Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение   
высшего образования «Национальный исследовательский университет   
«Московский институт электронной техники»

Кафедра «Информационная безопасность»

ОТЧЕТ

по проведению лабораторной работы

ЛАБОРАТОРНАЯ РАБОТА №ТЕСТ

«Введение в криптографию»

|  |  |
| --- | --- |
| Выполнил студент учебной группы ИБ- | \_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_/  подпись (фамилия, инициалы) |
|  |
| Проверил | \_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_/  подпись (фамилия, инициалы) |
|  |

Москва, 2024 г.

СОДЕРЖАНИЕ

[ЦЕЛИ РАБОТЫ 3](#_Toc182920545)

[ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 3](#_Toc182920546)

[1 Введение 3](#_Toc182920547)

[2 Шифр Цезаря 4](#_Toc182920548)

[3 Шифр Вижинера 5](#_Toc182920549)

[4 Кодировки 8](#_Toc182920550)

[4.1 ASCII 8](#_Toc182920551)

[4.2 UTF-8 9](#_Toc182920552)

[5 Системы счисления 10](#_Toc182920553)

[5.1 Binary (Двоичная) 11](#_Toc182920554)

[5.2 Hex (Шестнадцатеричная) 13](#_Toc182920555)

[ПРАКТИЧЕСКАЯ ЧАСТЬ 14](#_Toc182920556)

[1 Азбука Морзе 14](#_Toc182920557)

[2 Строки хэша 15](#_Toc182920558)

# ЦЕЛИ РАБОТЫ

1. Проверить эффективность и результативность самостоятельной работы студентов над учебным материалом.

**Время на выполнение:** 4 учебных часа (180 минут).

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1 Введение

Криптография - это наука о методах обеспечения конфиденциальности, целостности данных, аутентификации, шифрования.

Криптографические методы защиты информации имеют древние корни. Одним из наиболее известных примеров является шифр Цезаря, в котором каждая буква заменяется другой, сдвинутой на определенное количество позиций в алфавите. Другим интересным методом был «скитальческий» шифр, используемый древними спартанцами для передачи зашифрованных сообщений. В Средние века криптографические техники применялись рыцарями, участвовавшими в крестовых походах, для защиты секретной информации. Значительный вклад в развитие криптографии внес Фрэнсис Бэкон, предложив свой метод, известный как «Шифр Бэкона».

История криптографии пережила период активного развития, особенно заметного в эпоху Второй мировой войны. Одним из ключевых событий стало взлом немецкой шифровальной машины «Энигма», что существенно повлияло на ход войны. С развитием вычислительной техники появилась необходимость в создании более сложных криптографических методов. Были разработаны алгоритмы с использованием мощных ключей шифрования, цифровой подписи, хэширования и других технологий для обеспечения информационной безопасности.

## 2 Шифр Цезаря

Шифр Цезаря представляет собой разновидность шифра подстановки, при котором каждая буква в открытом тексте заменяется символом, находящимся на фиксированное количество позиций дальше в алфавите. Этот метод получил свое название в честь древнеримского полководца Юлия Цезаря, который использовал его для защиты секретной переписки. Одной из наиболее известных модификаций шифра Цезаря является алгоритм ROT13 (от англ. "rotate" – "сдвиг на 13 позиций"), при котором буквы сдвигаются ровно на 13 позиций, обеспечивая простой и симметричный способ шифрования текста.

