УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №2

по предмету

Теория Информации

Вариант 16

Выполнил:

Ушаков А.Д.

Проверил:

Болтак С.В.

Группа 351001

Минск 2025

**Задание:**

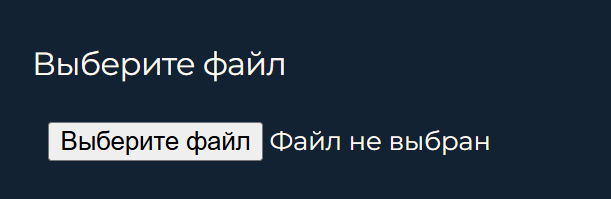
Реализовать систему потокового шифрования и дешифрования для файла с любым содержимым с помощью генератора ключевой последовательности на основе линейного сдвигового регистра с обратной связью LFSR1 (размерность регистра ***38***). Начальное состояние регистра ввести с клавиатуры. Поле для ввода состояния регистра должно игнорировать любые символы кроме 0 и 1. Вывести на экран сгенерированный ключ (последовательность из 0 и 1), исходный файл и зашифрованный файл в двоичном виде. Программа не должна быть написана в консольном режиме. Результат работы программы – зашифрованный/расшифрован- ный файл.

**Примитивный многочлен**

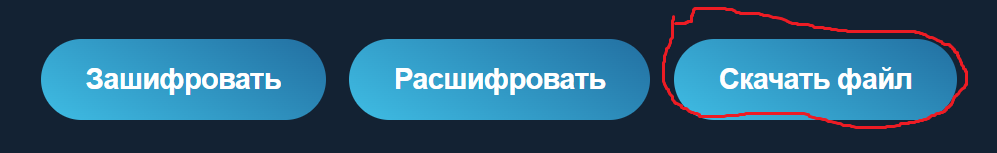
*x*38 + *x*6 + *x*5 + *x* + 1

**Работа с файлами**

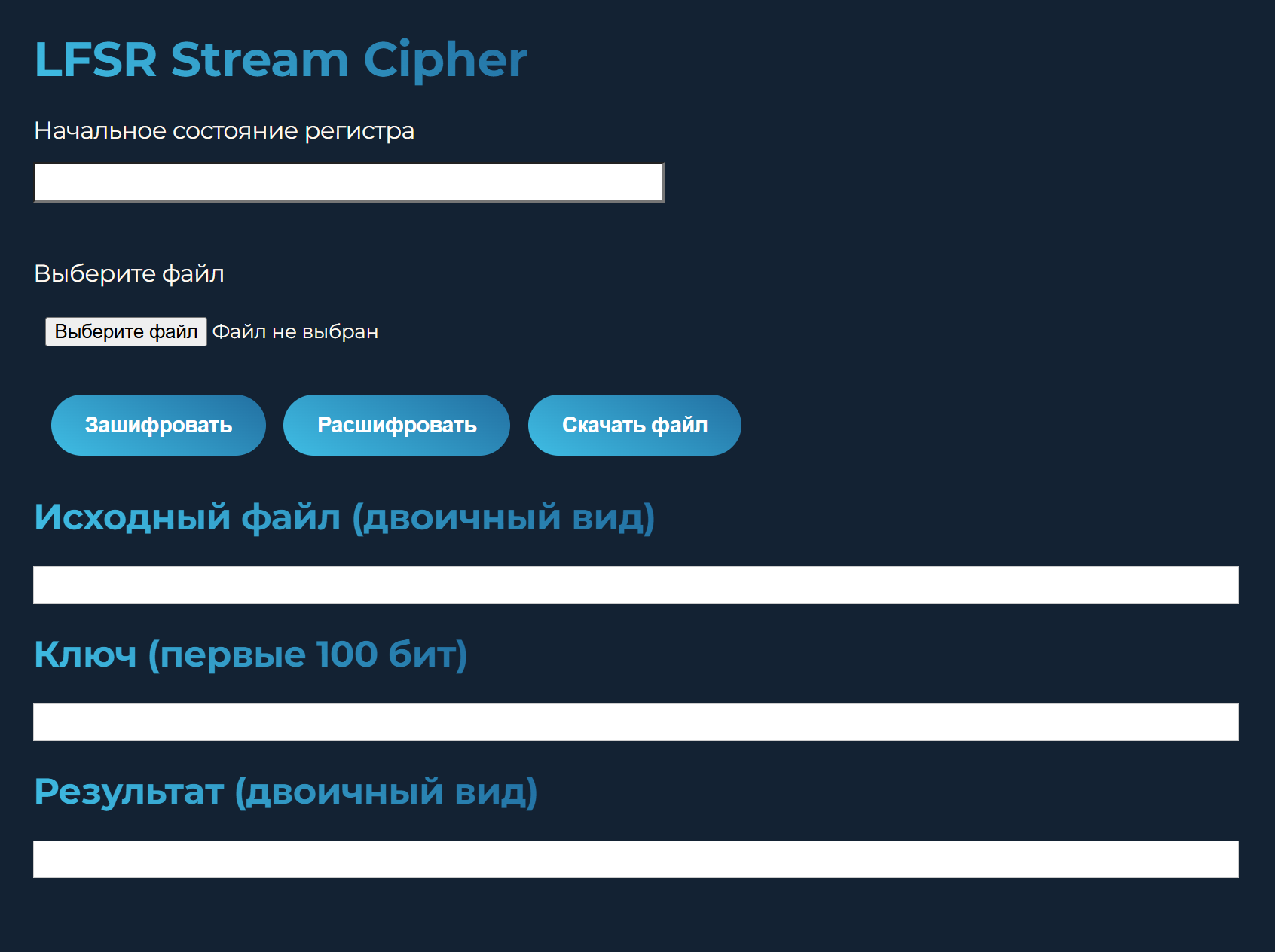
Взятие исходного файла:



Загрузка результата файла:

