Санкт-Петербургский государственный университет

Факультет прикладной математики – процессов управления

**Лабораторная работа №3**

**Отчет**

**по дисциплине «Системное программирование в Linux»**

Автор работы: Дацык Р.В.

Группа: 22.Б15-пу

Преподаватель: Дик А.Г.

Санкт-Петербург, 2023

**Оглавление**

[**1.** **Цель работы** 3](#_Toc151824592)

[**2.** **Задачи** 3](#_Toc151824593)

[**3.** **Введение** 3](#_Toc151824594)

[**4.** **Теоретическая часть** 3](#_Toc151824595)

[**5.** **Алгоритм метода** 4](#_Toc151824596)

[**6.** **Описание программы** 6](#_Toc151824597)

[**7.** **Рекомендации пользователю** 7](#_Toc151824598)

[**8.** **Рекомендации программисту** 8](#_Toc151824599)

[**9.** **Контрольный пример** 8](#_Toc151824600)

[**10.** **Вывод** 12](#_Toc151824601)

[**11.** **Список использованной литературы** 13](#_Toc151824602)

# **Цель работы**

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

# **Задачи**

1. Изучение особенностей шифрования телефонных номеров.
2. Разработка программы для деобезличивания данных.
3. Тестирование программы с различными хэш-функциями.
4. Анализ скорости расшифровки.

# **Введение**

В современном мире обеспечение конфиденциальности данных представляет собой одну из наиболее важных задач. Хеш-функции, играющие ключевую роль в обеспечении безопасности информации, напрямую влияют на эффективность механизмов защиты. В данной лабораторной работе мы проведем исследование хеш-функций, сосредотачивая внимание на различных вводных условиях, таких как типы солей и разнообразные хеш-функции. Мы также рассмотрим методы взлома, включая "Brute force", и изучим концепцию использования "соли" для повышения уровня защиты хранимых данных.

# **Теоретическая часть**

**Краткое описание хеш-функций, техники взлома Brute force и понятия “Соли” на основе лекционно-практического материала:**

1. ***MD5 (Message Digest Algorithm 5):***

Описание: MD5 генерирует фиксированный 128-битный хеш из входных данных определенной длины.

Показатель безопасности: устаревшая и уязвимая к коллизиям, когда разным входам соответствует одинаковый хеш.

1. ***SHA-1 (Secure Hash Algorithm 1):***

Описание: SHA-1 также создает фиксированный хеш длиной 160 бит.

Показатель безопасности: рассматривается как устаревшая и подверженная коллизиям. Рекомендуется избегать использования в критически важных системах.

1. ***SHA3-256 (Secure Hash Algorithm 3):***

Описание: SHA3-256 генерирует хеш длиной 256 бит.

Показатель безопасности: современная и безопасная хеш-функция, считается надежной для использования в настоящее время.

1. ***SHA2-512 (Secure Hash Algorithm 2 - 512 bit):***

Описание: SHA2-512 также создает фиксированный хеш, но длиной 512 бит.

Показатель безопасности: широко признана как безопасная хеш-функция на текущий момент, обладает высокой степенью стойкости к атакам.

Техники взлома:

1. ***Brute force:***

Описание: атакующая сторона осуществляет перебор всех возможных вариантов паролей или входных данных до тех пор, пока не будет найдено соответствие.

Применение: эффективна при использовании слабых паролей или устаревших хеш-алгоритмах.

Понятие "Соль":

1. ***Соль (Salt):***

Описание: «соль» представляет собой случайную последовательность бит, добавляемую к входным данным перед хешированием, чтобы повысить сложность атаки и усилить защиту хранящихся данных.

# **Алгоритм метода**

1. Программа использует Hashcat для деанонимизации хешей из файла «hashes.txt», тем самым получая список телефонных номеров. Список сохраняется в «phones.txt»
2. Получив «phones.txt» и сравнив с изначальными номерами, программа посчитает «соль»
3. С помощью программы создается 3 варианта датасета, основанных на 3 разных хеш-функциях и в каждую из которых добавлено три вида «соли»
4. Сравнение скорости выполнения процесса деанонимизации в зависимости от хеш-функции или вида «соли».

На рисунке 5.1 представлена блок-схема алгоритма.

