Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Информационная безопасность**

Студент: Козека Е. М.

ФИТ 3 курс 4 группа

Преподаватель: Нистюк О. А.

Минск 2025

**Лабораторная работа №9. Исследование асимметричных шифров**

**Цель:** изучение и приобретение практических навыков разработки и использования приложений для реализации асимметричных шифров.

**Задачи:**

1. Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости асимметричных шифров.
2. Разработать приложение для реализации указанных преподавателем методов генерации ключевой информации и ее использования для асимметричного зашифрования/расшифрования.
3. Выполнить анализ криптостойкости асимметричных шифров.
4. Оценить скорость зашифрования/расшифрования реализованных шифров.
5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**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.

*Односторонней функцией* 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:

* 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.

*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.

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, который легко использовать для зашифрования, но невозможно – для расшифрования. В качестве закрытого ключа 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.

Ранцевые криптосистемы не являются криптостойкими. А. Шамир и Р. Циппел обнаружили, что, зная числа а, a-1 и n, можно восстановить сверхвозрастающую последовательность по нормальной последовательности.

**Ход работы**

**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.

Функция generatePrivateKey принимает на вход начальное число последовательности и длину генерируемой последовательности. Начальное число последовательности – это случайное 100-битное число.

const generatePrivateKey = (initialNumber, z) => {

const sequence = [];

let element = initialNumber;

let sum = initialNumber;

for (let i = 0; i < z; i++) {

sequence.push(element);

element = sum.add(bigInt(z));

sum = sum.add(element);

}

return sequence;

};

Листинг 9.1 – Функция generatePrivateKey

Далее, на основе сгенерированного приватного ключа подбираются параметры n и a для публичного ключа. Значение модуля n должно быть больше суммы всех чисел последовательности; кроме того, НОД (а, n) = 1. Функция generatePublicKey принимает приватный ключ и параметры a и n и генерирует публичный ключ.

const generatePublicKey = (privateKey, a, n) => {

    const sequence = [];

    let d; e;

    for (let i = 0; i < privateKey.length; i++) {

        d = privateKey[i];

        e = d.multiply(a).mod(n);

        sequence.push(e);

    }

    return sequence;

};

Листинг 9.2 – Функция generatePublicKey

Функция шифрования encrypt принимает на вход публичный ключ, открытый текст и кодировку сообщения и выполняет зашифрования сообщения на основе алгоритма укладки ранца.

const encrypt = (publicKey, plaintext, encoding) => {

    const encryptedList = [];

    if (encoding === Encoding.BASE64) {

        plaintext = base64Encode(plaintext);

    }

    plaintext.split('').forEach((b, index) => {

        let binaryString;

        if (encoding === Encoding.ASCII) {

            binaryString =

plaintext.charCodeAt(index).toString(2).padStart(8, '0');

        }

        else {

            binaryString = convertBase64ToBinary(plaintext[index]);

        }

        const positions = [];

        for (let i = 0; i < binaryString.length; i++) {

            if (binaryString[i] === '1') { positions.push(i); }

        }

        let sum = bigInt.zero;

        positions.forEach(position => {

            if (position < publicKey.length) {

                sum = sum.add(publicKey[position]);

            }

        });

        encryptedList.push(sum);

    });

    return encryptedList;

};

Листинг 9.3 – Функция encrypt

Функция дешифрования decrypt принимает на вход приватный ключ, зашифрованный текст и параметры a и n.

const decrypt = (privateKey, encryptedText, a, n) => {

    let decryptedBytes = []; binaryResult = "";

    let inverse = getInverseNumber(a, n);

    for (let cipher of encryptedText) {

        let decryptedValue = cipher.times(inverse).mod(n);

        let binaryString = getDecryptedBinary(decryptedValue,

privateKey);

        binaryResult += binaryString;

        let decryptedByte = parseInt(binaryString, 2);

        decryptedBytes.push(decryptedByte);

    }

    return {decoded: new Uint8Array(decryptedBytes), binary:

binaryResult};

};

Листинг 9.4 – Функция decrypt

Результат работы приложения с сообщением «Kozeka Elizaveta Maksimovna», начальным числом 1264596768217202706724070384462 и длиной z = 8 (для кодов ASCII) приведен на рисунке 9.1.



Рисунок 9.1 – Результат работы приложения для кодировки ASCII

Результат работы приложения с сообщением «Kozeka Elizaveta Maksimovna», начальным числом 103794838703260433838991903124 и длиной z = 6 (для кодов base64) приведен на рисунке 9.2.