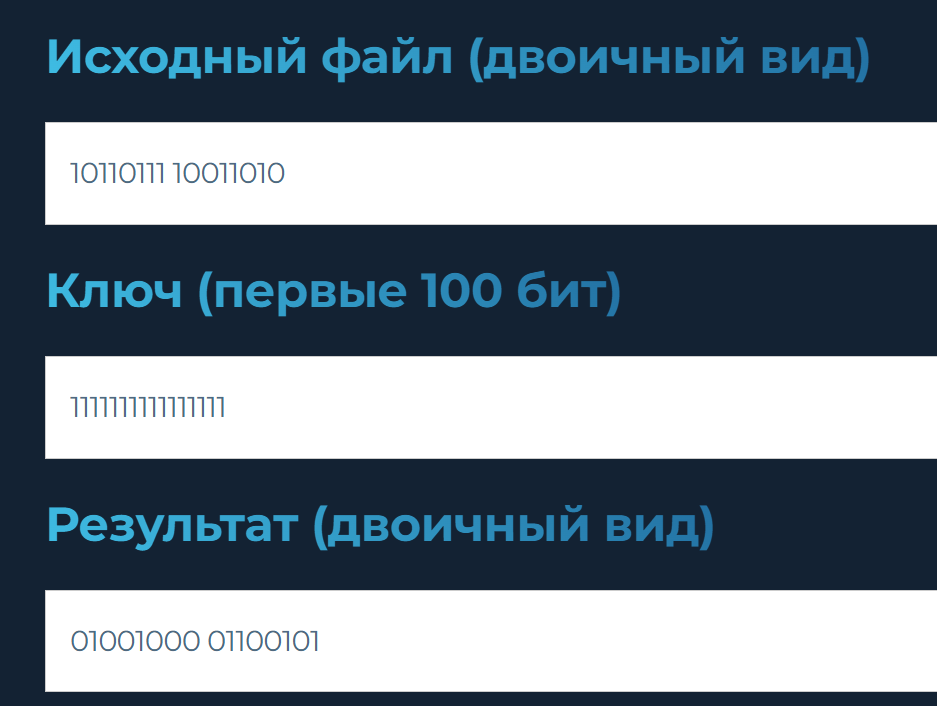
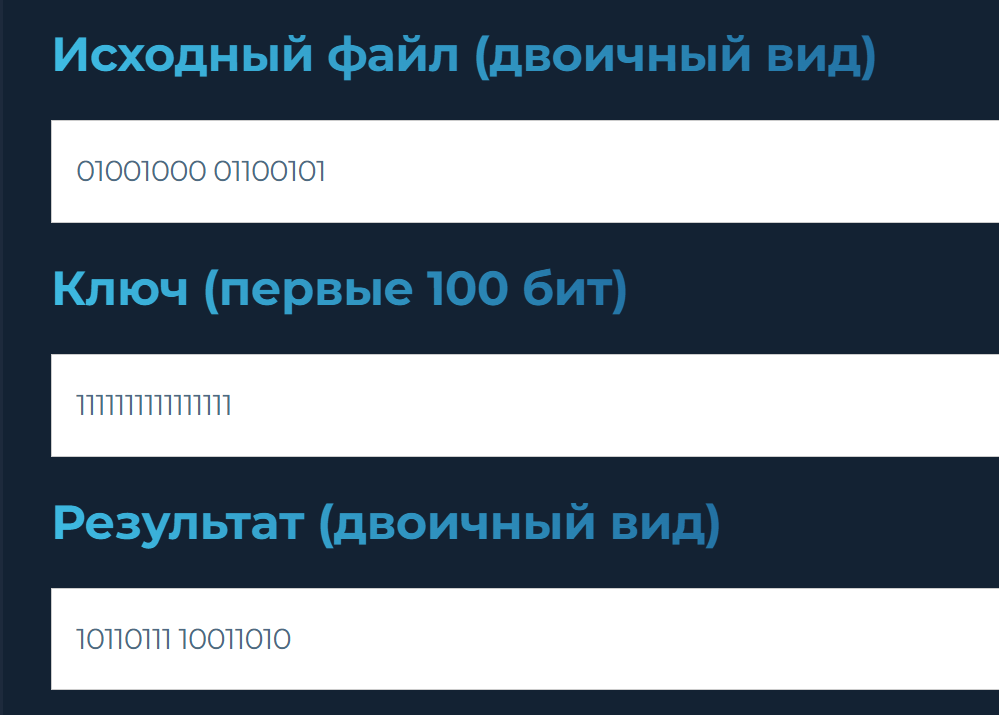


Общий внешний вид страницы:

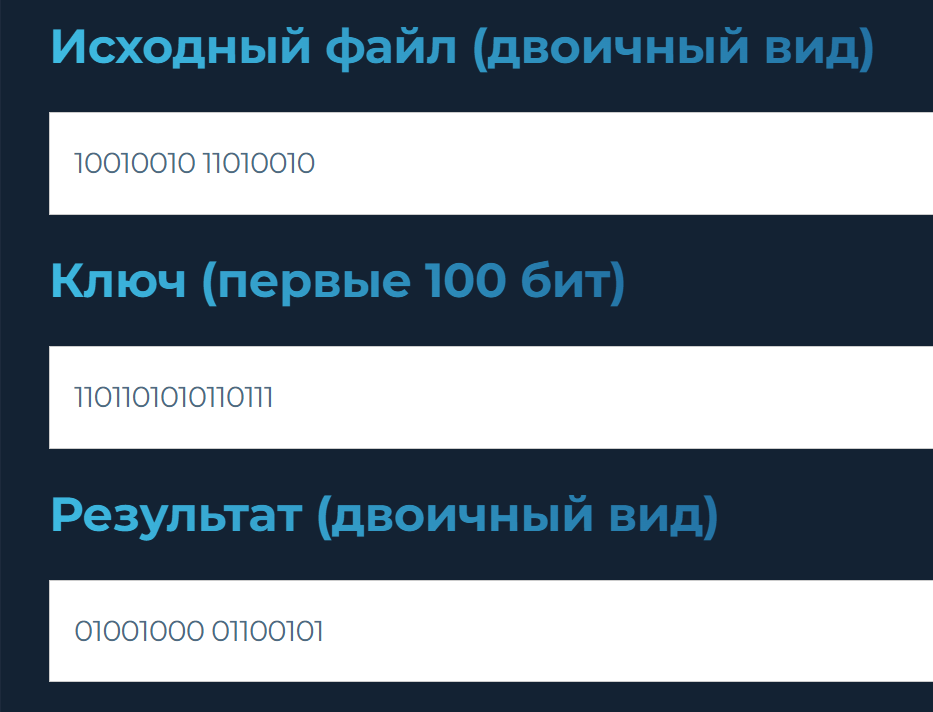
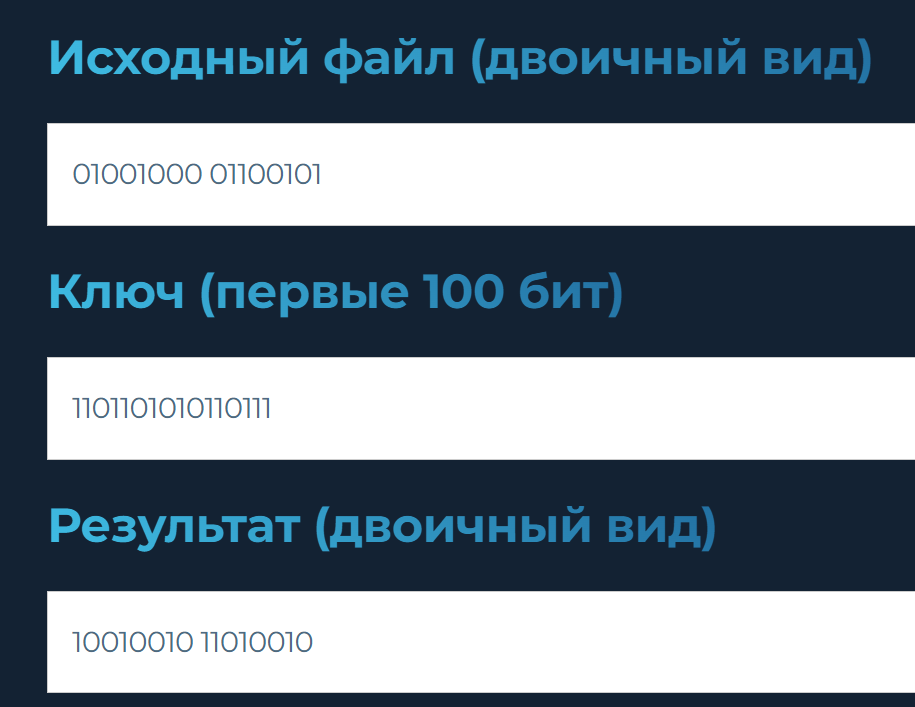


