МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образование «Белорусский государственный технологический университет»

Кафедра информационных систем и технологий

**«Исследование асимметричных шифров»**

Студент:

Септилко Анастасия Антоновна

Вариант 8

Преподаватель:

Блинова Евгения Александровна

Минск 2020

**ЗАДАНИЕ:**

**1.** Разработать авторское оконное приложение в соответствии с целью лабораторной работы. При этом можно воспользоваться доступными библиотеками либо программными кодами.

В основе вычислений – кодировочные таблицы Base64 и ASCII.

Приложение 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. **2.**Проанализировать время выполнения операций зашифрования/расшифрования при увеличении числа членов ключевой последовательности: при использовании разных таблиц колировки.

**3.** Результаты оформить в виде отчета по установленным правилам.

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.

Ранцевый (рюкзачный) вектор S = (s1, . . ., sz) – это упорядоченный набор из z, z ≥ 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.

S = b1s1 + b2s2+... + bzsz. (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 z элементов: d1, d2,…, dz: d = {di}, i = 1,…, z.

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

Пример 1. Последовательность {2, 3, 6, 13, 27, 52, 105, 210} (z = 8) является сверхвозрастающей, а {1, 3, 4, 9, 15, 25, 48, 76} – 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,…, z) все значения закрытого ключа умножаются на некоторое число a по модулю n:

ei = di a (mod n).

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

**Чтобы легче было рассчитать модуль я привела таблицу простых чисел от 953 до 4999.**



Для начала работы нам необходимо сгенерировать сверхвозрастающую последовательность чисел, за это отвечает функция Generate, которая показана на рисунке 1.1а, обработчик на кнопку Generate представлен на рисунке 1.1б.

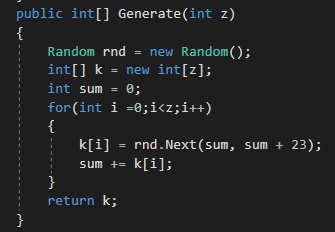


Рисунок 1.1а – Функция Generate()

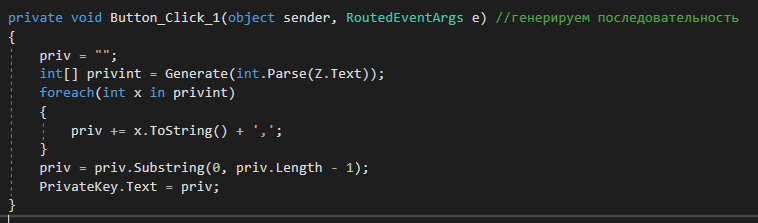


Рисунок 1.1б – Обработчик на кнопку Generate()

Чтобы последовательность сгенерировалась нам необходимо ввести число z. Работа функции генерирования сверхвозрастающей последовательности показана на рисунке 1.2.

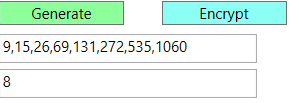


Рисунок 1.2 – Итог работы функции Generate()

Для реализации всех необходимых операций касаемые алгоритма о рюкзаке вынесены в отдельные класс Knapsack, основные функции продемонстрированы на рисунках ниже.

Функция *getcipher* предназначена для зашифрования сообщений, формирует S(1) и она продемонстрирована на рисунке 1.3. Здесь мы принимаем три параметра: публичный ключ, строку data и число z.

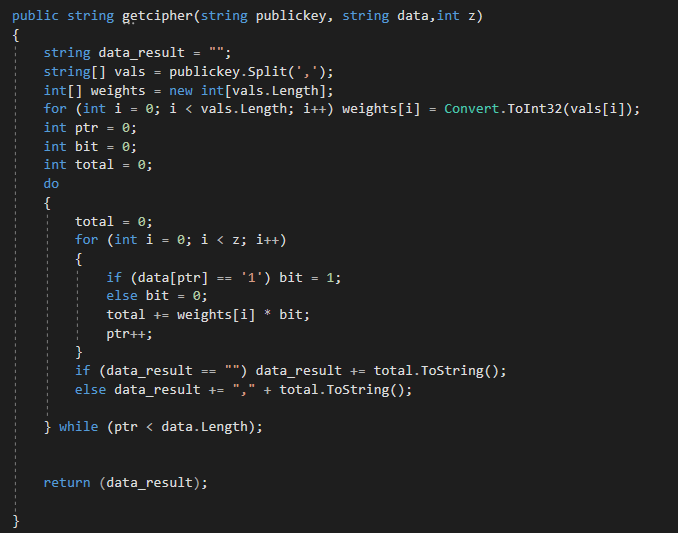


Рисунок 1.3 – Функция getcipher()

Обработчик на кнопку Encrypt показана на рисунке 1.4.

