Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Лабораторная работа № 8**

«Исследование асимметричных шифров»

Студент: Лэ Н.З.

ФИТ 2 курс 4 группа

Преподаватель: Берников О.В

Минск 2020

1. **Теоретические сведения**

В основу асимметричной криптографии положена идея использовать ключи парами: один – для зашифрования (открытый или публичный ключ), другой – для расшифрования (тайный ключ). Отметим, что указанная пара ключей принадлежит получателю зашифрованного сообщения.

Все алгоритмы шифрования с открытым ключом основаны на использовании односторонних функций, к числу которых, как известно, относится вычисление дискретного логарифма.

Определение 1. Односторонней функцией (one-way function) называется математическая функция, которую относительно легко вычислить, но трудно найти по значению функции соответствующее значение аргумента, т. е., зная х, легко вычислить f(x), но по известному f(x) трудно найти подходящее значение x.

Алгоритмы шифрования с открытым ключом можно использовать для решения следующих задач:

* зашифрования/расшифрования передаваемых и хранимых данных в целях их защиты от несанкционированного доступа,
* формирования цифровой подписи под электронными документами,
* распределения секретных ключей, используемых далее при шифровании документов симметричными методами.

В данной работе мы будем работать над аспектами решения первой из указанных задач.

По мнению Диффи и Хеллмана алгоритм шифрования с открытым ключом, должен:

* вычислительно легко создавать пару (открытый ключ, e – закрытый ключ, d),
* вычислительно легко зашифровывать сообщение Mi открытым ключом,
* вычислительно легко расшифровывать сообщение Ci, используя закрытый ключ,
* обеспечивать непреодолимую вычислительную сложность определения соответствующего закрытого ключа при известном открытом ключе,
* обеспечивать непреодолимую вычислительную сложность восстановления исходного (открытого сообщения, Mi) зная только открытый ключ и зашифрованное сообщение, Ci.

Определение 2. Ранцевый (рюкзачный) вектор S = (s1, . . ., sn) – это упорядоченный набор из n, n ≥ 3, различных натуральных чисел si. Входом задачи о ранце (рюкзаке) называем пару (S, S), где S – рюкзачный вектор, а S – натуральное число.

Решением для входа (S, S) будет такое подмножество из S, сумма элементов которого равняется S.

В наиболее известном варианте задачи о ранце требуется выяснить, обладает или нет данный вход (S, S) решением. В варианте, используемом в криптографии, нужно для данного входа (S, S) построить решение, зная, что такое решение существует. Оба эти варианта являются NP-полными. Имеются также варианты этой задачи, которые не лежат даже в классе NP.

Как видим, проблема укладки ранца формулируется просто. Дано множество предметов общим числом n различного веса. Спрашивается, можно ли положить некоторые из этих предметов в ранец так, чтобы его вес стал равен определенному значению S? Более формально задача формулируется так: дан набор значений s1, s2, …, sn и суммарное значение S. Требуется вычислить значения si такие, что

S = b1\*s1 + b2\*s2+... + bn\*sn. (1.1)

Здесь bi может быть либо нулем, либо единицей. Значение bi = 1 означает, что предмет mi кладут в рюкзак, а bi = 0 – не кладут.

Суть метода для шифрования состоит в том, что существуют две различные задачи укладки ранца: одна из них решается легко и характеризуется линейным ростом трудоемкости, а другая решается трудно. Легкий для укладки ранец можно трансформировать в трудный.

Трудный для укладки ранец применяется в качестве открытого ключа е, который легко использовать для зашифрования, но невозможно – для расшифрования. В качестве закрытого ключа d применяется легкий для укладки ранец, который предоставляет простой способ расшифрования сообщения.

В качестве закрытого ключа d (легкого для укладки ранца) используется сверхвозрастающая последовательность, состоящая из n элементов: d1, d2, …, dz: d = {di}, i = 1, …, n.

Определение 3. Сверхвозрастающей называется последовательность, в которой каждый последующий член больше суммы всех предыдущих.

Необходимо по очереди анализировать некоторый «текущий вес» S предметов, составляющих сверхвозрастающую последовательность; в результате анализа нужно упаковать (доупаковать) ранец.

1. В качестве текущего выбирается число S, которое сравнивается с «весом» самого тяжелого предмета (dn); если текущий вес меньше веса данного предмета, то его в ранец не кладут (0), в противном случае его укладывают (bn = 1) в ранец и переходят к анализу очередного (в общем случае – i-го предмета).

2. Если на предыдущем (i-м шаге) предмет пополнил ранец, то текущий вес уменьшают на вес положенного предмета (S = S – di); переходят к следующему по весу предмету в последовательности: di-1.

Шаги повторяются до тех пор, пока процесс не закончится.

