САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ – ПРОЦЕССОВ УПРАВЛЕНИЯ

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Алгоритмы и структуры данных»

на тему «Исследование хеш-функций»

Вариант 12

Выполнила
студентка 2 курса
группы 21-Б15.ПУ
Павлова Ксения Андреевна

Преподаватель
Дик Александр Геннадьевич

Санкт-Петербург
2023

СОДЕРЖАНИЕ

Цель: расшифровать набор данных с номерами телефона, зашифрованный с помощью хеш-функции, и исследовать трудоёмкость расшифровки в зависимости от модификатора входа – соли, и от метода шифрования.

Формулировка задачи: имеется набор входных данных, состоящий из 50 тысяч значений хеш-функции. Известно, что номера телефонов захешированы с помощью алгоритма md5. Статическая соль добавлялась к каждому номеру перед шифрованием. Для определения соли даны 5 номеров, которые точно содержатся в наборе данных в зашифрованном виде.

Задачи:

1. Расшифровать хеш-функции с помощью инструмента «Hashcat»;
2. Выявить модификатор входа;
3. Представить набор данных с «чистыми» номерами;
4. Исследовать трудоёмкость расшифровки в зависимости от модификатора входа и метода шифрования (использовать 3 метода хэширования).

Ход работы.

Расшифровка хеш-функции.

Первичная расшифровка проводилась с помощью атаки по маске из 11 символов (цифры) с использованием инструмента «Hashcat».

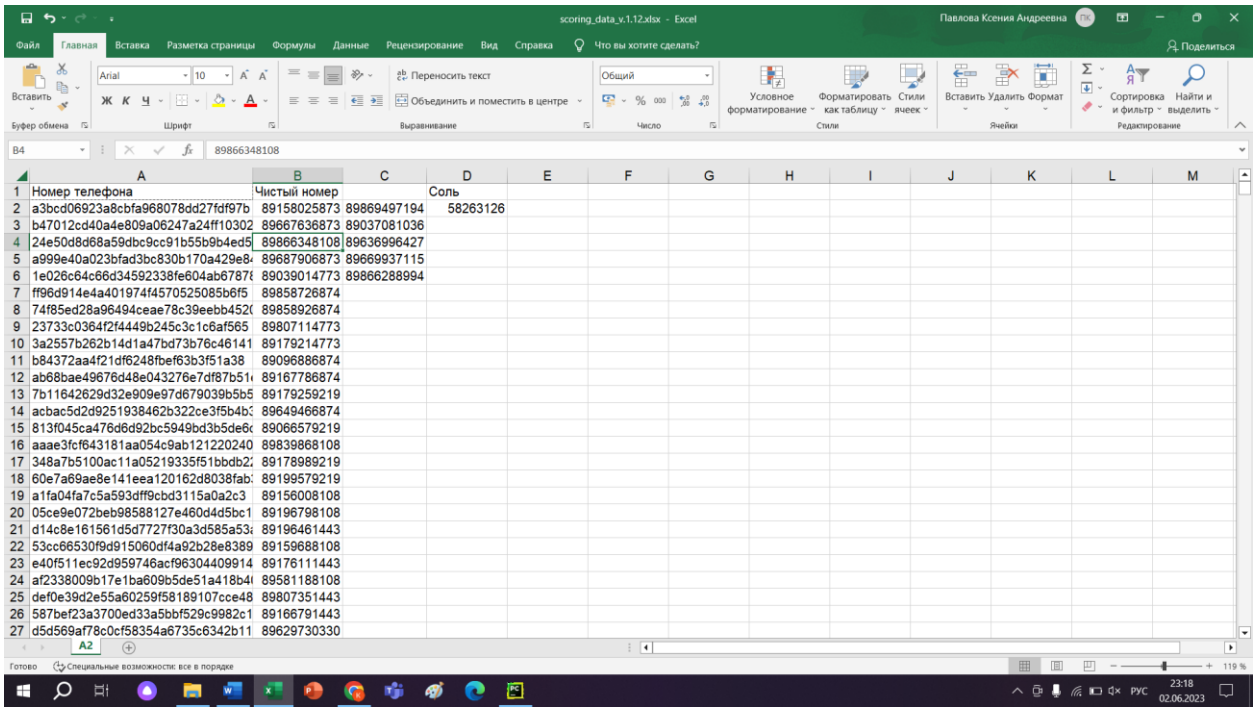
Команда для расшифровки: < hashcat -m 0 -a 3 hash.txt -o dehash.txt ?d?d?d?d?d?d?d?d?d?d?d --force>.

Поиск модификатора входа хеш-функции и восстановление датасета.

На входе имеется набор номеров телефонов с «солью». Путём вычитания из каждого номера из набора каждого из пяти номеров, гарантированно имеющихся в датасете, и составления словаря, где ключами являются полученные данные найдём «соль». Ей является единственный ключ со значением 5.

Данным методом было найдено значение «соли»: 58263126.

Вычтем полученное значение «соли» из каждого номера и с помощью библиотеки openpyxl для языка программирования Python запишем «чистые» номера в таблицу.



