**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**Факультет прикладной математики – процессов управления**

**отчет**

**по лабораторной работе**

**по дисциплине «Алгоритмы и структуры данных»**

**на тему «Исследование хеш-функций с различными вводными условиями»**

**Вариант 3**

|  |  |  |
| --- | --- | --- |
| Студент гр. 23.Б16 |  | Барафанов А.А. |
| Преподаватель |  | Дик А.Г. |

**Санкт-Петербург**

**2024 г.**

**Оглавление**

[**1.** **Цель работы** 2](#_Toc180713705)

[**2.** **Задача (формализация)** 3](#_Toc180713706)

[**3.** **Теоретическая часть** 3](#_Toc180713707)

[**4.** **Описание алгоритма** 4](#_Toc180713708)

[**5.** **Описание программы** 9](#_Toc180713709)

[**5.1** **Описание функций** 9](#_Toc180713710)

[**6.** **Рекомендации пользователю** 10](#_Toc180713711)

[**8.** **Контрольный пример** 11](#_Toc180713712)

[**9.** **Анализ результатов работы алгоритма и входных условий** 13](#_Toc180713713)

# **Цель работы**

Расшифровать набор данных, зашифрованный с помощью хеш-функции с использованием модификатора входа – соли, а также проанализировать решение аналогичной задачи при различных условиях.

# **Задача (формализация)**

Расшифровать таблицу xlsx с захешированными номерами телефонов (кол-во 50000 номеров), используя программу Hashcat. Определить соль, добавленную к номерам телефонов и провести эксперименты с разной солью и хеш-функциями для анализа эффективности тех или иных входных условий.

# **Теоретическая часть**

*Хеш-функция* — это математическая функция, которая преобразует входные данные произвольной длины в выходные данные фиксированной длины, обычно представленные в виде строки символов. Основные свойства хеш-функций включают детерминированность (одинаковые входные данные всегда дают одинаковый выход), быстроту вычисления, а также трудность восстановления исходных данных по хешу (односторонность).

*Хеш-функция MD5 (Message-Digest Algorithm 5)* — это криптографическая хеш-функция, разработанная Винсентом Риедом в 1991 году. Она принимает входные данные произвольной длины и преобразует их в фиксированный хеш-значение длиной 128 бит (16 байт). MD5 используется для проверки целостности данных и создания цифровых подписей.

*Хеш-функция SHA-1 (Secure Hash Algorithm 1)* — это криптографическая хеш-функция, разработанная Национальным институтом стандартов и технологий (NIST) США в 1995 году. Она принимает входные данные произвольной длины и преобразует их в фиксированное хеш-значение длиной 160 бит (20 байт).

*SHA-256 (Secure Hash Algorithm 256)* — это криптографическая хеш-функция, входящая в семейство SHA-2, разработанная Национальным институтом стандартов и технологий (NIST). Она предназначена для генерации хеша фиксированной длины 256 бит (32 байта) из произвольного объема входных данных. SHA-256 обеспечивает защиту целостности и аутентичности данных и находит широкое применение в различных криптографических приложениях, таких как цифровые подписи, сертификаты и системы проверки целостности.

*Хеш-функция SHA-512 (Secure Hash Algorithm 512)* — это криптографическая хеш-функция, которая принимает на вход произвольное количество данных и возвращает фиксированное значение хеша длиной 512 бит (64 байта). SHA-512 является частью семейства алгоритмов SHA-2, разработанного Национальным институтом стандартов и технологий США (NIST) в 2001 году.

*Соль* — это случайно сгенерированная последовательность данных, которая добавляется к паролю перед его хешированием. Основная цель использования соли заключается в повышении безопасности хранения паролей. Соль предотвращает атаки с использованием заранее вычисленных хешей (например, атак с радужными таблицами), так как даже одинаковые пароли будут иметь разные хеши из-за уникальных солей. Соль обычно хранится вместе с хешем пароля, чтобы при проверке пароля можно было восстановить оригинальную соль и выполнить хеширование введенного пароля для сравнения.