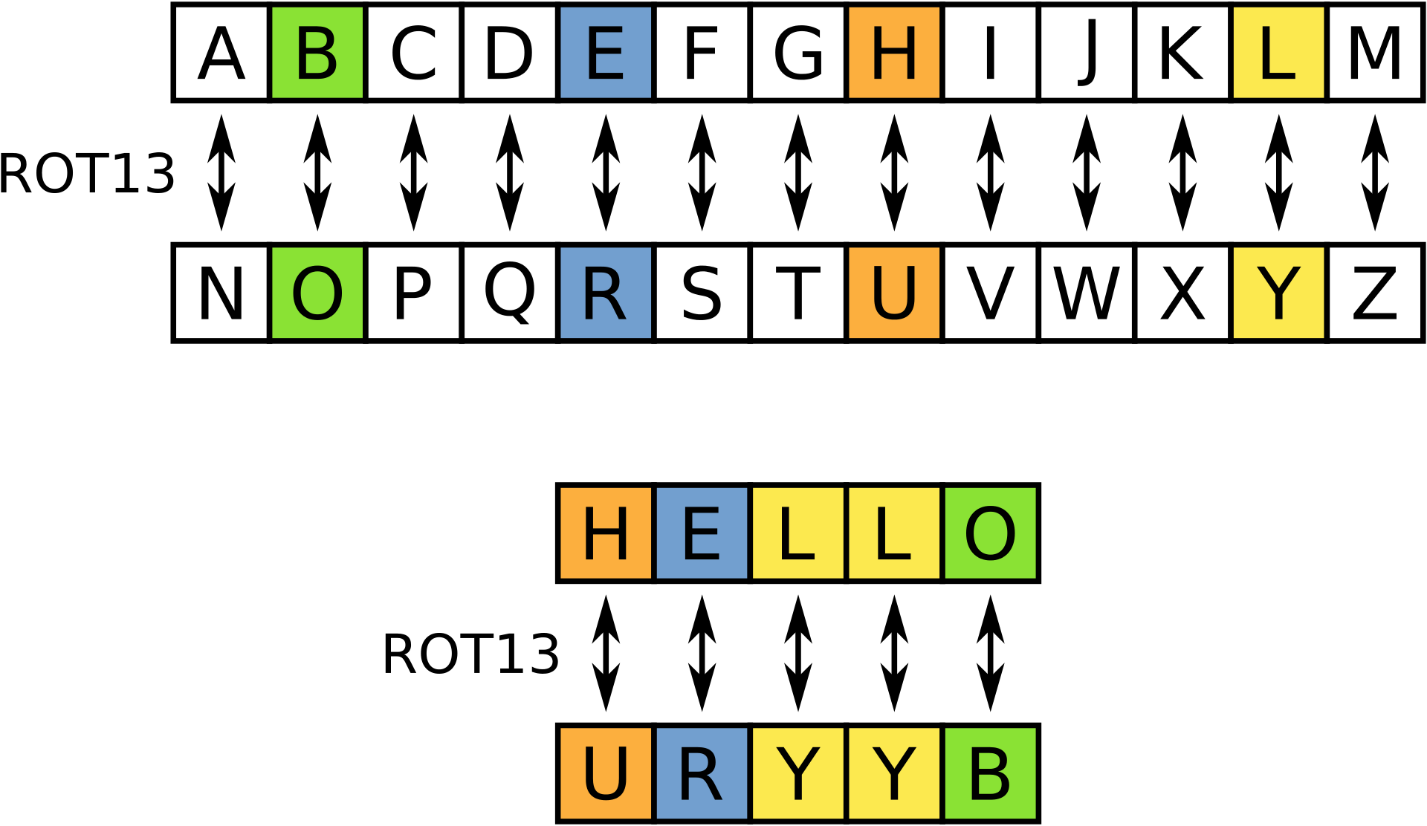


Рисунок 2.1. Иллюстрация шифра Цезаря ROT13.

Допустим у нас есть строка “ABCDEF”, которую нужно зашифровать. Смещаем последовательно каждый символ на 13 позиций:

| 1 | 2 | 3 | 4 | 5 | 6 |
| --- | --- | --- | --- | --- | --- |
| a | b | c | d | e | f |
| 7 | 8 | 9 | 10 | 11 | 12 |
| g | h | i | j | k | l |
| 13 | 14 | 15 | 16 | 17 | 18 |
| m | n | o | p | q | r |
| 19 | 20 | 21 | 22 | 23 | 24 |
| s | t | u | v | w | x |
| 25 | 26 |  |  |  |  |
| y | z |  |  |  |  |

К порядковому номеру изначальной буквы нужно прибавить 13 и заменить её на новую букву: “a” заменяется на букву “n”, “b” на “o”, “c” на “p” и так далее. Получается, что шифруя строку “ABCDEF”, мы получаем новую строку “NOPQRS”.

