Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы защиты информации

ОТЧЁТ
к лабораторной работе №5
на тему

ХЭШ-ФУНКЦИИ

Выполнил: студент гр.253505
Снежко М.А.

Проверил: ассистент кафедры информатики
Герчик А.В.

Минск 2025

СОДЕРЖАНИЕ

1 Цель работы 3
2 Теоретические сведения 4
3 Ход работы 6
Заключение 8
Приложение А (обязательное) Листинг программного кода 9

1 ЦЕЛЬ РАБОТЫ

Целью данной лабораторной работы является изучение принципов построения и практическая реализация криптографических хеш-функций – одного из ключевых инструментов обеспечения информационной безопасности. В ходе работы рассматриваются два алгоритма: отечественный стандарт ГОСТ 34.11-2012 и международный алгоритм *SHA*-1. Эти функции иллюстрируют разные подходы к проектированию хеш-алгоритмов и обладают существенными различиями в архитектуре, криптостойкости и сферах применения.

Актуальность исследования обусловлена фундаментальной ролью хеш-функций в современных криптографических системах. Они лежат в основе механизмов обеспечения целостности данных, аутентификации сообщений, цифровых подписей, блокчейн-технологий и множества других протоколов, где критически важно подтвердить неизменность информации или её подлинность.

Алгоритм ГОСТ 34.11 представляет собой российский криптографический стандарт, обеспечивающий высокий уровень безопасности за счёт 256-битного хеш-значения и сложной внутренней структуры. Он основан на модифицированной сети Фейстеля и использует восемь уникальных *S*-блоков для нелинейных преобразований, что повышает устойчивость к известным методам криптоанализа. Особенностью стандарта является также применение сигма-суммы для дополнительного контроля целостности обрабатываемых данных.

Алгоритм *SHA*-1, несмотря на признанную уязвимость к коллизиям и официальный отказ от его использования в криптографических целях, остаётся важным объектом изучения с образовательной и исторической точек зрения. Он служит основой для понимания эволюции хеш-функций, включая более современные и безопасные стандарты семейств *SHA*-2 и *SHA*-3.

В практической части работы выполняется программная реализация обоих алгоритмов на языке *Javascript*. Особое внимание уделяется, разбиению на блоки, итеративной обработке и формированию итогового хеш-значения.

2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Основное назначение криптографических функций заключается в обеспечении целостности информации, аутентификации данных и построении сложных криптографических протоколов. Криптографически стойкая хеш-функция должна удовлетворять ряду строгих требований, включая детерминированность результатов вычислений, выраженный лавинный эффект, устойчивость к коллизиям и вычислительную необратимость. Детерминированность гарантирует, что идентичные входные данные всегда приводят к одинаковому хеш-значению, что является основой для процедур проверки целостности. Лавинный эффект обеспечивает кардинальное изменение выходного хеша при малейшей модификации входных данных, что исключает возможность прогнозирования результатов. Стойкость к коллизиям подразумевает практическую невозможность нахождения двух различных сообщений с одинаковым хеш-значением, а свойство необратимости делает вычислительно невыполнимым восстановление исходных данных по их хешу.

Алгоритм ГОСТ 34.11, являющийся российским национальным стандартом, демонстрирует сложную многоуровневую архитектуру, основанную на обработке данных блоками по 512 бит с генерацией 256-битного хеш-значения. Уникальной характеристикой стандарта является вычисление сигма-суммы – накопительной суммы всех обрабатываемых блоков данных, обеспечивающей дополнительную защиту от атак, связанных с модификацией порядка блоков.

Алгоритм *SHA*-1, разработанный Агентством национальной безопасности США. Алгоритм обрабатывает входные данные блоками по 512 бит, производя 160-битное хеш-значение через 80 раундов преобразований. Ключевой особенностью является процедура расширения сообщения, при которой каждый блок преобразуется в 80 32-битных слов с использованием рекуррентных формул. Структура алгоритма разделена на четыре этапа по 20 раундов, каждый из которых использует уникальную логическую функцию и константу. Пять 32-битных регистров состояния последовательно обновляются в процессе вычислений, аккумулируя промежуточные результаты, а финальное хеш-значение формируется путем конкатенации этих регистров после обработки всех блоков сообщения.

Таким образом, сравнительный анализ рассмотренных алгоритмов показывает существенные различия в их архитектуре и криптографических свойствах. ГОСТ 34.11 демонстрирует более высокий уровень безопасности благодаря увеличенной длине хеша и сложной структуре преобразований, включая механизм сигма-суммы и множественные *S*-блоки. *SHA*-1, несмотря на историческую значимость и широкое распространение в прошлом, в настоящее время считается криптографически нестойким из-за обнаруженных уязвимостей к атакам поиска коллизий.