**Тесты**

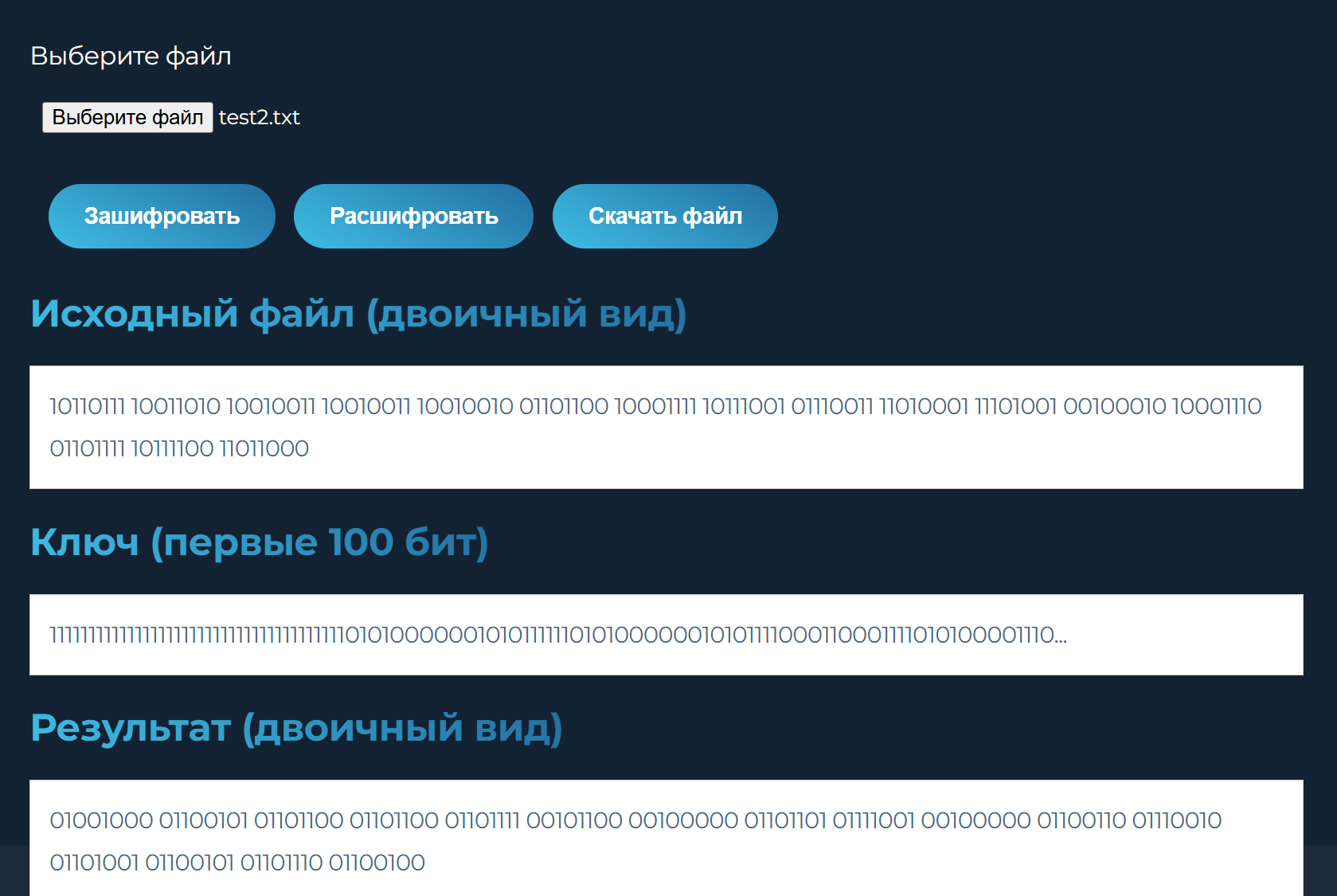
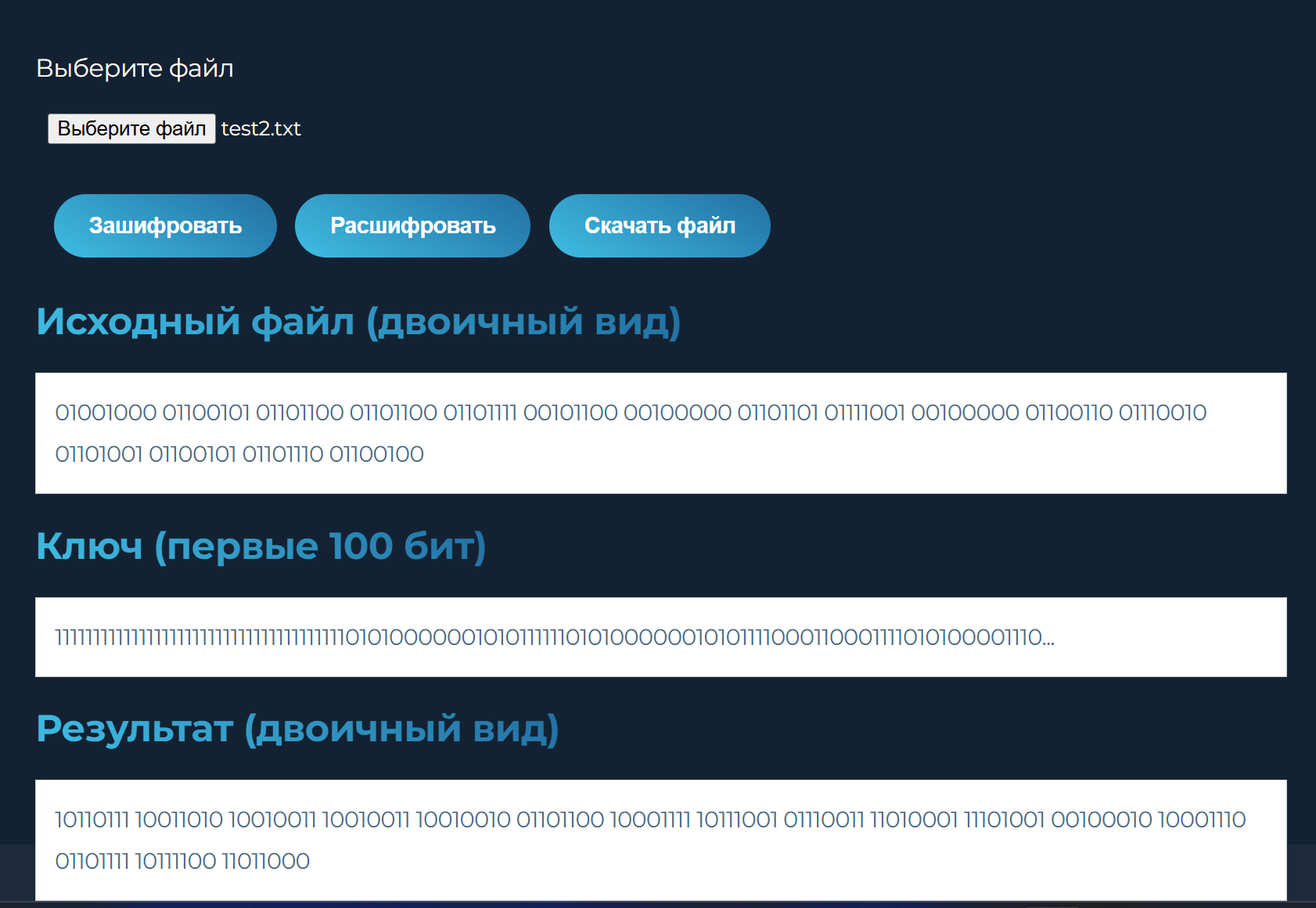
1. Легкий ключ, маленький текст. (зашифрованный/расшифрованный)