*Техника взлома "грубой силы" (brute force)* — это метод атаки на системы безопасности, при котором злоумышленник систематически перебирает все возможные комбинации паролей или ключей шифрования до тех пор, пока не найдет правильный. Этот метод не требует знания о системе или ее уязвимостях, но может занять значительное время и ресурсы, особенно если используются длинные и сложные пароли. Атаки грубой силы являются одной из самых примитивных, но в то же время эффективных техник, особенно против слабых или предсказуемых паролей.

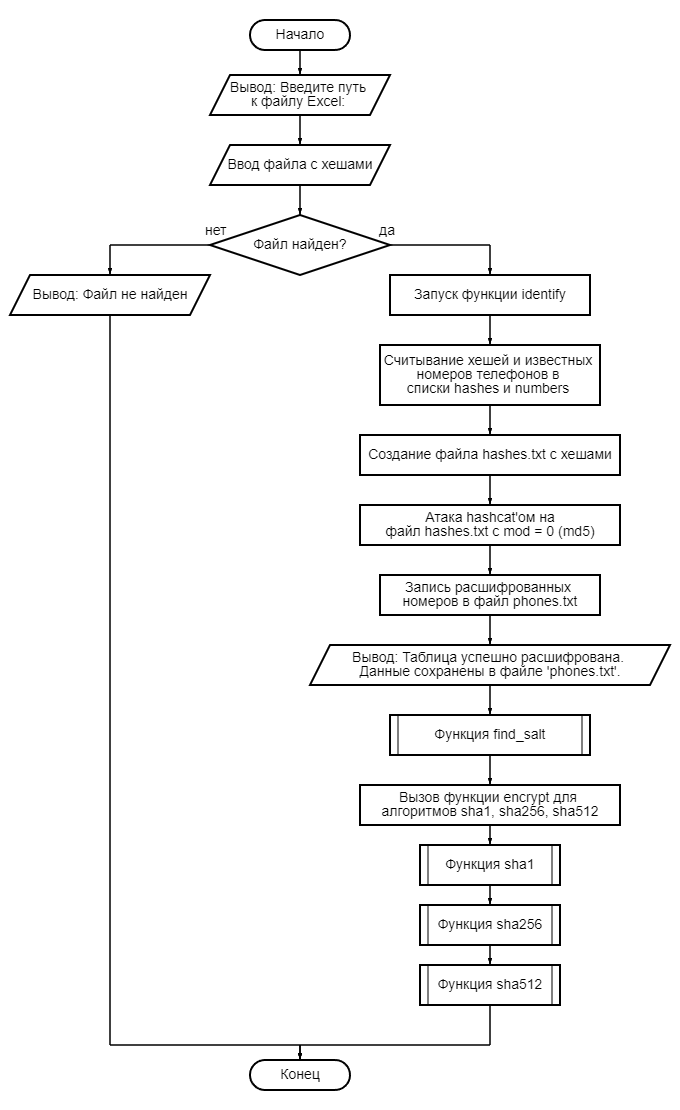
# **Описание алгоритма**

1. Открытие файла xlsx с хешированными данными;
2. Чтение строк в файл hashes.txt;
3. Запуск Hashcat для расшифровки файла hashes.txt с модификатором атаки 0 (хеш-функция md5) и запись результата в файл phones.txt;
4. Вычисление соли в функциях find\_salt и compute\_salt;
5. Перезапись результатов в файле phones.txt, учитывая найденную соль;
6. Тесты с другими семействами хеш-функций (шифрование и повторное деобезличивание):