Шифр Цезаря (ROT13) подразумевает сдвиг букв на 13 позиций, но есть ещё и другие варианты сдвига, например: ROT1, ROT2, ROT3, ..., ROT25. Сдвинуть можно на любое количество символов, но есть только 1 ограничение - размер алфавита. Например, в английском алфавите сдвинуть можно на 25 символов, а в русском на 32.

## 3 Шифр Вижинера

Шифр Виженера - метод полиалфавитного шифрования буквенного текста с использованием ключевого слова. Этот метод является простой формой многоалфавитной замены. Шифр Виженера изобретался многократно. Впервые этот метод описал Джованни Баттиста Беллазо в книге La cifra del. Sig. Giovan Battista Bellasо в 1553 году, однако в XIX веке получил имя Блеза Виженера, французского дипломата. Метод прост для понимания и реализации, но является недоступным для простых методов криптоанализа.

Чтобы зашифровать или расшифровать сообщение требуется сгенерированный автоматически или придуманный ключ. Предположим, что нам нужно зашифровать строку “codeby games” с помощью ключа “secret”. Для начала нам нужно выписать строку, которую нужно зашифровать, а затем под каждой буквой разместить аналогично буквы ключа:

| c | o | d | e | b | y |  | g | a | m | e | s |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| s | e | c | r | e | t |  | s | e | c | r | e |

Теперь потребуется таблица из английских букв (рис. 3.1)

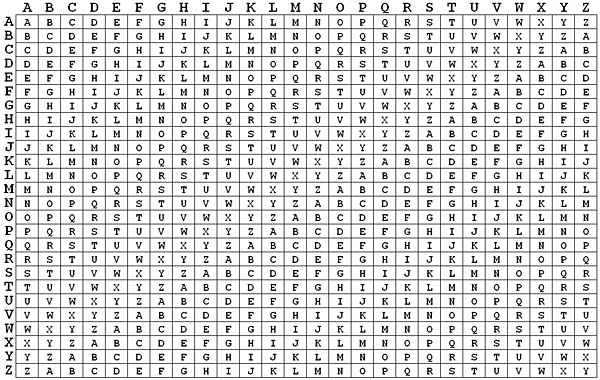


Рисунок 3.1. Таблица для преобразования шифром Виженера.

По верхней строке ищем букву “c”, а по левой букву “s”. Берём букву, которая располагается в середине пересечения - “u” (рис. 3.2)