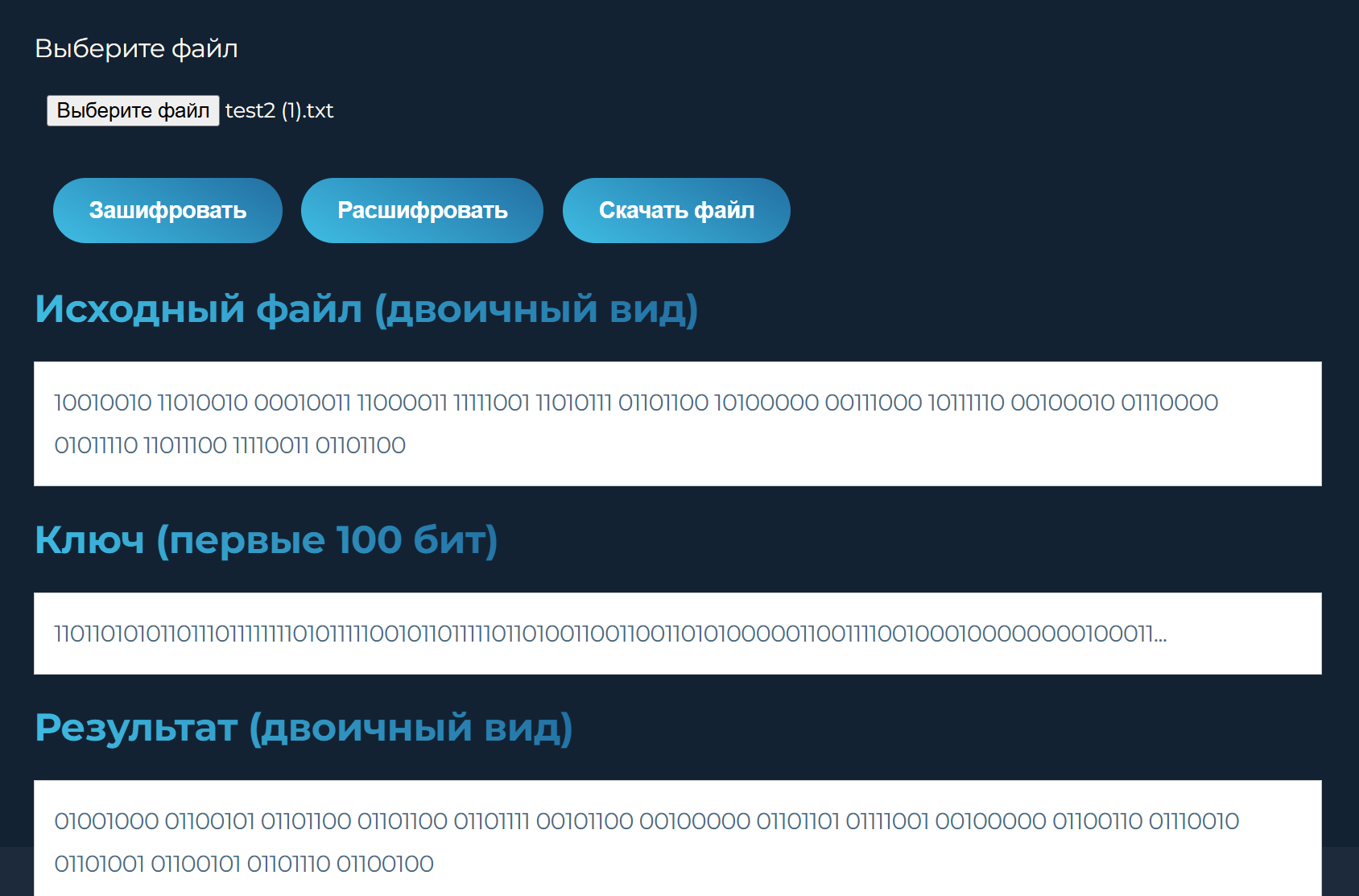
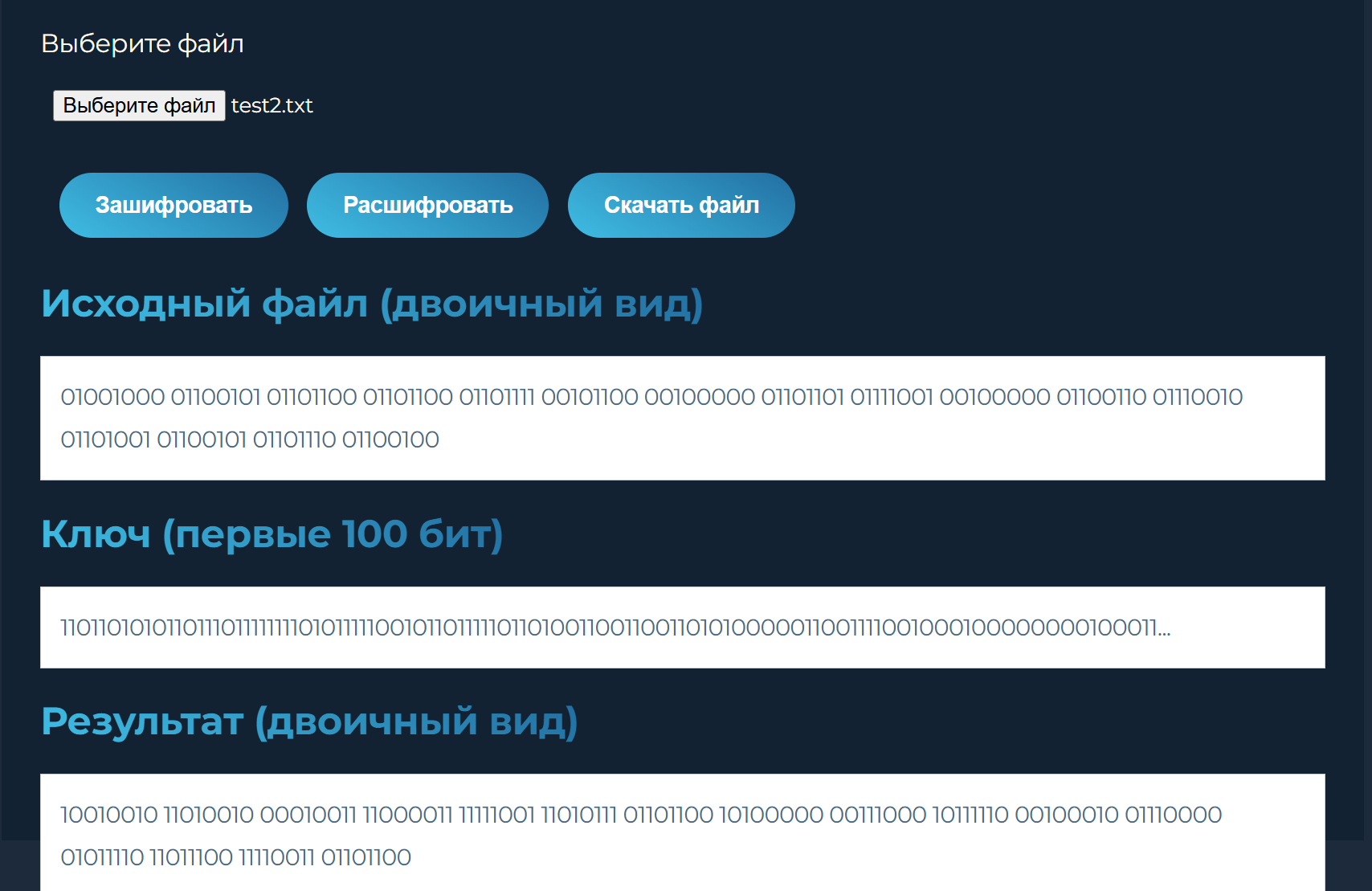
1. Сложный ключ, маленький текст. (зашифрованный/расшифрованный)