3 ХОД РАБОТЫ

Программное средство реализовано при помощи языка программирования *Javascript*. На рисунке 3.1 изображен процесс генерации хэшей двух сообщений, которые отличаются на 1 символ, использован алгоритм ГОСТ 34.11 (512 бит).

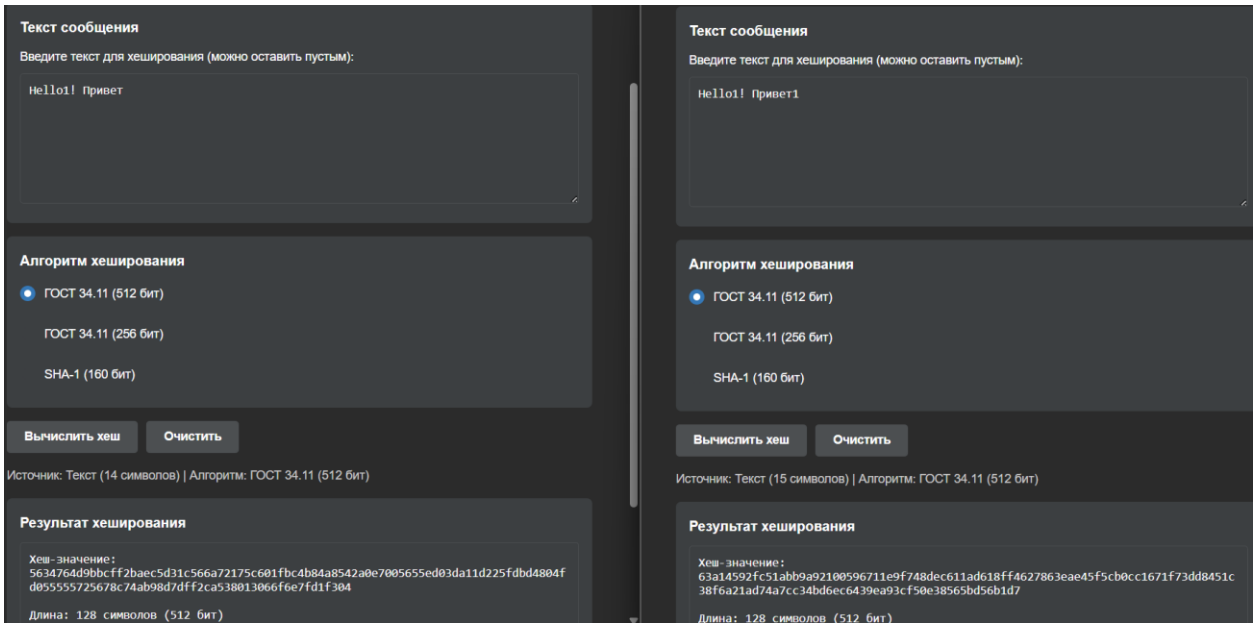

Рисунок 3.1 – Хэши при использовании алгоритма ГОСТ 34.11 (512 бит)

На рисунке 3.2 изображен процесс генерации хэшей двух сообщений, которые отличаются на 1 символ, использован алгоритм *SHA*-1 (160 бит).