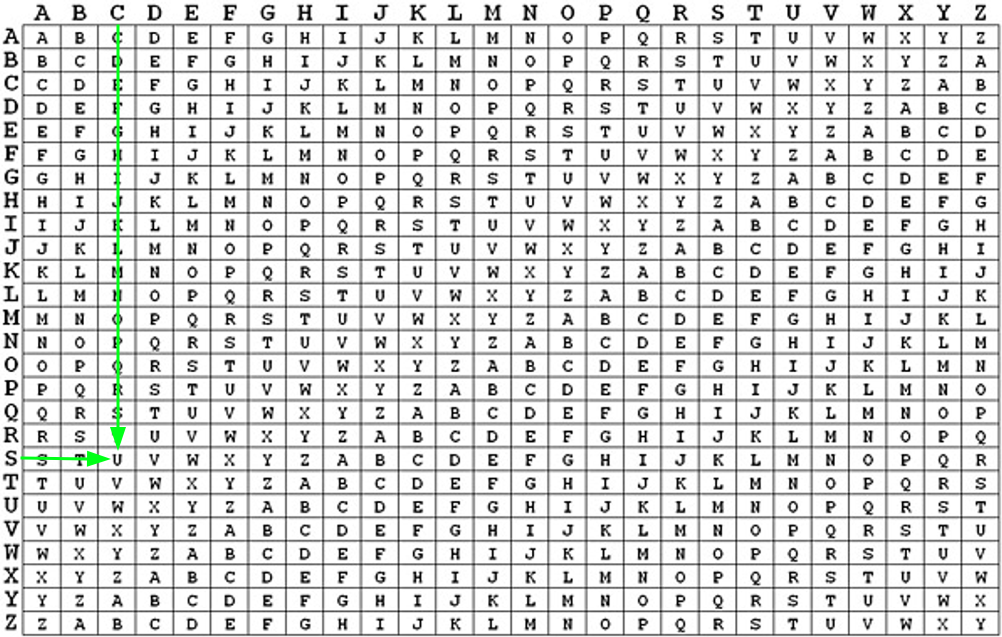


Рисунок 3.2. Пример использования таблицы Вижинера.

Действуя таким образом мы получаем с помощью шифра Вижинера зашифрованную строку - “usfvfr yeovw”.

Теперь расшифруем её вручную. Создаём такую же таблицу, где в верхней строке указываем шифр, а под ним ключ:

| u | s | f | v | f | r |  | y | e | o | v | w |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| s | e | c | r | e | t |  | s | e | c | r | e |

Делаем обратные действия, чтобы получить правильную расшифровку. Для начала берём первые две буквы - “u” и “s”. Затем в верхней строке ищем букву “s”, а под ней ищем букву “u” - в крайнем левом столбце будет исходная буква при шифровании “c” (рис. 3.3)

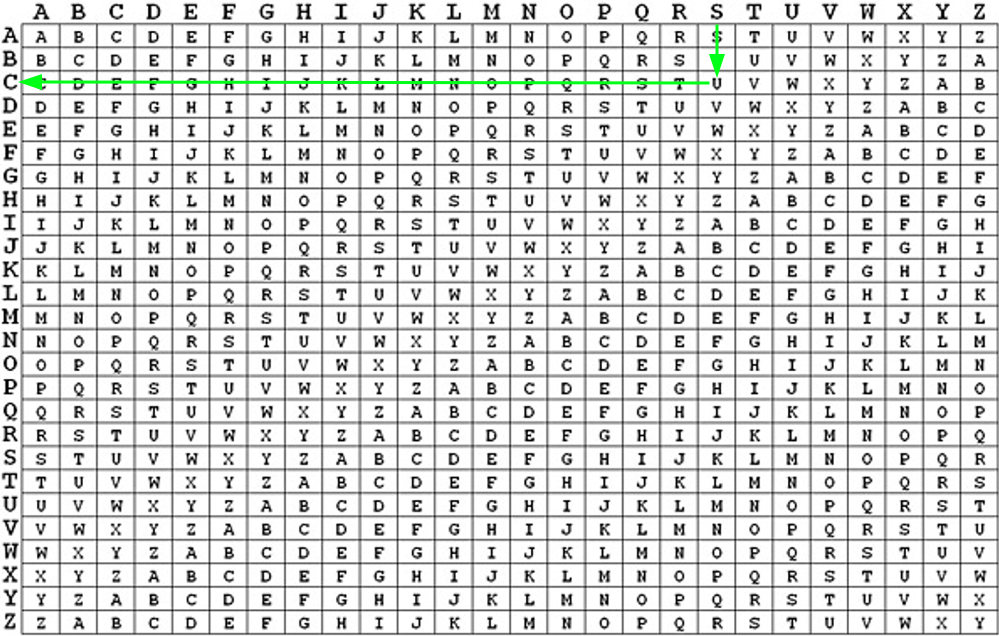


Рисунок 3.3. Пример расшифровывания шифра Вижинера.

В итоге получаем исходную строку - “codeby games”.

## 4 Кодировки

Кодировка (набор символов, англ. character set) - таблица, задающая кодировку конечного множества символов алфавита (обычно элементов текста: букв, цифр, знаков препинания). Такая таблица сопоставляет каждому символу последовательность длиной в один или несколько символов другого алфавита (точек и тире в коде Морзе, сигнальных флагов на флоте, нулей и единиц (битов) в компьютере).