Если текущий вес уменьшится до нуля (S = 0), то решение найдено. В противном случае – нет

Открытый ключ e представляет собой нормальную (не сверхвозрастающую) последовательность. Он формируется на основе закрытого ключа и не позволяет легко решить задачу об укладке ранца.

Для получения открытого ключа e (e = {ei}, i = 1, …, n) все значения закрытого ключа умножаются на некоторое число a по модулю n:

ei = di \* a (mod n). (1.2)

Значение модуля n должно быть больше суммы всех чисел последовательности; кроме того, НОД (а, n) = 1.

Для зашифрования сообщения (М) оно сначала разбивается на блоки, по размерам равные числу (z) элементов последовательности в ранце. Затем, считая, что 1 указывает на присутствие элемента последовательности в ранце, а 0 – на его отсутствие, вычисляются полные веса рюкзаков (Si, i = 1, . . ., z): по одному ранцу для каждого блока сообщения с использованием открытого ключа получателя, e.

Для расшифрования сообщения получатель (использует свой тайный ключ, d: сверхвозрастающую последовательность) должен сначала определить обратное к а число: а-1, такое что:

а \* а-1 (mod n) = 1. (1.3)

Для вычисления обратных чисел по модулю можно использовать известный нам расширенный алгоритм Евклида.

После определения обратного числа каждое значение шифрограммы (ci) преобразуется в соответствии со следующим соотношением:

Si = ci \* а-1 mod n. (1.4)

Полученное на основании последней формулы для каждого блока число далее рассматривается как заданный вес ранца, который следует упаковать по изложенному выше алгоритму, используя сверхвозрастающую последовательность (тайный ключ получателя).

Криптостойкость алгоритма во многом определяется скоростью (временем) поиска нужного варианта укладки ранца. Понятно, что для последовательности из шести-десяти или немногим более того элементов нетрудно решить задачу укладки ранца, даже если последовательность не является сверхвозрастающей. При практической же реализации алгоритма ранец должен содержать не менее нескольких сотен элементов. Длина каждого члена сверхвозрастающей последовательности должна быть несколько сотен бит, а длина числа n – от 100 до 200 бит. Для получения этих значений практические реализации алгоритма используют генераторы ПСП.

С другой стороны, известный способ определения, какие предметы кладутся в ранец, является проверка возможных решений до получения правильного. Самый быстрый алгоритм, принимая во внимание различную эвристику, имеет экспоненциальную зависимость от числа возможных предметов. Если добавить к последовательности весов еще один член, то найти решение станет вдвое труднее. Это намного труднее сверхвозрастающего ранца, где, при добавлении к последовательности одного элемента, поиск решения увеличивается на одну операцию.

Ранцевые криптосистемы не являются криптостойкими. А. Шамир и Р. Циппел обнаружили, что, зная числа а, a-1 и n («секретную лазейку»), можно восстановить сверхвозрастающую последовательность по нормальной последовательности. Важно то, что числа а и n («секретная пара») не обязательно должны быть теми же, что использовались при создании системы легальным пользователем.

1. **Практическая часть**

В данной лабораторной работе необходимо разработать пользовательское приложение, которое должно реализовывать следующие операции:

* генерация сверхвозрастающей последовательности (тайного ключа); старший член последовательности – 100-битное число; в простейшем случае принимается z = 6 (для кодировки Base64) и z = 8 (для кодировки ASCII);
* вычисление нормальной последовательности (открытого ключа);
* зашифрование сообщения, состоящего из собственных фамилии, имени и отчества;
* расшифрование сообщения;
* оценка времени выполнения операций зашифрования и расшифрования.

Были написаны 4 функции, которые позволяют имитировать работу рюкзачного шифра: функция генерации ключей, функции зашифрования и расшифрования, а также функция очистки полей.

Функция Generate получает на вход два параметра: стартовое число последовательности (первый textbox), а также длина последовательности (можно выбрать из первого combobox).

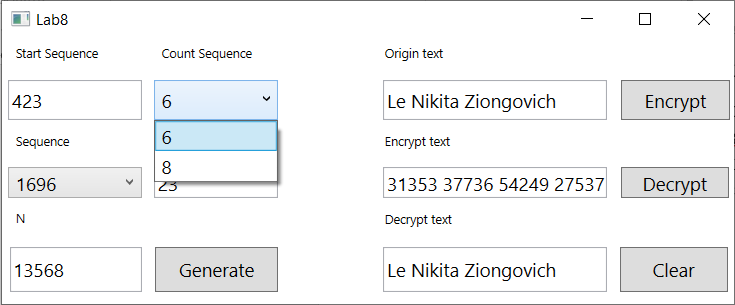


Рисунок 2.1 – Длина последовательности

После нажатия на кнопку generate на основе предыдущих параметров генерируется сверхвозрастающая последовательность (второй combobox), а также подсчитывается сумма всех элементов последовательности + 1 (N -третий textbox). Данное число можно изменить, но меньше этого числа писать нельзя.

