Symmetric & Asymmetric

Code:

```
#include <iostream>

#include <string>

#include <algorithm>

using namespace std;


string encryptCaesar(string text, int shift) {

    string result = "";

    for (char c : text) {

        if (isalpha(c)) {

            char base = islower(c) ? 'a' : 'A';

            result += (c - base + shift) % 26 + base;

        } else {

            result += c;

        }

    }


    return result;

}

string decryptCaesar(string text, int shift) {

    return encryptCaesar(text, 26 - shift);

}


string transposeRows(string text, int rows, int cols) {

    string transposedText = "";


    for (int col = 0; col < cols; ++col) {

        for (int row = 0; row < rows; ++row) {

            transposedText += text[row * cols + col];
        }
    }

    return transposedText;
}

string transposeColumns(string text, int rows, int cols) {
    string transposedText = "";

    for (int row = 0; row < rows; ++row) {
        for (int col = 0; col < cols; ++col) {
            transposedText += text[row + col * rows];
        }
    }

    return transposedText;
}

string transposeBoth(string text, int rows, int cols) {
    string transposedText = transposeRows(text, rows, cols);
    return transposeColumns(transposedText, cols, rows);
}

int power(int base, int exp, int mod) {
    if (exp == 0) return 1;
    long long temp = power(base, exp / 2, mod);
    long long result = (temp * temp) % mod;
    if (exp % 2 == 1) result = (result * base) % mod;
    return static_cast<int>((result + mod) % mod);
```

```
}

int diffieHellman(int base, int prime) {
    int privateA, privateB;
    cout << "Enter Alice's private key: ";
    cin >> privateA;
    cout << "Enter Bob's private key: ";
    cin >> privateB;

    int publicA = power(base, privateA, prime);
    int publicB = power(base, privateB, prime);

    int secretKeyA = power(publicB, privateA, prime);
    int secretKeyB = power(publicA, privateB, prime);

    if (secretKeyA == secretKeyB) {
        cout << "Shared Secret Key: " << secretKeyA << endl;
        return secretKeyA;
    } else {
        cout << "Error in key exchange!" << endl;
        return -1;
    }
}

string encryptVigenere(string plaintext, string keyword) {
    string ciphertext = "";
    int keyLength = keyword.length();
    int textLength = plaintext.length();
    char encryptedChar;
    for (int i = 0; i < textLength; ++i) {
        char plainChar = plaintext[i];
        char keyChar = keyword[i % keyLength];

        if (isalpha(plainChar)) {
            plainChar = toupper(plainChar);
            keyChar = toupper(keyChar);

            encryptedChar = 'A' + ((plainChar - 'A' + keyChar - 'A') % 26);
        } else {
            encryptedChar = plainChar;
        }

        ciphertext += encryptedChar;
    }

    return ciphertext;
}

void encryptionMenu() {
    int choice;
    string text, keyword;
    int shift, base, prime;

    while (true) {
        cout << "Choose encryption method:\n";
        cout << "1. Caesar Cipher\n";
        cout << "2. Transpose Cipher\n";
        cout << "3. Diffie-Hellman Key Exchange\n";
        cout << "4. Polyalphabetic Cipher\n";
        cout << "5. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        cin.ignore();

        switch (choice) {
            case 1:
                cout << "Enter the text to encrypt/decrypt: ";
                getline(cin, text);

                cout << "Enter the shift value: ";
                cin >> shift;

                text = encryptCaesar(text, shift);
                cout << "Processed text: " << text << endl;
                break;

            case 2:
                cout << "Enter the text to transpose: ";
                getline(cin, text);

                int transposeChoice;
                int rows, cols;

                cout << "Choose transpose method:\n";
                cout << "1. Transpose with Rows\n";
                cout << "2. Transpose with Columns\n";
        cout << "3. Transpose with Both Rows and Columns\n";
        cout << "Enter your choice: ";
        cin >> transposeChoice;
        cin.ignore();

        switch (transposeChoice) {
            case 1:
                cout << "Enter the number of rows: ";
                cin >> rows;
                cout << "Enter the number of columns: ";
                cin >> cols;
                text = transposeRows(text, rows, cols);
                cout << "Transposed text with rows: " << text << endl;
                break;

            case 2:
                cout << "Enter the number of rows: ";
                cin >> rows;
                cout << "Enter the number of columns: ";
                cin >> cols;
                text = transposeColumns(text, rows, cols);
                cout << "Transposed text with columns: " << text << endl;
                break;

            case 3:
                cout << "Enter the number of rows: ";
                cin >> rows;
                cout << "Enter the number of columns: ";
                cin >> cols;
```

```
      text = transposeBoth(text, rows, cols);
      cout << "Transposed text with both rows and columns: " << text << endl;
      break;

    default:
      cout << "Invalid choice. Please enter a valid option.\n";
  }
  break;

case 3:
  cout << "Enter a prime number (modulus): ";
  cin >> prime;
  cout << "Enter a primitive root modulo " << prime << ": ";
  cin >> base;
  diffieHellman(base, prime);
  break;

case 4:
  cout << "Enter the text to encrypt: ";
  cin.ignore();
  getline(cin, text);
  cout << "Enter the keyword: ";
  getline(cin, keyword);
  text = encryptVigenere(text, keyword);
  cout << "Encrypted Text: " << text << endl;
  break;

case 5:
  cout << "Exiting the program.\n";
```

```
                return;

            default:
                cout << "Invalid choice. Please enter a valid option.\n";
        }
    }
}

int main() {
    encryptionMenu();

    return 0;
}
```