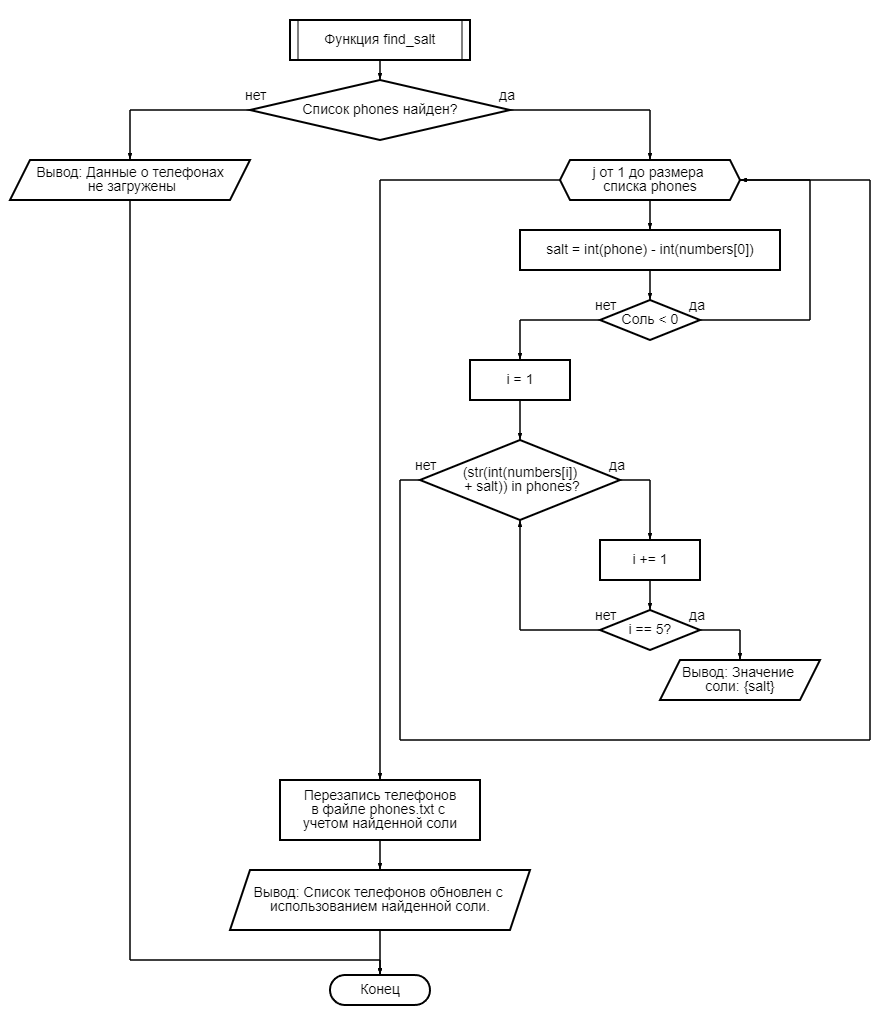
* Запуск функции sha1;
* Запуск функции sha256;
* Запуск функции sha512;

1. Сохранение результатов в файлы*: sha1.txt, sha256.txt, sha512.txt, decrypt\_sha1.txt, decrypt\_sha256.txt, decrypt\_sha512.txt*

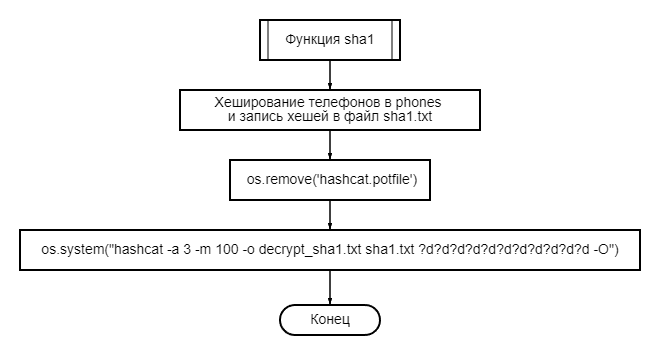
*Блок-схемы, иллюстрирующие шаги алгоритма показаны на рисунках 4.1-4.3.*