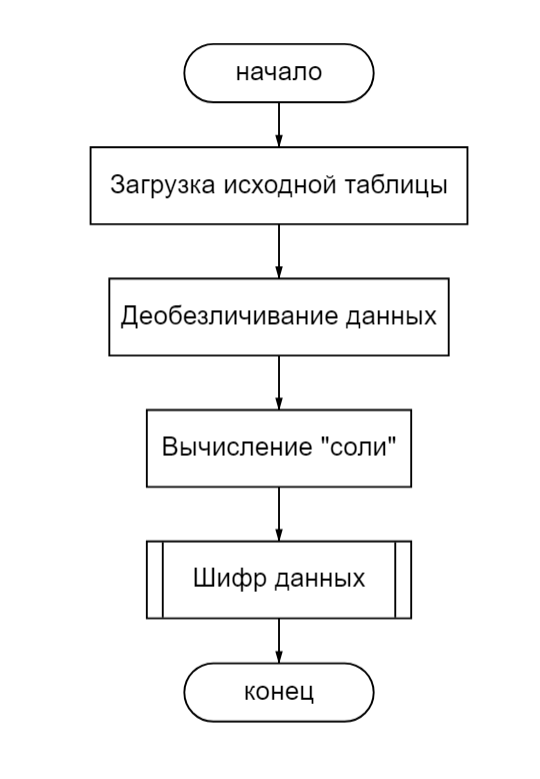


Рисунок 5*.*1 Блок-схема программы.

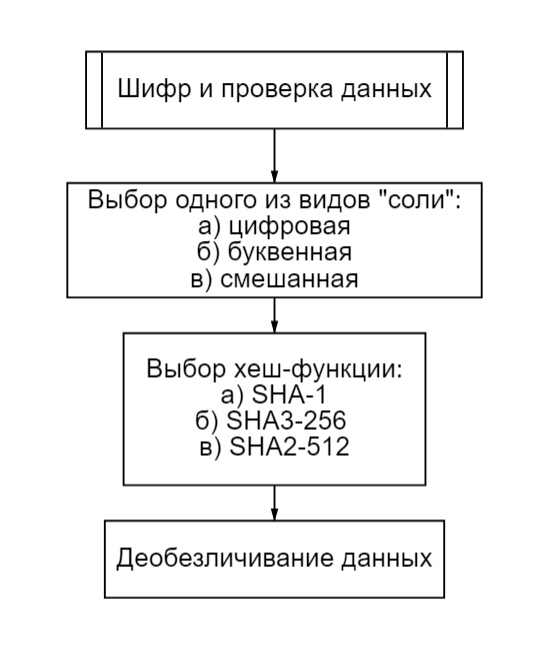


Рисунок 5*.2* Блок-схема подпрограммы.

# **Описание программы**

Алгоритм реализован на языке python 3.10 с использованием следующих пакетов: pandas, random, datetime.

В программе используются 8 функций: 7 связанных с обработкой хэш-функций и одна для обработки интерфейса. В таблице 6.1 представлено описание всех функций.

Таблица 6.1. Описание функций

|  |  |  |  |
| --- | --- | --- | --- |
| Имя функции | Аргумент | Возвращаемое значение | Условия формирования данных |
| hash\_phone | Номер телефона для хэширования, Соль, добавляемая к номеру телефона перед хэшированием. | Кортеж из трех хэшей: SHA-1, MD5 и SHA-256. | Функция принимает номер телефона и соль, применяет к ним три различные хэш-функции и возвращает результат в виде кортежа. |
| save\_hashes | Имя файла, в который будут сохранены хэши, Список хэшей для сохранения. | файл | Функция сохраняет хэши в указанный файл. |
| identify | None | 2 файла | Функция читает данные из файла Excel, использует hashcat для расшифровки номеров телефонов, сохраняет их в файл 'phones.txt' и исходные хэши в файл 'hashes.txt'. |
| find\_salt | None | Сообщение с найденными значениями «соли» | Функция вычисляет «соль» на основе расшифрованных номеров телефонов и их частей. |
| compute\_salt | Список расшифрованных номеров телефонов, Список частей номеров телефонов, Тип «соли» ("Numeric", "Alphabetic", "Mixed"). | Список значений «соли». | Функция вычисляет «соль» на основе расшифрованных номеров телефонов и их частей, в зависимости от выбранного типа «соли». |
| encrypt | Выбранный алгоритм шифрования ("sha1", "sha256", "sha512"). | файлы с зашифрованными данными. | Функция выбирает и выполняет шифрование с использованием выбранного алгоритма, сохраняя результат в соответствующем файле. |

