МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Информационной безопасности

ОТЧЕТ

по лабораторной работе №6
по дисциплине «Криптография и защита информации»
Тема: Изучение хэш-функций

Студент гр.8382 _____ Нечепуренко Н.А.
Преподаватель _____ Племянников А.К.

Санкт-Петербург
2021

Цели работы.

Исследование хэш-функций MD5, SHA-256, SHA-512, SHA-3, кода контроля целостности HMAC и анализ атак дополнительной коллизии на хэш-функцию. Получить практические навыки работы с хэшфункциями и атакой на них, в том числе и в программном продукте Cryptool 1 и 2.

Исследование лавинного эффекта MD5, SHA-1, SHA-256, SHA512.

Задание.

1. Открыть текст не менее 1000 знаков. Добавить свое ФИО последней строкой. Перейти к утилите Indiv.Procedures->Hash->Hash Demonstration.
2. Задать хэш-функцию, подлежащую исследованию: MD5, SHA-1, SHA-256, SHA-512.
3. Для каждой хэш-функции повторить следующие действия:
    - Измените (добавлением, заменой, удалением символа) исходный файле
    - Зафиксировать количество измененных битов в дайджесте модифицированного сообщения.
    - Вернуть сообщение в исходное состояние.
4. Выполните процедуру 3 раза (добавлением, заменой, удалением символа) и подсчитайте среднее количество измененных бит дайджеста. Зафиксировать результаты в таблице.

Основные теоретические положения.

Хэш-функцией (hashfunction) называется математическая или иная функция, которая для строки произвольной длины вычисляет некоторое целое значение или некоторую другую строку фиксированной длины.

Хэш-значение может также называться дайджестом (digest) или отпе-

чатком (fingerprint) сообщения.

Алгоритмы хэширования также называют бесключевыми дайджестами сообщений (nonkeyedmessagedigest).

Однонаправленная функция H(M) применяется к сообщению длины M и возвращает значение фиксированной длины h. ($h = H(M)$, где h имеет длину m). Однонаправленные функции имеют дополнительные свойства, позволяющие отличать их от обычных функций, которые вычисляют значение фиксированной длины по входным данным:

1. Зная M, легко вычислить h.
2. Зная H, трудно определить M, для которого $H(M) = h$,
3. Зная M, трудно определить другое сообщение, M1, для которого $H(M) = H(M1)$.

Исследование лавинного эффекта MD5, SHA-1, SHA-256, SHA512 в Cryptool 1.

Сгенерируем текст порядка 1000 символов. Добавим в конец текста ФИО Nechepurenko Nikita Aleksandrovich. Откроем утилиту Hash Demonstration в Cryptool 1 (см. рис. 1).

