АСД-122-23

Дюба Ігор

Звід до лабораторного практикуму з дисципліни

**Методи і моделі захисту в комп’ютерних системах**

Виконані роботи 1, 3

Код завантажено на гітхаб: <https://github.com/idyuba/SecurLabs>

**Лабораторна робота №1**

Потокове шифрування на базі РЗЛЗЗ

**Мета**: отримати навички у створенні програмної реалізації алгоритму потокового шифрування на базі регістра зсуву з лінійним зворотним зв'язком (РЗЛЗЗ).

**Завдання**: Розробити на обраній мові програмування консольний або віконний додаток, що реалізує описаний алгоритм потокового шифрування на базі РЗЛЗЗ для шифрування вмісту текстового або виконуваного файлу.

**Хід виконання роботи.**

Використовуючи мову програмування С++ розроблено консольний застосунок, що реалізує алгоритм потокового шифрування на базі РЗЛЗЗ

Код протестована на поліномах (8, 4, 3, 2, 0) (59, 6, 5, 4, 3, 1, 0) (64, 4, 3, 1, 0)

Код програми

|  |
| --- |
| #include <iostream>  #include <bitset>  #include <vector>  // polynom (8, 4, 3, 2, 0)  //const size\_t bitsCnt = 8;  //std::bitset<bitsCnt> reg;  //std::vector<int> bitsPositions{ 4, 3, 2, 0 };  // (59, 6, 5, 4, 3, 1, 0)  const size\_t bitsCnt = 59;  std::bitset<bitsCnt> reg;  std::vector<int> bitsPositions{ 6, 5, 4, 3, 1, 0 };  // (64, 4, 3, 1, 0)  //const size\_t bitsCnt = 64;  //std::bitset<bitsCnt> reg;  //std::vector<int> bitsPositions{ 4, 3, 1, 0 };  void findCodeKey() {  bool xorValue = false;  for (const auto& bit : bitsPositions) {  xorValue ^= reg[bit];  }  reg >>= 1;  reg[bitsCnt - 1] = xorValue;  }  std::string codeText(const std::string& str)  {  std::string output;  for (const char& c : str) {  findCodeKey();  char codedSympol = c ^ static\_cast<char>(reg.to\_ullong());  output += codedSympol;  }  return output;  }  int main() {    std::string initialText = "ABC def 123 !@#";  std::cout << "Initial text: " << initialText << "\n";  reg.set(); // all bits 1  std::string codedText = codeText(initialText);  std::cout << "Coded text: " << codedText << "\n";  reg.set(); // all bits 1  std::string deCodeText = codeText(codedText);  std::cout << "DeCoded text: " << deCodeText << "\n";  return 0;  } |

Результат виконання

A black and white screen with white text

Description automatically generated

**Висновок**: в ході виконання лабораторної роботи розроблено консольний застосунок, що реалізує алгоритм потокового шифрування на базі РЗЛЗЗ

**Лабораторна робота № 3**

Схеми розподілу таємних ключів між користувачами.

**Мета**: Ознайомитися з основними поняттями, присвяченими принципам обміну таємними ключами симетричних систем. Розглянути і реалізувати програмно базові схеми управління ключами.

Завдання:

1. Вивчити основні теоретичні положення стосовно принципів обміну ключами симетричного шифрування.

2. Реалізувати програмно схему шифрування та дешифрування управляючого вектора. Деталі алгоритму:

- сеансовий ключ, призначений для обміну даними генерується з використанням алгоритму Лемера . Розмірність ключа визначається розрядністю блоку даних . Сеансовий ключ зберігається у файлі "s\_key.txt";

- управляючий вектор генерується за допомогою алгоритму Лемера. Розрядність вектора визначається параметром ;

- в якості функції хешування застосовується метод RXOR, розрядність хэша складає ;

- основний ключ вводиться з клавіатури. Довжина основного ключа складає біт;

- результат складання по модулю 2 основного ключа і хеш-функції управляючого вектора, ключ зберігається у файлі "m\_key.txt";

- шифрування сеансового ключа проводиться згідно симетричного алгоритму, згідно з варіантом. Текстом для шифрування в даному випадку являється вміст файлу "s\_key.txt", а ключем шифрування - вміст "m\_key.txt";

- результат вивести на екран.