### 4.1 ASCII

ASCII (англ. American standard code for information interchange) - стандарт кодирования знаков латинского алфавита, цифр, некоторых специальных знаков и управляющих последовательностей, принятый в 1963 году Американской ассоциацией стандартов как основной способ представления текстовых данных в ЭВМ. Изначально кодировка вмещала в себя 128 символов, однако со временем её расширили до 256. Таблицу с символами кодировки можно найти в поисковике, либо использовать утилиту ascii в терминале. Установить её можно с помощью следующей команды:

sudo apt install ascii

Чтобы посмотреть список ASCII символов достаточно указать название утилиты и опцию -d.

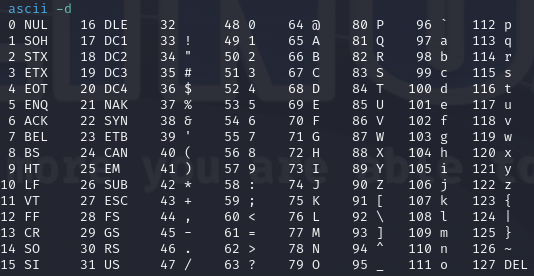


Рисунок 4.1.1. Таблица ASCII.

### 4.2 UTF-8

UTF-8 (от англ. Unicode Transformation Format, 8-bit - “формат преобразования Юникода, 8-бит”) - распространённый стандарт кодирования символов, позволяющий более компактно хранить и передавать символы Юникода, используя переменное количество байт (от 1 до 4), и обеспечивающий полную обратную совместимость с 7-битной кодировкой ASCII. Стандарт UTF-8 официально закреплён в документах RFC 3629 и ISO/IEC 10646 Annex D. В данной кодировке 1 114 112 символов и она может представлять буквы практически всех письменных языков мира. UTF-8 использует от 8 до 32 бит для представления символов, что позволяет кодировать огромное количество символов, включая эмодзи, редкие и иероглифы.

## 5 Системы счисления

Система счисления (англ. numeral system или system of numeration) - символический метод записи чисел, представление чисел с помощью письменных знаков.

Наиболее часто употребляемыми в настоящее время позиционными системами являются:

* Двоичная (в дискретной математике, информатике, программировании)
* Троичная
* Восьмеричная
* Десятичная (используется повсеместно)
* Двенадцатеричная (счёт дюжинами)
* Шестнадцатеричная (используется в программировании, информатике)
* Двадцатеричная

С помощью систем счисления можно записать одно число по-разному, например, число “1337” в десятичной системе счисления представляется, как “1337”, однако в двоичной системе счисления число “1337” представляется, как “10100111001”.

Таблица 5.1

|  |  |
| --- | --- |
| Двоичная | 10100111001 |
| Троичная | 1211112 |
| Восьмеричная | 2471 |
| Десятичная | 1337 |
| Двенадцатеричная | 935 |
| Шестнадцатеричная | 539 |
| Двадцатеричная | 36H |

Наиболее употребляемые в CTF системы счисления - это двоичные (binary) и шестнадцатеричные (hex).

### 5.1 Binary (Двоичная)

Двоичная система счисления - это система счисления, основанная на двух цифрах: 0 и 1. Каждая цифра в двоичной системе называется битом. Бит - это минимальная единица информации в компьютерах, именно из-за этого двоичная система очень важна в вычислительной технике. Ниже представлена таблица размерности от бита до петабайта (таблица 5.1.1)

Таблица 5.1.1

|  |  |
| --- | --- |
| 1 байт | 8 бит |
| 1 килобайт | 1024 байта |
| 1 мегабайт | 1024 килобайт |
| 1 гигабайт | 1024 мегабайта |
| 1 терабайт | 1024 гигабайта |
| 1 петабайт | 1024 терабайта |

Чтобы перевести число из десятичной системы в двоичную, нужно делить число на 2 и запоминать остатки. Этот процесс повторяется до тех пор, пока число не станет равным 0. Затем записываются полученные остатки в обратном порядке, образуя двоичное представление числа.