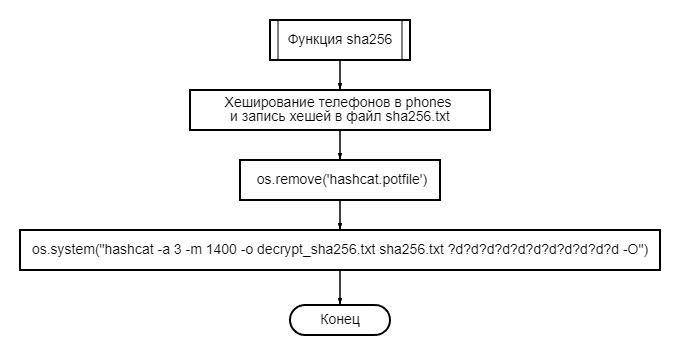


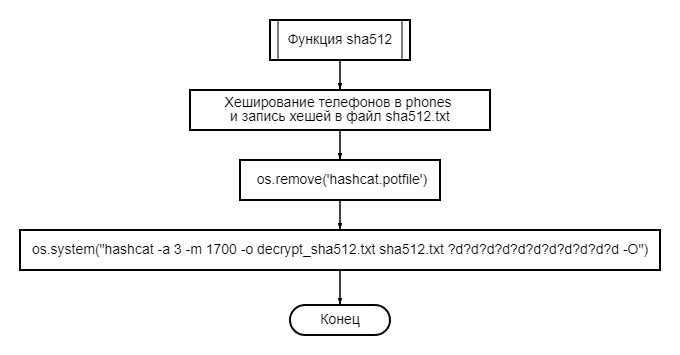
*Рисунок 4.1. Блок-схема основной программы.*



*Рисунок 4.2. Блок-схема функции find\_salt.*

**

**

**

*Рисунок 4.2. Блок-схемы других подпрограмм.*

# **Описание программы**

## **Описание функций**

В программе используется 8 функций. В таблице 5.1 представлено их описание.

*Таблица 5.1. Описание функций*

|  |  |  |
| --- | --- | --- |
| Имя функции | Входные данные | Описание функции |
| main | - | Главная функция, запускающая все остальные подпрограммы |
| decrypt\_without\_salt | Путь к файлу с хешами | Первичное декодирование исходного файла, создание файлов hashes.txt и phones.txt |
| find\_salt | - | Функция, запускающая процесс нахождения соли, выводящая результат в консоль и обновляющая список телефонов с учетом найденной соли |
| compute\_salt | Список первично-декодированных телефонов и список заранее известных номеров | Вычисление соли |
| encrypt\_alg | Название алгоритма | Функция, запускающая кодирование файла той или иной хеш-функцией |
| sha1 | Список номеров для кодирования | Кодирование алгоритмом sha1 и его последующее декодирование |
| sha256 | Список номеров для кодирования | Кодирование алгоритмом sha256 и его последующее декодирование |
| sha512 | Список номеров для кодирования | Кодирование алгоритмом sha512 и его последующее декодирование |

Описание глобальных переменных программы представлено в таблице 5.2.

*Таблица 5.2. Описание глобальных переменных*

|  |  |  |
| --- | --- | --- |
| Тип | Имя | Описание |
| global string | file\_path | Путь к файлу xlsx с хешированными номерами телефонов |
| global list | phones | Список с деобезличенными номерами |
| global list | numbers | Список с достоверно известными номерами |

# **Рекомендации пользователю**

* Убедитесь, что у вас установлена программа Hashcat.
* Убедитесь, что у вас установлен интерпретатор языка Python и
* библиотеки os, pandas и hashlib.
* Убедитесь, что в файле xlsx хеши номеров телефонов хранятся в первом столбце таблицы, а достоверно известные номера в третьем столбце таблицы.
* Правильно введите путь к файлу xlsx с хешированными номерами, иначе программа не найдет его.
* В случае возникновения проблем с запуском Hashcat запускайте код программы в папке установленной версии Hashcat. Если это не помогло, попробуйте обновить драйвера видеокарты.
* Найденная соль будет выведена в консоли, а результаты декодирования в соответствующих файлах в папке с Hashcat.

