//仿射密码

#include <iostream>

#include <string>

#include <cstdlib>

#include <ctime>

#include <algorithm>

using namespace std;

// 计算最大公约数

int gcd(int a, int b) {

while (b != 0) {

int temp = b;

b = a % b;

a = temp;

}

return a;

}

// 计算模逆元

int modInverse(int a, int m) {

a = a % m;

for (int x = 1; x < m; x++) {

if ((a \* x) % m == 1) {

return x;

}

}

return -1; // 如果不存在逆元

}

// 生成仿射密码的密钥

void generateAffineKey(int& a, int& b, int& a\_inv) {

srand(time(0));

while (true) {

a = rand() % 26;

if (gcd(a, 26) == 1) {

break;

}

}

b = rand() % 26;

a\_inv = modInverse(a, 26);

}

// 仿射加密

string affineEncrypt(const string& plaintext, int a, int b) {

string ciphertext = "";

for (char ch : plaintext) {

if (isalpha(ch)) {

char base = isupper(ch) ? 'A' : 'a';

int P = ch - base;

int C = (a \* P + b) % 26;

ciphertext += (C + base);

}

else {

ciphertext += ch; // 非字母字符直接保留

}

}

return ciphertext;

}

// 仿射解密

string affineDecrypt(const string& ciphertext, int a\_inv, int b) {

string plaintext = "";

for (char ch : ciphertext) {

if (isalpha(ch)) {

char base = isupper(ch) ? 'A' : 'a';

int C = ch - base;

int P = (a\_inv \* (C - b + 26)) % 26; // 避免负数

plaintext += (P + base);

}

else {

plaintext += ch; // 非字母字符直接保留

}

}

return plaintext;

}

int main() {

int a, b, a\_inv;

generateAffineKey(a, b, a\_inv);

cout << "生成的密钥: a = " << a << ", b = " << b << ", a的逆元 = " << a\_inv << endl;

string plaintext = "Hello, World!";

string ciphertext = affineEncrypt(plaintext, a, b);

string decryptedText = affineDecrypt(ciphertext, a\_inv, b);

cout << "明文: " << plaintext << endl;

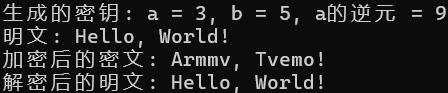
cout << "加密后的密文: " << ciphertext << endl;

cout << "解密后的明文: " << decryptedText << endl;

return 0;

}

//运行示意图



//单表代换密码

#include <iostream>

#include <string>

#include <algorithm>

#include <ctime>

#include <cstdlib>

using namespace std;

// 生成随机单表代换密钥

string generateSubstitutionKey() {

string key = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

srand(time(0));

random\_shuffle(key.begin(), key.end());

return key;

}

// 单表代换加密

string substitutionEncrypt(const string& plaintext, const string& key) {

string ciphertext = "";

for (char ch : plaintext) {

if (isalpha(ch)) {

char base = isupper(ch) ? 'A' : 'a';

int index = toupper(ch) - 'A';

ciphertext += (isupper(ch) ? key[index] : tolower(key[index]));

}

else {

ciphertext += ch; // 非字母字符直接保留

}

}

return ciphertext;

}

// 单表代换解密

string substitutionDecrypt(const string& ciphertext, const string& key) {

string plaintext = "";

for (char ch : ciphertext) {

if (isalpha(ch)) {

char base = isupper(ch) ? 'A' : 'a';

char upperCh = toupper(ch);

size\_t index = key.find(upperCh);

plaintext += (isupper(ch) ? ('A' + index) : ('a' + index));

}

else {

plaintext += ch; // 非字母字符直接保留

}

}

return plaintext;

}

int main() {

string key = generateSubstitutionKey();

cout << "生成的密钥: " << key << endl;

string plaintext = "Hello, World!";

string ciphertext = substitutionEncrypt(plaintext, key);

string decryptedText = substitutionDecrypt(ciphertext, key);

cout << "明文: " << plaintext << endl;

cout << "加密后的密文: " << ciphertext << endl;

cout << "解密后的明文: " << decryptedText << endl;

return 0;

}

//运行示意图