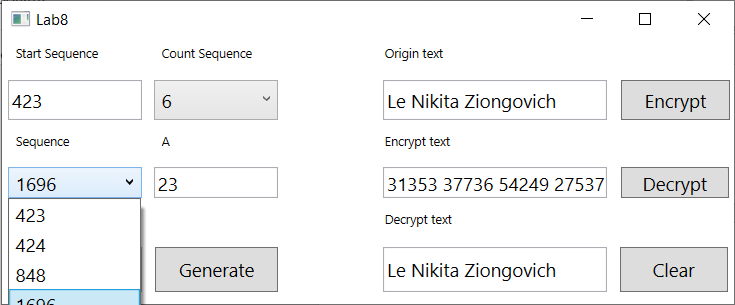
****

Рисунок 2.2 – Сверхвозрастающая последовательность

Вторая функция – Encrypt, требует для своей работы еще один параметр: число A (второй textbox), взаимнопростое с N. Текст, введенный в textbox для сообщения, посимвольно переводится в двоичный код. Если длина последовательности была равна 6 – переводится согласно Base64 (максимальное число – 2^6 – 1 = 63 = 111111), а если 8 – то используется Ascii (максимальное число – 2^8 – 1 = 255 = 11111111).

Далее на основе сверхвозрастающей последовательности (закрытого ключа) вычисляется открытый ключ. Для этого вычисляется остаток от деления произведения элемента последовательности и числа A по модулю N.

После чего входная строка посимвольно (по 6- или 8-битным блокам) шифруется следующим образом: берем первый бит блока, если он равен единице – берем первый элемент вектора открытого ключа и прибавляем его к переменной, которая считает общую сумму; берем следующий бит, опять сравниваем с единицей, снова прибавляем соответствующий элемент вектора открытого ключа, если бит равен единице. Повторяем операция для всех битов всех блоков сообщения. Шифротекстом является массив полученных сумм:

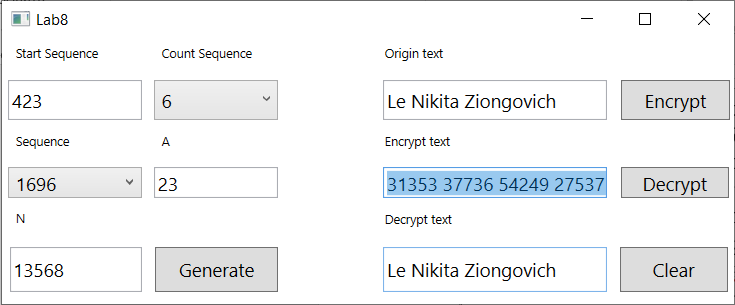
****

Рисунок 2.3 – Зашифрованное сообщение

Для дешифрации сообщения нам необходимо найти число, обратное A по модулю N. После нахождения A-1 нам поэлементно найти остаток от деления произведения числа шифротекста и A-1 по модулю N. Мы получим какое-то число. Это число является суммой элементов изначальной последовательности, на месте которых была единичка в 6- или 8-битном блоке сообщения. Для того, чтобы восстановить полученные блоки, необходимо циклично пройти все всем элементам изначальной последовательности от самого большого к самому мало отнимая элемент от числа, если число больше либо равно этому числу, при этом записывая единицу, когда число было больше либо равно, и ноль, если меньше. Тем самым мы проходим первоначальный алгоритм зашифрования в обратном порядке.

После получения 6- или 8-битных блоков можно перевести полученные двоичные числа в символ (Base64, если блоки 6-битные, Ascii, если 8-битные).

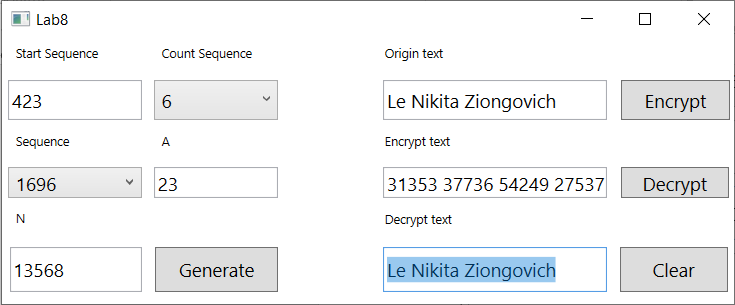
****

Рисунок 2.4 – Расшифрованное сообщение

**Вывод**

В данной лабораторной работе я закрепил теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости асимметричных шифров. Разработал приложения для реализации указанных преподавателем методов генерации ключевой информации и ее использования для асимметричного зашифрования/расшифрования. Выполнил анализ криптостойкости асимметричных шифров. А также оценил скорость зашифрования/расшифрования реализованных шифров.