1. **Рекомендации программисту**

Для работы с программой должен быть установлен интерпретатор язык Python, а также же интегрированная среда разработки (Visual Studio). Поддерживайте актуальные версии интерпретатора и сред разработки. Установите необходимые библиотеки языка python в случае их

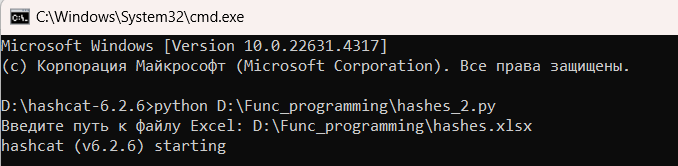
отсутствия: os, pandas, hashlib.

Также для работы с программой потребуется разархивированная папка с бинарниками Hashcat’а последней версии. При возникновении проблем с Hashcat внимательно просмотрите запущенные команды в консоли и удостоверьтесь в отсутствии ошибок (соответствие идентификаторов хеш-функций, атаки по маске и тд). Если проблемы остались, просмотрите целостность папки: она должна включать hashcat.exe, а также папки OpenCl и другие. В ряде случаев стоит обновить драйвера видеокарты или прописать путь к папке с hashcat.exe в переменную среды PATH. За последующей информацией об оставшихся проблемах обращайтесь к документации программы.

При желании поэкспериментировать с другими семействами хеш-функций требуется установка дополнительных библиотек, например, pycryptodome. Для запуска программы рекомендуется 64-битная операционная система Windows.

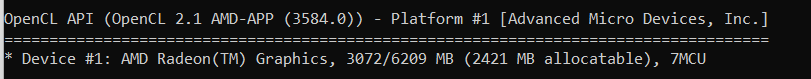
1. **Контрольный пример**

В этом разделе продемонстрирован контрольный пример, показывающий вид консоли и полученные результаты при запуске hashcat. Код запускается из папки с hashcat.exe, командой *python путь\_к\_файлу*. После этого нужно прописать путь к xlsx файлу и нажать *Enter*. При отсутствии проблем появится надпись *hashcat (v6.2.6) starting*. Пример ввода данных показан на рисунке 8.1



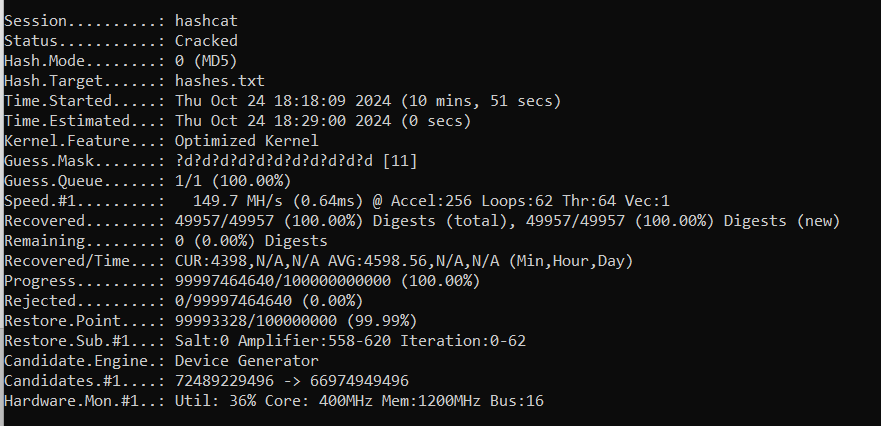
*Рисунок 8.1 Ввод данных*

При работе hashcat в консоль будет выводиться системная информация и информация о работе hashcat. Например, на экран будет выведен процессор/графический процессор, который был найден hashcat’ом (рисунок 8.2), а также иные данные.