**Хід виконання роботи**

Написано клас для алгоритму Лемера

Написані додаткові функції для генерації сеансового ключа, контрольного вектору, хеш функції та кодування

Код програми

|  |
| --- |
| #include <iostream>  #include <fstream>  #include <vector>  #include <iomanip>  #include <string>  #include <cstring>  #include <chrono>  class LemerAlgr {  public:  LemerAlgr() {  current = std::chrono::system\_clock::now().time\_since\_epoch().count();  }  void reset() {  current = std::chrono::system\_clock::now().time\_since\_epoch().count();  }  unsigned long next() {  current = (a \* current + c) % m;  return current;  }  private:  static const unsigned long a = 1664525;  static const unsigned long c = 1013904223;  static const unsigned long m = 4294967295;  unsigned long current;  };  std::vector<unsigned char> generateSessionKey(LemerAlgr& lcg, size\_t keyBits) {  size\_t keyBytes = keyBits / 8;  std::vector<unsigned char> key(keyBytes);  for (size\_t i = 0; i < keyBytes; ++i) {  key[i] = static\_cast<unsigned char>(lcg.next() % 256);  }  return key;  }  std::vector<unsigned long> generateControlVector(LemerAlgr& lcg, size\_t vectorBits) {  size\_t vectorSize = vectorBits / 32;  std::vector<unsigned long> controlVector(vectorSize);  for (size\_t i = 0; i < vectorSize; ++i) {  controlVector[i] = lcg.next();  }  return controlVector;  }  std::vector<unsigned char> rxor(const std::vector<unsigned char>& data, const std::vector<unsigned char>& key) {  std::vector<unsigned char> result(data.size());  for (size\_t i = 0; i < data.size(); ++i) {  result[i] = data[i] ^ key[i % key.size()];  }  return result;  }  void saveToFile(const std::string& filename, const std::vector<unsigned char>& key) {  std::ofstream file(filename, std::ios::binary);  if (file.is\_open()) {  file.write(reinterpret\_cast<const char\*>(key.data()), key.size());  file.close();  }  else {  std::cerr << "Error: " << filename << std::endl;  }  }  std::vector<unsigned char> encrypt(const std::vector<unsigned char>& data, const std::vector<unsigned char>& key) {  return rxor(data, key);  }  int main() {  size\_t keyBits = 128;  LemerAlgr lemerAlgr;  std::vector<unsigned char> sessionKey = generateSessionKey(lemerAlgr, keyBits);  for (unsigned char byte : sessionKey) {  std::cout << std::hex << std::setw(2) << std::setfill('0') << static\_cast<int>(byte) << " ";  }  std::cout << std::dec << std::endl;  //saveToFile("s\_key.txt", sessionKey);  lemerAlgr.reset();  size\_t vectorBits = 128;  std::vector<unsigned long> controlVector = generateControlVector(lemerAlgr, vectorBits);  std::vector<unsigned char> hash(controlVector.size() \* sizeof(unsigned long));  std::memcpy(hash.data(), controlVector.data(), hash.size());  std::vector<unsigned char> mKey = rxor(hash, sessionKey);  for (unsigned char byte : mKey) {  std::cout << std::hex << std::setw(2) << std::setfill('0') << static\_cast<int>(byte) << " ";  }  std::cout << std::dec << std::endl;    //saveToFile("m\_key.txt", mKey);  //std::ifstream inFile("s\_key.txt", std::ios::binary);  //std::vector<unsigned char> data((std::istreambuf\_iterator<char>(inFile)), std::istreambuf\_iterator<char>());  // inFile.close();  std::vector<unsigned char> encryptedData = encrypt(sessionKey, mKey);  std::cout << "Encrypted data: ";  for (unsigned char byte : encryptedData) {  std::cout << std::hex << std::setw(2) << std::setfill('0') << static\_cast<int>(byte) << " ";  }  std::cout << std::dec << std::endl;  return 0;  } |

Результат виконання

A screenshot of a computer

Description automatically generated

**Висновок**: в ході виконання роботи реалізовано програмно схему шифрування та управляючого вектору.