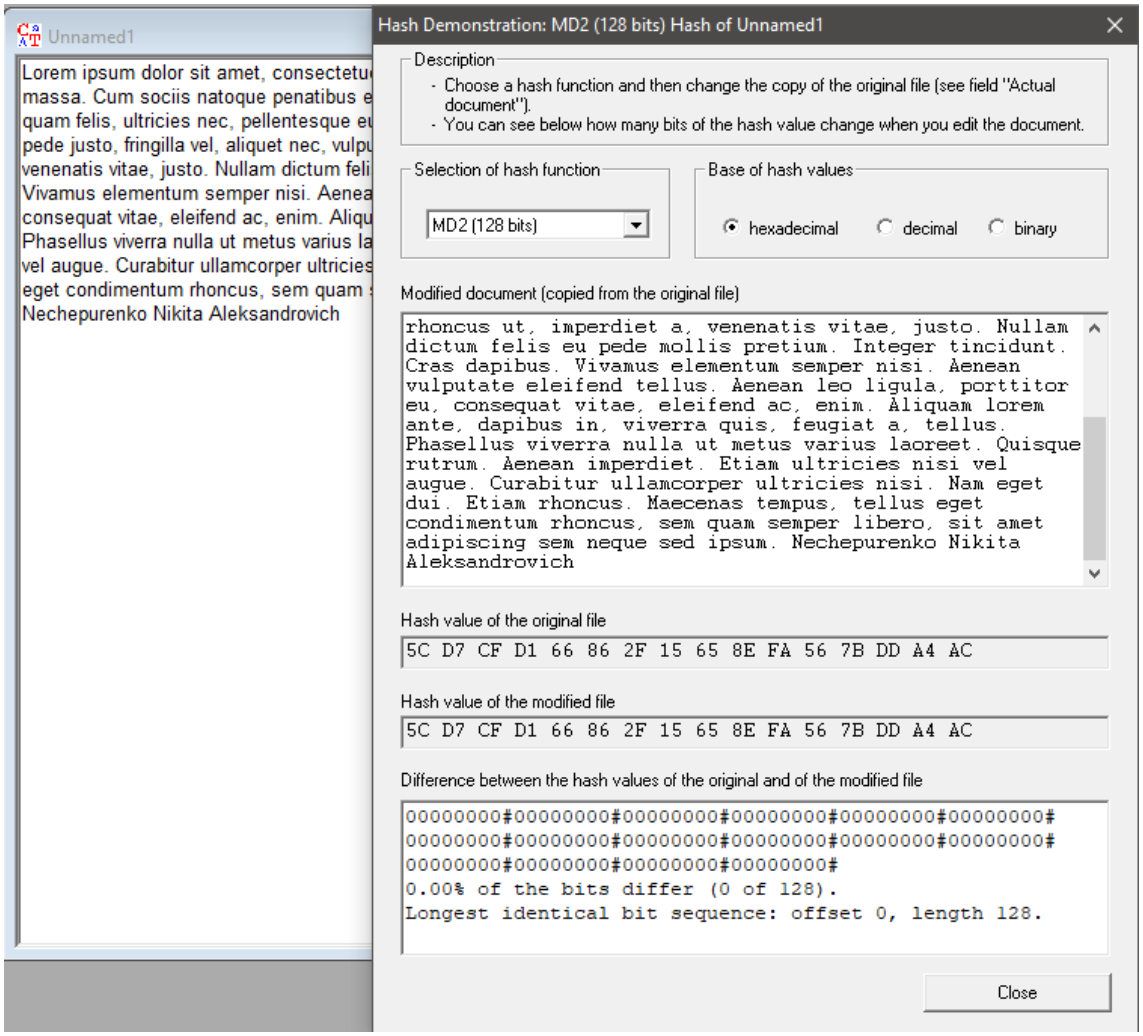

Рисунок 1 – Интерфейс утилиты Hash Demonstration

Изменяя исходный текст, путем добавления, удаления или замены символа, можно пронаблюдать лавинный эффект изменения получаемого хэш-значения. Добавим в конец сообщения символ !, результаты приведены на рисунке 2.