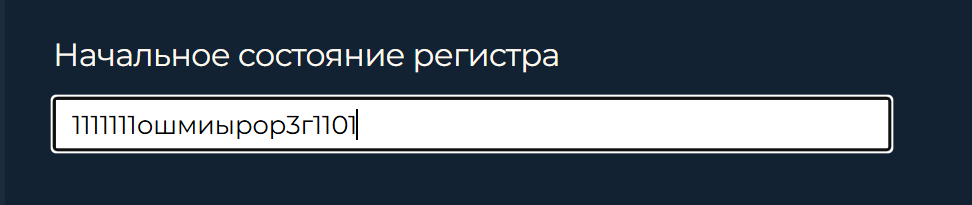
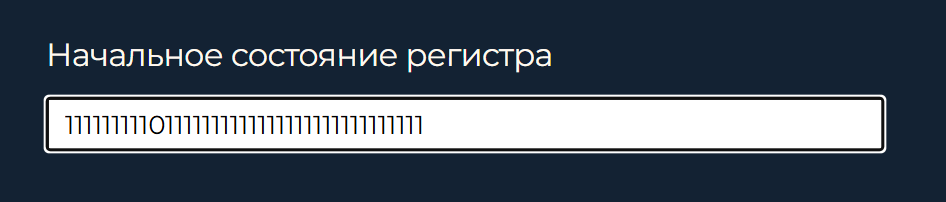
1. Легкий ключ, большой текст. (зашифрованный/расшифрованный)



1. Сложный ключ, большой текст. (зашифрованный/расшифрованный)

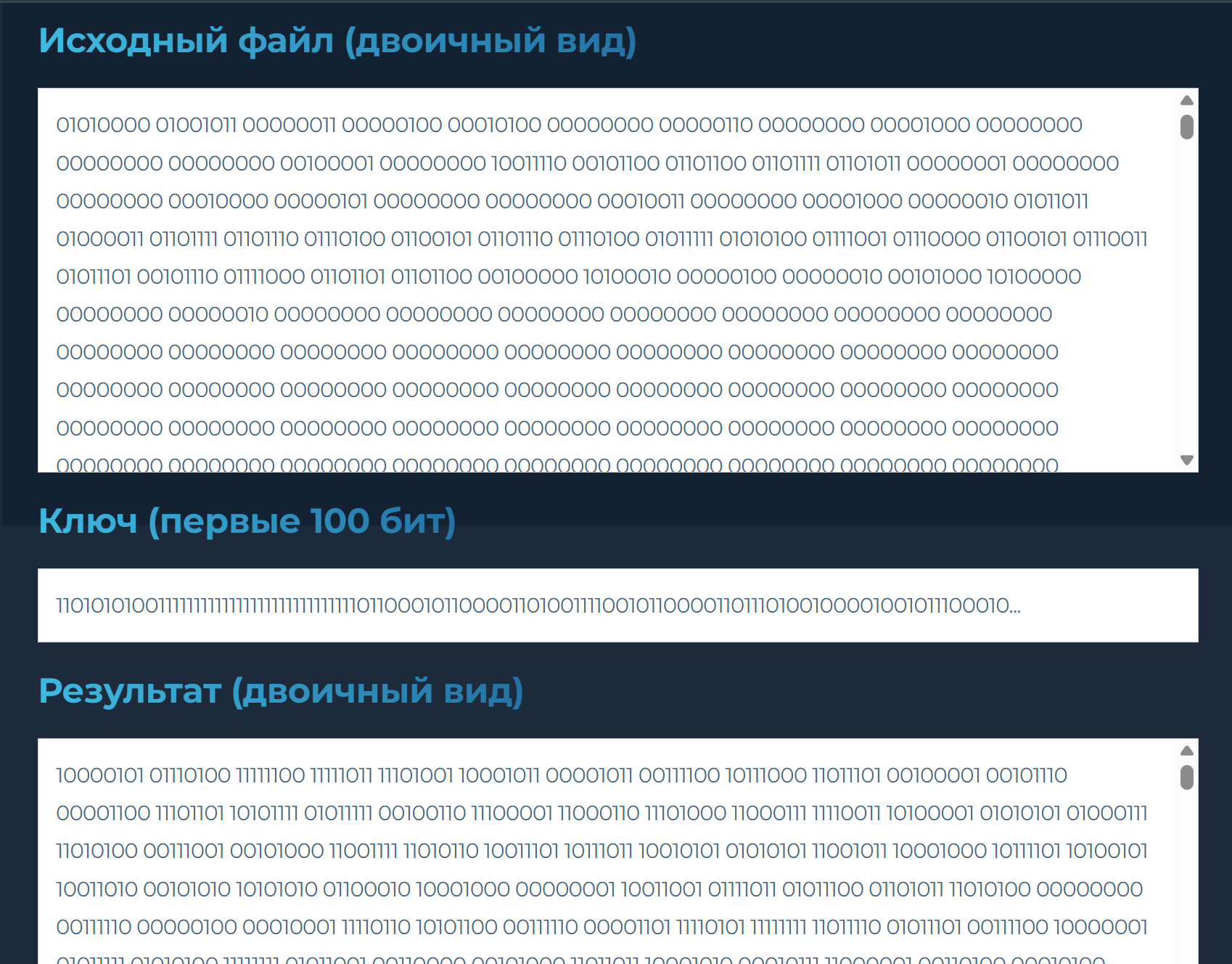
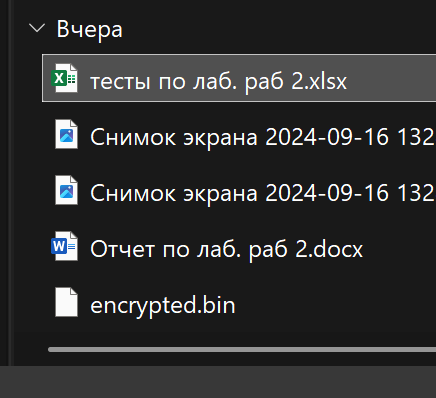


1. Некорректные символы в начальном состоянии LFSR вместе с нехваткой символов до разрядности регистра (до/после)

