Презентация по лабораторной работе №8

НКНбд-01-21

Юсупов Эмиль Артурович

Введение

Цель работы

- Освоить на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом

Выполнение работы

Выполнение работы

1. Проанализировали паттерны работы самого шифрования/дешифрования.
2. В программе мы занесли ASCII таблицу в вектор.
3. Сделали генератор случайного ключа.
4. Написали функции шифрования/дешифрования.
5. Прописали главную функцию со всей логикой.
6. Получили в консоль информацию.

Листинг программы

ASCII to Vector

```
void pushToVec(vector<char>* v) {
    for (int i = 0; i < 128; i++) {
        v->push_back(char(i));
    }
}
```

Figure 1: ASCII to Vector

Random key generator

```cpp
vector<char> generateRandomKey(const vector<char> *v, size_t len) {
    random_device rd;
    mt19937 mt(rd());
    uniform_int_distribution<> dist(0, v->size()-1);

    vector<char> key;

    for (int i = 0; i < len; i++) {
        key.push_back((*v)[dist(mt)]);
    }

    return key;
}
```

Figure 2: Random Key Generator

Encryption/Decryption methods

```cpp
vector<char> xorEncryption(vector<char> p, vector<char> k) {
    vector<char> enc;
    if (p.size() == k.size()) {
        for (int i = 0; i < p.size(); i++) {
            enc.push_back(p[i] ^ k[i]);
        }
    }
    return enc;
}


vector<char> findKey(const vector<char>& p, const vector<char>& enc) {
    return xorEncryption(p, enc);
}

vector<char> xorDecryption(const vector<char>& enc, const vector<char>& k) {
    return xorEncryption(enc, k);
}
```

Figure 3: Encryption/Decryption methods

Main

```cpp
int main()
{
    vector<char> v;
    // 97 - a, z - 122
    pushToVec(&v);

    std::string str1("HelloWorldAndMyNewFr");
    std::string str2("AndYouTooMyOldFriend");

    if (str1.size() == str2.size()) {
        vector<char> key = generateRandomKey(&v, str1.size());

        vector<char> arr1(str1.begin(), str1.end());
        vector<char> arr2(str2.begin(), str2.end());

        vector<char> enc1 = xorEncryption(arr1, key);
        vector<char> enc2 = xorEncryption(arr2, key);


        cout << "Original str1: " << str1 << endl;
        cout << "Original str2: " << str2 << endl;

        cout << endl << "Generated: ";
        for (char c : key) { cout << c; }

        vector<char> enc1_enc2 = xorDecryption(enc1, enc2);
        cout << endl << "enc1 ^ enc2: ";
        for (char c : enc1_enc2) { cout << c; }


        vector<char> enc1_enc2_p1 = xorDecryption(enc1_enc2, arr1);
        cout << endl << " Decrypted by two encryption keys: ";
        for (char c : enc1_enc2_p1) { cout << c; }
        vector<char> enc1_enc2_p2 = xorDecryption(enc1_enc2, arr2);
        cout << endl << " Decrypted by two encryption keys: ";
        for (char c : enc1_enc2_p2) { cout << c; }
    }
    return 0;
}
```

Output

```
Original str1: HelloWorldAndMyNewFr
Original str2: AndYouTooMyOldFriend

0enerate}89~ʕ!3qzXU
enc1 ^ enc2:
5";)8)?<
(
 Decrypted by two encryption keys: AndYouTooMyOldFriend
 Decrypted by two encryption keys: HelloWorldAndMyNewFr
```

Figure 5: Console output

Выводы

Выводы

Во время выполнения работы, мы получили навыки работы с режимом однократного гаммирования и поняли, что в режиме однократного гамирования получить второй текст можно легко узнать чужой исходный текст, зная, например, свой зашифрованный текст и исходный текст и чужой зашифрованный текст.

Спасибо за вниманиие