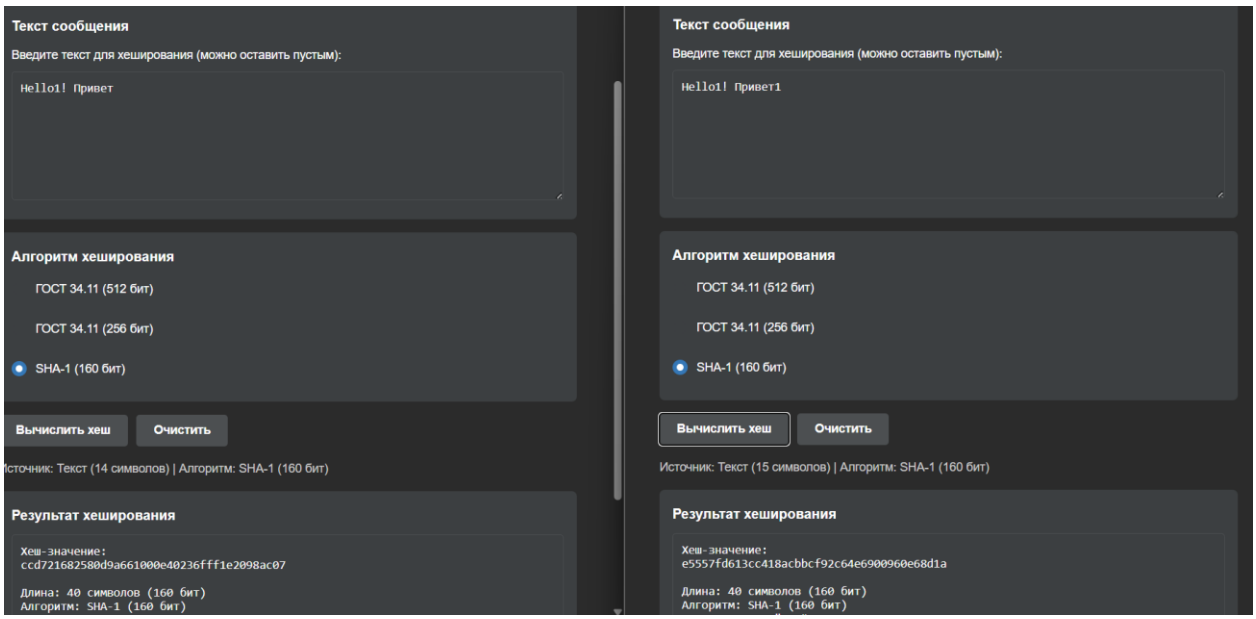

Рисунок 3.2 – Хэши при использовании алгоритма SHA-1 (160 бит)

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были успешно реализованы и исследованы две криптографические хеш-функции: российский стандарт ГОСТ 34.11-2012 и международный алгоритм *SHA*-1. Практическая реализация на языке *Javascript* подтвердила корректность работы обоих алгоритмов: они корректно обрабатывали произвольные пользовательские входные данные. При этом строго соблюдались все требования спецификаций – от дополнения исходного сообщения до пошаговой обработки блоков и формирования итогового хеш-значения.

Известно, что рассмотренные алгоритмы существенно различаются по своим криптографическим свойствам и производительности. Стандарт ГОСТ 34.11 обеспечивает высокий уровень безопасности благодаря 256-битному хешу, сложной структуре преобразований, использованию восьми уникальных *S*-блоков и механизму сигма-суммы. Однако эти особенности обуславливают повышенные вычислительные затраты и большую сложность реализации. В то же время алгоритм *SHA*-1, несмотря на более высокую скорость обработки данных, давно признан криптографически нестойким из-за уязвимости к атакам нахождения коллизий, что делает его непригодным для использования в современных системах, требующих гарантированной защиты информации.

Практическая ценность работы заключается в создании верифицированных, корректно работающих реализаций хеш-функций, которые могут использоваться в образовательных целях для изучения принципов криптографического хеширования, а также служить основой для разработки более сложных криптографических приложений. Особенно актуальной является реализация ГОСТ 34.11, широко применяемого в российских системах защиты информации и представляющего значительный интерес с точки зрения изучения отечественных криптографических стандартов.

Проведённое исследование наглядно иллюстрирует эволюцию подходов к проектированию хеш-функций и подчёркивает необходимость своевременного перехода на современные, криптографически стойкие алгоритмы в условиях непрерывного развития методов криптоанализа.

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг программного кода

```
class HashApp {
    constructor() {
        this.gost_512 = new GOST3411(512);
        this.gost_256 = new GOST3411(256);
        this.sha1 = new SHA1();
        this.lastResult = null;

        this.initializeEventListeners();
    }
    calculateHash() {
        try {
            const data = document.getElementById('text-input').value;
            const                          algorithm                                =
document.querySelector('input[name="algorithm"]:checked').value;
            let hashResult, algoName;
            switch (algorithm) {
                case "gost_512":
                    hashResult = this.gost_512.hashHex(data);
                    algoName = "ГОСТ 34.11 (512 бит)";
                    break;
                case "gost_256":
                    hashResult = this.gost_256.hashHex(data);
                    algoName = "ГОСТ 34.11 (256 бит)";
                    break;
                case "sha1":
                    hashResult = this.sha1.hashHex(data);
                    algoName = "SHA-1 (160 бит)";
                    break;
            }
            document.getElementById('info-label').textContent = `${sourceInfo}
| Алгоритм: ${algoName}`;
            const resultText = `Хеш-значение:\n${hashResult}\n\n` +
                               `Длина:       ${hashResult.length}       символов
(${hashResult.length * 4} бит)\n` +
                               `Алгоритм: ${algoName}\n` +
                               `Источник: ${!data ? 'пустая строка' : 'введённый
текст'}`;
            document.getElementById('result-text').value = resultText;
            this.lastResult = {
                hash: hashResult,
                algorithm: algoName,
                source: sourceInfo,
                data: data
            };
        } catch (error) {
            alert(`Ошибка при вычислении хеша: ${error.message}`); }
    }
    clearAll() {
        document.getElementById('text-input').value = '';
        document.getElementById('result-text').value = '';
        document.getElementById('info-label').textContent = '';
        this.lastResult = null;
    }
}
document.addEventListener('DOMContentLoaded', () => {
    new HashApp();
});
```