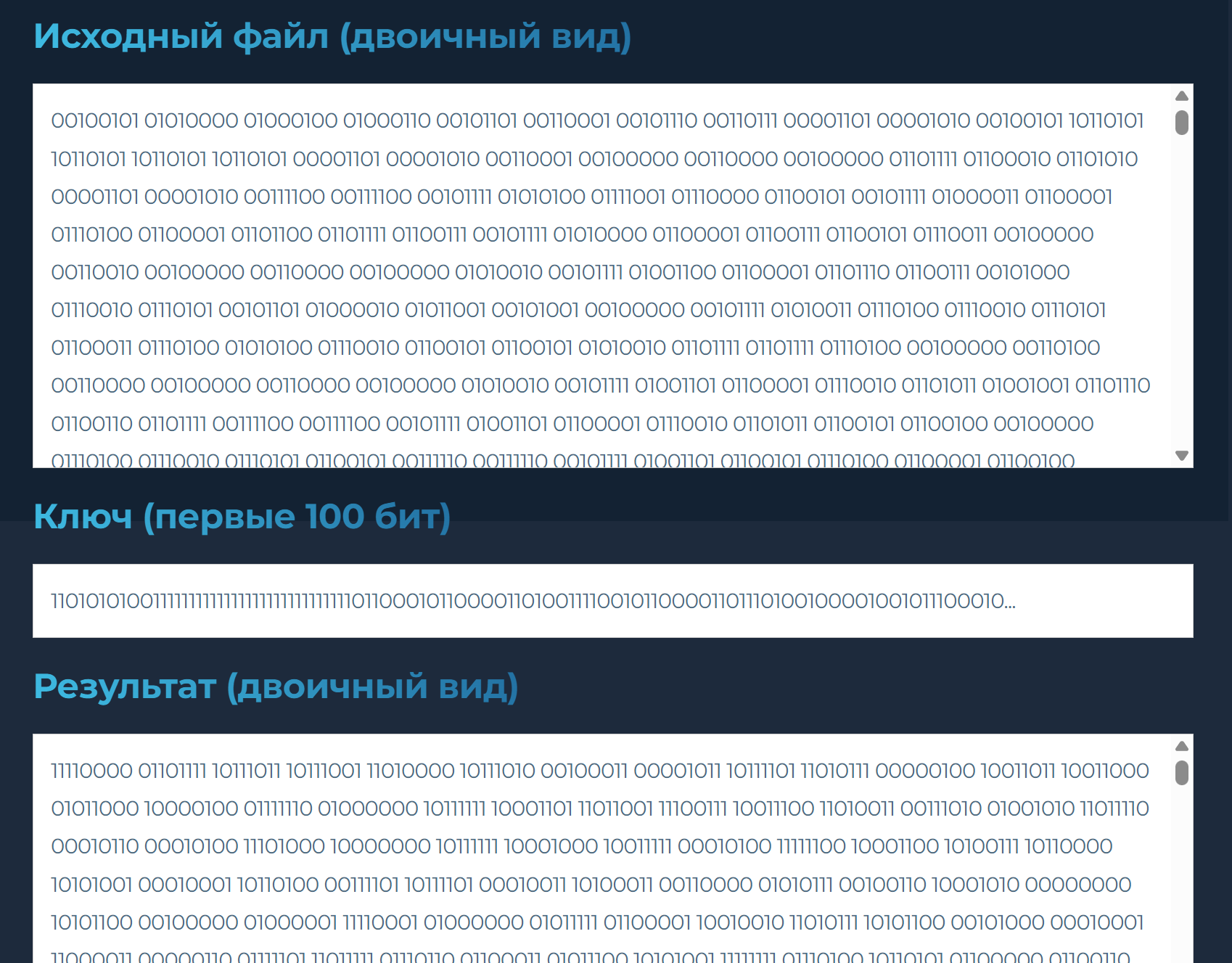
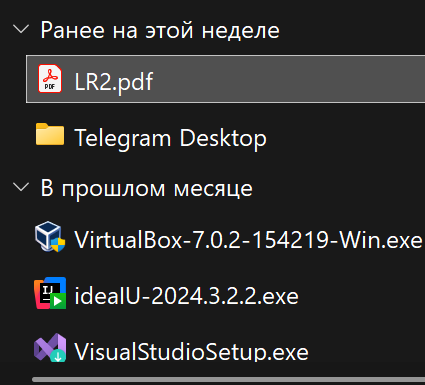
 

**Проверка на разные типы файлов**

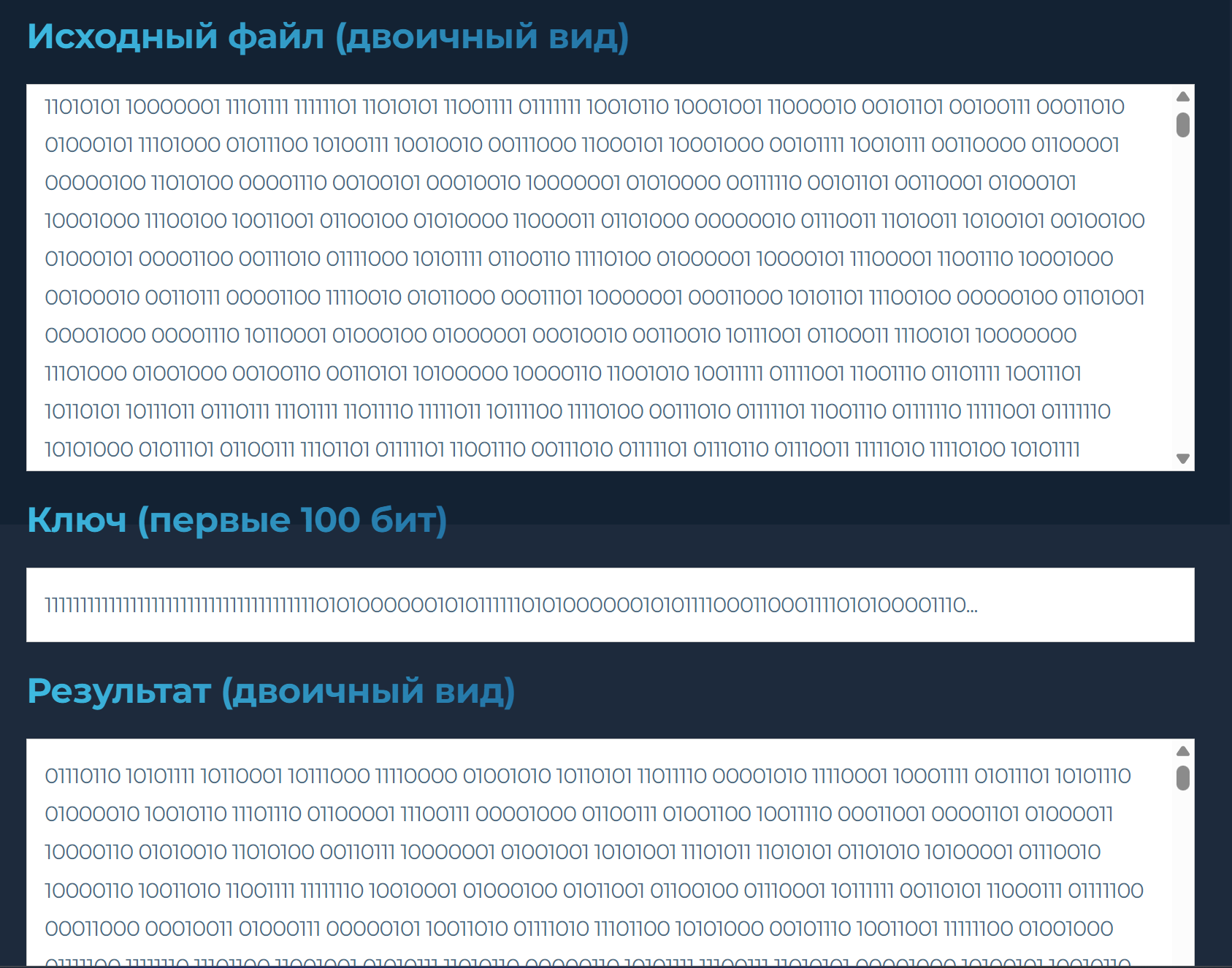
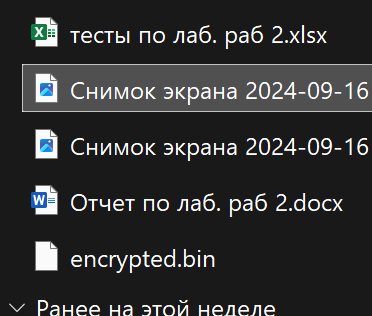
1. Обработка Excel файла (тест пройден успешно):