Например, нужно перевести число “1337” в двоичную систему счисления. Выполним следующие действия:

1337 разделим на 2 - получается 668, а в остатке 1.

668 разделим на 2 - получается 334, а в остатке 0.

334 разделим на 2 - получается 167, а в остатке 0.

167 разделим на 2 - получается 83, а в остатке 1.

83 разделим на 2 - получается 41, а в остатке 1.

41 разделим на 2 - получается 20, а в остатке 1.

20 разделим на 2 - получается 10, а в остатке 0.

10 разделим на 2 - получается 5, а в остатке 0.

5 разделим на 2 - получается 2, а в остатке 1.

2 разделим на 2 - получается 1, а в остатке 0.

1 разделим на 2 - получается 0, а в остатке 1.

Наглядные вычисления (таблица 5.1.2)

Таблица 5.1.2

|  |  |  |
| --- | --- | --- |
| **Операция** | **Результат** | **Остаток** |
| 1337 / 2 | 668 | 1 |
| 668 / 2 | 334 | 0 |
| 334 / 2 | 167 | 0 |
| 167 / 2 | 83 | 1 |
| 83 / 2 | 41 | 1 |
| 41 / 2 | 20 | 1 |
| 20 / 2 | 10 | 0 |

Полученные единицы и нули записываем в обратном порядке - “10011100101”. Ручные вычисления всегда можно проверить через онлайн сервисы или языки программирования.

Теперь переведём число “10011100101” из двоичной системы счисления в десятичную. Для этого нужно каждый разряд двоичного числа умножить на его основание в степени n, где n - это номер разряда, а 0 - это самый младший целый разряд:

(1 \* 2^10) + (0 \* 2^9) + (1 \* 2^8) + (0 \* 2^7) + (0 \* 2^6) + (1 \* 2^5) + (1 \* 2^4) + (1 \* 2^3) + (0 \* 2^2) + (0 \* 2^1) + (1 \* 2^0) = 1024 + 0 + 256 + 0 + 0 + 32 + 16 + 8 + 0 + 0 + 1 = 1337

Таким образом из двоичного числа “10011100101” мы получили десятичное “1337”.

### 5.2 Hex (Шестнадцатеричная)

Перевод из десятичной системы счисления в шестнадцатеричную и наоборот аналогичен переводу из десятичной в двоичную, однако деление производится на число 16.

Например, нам нужно перевести число 1337 в шестнадцатеричную:

* 1337 разделим на 16 - получается 83, а в остатке 9.
* 83 разделим на 16 - получается 5, а в остатке 3.
* 5 разделим на 16 - получается 0, а в остатке 5.

Наглядные вычисления (таблица 5.2.1)

|  |  |  |
| --- | --- | --- |
| **Операция** | **Результат** | **Остаток** |
| 1337 / 16 | 83 | 9 |
| 83 / 16 | 5 | 3 |
| 5 / 16 | 0 | 5 |

Таким образом мы из десятичного числа “1337” получили шестнадцатеричное число “539”.

Теперь по аналогии с переводом двоичной системы счисления в десятичную переведём из шестнадцатеричной в десятичную:

(5 \* 16^2) + (3 \* 16^1) + (9 \* 16^0) = 1280 + 48 + 9 = 1337

Мы получили из шестнадцатеричного числа “539” десятичное число “1337”.