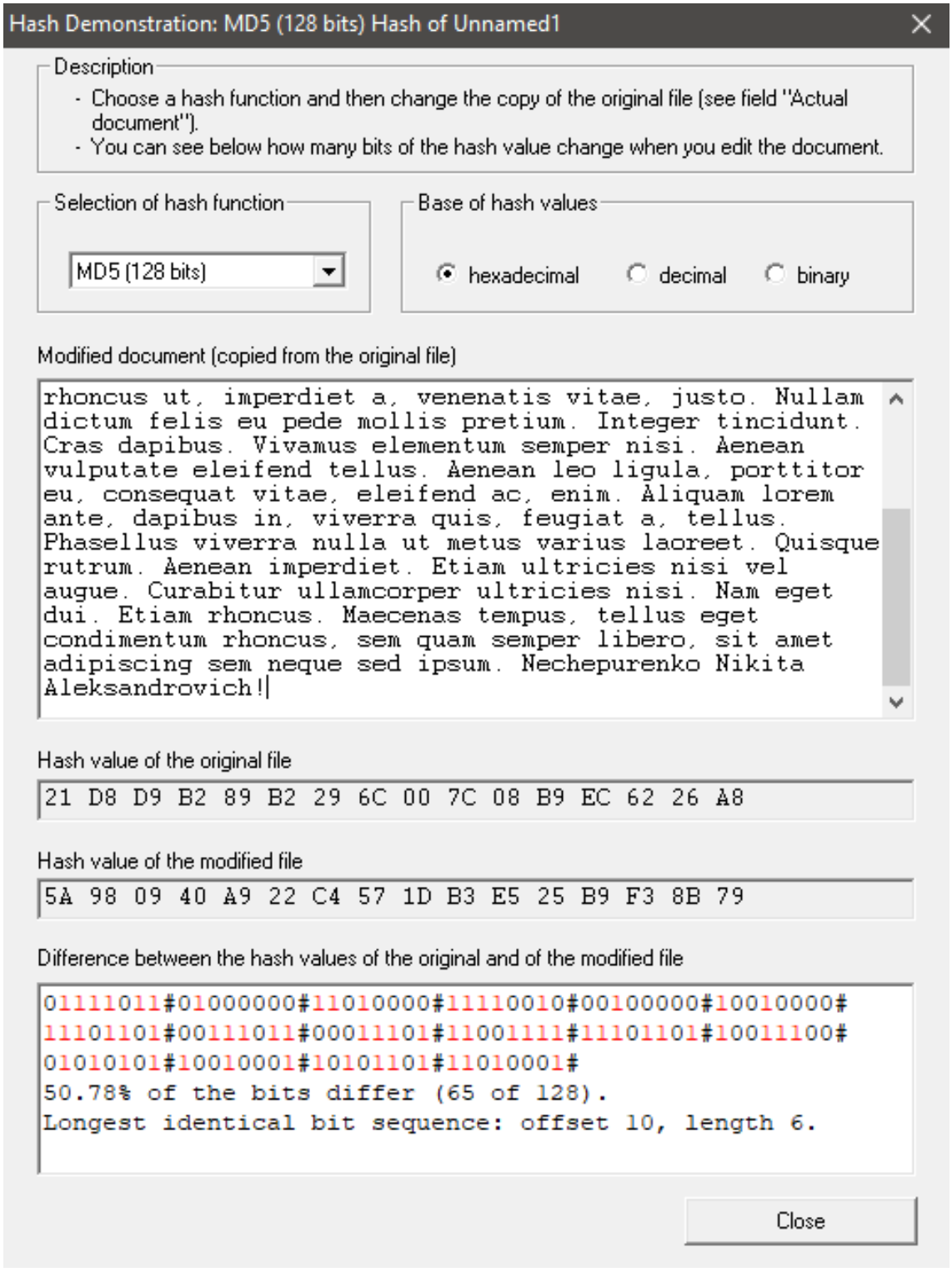

Рисунок 2 – Сравнения полученных хэш-значений после добавления символа

Сравним количество измененных битов в хэш-значении после удаления,

добавления, замены символа в исходном сообщении для алгоритмов MD5, SHA-1, SHA-256, SHA512. Результаты сравнения приведены в таблице 1.

Таблица 1 – Сравнение лавинного эффекта алгоритмов MD5, SHA-1, SHA-256, SHA512

	MD5	SHA-1	SHA-256	SHA-512
Добавление (!)	65 of 128	76 of 160	136 of 256	260 of 512
Изменение (H)	52 of 128	85 of 160	124 of 256	261 of 512
Удаление (h)	56 of 128	89 of 160	122 of 256	264 of 512
Среднее в битах	57.7	83.3	127.3	261.7
Среднее в процентах	45%	52%	49.7%	51.1%

При операции добавление приписывался восклицательный знак в конец текста, при изменении последняя буква заменялась на H, при удалении удалялась последняя буква (h).

Вывод.

В среднем операции, изменяющие один символ (добавление, замена, удаление) исходного текста размером более 1000 символов приводят к изменению примерно половины битов дайджеста. Отклонение среднего в процентах от значения в 50% лежит в пределах статистической погрешности и может быть вызвано особенностями выбора добавляемого символа и символа замены.