Рисунок 9.2 – Результат работы приложения для кодировки base64

Если попробовать расшифровать полученное сообщение, то получим исходное (рисунок 9.3).

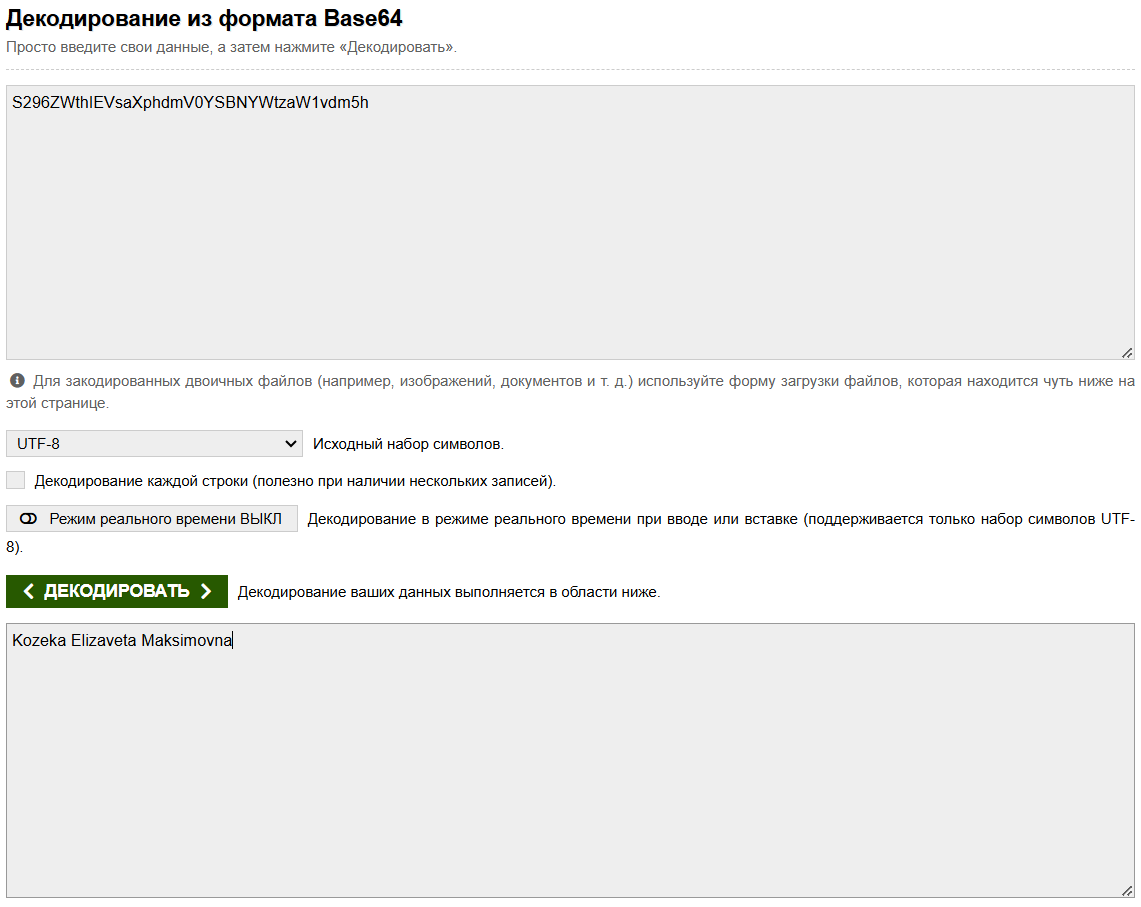


Рисунок 9.3 – Декодирование полученного сообщения из base64

Кроме того, приложение выполняет оценку скорости шифрования и дешифрования. Можно заметить, что шифрование сообщения, представленного в кодах ASCII и base64 выполняется примерно за одно и тоже время. Дешифрование сообщения, представленного в кодах ASCII, выполняется немного дольше, чем дешифрование сообщения в кодах base64. При увеличении числа членов ключевой последовательно время шифрования и дешифрования растет как для текста, представленного в кодах ASCII, так и для текста, представленного в кодах base64.

**Вывод:** в ходе выполнения лабораторной работы был изучен принцип реализации асимметричных шифров. Также было разработано приложение, выполняющее шифрование и расшифрование с помощью шифра, основанного на алгоритме об укладке ранца. Была оценена скорость шифрования и дешифрования сообщений, представленных в кодах ASCII и base64.