Исследование потоковых шифров

Выполнил: Макаров Алексей Игоревич, 3 курс 4 группа 2 подгруппа

2024

**Теоритические сведения**

Потоковый шифр (иногда говорят «поточный») – симметричный шифр, преобразующий каждый символ mi открытого текста в символ шифрованного ci, зависящий от ключа и расположения символа в тексте.

Термин «потоковый шифр» обычно используется в том случае, когда шифруемые символы открытого текста представляются одной буквой, битом или реже – байтом.

Все потоковые шифры делятся на 2 класса: синхронные и асинхронные (или самосинхронизирующиеся).

Основной задачей потоковых шифров является выработка некоторой последовательности (гаммы) для зашифрования, т. е. выходная гамма является ключевым потоком (ключом) для сообщения.

Синхронные потоковые шифры (СПШ) характеризуются тем, что поток ключей генерируется независимо от открытого текста и шифртекста. Главное свойство СПШ – нераспространение ошибок. Ошибки отсутствуют, пока работают синхронно шифровальное и дешифровальное устройства отправителя и получателя информации.

Синхронные потоковые шифры уязвимы к атакам на основе изменения отдельных битов шифртекста.

Недостаток этих потоковых шифров – распространение ошибок, так как искажение одного бита в процессе передачи шифртекста приведет к искажению нескольких битов гаммы и, соответственно, расшифрованного сообщения.

Ключевые последовательности (случайные последовательности (СП), либо псевдослучайные последовательности (ПСП)) вырабатываются специальными блоками систем потокового шифрования – генераторами

Указанный стандарт определяет базовые понятия в рассматриваемой предметной области:

• случайные числа (последовательности) – последовательность элементов, каждый из которых не может быть предсказан (вычислен) только на основе знания предшествующих ему элементов данной последовательности;

• псевдослучайные числа – последовательность элементов, полученная в результате выполнения некоторого алгоритма и используемая в конкретном случае вместо последовательности случайных чисел

Алгоритм RSA разработан для систем асимметричного зашифрования/расшифрования и будет более детально рассмотрен с практической точки зрения ниже. Генератор же ПСП на основе RSA устроен следующим образом. Последовательность генерируется с использованием соотношения



Безопасность генератора опирается на сложность взлома алгоритма RSA, т. е. на разложение числа n на простые сомножители.

Широкое распространение получил алгоритм генерации ПСП, называемый алгоритмом BBS (от фамилий авторов: L. Blum, M. Blum, M. Shub) или генератором на основе квадратичных вычетов. Для целей криптографии этот метод предложен в 1986 г. Начальное значение x0 генератора вычисляется на основе соотношения



где n, как и в генераторе на основе RSA, является произведением простых чисел p и q, однако в нашем случае эти простые числа должны быть сравнимы с числом 3 по модулю 4, т. е. при делении p и q на 4 должен получаться одинаковый остаток 3; число x должно быть взаимно простым с n; число n называют числом Блюма.

Выходом генератора на t-м шаге является младший бит числа xt:



Алгоритм RC4 разработан Р. Ривестом в 1987 г. Он представляет собой потоковый шифр с переменным размером ключа. Алгоритм RC4, как и любой потоковый шифр, строится на основе генератора псевдослучайных битов (генератора ПСП). На вход генератора записывается ключ, а на выходе читаются псевдослучайные биты. Длина ключа может составлять от 40 до 2048 битов.

Генератор ключевого потока RC4 переставляет значения, хранящиеся в S, и каждый раз выбирает различное значение из S в качестве результата. В одном цикле RC4 определяется одно n-битное слово K из ключевого потока, которое в последующем суммируется с исходным текстом для получения зашифрованного текста. Эта часть алгоритма называется генератором ПСП.

**Ход работы**

Вариант 8

В данной работе было необходимо реализовывать генерацию ПСП в соответствии с вариантом.

Таблица 1 – Вариант лабораторной работы

|  |  |  |
| --- | --- | --- |
| Вариант | Алгоритм генерации ПСП | Параметры |
| 2 | RSA | p, q, e – 256 бит |

Код программы представлен в листинге 1.

|  |
| --- |
| from Crypto.PublicKey import RSA  import Crypto.Util.number  # Генерация ключей RSA  def generate\_RSA\_keys():  # Генерация случайного простого числа p и q  p = Crypto.Util.number.getPrime(256)  q = Crypto.Util.number.getPrime(256)  # Выбор открытой экспоненты e  e = 65537  # Вычисление модуля n  n = p \* q  # Вычисление функции Эйлера  phi = (p - 1) \* (q - 1)  # Вычисление секретной экспоненты d  d = Crypto.Util.number.inverse(e, phi)  # Создание объекта ключа RSA  key = RSA.construct((n, e, d))  return key  # Пример использования  key = generate\_RSA\_keys()  print("Открытый ключ (n, e):", key.n, key.e)  print("Закрытый ключ (d):", key.d) |

Листинг 1 – Программная реализация приложения

Результат работы программы представлен в листинге 2:

|  |
| --- |
| 10011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001100110011001 |

Листинг 2 – Результат работы приложения

Приложение 2 должно реализовывать алгоритм RC4 в соответствии с вариантом

|  |  |  |
| --- | --- | --- |
| Вариант | n | Ключ (в виде десятичных чисел) |
| 4 | 8 | 12, 13, 90, 91, 240 |

Код программы представлен в листинге 3.

|  |
| --- |
| from Crypto.Cipher import ARC4 import time  *# Алгоритм RC4* def rc4\_encrypt(key, plaintext):  *cipher* = ARC4.new(key)  return *cipher*.encrypt(plaintext)  *# Оценка скорости выполнения операций генерации ПСП* def evaluate\_rc4\_speed():  *key* = b'\x14\x15\x16\x17\x3C\x3D'  *plaintext* = b'Hello, world!'   *start\_time* = time.time()  rc4\_encrypt(*key*, *plaintext*)  *end\_time* = time.time()   return *end\_time* - *start\_time  # Пример использования* encryption\_time = evaluate\_rc4\_speed() print("Время выполнения операции генерации ПСП:", encryption\_time, "секунд") |

Листинг 3 – Программная реализация приложения

Результат работы программы представлен в листинге 4:

|  |
| --- |
| Засшифрованные данные: TGUF|ÿ@ÄI/î§"Jõ,ÂèMÚ2¶©mR} Îg  Время шифрования: 0.0 секунд  Время расшифровки: 0.0009856224060058594 секунд  Расшифрованные данные: The quick brown fox jumps over the lazy dog |

Листинг 4 – Результат работы приложения

**Вывод**: таким образом, были закреплены теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости потоковых шифров.