Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Направление 09.03.01 – «Информатика и вычислительная техника»

Дисциплина: «Защита информации»

Профиль: «Автоматизированные системы обработки информации и управления»

Семестр 7

ОТЧЕТ

по лабораторной работе №7

Тема: «Методы хеширования паролей»

Выполнил: студент группы АСУ-17-1б

Хохряков Денис

Проверил: доцент кафедры ИТАС

Шереметьев В.Г.\_\_\_\_\_\_\_\_\_\_

Дата\_\_\_\_\_\_

# ЦЕЛЬ РАБОТЫ

Получить практические навыки по созданию алгоритмов хеширования паролей. Создать свой алгоритм на основе блочного шифрования из лабораторной работы 4.

# ЗАДАНИЕ

Написать программу, реализующую методику хеширования паролей, используя в качестве блочного шифра DES с длинной блоков 32 бит и длиной ключа 16 бит.

# КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ

Разработчикам приложений часто приходится заниматься разработкой систем учетных записей пользователей. Самый важный аспект системы учетных записей — защита паролей пользователей. Взлом баз данных — не самое редкое явление, поэтому нужно что-то предпринимать, чтобы обеспечить защиту личных данных пользователей, в том числе паролей, в подобных случаях. Самый лучший способ обеспечения безопасности паролей — их хеширование.

Хеширование — процесс необратимый. Это означает, что, имея хеш некоторой сущности, невозможно восстановить саму сущность. Или простым языком: нельзя получить исходный пароль при наличии его хеша.

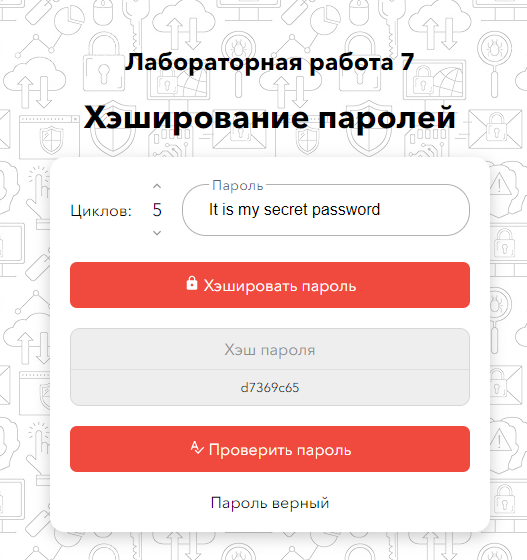
Алгоритм хеширования превращает исходную строку в другую строку фиксированного размера, которую можно рассматривать как ее “отпечаток пальца” — единственный и неповторимый, принадлежащий только это строке. Это отличная защита для паролей. Даже если база данных с паролями вдруг будет взломана, злоумышленник не сможет заполучить сами пароли: ему будут доступны только хешы, с которыми далеко не уедешь.

# ХОД РАБОТЫ

1. На первом шаге преобразовываем строку в набор байтов
2. Каждые два символа складываем в один блок и используем в качестве ключа. Дальше шифруем нулевой блок этим ключом шифром DES
3. Складываем все зашифрованные блоки вместе
4. В итоге получается вполне себе алгоритм хеширования, который крайне сложно разгадать, а любое, даже малейшее изменение, приводит к изменению всего хеша.

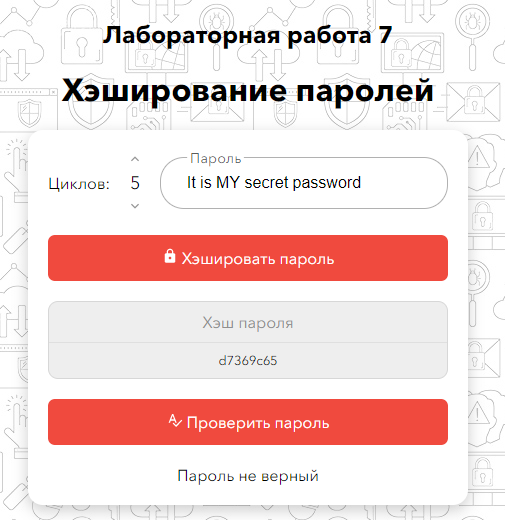
# РАБОТА ПРОГРАММЫ

Мы вводим пароль и нажимаем на кнопку «хешировать пароль», после чего наш пароль хешируется блочным шифром DES с указанным кол-вом циклов.



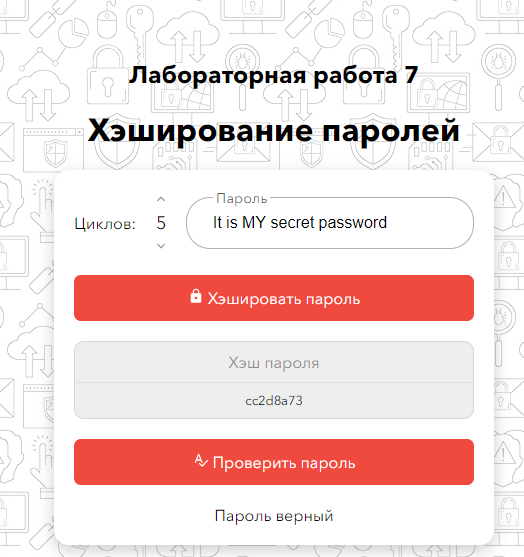
1. Хеширование пароля

Теперь, если мы немного изменим пароль и нажмем на кнопку «проверить пароль», то программа выдаст, что пароль не верный



. Проверка пароля

И если мы повторно хешируем новый пароль, то увидим, что хеш изменился полностью (был - d7369c65, а стал - cc2d8a73)



3. Новый хеш для нового пароля

# ВЫВОД

На самом деле данный алгоритм является не более чем учебным примером, чтобы дать теоритические основы о хешировании. На практике же используются проверенные алгоритмы, такие как SHA256, и, чаще всего, bcrypt.

При этом хешируется не только один пароль, а конкатенация пароля и соли. Соль позволяет защитить хеш пароля от «радужных таблиц» - больших баз данных с различными наборами хешей большинства паролей. При этом если хеши паролей были скомпрометированы вместе с «солью», то злоумышленнику все равно остается только перебор всех вариантов паролей, что является довольно затратной операцией.

# КОД ПРОГРАММЫ (ФУНКЦИЯ ХЕШИРОВАНИЯ)

# (Язык программирования – JavaScript)

//Функция вычисления хеша

const getHash = (str, cycles) => {

const utf8 = [];

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

const char = str.charCodeAt(i);

while(char !== 0){

utf8.push(char & ~(~0<<8));

char = char >>> 8;

}

}

let lastA = 0;

for(let i = 0; i < utf8.length; i+=2){

const key = (utf8[i] << 8) + (utf8[i+1] || 0);

//Функция cipher (str, key, cycles) взята из 4 лабораторной

const a = cipher(0, key, cycles);

lastA = lastA ^ a;

}

if(lastA < 0)

lastA = 0xFFFFFFFF + lastA + 1;

return lastA;

}