	A	B	C	D
1	Номер телефона	Чистый номер		Соль
2	a3bcd06923a8cbfa968078dd27fdf97b	89158025873	89869497194	58263126
3	b47012cd40a4e809a06247a24ff10302	89667636873	89037081036	
4	24e50d8d68a59dbc9cc91b55b9b4ed5	89866348108	89636996427	
5	a999e40a023bfad3bc830b170a429e8	89687906873	89669937115	
6	1e026c64c66d34592338fe604ab6787	89039014773	89866288994	
7	ff96d914e4a401974f4570525085b6f5	89858726874		
8	74f85ed28a96494ceae78c39eebb452	89858926874		
9	23733c0364f2f4449b245c3c1c6af565	89807114773		
10	3a2557b262b14d1a47bd73b76c46141	89179214773		
11	b84372aa4f21df6248fbef63b3f51a38	89096886874		
12	ab68bae49676d48e043276e7df87b51	89167786874		
13	7b11642629d32e909e97d679039b5b5	89179259219		
14	acbac5d2d9251938462b322ce3f5b4b3	89649466874		
15	813f045ca476d6d92bc5949bd3b5de6	89066579219		
16	aaae3fcf643181aa054c9ab121220240	89839868108		
17	348a7b5100ac11a05219335f51bbdb2	89178989219		
18	60e7a69ae8e141eea120162d8038fab	89199579219		
19	a1fa04fa7c5a593dff9cbd3115a0a2c3	89156008108		
20	05ce9e072beb98588127e460d4d5bc1	89196798108		
21	d14c8e161561d5d7727f30a3d585a53	89196461443		
22	53cc66530f9d915060df4a92b28e8389	89159688108		
23	e40f511ec92d959746acf96304409914	89176111443		
24	af2338009b17e1ba609b5de51a418b4	89581188108		
25	def0e39d2e55a60259f58189107cce48	89807351443		
26	587bef23a3700ed33a5bbf529c9982c1	89166791443		
27	d5d569af78c0cf58354a6735c6342b11	89629730330		

Рис.1. Таблица с «чистыми» номерами

Шифрование другими методами.

Для исследования были выбраны методы хэширования md5, SHA-1 и SHA-384. Хэширование производилось с помощью библиотеки hashlib для языка программирования Python.

Также были составлены наборы номеров с добавлением «соли». Использованы два модификатора входа: увеличенный числовой и буквенный. Увеличенный числовой модификатор 1023458978 преобразует часть исходных номеров в 12-тизначные числа, что увеличивает полный перебор на этапе расшифровки. Буквенный модификатор представляет собой три случайных нефиксированных буквы латинского алфавита и добавляется в конце номера телефона.

Исследование трудоёмкости расшифровки.

Расшифровка снова проводилась с помощью инструмента «Hashcat».

Команды для расшифровки:

Наборов с различными модификаторами:

Увеличенный числовой: <hashcat -m 0 -a 3 digit.txt -o digit1.txt ?d?d?d?d?d?d?d?d?d?d?d --force>

Буквенный: <hashcat -m 0 -a 3 letter.txt -o letter1.txt ?d?d?d?d?d?d?d?d?d?d?d?l?l?l --force>

Различных алгоритмов хэширования:

md5: < hashcat -m 0 -a 3 md5c.txt -o md5c1.txt ?d?d?d?d?d?d?d?d?d?d?d --force>

SHA-1: < hashcat -m 100 -a 3 sha1.txt -o sha11.txt ?d?d?d?d?d?d?d?d?d?d?d --force>

SHA3-384: <hashcat -m 10800 -a 3 sha384.txt -o sha3841.txt ?d?d?d?d?d?d?d?d?d?d?d --force>

Результаты потенциально затрачиваемого времени на расшифровку в зависимости от изменения модификаторов и методов хэширования приведены в таблицах 1 и 2.

Таблица 1. Зависимость трудоёмкости расшифровки от модификатора

Модификатор	Первичный	Увеличенный числовой	Буквенный (динамический)
Время	5 мин 45 с	2 дн 4 ч	95 дн 16 ч

Использовался алгоритм md5. Как видно из таблицы 1, и увеличение числового модификатора, и использование динамического модификатора приводит к усложнению задачи расшифровки, однако влияние динамического модификатора более значительно.

Алгоритм хэширования	md5	SHA-1	SHA3-384
Время	5 мин 21 с	7 мин 38 с	10 мин 15 с

Таблица 2. Зависимость трудоёмкости расшифровки от алгоритма хэширования

Использовались первичные данные. Как видно из таблицы 2, выбор алгоритма хэширования влияет на время расшифровки незначительно. Выбор более сложной функции снижает скорость обработки, однако этого падения недостаточно, чтобы затрачиваемое на решение время стало неприемлемым.

Заключение.

В ходе работы изучены и реализованы несколько методов хэширования данных, дешифрован обрабатываемый датасет, а также проведено исследование трудоёмкости этих методов. Из полученных данных сделаны выводы:

- Модификатор входа исходного набора 58263126;
- Выбор хэш-функции влияет на время расшифровки незначительно;
- Динамические модификаторы входа увеличивают время расшифровки до неприемлемого.

Список литературы.

1. Introducing Python. Modern computing in simple packages / Bill Lobanovic.
2. https://hackware.ru/?p=4830
3. https://docs.python.org/3/library/hashlib.html
4. Ссылка на код https://github.com/ksenkap/hash