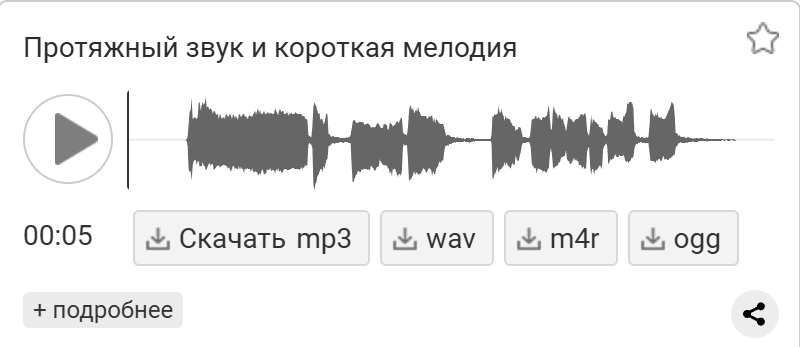
1. Обработка pdf-файла (тест пройден успешно):

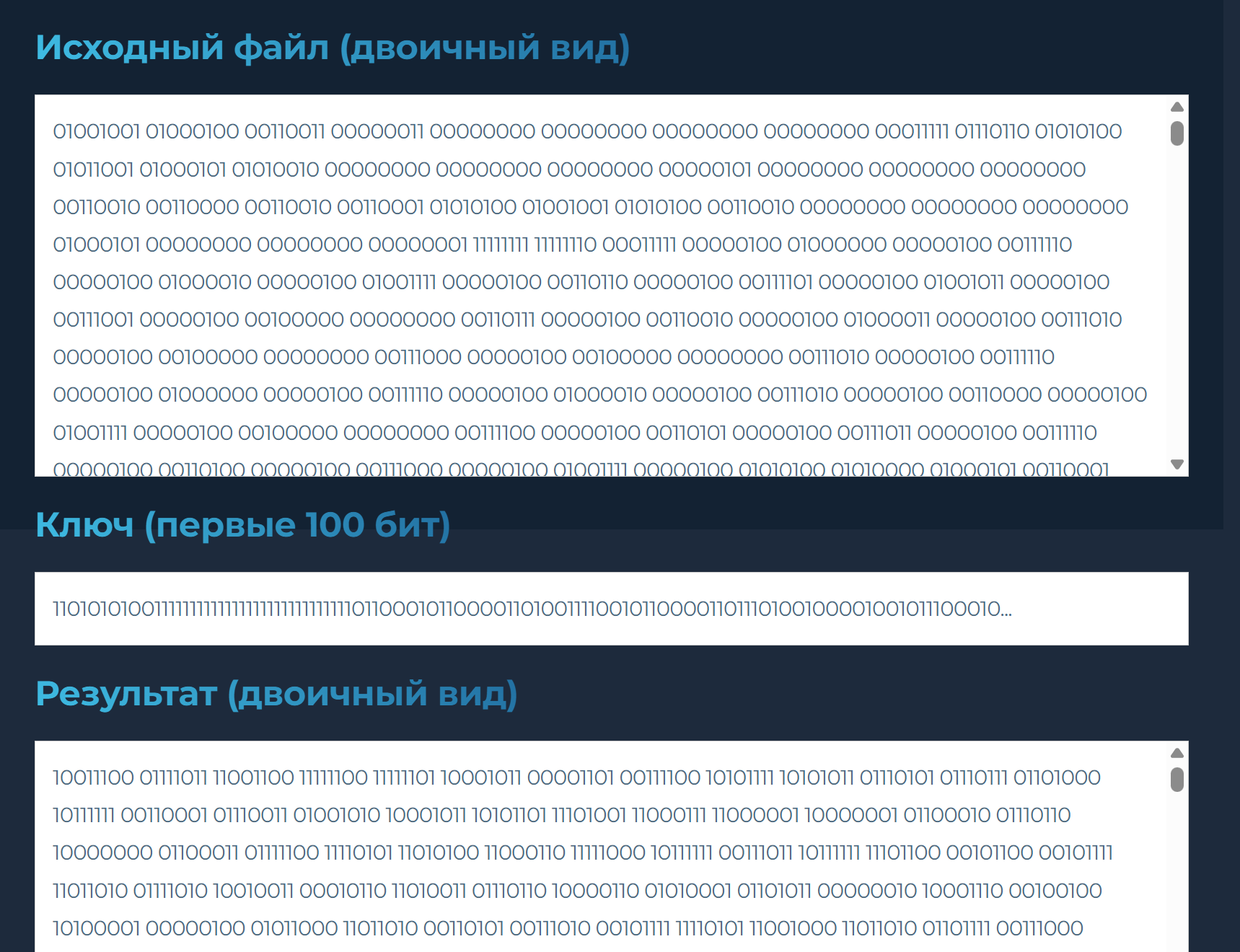
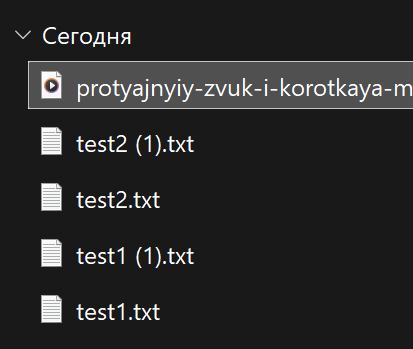


1. Обработка png-файла (тест пройден успешно):



1. Обработка аудио (тест пройден успешно):





**ОБРАТИТЕ ВНИМАНИЕ!**

Лабораторная работа выполнена на JS, который является интерпретируемым языком программирования, а значит, он по сравнению с компилируемыми языками более медленный

Если вы не желаете долго ожидать результат, выбирайте файлы размером не больше 500-1000 КБ, то есть не больше одного мегабайта.