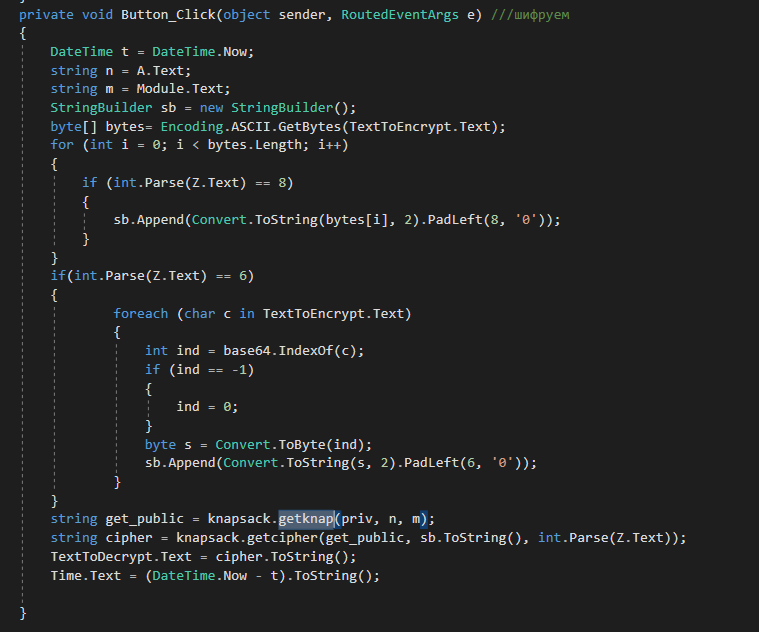
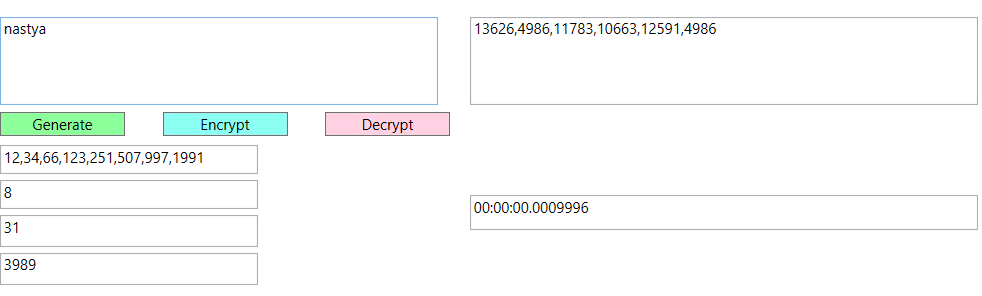


Рисунок 1.4 – Обработчик на кнопку Encrypt

Результат выполнения шифрования сообщения представлен ниже.



Для расшифровки сообщения разработано несколько функций. После того как мы нажмем на кнопку Decrypt сразу же вызовется обработчик на эту кнопку Button\_Click\_2 (рисунок 1.5).

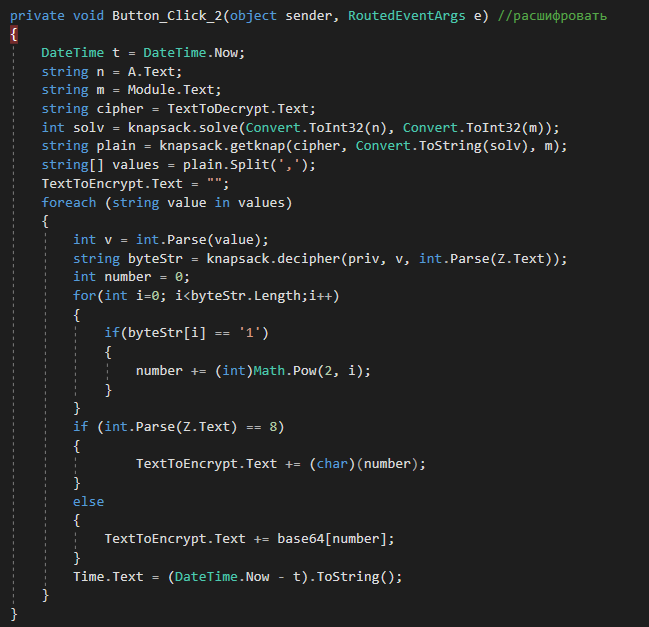


Рисунок 1.5 – Обработчик кнопки Decrypt

Мы идем плавно по коду и попадаем в функцию solve(). В этой функции мы выполняем необходимое действие, показано на рисунке 1.6. Для расшифрования нам необходимо найти число обратное по модулю числу а.

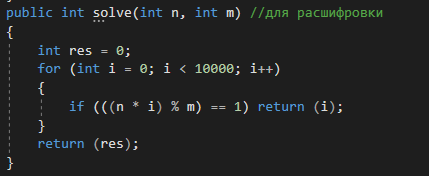


Рисунок 1.6 – Функция solve()

После чего мы попадаем еще в одну функцию класса Knapsack, которая называется getknap(). В этой функции мы формируем последовательность открытого ключа, пример показан на рисунке 1.7.

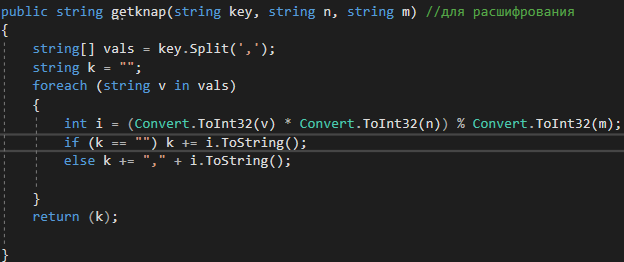


Рисунок 1.7 – Функция getknap()

Ну и после того как мы получили последовательность нам необходимо снова применить алгоритм укладки ранца, функция показана на рисунке 1.8.

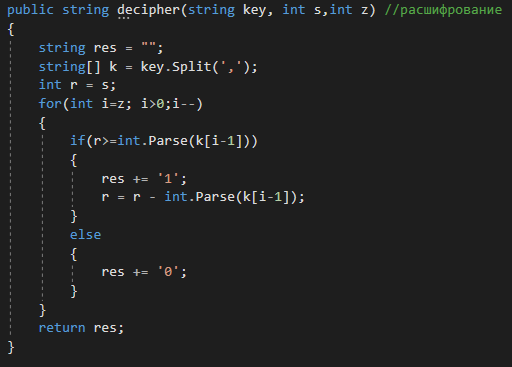


Рисунок 1.8 – Функция decipher()

Вот результат выполнения функции расшифрования сообщения. Рисунок продемонстрирован ниже.