# 

# **Рекомендации пользователю**

Для успешного запуска программы необходимо устройство с операционной системой Linux, macOS или Windows, а также среда разработки, поддерживающей запуск python 3.10

Для запуска деанонимизации достаточно указать путь к вашему excel файлу и нажать кнопку деобезличить. Как только таблица будет обработана, появится уведомление об успешном завершении процесса. Для поиска «соли» достаточно нажать на кнопку «вычислить соль» (функция доступна только после обезличивания данных). Файл с номерами будет сохранен в директорию, где находится код программы. Для создания зашифрованного датасета нужно с помощью меня выбрать интересующие вас параметры: вид «соли» и хеш-функции.

# **Рекомендации программисту**

Для запуска программы необходима 64-битная операционная система Windows, Linux или macOS. Для работы с кодом необходима среда разработки, совместимая с python 3.10 и библиотеки hashlib, os, pandas, tkinter, а также приложение Hashcat

Исходный код программы и необходимые текстовые файлы доступны по ссылке:

<https://github.com/CapTopGrade/Algorithms/blob/main/3%20lab.py>

# **Контрольный пример**

Было проведено хеширование телефонных номеров с помощью 3 разных хеш-функций: SHA1, SHA2\_512, SHA3\_256. Также было использовано 3 разных вида соли: числовая, алфавитная, смешанная. Используя hashcat все номера были расшифрованы, однако за разное количество времени. Заметим, что помимо хеш-функции на скорость взлома влияет и сложность «соли». При первом деобезличивании значение «соли» равно 58644554.

Все иллюстрации представлены ниже в контрольном примере.