# ПРАКТИЧЕСКАЯ ЧАСТЬ

## 1 Азбука Морзе

Дан текстовый файл «task1.txt», содержащий текст

яблоко-- банан.. апельсин. виноград- банан{ апельсин-.. виноград.---- ананас-.. яблоко..--.- банан-.-- апельсин----- виноград..- ананас..--.- яблоко.-. банан...-- апельсин-.-. виноград----- ананас--. яблоко-. банан.---- апельсин--.. виноград...-- ананас..--.- яблоко-- банан----- апельсин.-. виноград... ананас...-- яблоко..--.- банан-.-. апельсин----- виноград-.. ананас...-- яблоко..--.. банан}

Здесь видны слова и символы, которые напоминают шифрование Азбукой Морзе. Необходимо проверить, для этого ищем в интернете алфавит и меняем символы на буквы (рис. 6.1.1)

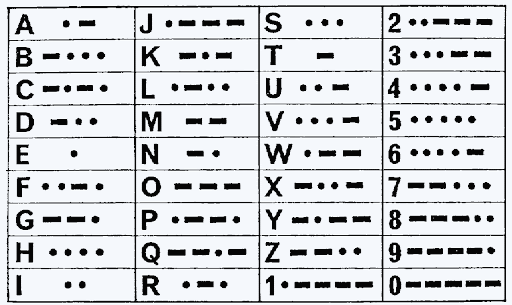


Рисунок 6.1.1. Алфавит Азбуки Морзе.

Поняв первые 4 символа получается открыть небольшую часть сообщения

яблокоM бананI апельсинE виноградT

Можно предположить, что слова «яблоко», «банан», «апельсин», «виноград» и «ананас» не имеют смыла и их можно убрать. Теперь имеется сообщение в формате флага

MIET{ -.. .---- -.. ..--.- -.-- ----- ..- ..--.- .-. ...-- -.-. ----- --. -. .---- --.. ...-- ..--.- -- ----- .-. ... ...-- ..--.- -.-. ----- -.. ...-- ..--.. }

Для упрощения вскрытия шифра можно воспользоваться языком программирования или онлайн переводчиком текста (рис. 6.1.2)

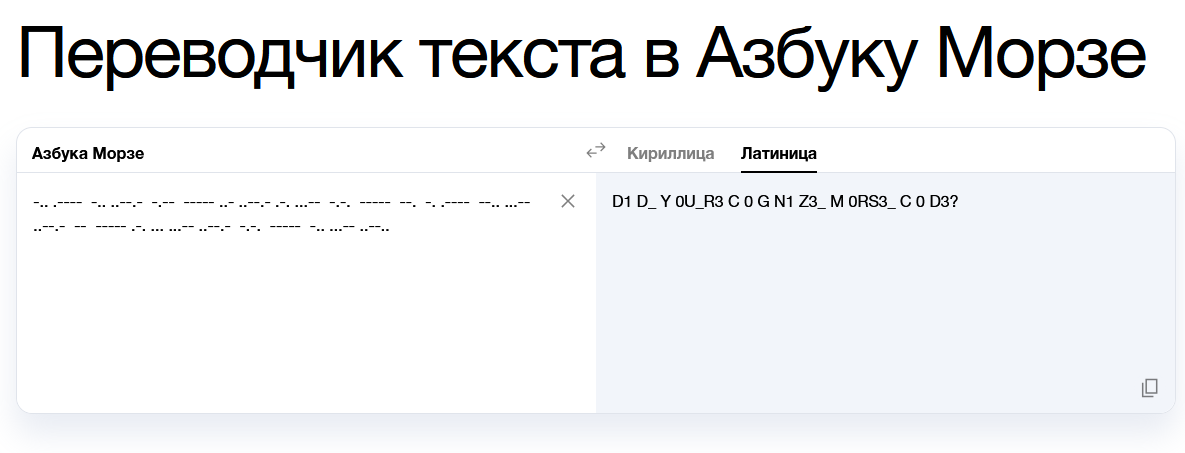


Рисунок 6.1.2. Онлайн перевод текста на Азбуке Морзе.