Output:

```
Choose encryption method:
1. Caesar Cipher
2. Transpose Cipher
3. Diffie-Hellman Key Exchange
4. Polyalphabetic Cipher
5. Exit
Enter your choice: 1
Enter the text to encrypt/decrypt: Hello
Enter the shift value: 1
Processed text: Ifmmp

Choose encryption method:
1. Caesar Cipher
2. Transpose Cipher
3. Diffie-Hellman Key Exchange
4. Polyalphabetic Cipher
5. Exit
Enter your choice: 2
Enter the text to transpose: Hello
Choose transpose method:
1. Transpose with Rows
2. Transpose with Columns
3. Transpose with Both Rows and Columns
Enter your choice: 3
Enter the number of rows: 1
Enter the number of columns: 2
Transposed text with both rows and columns: He

Choose encryption method:
1. Caesar Cipher
```

```
2. Transpose Cipher
3. Diffie-Hellman Key Exchange
4. Polyalphabetic Cipher
5. Exit
Enter your choice: 3
Enter a prime number (modulus): 3
Enter a primitive root modulo 3: 17
Enter Alice's private key: 6
Enter Bob's private key: 15
Shared Secret Key: 1

Choose encryption method:
1. Caesar Cipher
2. Transpose Cipher
3. Diffie-Hellman Key Exchange
4. Polyalphabetic Cipher
5. Exit
Enter your choice: 4
Enter the text to encrypt: Hello
Enter the keyword: Hi
Encrypted Text: LTSW

Choose encryption method:
1. Caesar Cipher
2. Transpose Cipher
3. Diffie-Hellman Key Exchange
4. Polyalphabetic Cipher
5. Exit
Enter your choice: 5
Exiting the program.
```

Explaination:

- **Caesar Cipher (encryptCaesar and decryptCaesar functions):** Performs encryption and decryption using a shift-based substitution technique. It shifts each letter of the input text by a fixed number of positions in the alphabet.
- **Transpose Cipher (transposeRows, transposeColumns, transposeBoth functions):** Provides options to transpose the input text either by rows, columns, or both. It rearranges the characters in the input text based on the specified number of rows and columns.
- **Diffie-Hellman Key Exchange (diffieHellman function):** Implements the Diffie-Hellman key exchange algorithm, enabling two parties (Alice and Bob) to securely establish a shared secret key over an insecure channel using modular arithmetic.
- **Polyalphabetic Cipher (encryptVigenere function)**: Implements the Polyalphabetic substitution cipher that encrypts text using a keyword. It shifts each character of the input text by the corresponding character in the keyword