**Исходный код**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>LFSR Stream Cipher</title>

<style>

@import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@100..900&display=swap');

body {

font-family: "Montserrat", serif;

font-size: 20px;

line-height: 163%;

letter-spacing: -0.02em;

color: floralwhite;

margin: 0 auto;

background-color: rgb(29, 42, 60);

}

textarea {

width: 100%;

height: 100px;

resize: none;

}

.container {

max-width: 1000px; /\* Увеличиваем максимальную ширину \*/

margin: 0 auto;

padding: 10px 30px;

background-color: rgb(19, 34, 51);

height: 100vh;

}

input[type="text"], input[type="file"] {

padding: 10px;

margin: 10px 0;

font-size: 16px; /\* Увеличиваем размер шрифта \*/

font-family: "Montserrat", serif;

}

input[type="text"] {

width: 500px;

height: 10px;

background-color: white;

}

input[type="file"] {

width: auto;

}

button {

padding: 10px 20px;

font-size: 16px; /\* Увеличиваем размер шрифта \*/

margin: 10px 5px;

}

.output {

border: 1px solid #ccc;

padding: 15px;

margin-top: 15px;

max-height: 300px; /\* Увеличиваем высоту \*/

overflow-y: auto;

font-size: 18px; /\* Увеличиваем размер шрифта \*/

background-color: white;

color: rgba(0, 38, 71, 0.69);

}

.text-gradient {

background: linear-gradient(223deg, #226fa1 0%, #3fbde4 100%);

background-clip: text;

-webkit-background-clip: text;

-webkit-text-fill-color: transparent;

}

.button {

padding: 15px 28px;

border-radius: 62px;

border: none;

background: linear-gradient(210deg, #226FA1FF 0%, #3FBDE4FF 100%);

font-weight: 639;

font-size: 18px;

letter-spacing: -0.02em;

color: #fff;

transition: 1s;

max-width: 336px;

}

.file {

width: 720px;

}

.button-container {

margin: 10px;

}

</style>

</head>

<body>

<div class="container">

<h1><span class="text-gradient">LFSR Stream Cipher</span></h1>

<form id="cipherForm">

<div class="file">

<label for="initialState">Начальное состояние регистра</label>

</div>

<input type="text" id="initialState" pattern="[01]{38}" required>

<br><br>

<div class="file">

<label for="fileInput">Выберите файл</label>

</div>

<input type="file" id="fileInput" required>

<br>

<div class="button-container">

<button class="button" type="button" id="encryptBtn">Зашифровать</button>

<button class="button" type="button" id="decryptBtn">Расшифровать</button>

<button class="button" type="button" id="downloadBtn">Скачать файл</button>

</div>

</form>

<h2><span class="text-gradient">Исходный файл (двоичный вид)</span></h2>

<div id="originalFile" class="output"></div>

<h2><span class="text-gradient">Ключ (первые 100 бит)</span></h2>

<div id="keyStream" class="output"></div>

<h2><span class="text-gradient">Результат (двоичный вид)</span></h2>

<div id="processedFile" class="output"></div>

</div>

<script>

class LFSRCipher {

constructor(initialState) {

// Преобразуем начальное состояние в массив бит

this.state = Array.from(initialState).map(bit => parseInt(bit));

this.taps = [38, 6, 5, 1, 0]; // Позиции отводов для x^38 + x^6 + x^5 + x + 1

}

// Метод для генерации ключевой последовательности заданной длины

generateKeyStream(length) {

const keyStream = [];

const curState = [...this.state]; // Копируем текущее состояние

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

// Добавляем старший бит в результат

keyStream.push(curState[0]);

// Вычисляем новый бит для обратной связи

let newBit = 0;

for (const tap of this.taps) {

newBit ^= curState[curState.length - tap];

}

// Сдвигаем все биты, кроме первого

for (let j = 0; j < curState.length - 1; j++) {

curState[j] = curState[j + 1];

}

// Добавляем новый бит в конец

curState[curState.length - 1] = newBit;

}

return keyStream;

}

// Метод для шифрования/дешифрования данных

processData(data) {

const dataBits = this.bytesToBits(data); // Преобразуем данные в биты

const keyStream = this.generateKeyStream(dataBits.length); // Генерируем ключевую последовательность

const processedBits = [];

// Применяем XOR между битами данных и ключевой последовательностью

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

processedBits.push(dataBits[i] ^ keyStream[i]);

}

return this.bitsToBytes(processedBits); // Преобразуем биты обратно в байты

}

// Метод для шифрования данных

encrypt(data) {

return this.processData(data);

}

// Метод для дешифрования данных

decrypt(data) {

return this.processData(data);

}

// Вспомогательный метод для преобразования массива байт в массив бит

bytesToBits(bytes) {

let bits = [];

for (let byte of bytes) {

for (let i = 7; i >= 0; i--) {

bits.push((byte >> i) & 1);

}

}

return bits;

}

// Вспомогательный метод для преобразования массива бит в массив байт

bitsToBytes(bits) {

let bytes = [];

for (let i = 0; i < bits.length; i += 8) {

let byte = 0;

for (let j = 0; j < 8; j++) {

byte |= (bits[i + j] << (7 - j));

}

bytes.push(byte);

}

return new Uint8Array(bytes);

}

// Метод для получения ключевой последовательности в виде строки из 0 и 1

getKeyStreamAsString(length) {

const keyStream = this.generateKeyStream(length);

const keyStreamStr = keyStream.join('');

return keyStreamStr;

}

}

// Автодополнение до 38 бит единицами при потере фокуса

document.getElementById('initialState').addEventListener('blur', function (e) {

// Удаляем невалидные символы (оставляем только 0 и 1)

this.value = this.value.replace(/[^01]/g, '');

// Если длина больше 38, выводим ошибку и очищаем поле

if (this.value.length > 38) {

alert('Начальное состояние регистра должно быть не более 38 бит.');

this.value = '';

return; // Прекращаем выполнение, чтобы не дополнять единицами

}

// Если длина меньше 38, дополняем единицами

if (this.value.length < 38) {

this.value = this.value.padEnd(38, '1');

}

});

let lfsr;

let originalData;

let processedData;

document.getElementById('encryptBtn').addEventListener('click', async () => {

await processFile(true);

});

document.getElementById('decryptBtn').addEventListener('click', async () => {

await processFile(false);

});

document.getElementById('downloadBtn').addEventListener('click', () => {

if (!processedData) {

alert('Сначала зашифруйте или расшифруйте файл.');

return;

}

downloadFile(processedData, document.getElementById('fileInput').files[0].name);

});

// Функция для обработки файла

async function processFile(isEncrypting) {

const initialState = document.getElementById('initialState').value;

if (!initialState || initialState.length !== 38) {

alert('Начальное состояние регистра должно быть ровно 38 бит.');

return;

}

const file = document.getElementById('fileInput').files[0];

if (!file) {

alert('Выберите файл.');

return;

}

lfsr = new LFSRCipher(initialState);

originalData = new Uint8Array(await file.arrayBuffer());

processedData = isEncrypting ? lfsr.encrypt(originalData) : lfsr.decrypt(originalData);

// Отображаем двоичные данные

document.getElementById('originalFile').textContent = bytesToBinaryString(originalData);

// Отображаем ключ (первые 100 бит или весь ключ, если он короче)

const keyStream = lfsr.generateKeyStream(originalData.length \* 8);

const keyStreamStr = keyStream.join('');

document.getElementById('keyStream').textContent = keyStreamStr.length <= 100 ? keyStreamStr : keyStreamStr.slice(0, 100) + '...';

document.getElementById('processedFile').textContent = bytesToBinaryString(processedData);

}

// Функция для преобразования байт в двоичную строку

function bytesToBinaryString(bytes) {

return Array.from(bytes)

.map(byte => byte.toString(2).padStart(8, '0'))

.join(' ');

}

// Функция для скачивания файла

function downloadFile(data, fileName) {

const blob = new Blob([data], { type: 'application/octet-stream' });

const url = URL.createObjectURL(blob);

const a = document.createElement('a');

a.href = url;

a.download = fileName;

a.click();

URL.revokeObjectURL(url);

}

</script>

</body>

</html>