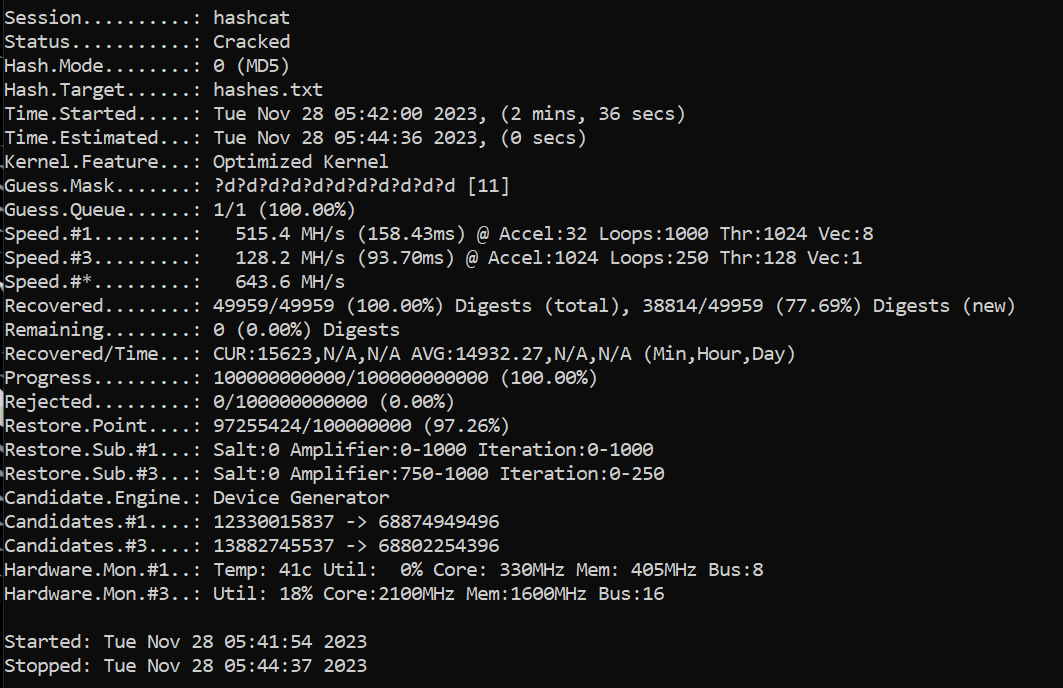


Рис. 9.1 Расшифровка исходного датасета по методу MD5

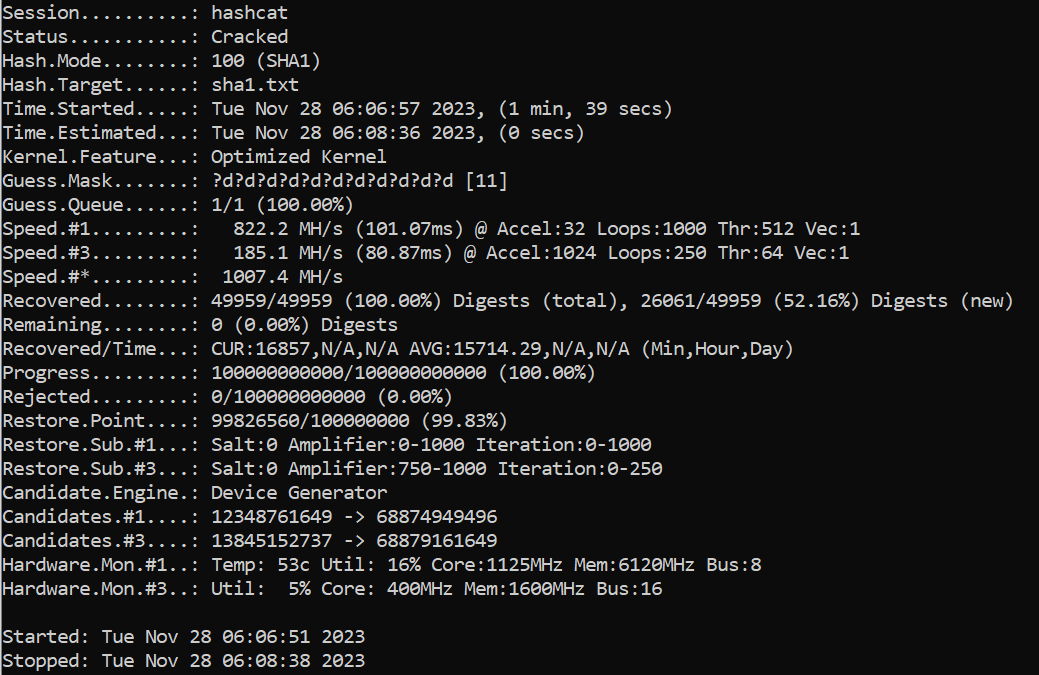


Рис. 9.2 Расшифровка датасета с добавлением числовой соли по методу SHA1

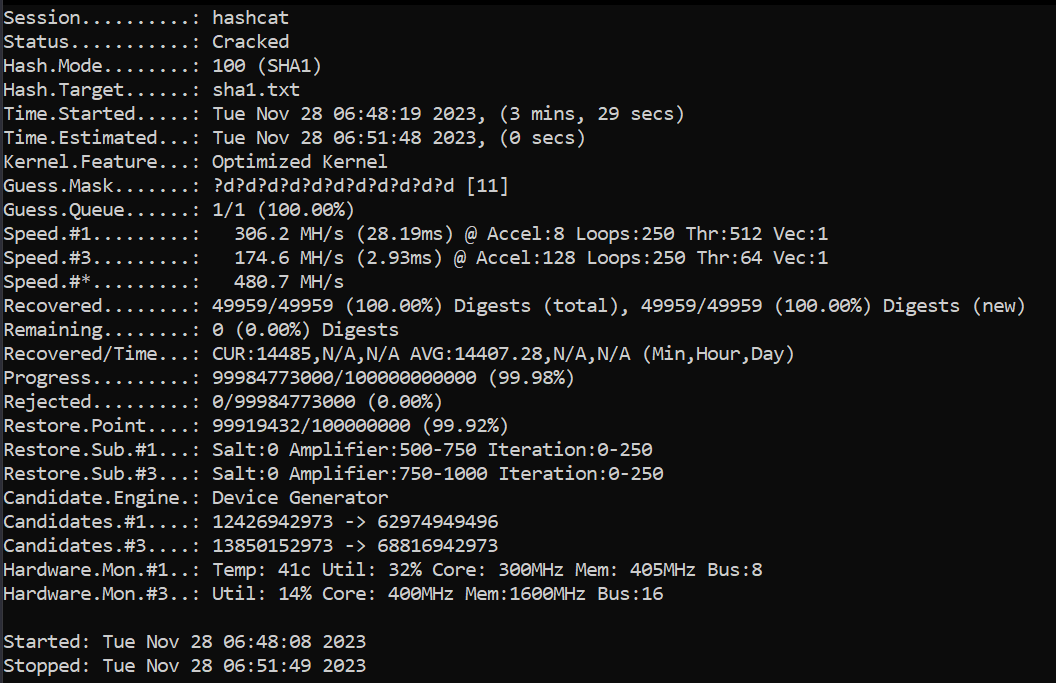


Рис. 9.3 Расшифровка датасета с добавлением смешанной соли по методу SHA1

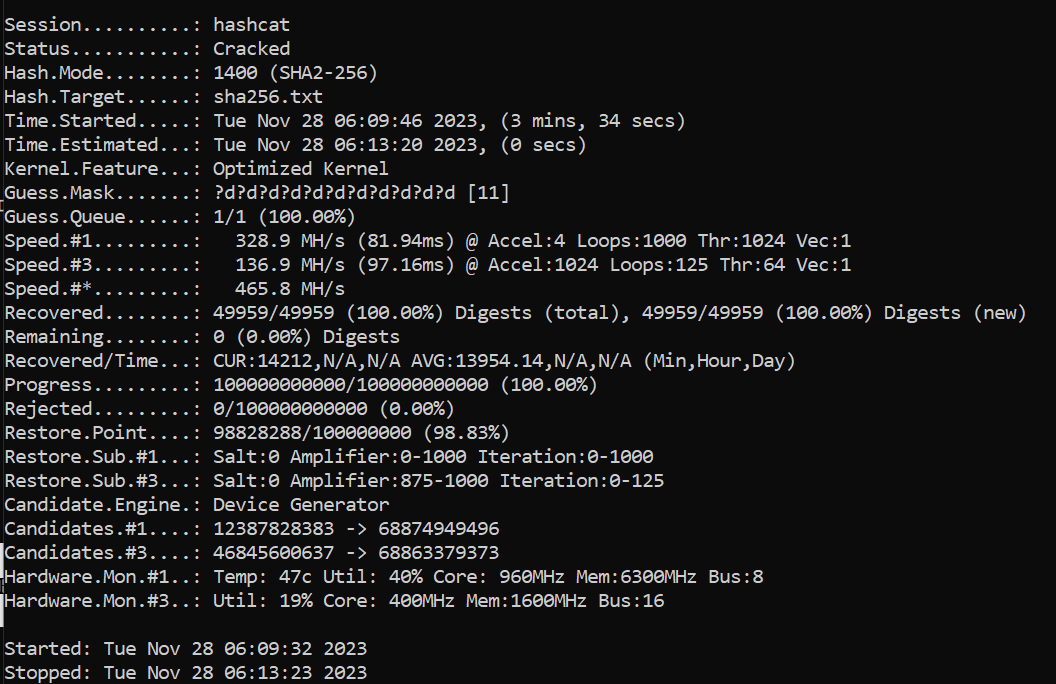


Рис. 9.4 Расшифровка датасета с добавлением числовой соли по методу SHA2\_256

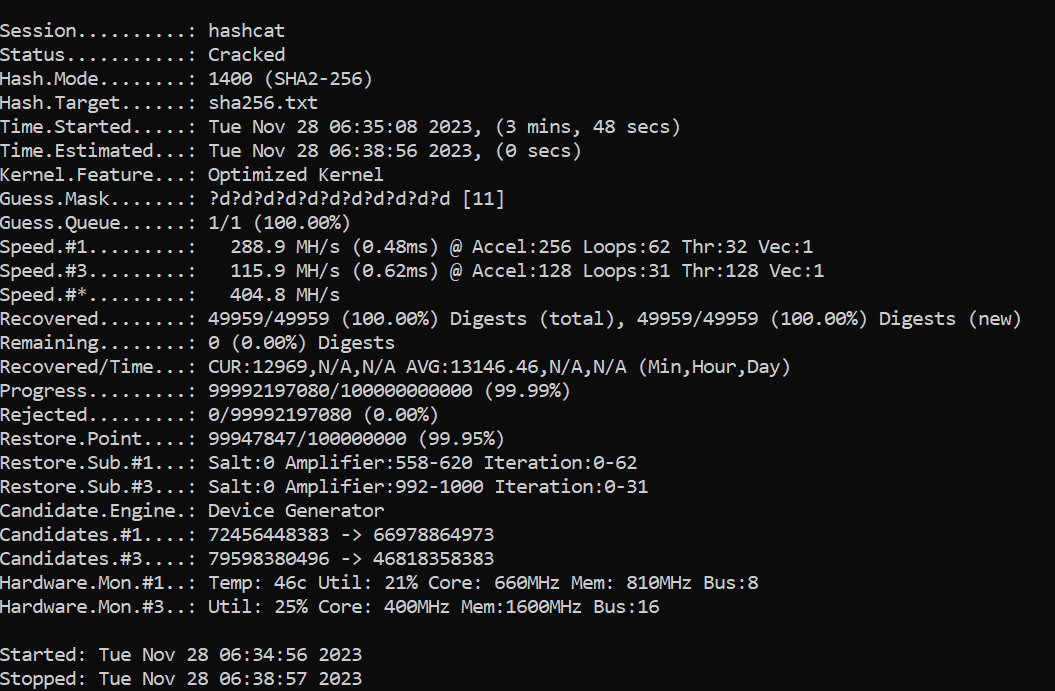


Рис. 9.5 Расшифровка датасета с добавлением смешанной соли по методу SHA2\_256

