Лабораторная работа №4
Криптосистема RSA

Дедлайн: 12.04.2021
Базовый балл за работу: 30 баллов

1 Введение

В данной лабораторной работе вам необходимо программно реализовать криптосистему RSA, стойкость которой связана с вычислительной сложностью задачи факторизации больших чисел. Дополнительную информацию об этой криптосистеме вы можете получить в презентации, на популярном сайте Википедия, а также в книгах по криптографии.

2 Условие лабораторной работы

Основные алгоритмы

Формально, как и для любой ассиметричной криптосистемы, для работы RSA необходимо реализовать три основных алгоритма, которые описаны в лекции по RSA:

1. *(8 баллов)* генерация ключей (Gen) – параграф 23.1;
2. *(3 балла)* алгоритм зашифрования ($Encr$) – параграф 23.1;
3. *(3 балла)* алгоритм расшифрования ($Decr$) – параграф 23.1.

Вспомогательные алгоритмы

Помимо этого, вам будет необходимо реализовать четыре вспомогательных алгоритма, которые необходимы для организации шифрования и описаны в лекциях по реализации RSA и генерации простых:

1. *(5 баллов)* расширенный алгоритм Евклида ($ExEu$) – параграф 24.3;
2. алгоритм возведения в степень по модулю ($ModPow$) – параграф 24.4:

- *(5 баллов)* без использования дополнительных алгоритмов;
- *(10 баллов)* с использованием алгоритма Монтгомери;

3. *(2 балла)* вариант А алгоритма генерации простых чисел ($GenPrime$) – параграф 25.1;
4. вероятностный алгоритм проверки числа на простоту ($IsPrime$) на основе:
   - *(4 балла)* теста Ферма – параграф 25.3;
   - *(7 баллов)* теста Рабина-Миллера – параграф 25.4;
   - *(7 баллов)* теста Соловея-Штрассена – смотреть тут.

Оптимизации RSA

Также, по желанию, возможна реализация оптимизаций RSA, которые описаны в параграфе 24.6 лекции по реализации RSA:

- *(4 балла)* использовать функцию Кармайкла вместо функции Эйлера;
- *(6 баллов)* использовать прием с сохранением p и q для ускорения расшифрования.

Особенности реализации лабораторной работы

- Под реализацией следует понимать программу, написанную на одном из языков программирования: {C/C++, C#, Java, Python[1].
- При реализации алгоритма $ModPow$ парни реализуют способ “справа налево“, а девушки – “слева направо“.
- При реализации алгоритма $IsPrime$ число раундов проверки числа на простоту у используемого алгоритма должно быть не менее 100.
- При реализации алгоритма $IsPrime$ допускается использование нескольких тестов одновременно, при этом число следует считать простым тогда и только тогда, когда все используемые алгоритмы считают число вероятно простым. Баллы за реализованные тесты суммируются.

[1]Но не Jupyter Notebook}, Kotlin, Ruby, Swift, Rust

- Помимо указанных алгоритмов, в реализации должен присутствовать функционал, демонстрирующий работоспособность каждого из реализованных алгоритмов, другими словами необходимо написать тесты, которые покрывают весь ваш исходный код.

- Все реализованные алгоритмы должны поддерживать "длинную арифметику", при этом допускается использование стандартных возможностей языка (например, *BigInteger* в Java) или подключение сторонних библиотек (например, *boost* в C++). При тестировании значение l считать равным 2048.

- Нельзя получать баллы и за реализацию базового алгоритма возведения в степень, и за реализацию возведения в степень с использованием алгоритма Монтгомери.

- Допускается использование любого стандартного источника случайности, который присутствует в выбранном языке программирования.

3 Бонусные задания

Бонусное задание №1

Реализовать бота в Telegram, который поддерживает выполнение следующих команд:

- /start – бот начинает общение с пользователем (например, отправляет приветственное сообщение);

- /help – бот отображает сообщение с помощью по командам;

- /gen $\{l\}$ – бот выполняет алгоритм *Gen* из лабораторной работы, при этом личный ключ (*privkey*), состоящий из модуля $n = pq$ и секретной экспоненты d, сохраняется на устройстве, а открытый ключ (*pubkey*), состоящий из модуля $n = pq$ и открытой экспоненты e, выводится пользователю на экран;

- /enc $\{pubkey\}$ $\{data\}$ – бот выполняет алгоритм *Encr* из лабораторной работы, при этом он зашифровывает данные *data* на открытом ключе *pubkey*;

- /dec $\{data\}$ – бот выполняет алгоритм *Decr* из лабораторной работы, при этом он расшифровывает данные *data* с помощью личного ключа *privkey*.

Данный набор команд является ориентировочным, допускается его изменение. Выбор формата представления данных и открытого ключа (числа, $base64$ строки, файлы) осуществляется автором самостоятельно и должен быть аргументирован.

Дополнительно можно реализовать прием "цифрового конверта", обеспечить безопасность хранения личного ключа (например, защитив его на пароле)и т.п..

Бонусное задание №2

Самостоятельно реализовать пять основных операций "длинной арифметики": сложение, вычитание, умножение, деление, возведение в степень (при этом нельзя использовать любые механизмы длинной арифметики, которые есть в языке программирования, даже стандартные!). Дополнительно можно реализовать другие операции: модульную арифметику, вычисление квадратного корня, перевод из одной системы счисления в другую и т.п.

Бонусное задание №3

Реализовать надежный генератор случайных чисел, который должен быть основан на комбинации криптостойкого генератора случайных чисел и внешних источников энтропии. Необходимо использовать не менее двух альтернативных разнотипных источников энтропии например, информацию о системных прерываниях и информацию о положении курсора на экране).

Бонусное задание №4

Реализовать протокол ЭЦП (электронно-цифровой подписи) с помощью алгоритма RSA. Должны быть реализрованы три основных алгоритма: Gen, $Sign$ и $Verify$. При этом подписывать необходимо не сообщение m, а его хэш-значение $h(m)$, причем в качестве алгоритма хэширования следует использовать алгоритм хэширования из СТБ 34.101.77 (при $l = 128$), подключив его реализацию из библиотеки *bee2* в свой проект. Использовать модуль RSA длиной не менее 3072 бит.

Особенности бонусных заданий

- За каждое бонусное задание можно получить от 15 до 45 баллов (в зависимости от сложности проделанной работы).
- Бонусные задания можно выполнить и сдать до конца семестра.
- Одно бонусное задание может выполнить не более восьми человек, при этом используемые языки программирования должны различаться.

4 Порядок сдачи лабораторной работы

- Вам необходимо создать архив формата «.zip», название которого должно иметь вид «Ivanov_4.zip», где «Ivanov» – ваша фамилия латинскими буквами. В архиве должен быть исходный код вашей реализации. При отправке бонусных заданий наименование файла сохраняется, лишь в самый конец дописывается символ «b».
- Не позже, чем за 24 часа до сдачи лабораторной работы преподавателю, вы должны отправить архив по одному из контактов, указанных в ответе на вопрос №3 в файле «FAQ.pdf».