**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський державний університет   
імені Петра Могили**

**Факультет комп’ютерних наук**

**Кафедра <Інтелектуальних інформаційних систем>**

**ЗВІТ**

*з лабораторної роботи № 2*

# "Відтворення алгоритму шифрування MD5"

Напрям підготовки: **Комп’ютерні науки**

122-ЛР.ПЗ.02-402.1610219

***Cтудентка*** *\_\_\_\_\_\_\_\_\_\_\_\_\_****С.О.******Рагуліна***

*(підпис)*

*\_\_22.09.2019 \_*

*(дата)*

***Викладач***

*\_\_\_\_\_\_\_\_\_\_\_\_\_* ***Тогоєв О. Р***

*(підпис)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

**Миколаїв – 2019**

# Лабораторна робота №2

# Тема: «Відтворення алгоритму шифрування MD5».

Завдання:

1. Розробити форму реєстрації з записом у БД за допомогою JS/PHP.
2. Зашифрувати дані форми MD5.
3. Оформити звіт.

**Теоретичні відомості**

MD5 - 128-бітний алгоритм хешування, розроблений професором Рональдом Л. Ривестом з Массачусетського технологічного інституту (Massachusetts Institute of Technology, MIT) в 1991 році. Призначений для створення «відбитків» або дайджестів повідомлення довільної довжини і подальшої перевірки їх достовірності. Широко застосовувався для перевірки цілісності інформації та зберігання хешів паролів.

На вхід алгоритму надходить вхідний потік даних, хеш якого необхідно знайти. Довжина повідомлення вимірюється в бітах і може бути будь-який (в тому числі нульовою). Запишемо довжину повідомлення в L. Це число ціле і невід'ємне. Кратність будь-яким числам необов'язкова. Після надходження даних йде процес підготовки потоку до обчислень.

Хеш-функція призначена для згортки вхідного масиву будь-якого розміру в бітову рядок, для MD5 довжина вихідний рядки дорівнює 128 бітам. Для чого це потрібно? Наприклад у вас є два масиви, а вам необхідно швидко порівняти їх на рівність, то хеш-функція може зробити це за вас, якщо у двох масивів хеші різні, то масиви гарантовано різні, а в разі рівності хеш - масиви швидше за все рівні.

Однак найчастіше хеш-функції використовуються для перевірки унікальності пароля, файлу, рядка і тд. Наприклад, завантажуючи файл з інтернету, ви часто бачите поруч з ним рядок виду b10a8db164e0754105b7a99be72e3fe5 - це і є хеш, прогнавши цей файл через алгоритм MD5 ви отримаєте такий рядок, і, якщо хеши рівні, можна з великою ймовірністю стверджувати що цей файл дійсно справжній (звичайно з деякими застереженнями, про які розповім далі).

Алгоритм складається з п'яти кроків:

1) Append Padding Bits

У вихідний рядок дописують одиничний байт 0х80, а потім дописують нульові біти, до тих пір, поки довжина повідомлення не буде порівнянна з 448 по модулю 512. Тобто дописуємо нулі до тих пір, поки довжина нового повідомлення не буде дорівнює [довжина] = ( 512 \* N + 448),

де N - будь-яке натуральне число, таке, що цей вислів буде найближче до довжини блоку.

2) Append Length

Далі в повідомлення дописується 64-бітове представлення довжини вихідного повідомлення.

3) Initialize MD Buffer

На цьому кроці инициализируется буффер

word A 01 23 45 67

word B: 89 ab cd ef

word C: fe dc ba 98

word D: 76 54 32 10

Як можна помітити буффер складається з чотирьох констант, призначений для збору хеша.

4) Process Message in 16-Word Blocks

На четвертому кроці в першу чергу визначається 4 допоміжні логічні функції, які перетворять вхідні 32-бітові слова, в, як не дивно, в 32-бітові вихідні.

F (X, Y, Z) = XY v not (X) Z

G (X, Y, Z) = XZ v Y not (Z)

H (X, Y, Z) = X xor Y xor Z

I (X, Y, Z) = Y xor (X v not (Z))

Також на цьому етапі реалізується так званий «білий шум» - посилення алгоритму, що складається 64 елементного масиву, що містить псевдовипадкові числа, залежні від синуса числа i:

T [i] = 4,294,967,296 \* abs (sin (i))

Далі починається «магія». Копіюємо кожен 16-бітний блок в масив X [16] і виробляємо маніпуляції:

AA = A

BB = B

CC = C

DD = D

Потім відбуваються «чудесні» перетворення-раунди, яких всього буде 4. Кожен раунд складається з 16 елементарних перетворень, які в загальному вигляді можна представити у вигляді [abcd ksi], яке, в свою чергу, можна уявити як A = B + (( A + F (B, C, D) + X [k] + T [i]) <<< s), де

A, B, C, D - регістри

F (B, C, D) - одна з логічних функцій

X [k] - k-тий елемент 16-бітного блоку.

T [i] - i-тий елемент таблиці «білого шуму»

<<< s - операція циклічного зсуву на s позицій вліво.

Наводити всі раунди не має сенсу, все їх можна подивитися тут

Ну і в кінці підсумовуємо результати обчислень:

A = A + AA

B = B + BB

C = C + CC

D = D + DD

5) Output

Виводячи побайтово буффер ABCD починаючи з A і закінчуючи D отримаємо наш хеш.