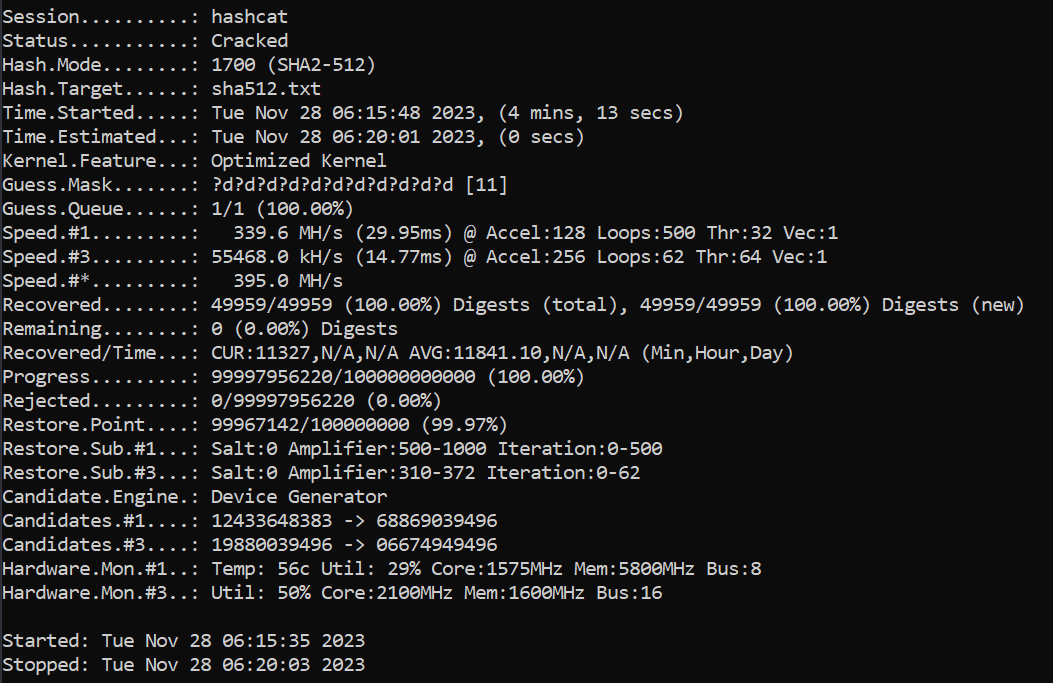


Рис. 9.6 Расшифровка датасета с добавлением числовой соли по методу SHA2\_512

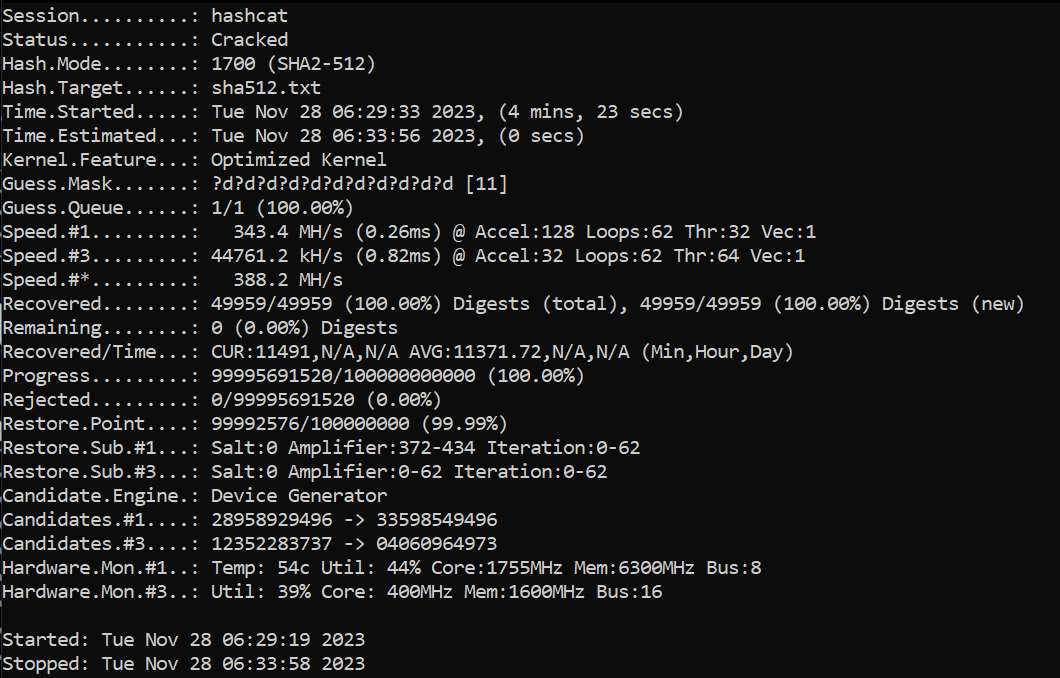


Рис. 9.7 Расшифровка датасета с добавлением смешанной соли по методу SHA2\_512

Полученные результаты показывают, что алгоритмы SHA3-256 и SHA2-512 являются более безопасными и устойчивыми к атакам по сравнению с SHA-1. Стоит отметить, что чем выше битовая длина хеш-значения, тем больше вычислительных ресурсов требуется для проведения атак brute-force. SHA3-256 и SHA2-512 имеют более длинные битовые хеши, а тем более с увеличением длины и сложности «соли» процесс расшифровки становится более затратным.

# **Вывод**

В результате выполнения работы раcшифрован набор данных, обезличенный с помощью хеш-функции и модификатора входа – «соли»(значение «соли» 58644554), и проанализировано решение аналогичной задачи при других алгоритмах sha1, sha256, sha512 и с различными видами «соли»

Изучение результатов, полученных с использованием Hashcat, позволило оценить уровень надежности шифрования и оценить эффективность защиты данных от возможных атак. Важно отметить, что обеспечение безопасности данных требует постоянного мониторинга и обновлений в свете появления новых методов взлома. Регулярное обновление используемых методов хеширования и солей играет ключевую роль в обеспечении безопасности информации.

# **Список использованной литературы**

[1] Документация библиотеки Hashcat: <https://hashcat.net/wiki/doku.php?id=hashcat>

[2] Документация библиотеки Tkinter: <https://docs.python.org/3/library/tkinter.html>

[3] Документация библиотеки Pandas: https://docs.python.org/3/library/pandas.html

[4] Статья о работе с tkinter: https://pythonru.com/uroki/obuchenie-python-gui-tkinter