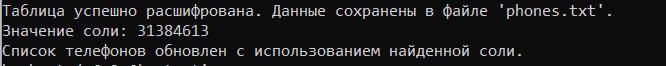
*Рисунок 8.2 Системная информация*

После этого следует дождаться завершения деобезличивания данных. В случае успеха, в консоли высветится надпись *Cracked* (рисунок 8.3)



*Рисунок 8.3 Успешный взлом*

После нахождения соли в консоли появится информирующее сообщение (рисунок 8.4) после чего начнутся тесты с другими хеш-функциями.



*Рисунок 8.4 Информация о соли*

1. **Анализ результатов работы алгоритма и входных условий**

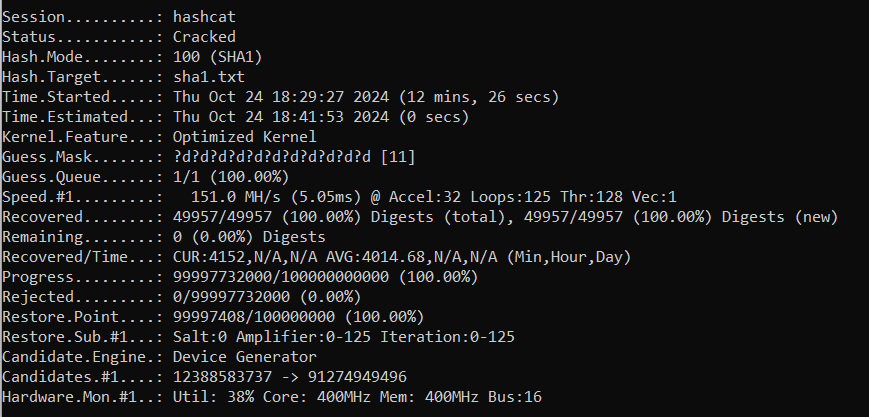
Анализ работы алгоритма проведем в нескольких блоках.

1. Деобезличивание входного файла.

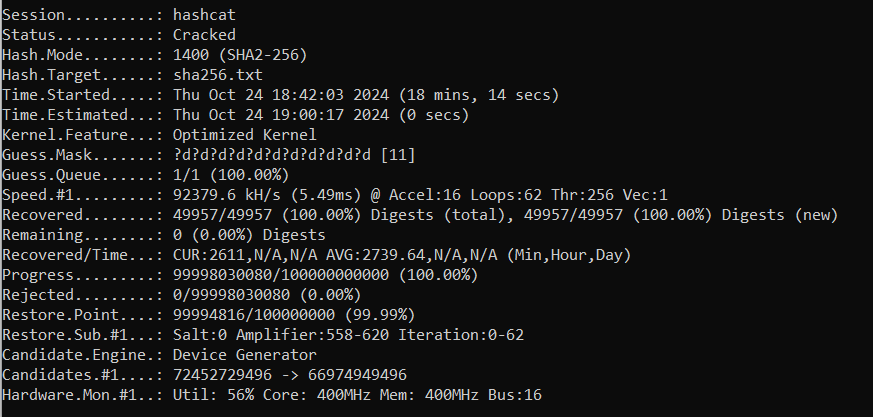
Было определено, что 1) номера захешированы алгоритмом **MD5**, 2) соль равна **31384613**. Расшифрованные номера были сохранены в файл **phones.txt**. Была проведена проверка путем поиска достоверных номеров в файле phones.txt. Все 5 достоверных номеров были получены в файле.

1. Тесты с различными хеш-функциями.