**Хід роботи**

Лабораторну роботу було виконано за допомогою мови PHP.

1. Створення форми реєстрації за допомогою HTML, яка містить два поля – email та пароль. Поле пароль буде хешуватися.

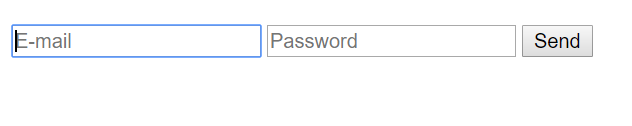


Рис 1. – Форма реєстрації.

1. При успішній обробці форми буде виведено повідомлення, в якому зазначено зашифрований пароль, і додано дані юзера до бази.

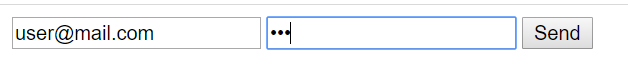


Рис 2. – Заповнення форми.



Рис 3. – Результат роботи.

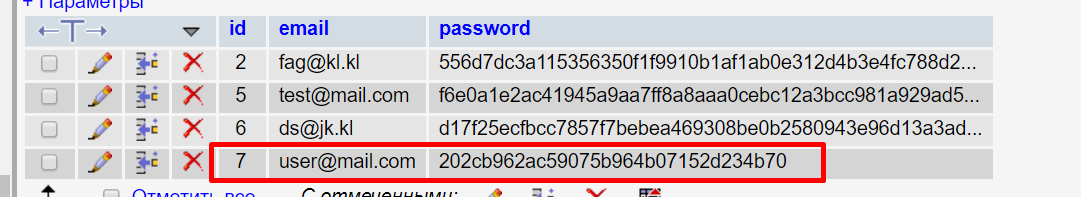


Рис 4. – Додавання інформації користувача до бази даних.

**Лістинг програми:**

Index.html

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8">  
 <title>  
 Register Form  
 </title>  
</head>  
<body>  
 <div class="form\_section">  
 <form action="./handler.php" method="post">  
 <input type="email" name="form\_\_email" required placeholder="E-mail">  
 <input type="password" name="form\_\_password" required placeholder="Password" >  
 <input type="submit" name="form\_\_submit" value="Send">  
 </form>  
 </div>  
</body>  
</html>

handler.php

**<?php  
require**("./src/Db.php");  
**require**("./src/UsersDbHandler.php");  
  
$email = **isset**($\_POST['form\_\_email']) ? $\_POST['form\_\_email'] : **null**;  
$password = **isset**($\_POST['form\_\_password']) ? $\_POST['form\_\_password'] : **null**;  
**if** (($email === **null**) || ($password === **null**)) {  
 **echo** "Data is not valid";  
 **die**();  
}  
$connection = Db::*getInstance*();  
$db = **new** UsersDbHandler($connection);  
$queryRes = $db->addUser($email, $password);  
$queryRes = json\_decode($queryRes, **true**);  
**if** (!$queryRes["status"]) {  
 **echo** $queryRes["message"];  
 **die**();  
}  
**echo** "User has been added, hashed password:" . $queryRes["password"];

Клас Db для встановлення з’єднання з базою

**<?php  
  
class** Db  
{  
 **static private** *$instance* = **null**;  
  
 **private function** \_\_construct()  
 {  
  
 }  
  
 **public static function** getInstance()  
 {  
 **if** (!**self**::*$instance*) {  
 **try** {  
 $host = "127.0.0.1";  
 $port = "3360";  
 $dbname = "tzi\_lr1";  
 $dbuser = "root";  
 $dbpassword = "";  
 $dsn = 'mysql:host=' . $host . ';port=' . $port . ';dbname=' . $dbname;  
 **self**::*$instance* = **new** PDO($dsn, $dbuser, $dbpassword);  
 } **catch** (PDOException $e) {  
 **echo** $e->getMessage();  
 **echo** json\_encode([  
 'status' => **false**,  
 'line' => *\_\_LINE\_\_*,  
 'body' => [  
 'error\_key' => 'PDOException'  
 ]  
 ]);  
 $instance = **null**;  
 }  
  
 }  
 **return self**::*$instance*;  
 }  
}

Клас UsersDbHandler для роботи з таблицею користувачів

**<?php  
  
class** UsersDbHandler  
{  
 **private** $instance;  
  
 **public function** \_\_construct($connection)  
 {  
 $this->instance = $connection;  
 }  
  
 **public function** addUser($email, $password)  
 {  
 $query = $this->instance->prepare('INSERT INTO users\_table (email, password) values (:email,:password)');  
 $hashedPassword = md5($password);

$query->bindParam(':email', $email);  
 $query->bindParam(':password', $hashedPassword);  
 **if** ($query->execute()) {  
 **return** json\_encode(  
 **array**(  
 "status" => **true**,  
 "password" => $hashedPassword  
 )  
 );  
 }  
 **return** json\_encode(  
 **array**(  
 "status" => **false**,  
 "message" => "Error occurred"  
 )  
 );  
  
 }  
}

**Висновок:** під час лабораторної роботи було освоєно алгоритм шифрування MD5. В ході роботи створено форму для користувачів, в результаті якої дані буде зашифровано і додано до бази даних. Закріплено навички роботи з мовою програмування PHP.