## 2 Строки хэша

В файле с заданием «task2.txt» содержится строка

0d61f8370cad1d412f80b84d143e1257 f186217753c37b9b9f958d906208506e f623e75af30e62bbd73d6df5b50bb7b5 3a3ea00cfc35332cedf6e5e9a32e94da 9d5ed678fe57bcca610140957afab571 57cec4137b614c87cb4e24a3d003a3e0 f95b70fdc3088560732a5ac135644506 0cc175b9c0f1b6a831c399e269772661 8fa14cdd754f91cc6554c9e71929cce7 e358efa489f58062f10dd7316b65649e eccbc87e4b5ce2fe28308fd9f2a7baf3 4b43b0aee35624cd95b910189b3dc231 b14a7b8059d9c055954c92674ce60032 0cc175b9c0f1b6a831c399e269772661 2db95e8e1a9267b7a1188556b2013b33 2db95e8e1a9267b7a1188556b2013b33 b14a7b8059d9c055954c92674ce60032 8f14e45fceea167a5a36dedd4bea2543 2510c39011c5be704182423e3a695e91 865c0c0b4ab0e063e5caa3387c1a8741 e4da3b7fbbce2345d7772b0674a318d5 b14a7b8059d9c055954c92674ce60032 e358efa489f58062f10dd7316b65649e c4ca4238a0b923820dcc509a6f75849b 6f8f57715090da2632453988d9a1501b e1671797c52e15f763380b45e841ec32 b14a7b8059d9c055954c92674ce60032 415290769594460e2e485922904f345d cfcd208495d565ef66e7dff9f98764da 7b774effe4a349c6dd82ad4f4f21d34c b14a7b8059d9c055954c92674ce60032 6f8f57715090da2632453988d9a1501b a87ff679a2f3e71d9181a67b7542122c 8277e0910d750195b448797616e091ad e1671797c52e15f763380b45e841ec32 b14a7b8059d9c055954c92674ce60032 0cc175b9c0f1b6a831c399e269772661 b14a7b8059d9c055954c92674ce60032 8fa14cdd754f91cc6554c9e71929cce7 2db95e8e1a9267b7a1188556b2013b33 a87ff679a2f3e71d9181a67b7542122c b2f5ff47436671b6e533d8dc3614845d cbb184dd8e05c9709e5dcaedaa0495cf

Слова напоминают хэш. Можно проверить, используя утилиту hash-identifier, которая установлена по умолчанию в дистрибутиве Kali Linux. Прогнав первые 3 слова из задания становится ясно, что символы преобразованы с использованием md5 (рис. 6.2.1)

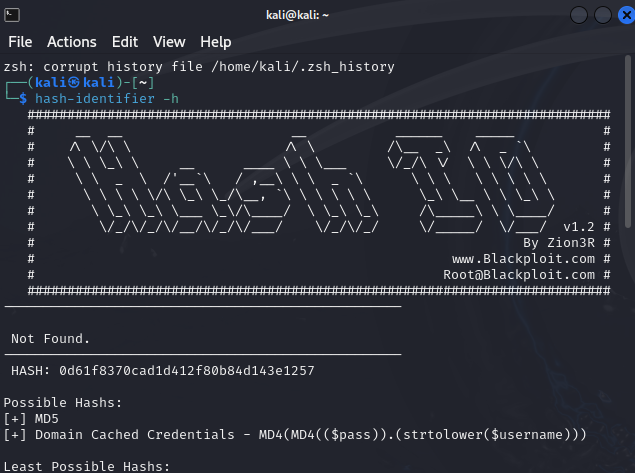


Рисунок 6.2.1. Распознавание хэша.

По определению взломать хэш невозможно, но можно сделать программу, которая создает хэш всех символов алфавита. Сравнивая с полученными словами, можно определить содержание зашифрованного текста.

Для хэширования используется библиотека «hashlib»

import hashlib

alpha = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890{}-\_.,' *# Алфавит*

encode\_alpha = {} *# Список для захэшированного алфавит*

for s in alpha:

    encode\_alpha[hashlib.md5(s.encode('UTF-8')).hexdigest()] = s *# Создание захэшированного алфавита*

f = open('task2.txt', 'r') *# Открытие файла с заданием*

fwords = f.read().split()

decoded\_text = '' *# Декодированный текс*

for i in fwords:

    decoded\_text = decoded\_text + encode\_alpha[i]

print(decoded\_text) *# Вывод флага*