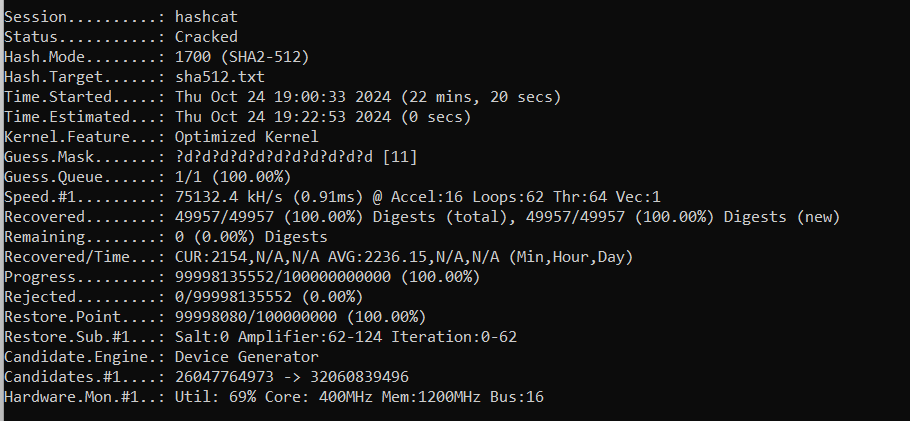
Были проведены дополнительные эксперименты с хеш-функциями **sha1, sha256** и **sha512**. Все функции были успешно взломаны (рисунки 9.1-9.3)



*Рисунок 9.1 Взлом sha1*



*Рисунок 9.2 Взлом sha256*



*Рисунок 9.3 Взлом sha512*

Тесты показали, для расшифровки различных хеш-функций требуется разное время. Результаты показаны в таблице:

|  |  |
| --- | --- |
| Хеш-функция | Время, потребовавшееся для расшифровки |
| MD5 | 10 мин 51 сек |
| SHA1 | 12 мин 26 сек |
| SHA256 | 18 мин 14 сек |
| SHA512 | 22 мин 20 сек |

Таким образом хеш-функция **SHA512**, принадлежащая семейству SHA3 показала наиболее эффективные результаты с точки зрения безопасности.

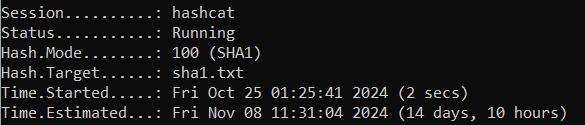
1. Тесты с различной длиной соли.

Для тестов использовалась функция SHA1 и три числовые соли различной длины: 111, 111111 (в два раза длиннее), 111111111 (в три раза длиннее), которые прибавлялись к номеру телефона. Время расшифровки практически не менялось, что означает, что длина прибавляемой (целочисленной) соли не влияет на скорость расшифровки. Это можно объяснить тем, что перебор по маске все равно проходит по одним и тем же десятичным цифрам и добавка в разных разрядах не приводит к увеличению перебора.

1. Тест с другим видом соли.

Для теста была выдвинута гипотеза, что соль – это строка из трех символов, прибавляемая в конец номера телефона. Таким образом, атака велась по маске: os.system("hashcat -a 3 -m 100 -o output\_sha1.txt sha1.txt ?d?d?d?d?d?d?d?d?d?d?d?h?h?h -O")

В этом тесте ожидаемое время расшифровки значительно увеличилось и составило около 14 дней (рисунок 9.4). Этот результат подтверждается теорией – комбинаторно можно вычислить, что количество возможных вариантов увеличилось в несколько тысяч раз.



*Рисунок 9.4 Ожидаемое время расшифровки символьной соли*

1. **Вывод**

В ходе работы было выяснено, что хеширование с солью значительно повышает безопасность телефонных номеров, затрудняя атаки методом подбора. Созданная программа для деобезличивания данных успешно восстановила исходные номера телефонов при определенных условиях.

Анализ трех различных хеш-функций показал, что скорость расшифровки зависит от сложности алгоритма и характеристик соли. Вид соли существенно влияет на время расшифровки. Таким образом, работа подчеркнула важность выбора алгоритмов шифрования и применения соли для защиты данных.