bijays-MacBook-Air new %
```

Fig: Encryption and Transmission of AES Key (Wrapped by RSA)

```
Received AES_Key: [1936, 2778, 1146, 1146, 1729, 2778, 2077, 2077, 3000,
 2778, 1936, 2457, 1146, 2077, 2077, 2269, 2269, 1361, 1729, 1936, 1146,
 1936, 2457, 2173, 2173, 858, 2173, 1936, 2457, 1430, 2173, 1729]
```

Fig: Received Encrypted AES Key at Server Side

Fig: The decrypted AES key successfully decrypted the subsequent message.