Хэш-функция SHA-3.

Задание.

1. Открыть шаблон Keccak Hash (SHA-3) в Cryptool 2
2. В модуле Keccak сделать следующие настройки:
    - Adjust manually=ON
    - Keccak version= SHA3-512
3. Загрузить файл из предыдущего задания
4. Запустить проигрывание шаблона в режиме ручного управления:
    - Сохранить скриншоты преобразований первого раунда
    - Сохранить скриншот заключительной фазы
    - Сохранить значение дайджеста
5. Вычислить значения дайджеста для модифицированных текстов из предыдущего задания
6. Подсчитать лавинный эффект с помощью самостоятельно разработанной автоматизированной процедуры

Основные теоретические положения.

В основе Keccak (SHA-3) лежит конструкция под названием Sponge – губка. Сам алгоритм состоит из 2-х этапов:

1. Впитывание – Absorbing. На каждом шаге очередной блок сообщения pi длиной r подмешивается к части внутреннего состояния S, которая затем целиком модифицируется функцией fмногораундовой бесключевой псевдослучайной перестановкой.
2. Отжатие – Squeezing. Чтобы получить хэш, функция f многократно применяется к состоянию, и на каждом шаге сохраняется кусок размера r до тех пор, пока не получим выход Z необходимой длины (путем конкатенации).

Обобщенная схема работы алгоритма представлена на рисунке ниже.

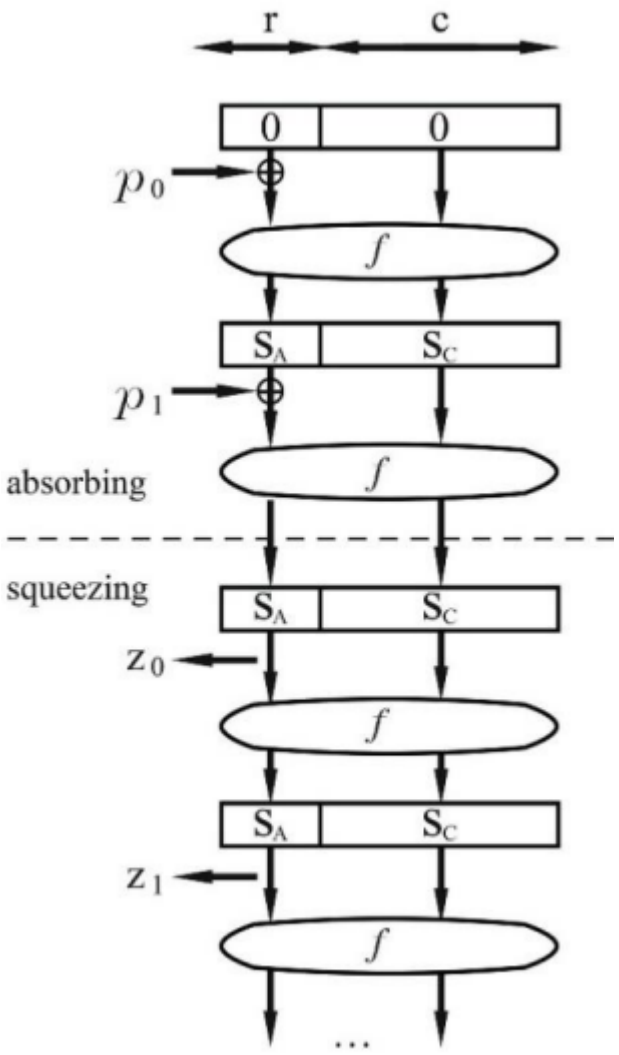

Рисунок 3 – Обобщенная схема работы алгоритма Keccak

Алгоритм Keccak в Cryptool 2.

Рассмотрим визуализацию процесса работы алгоритма Keccak в Cryptool 2. Рассмотрим один раунд хэширования. Скриншоты интерфейса программы приведены ниже.

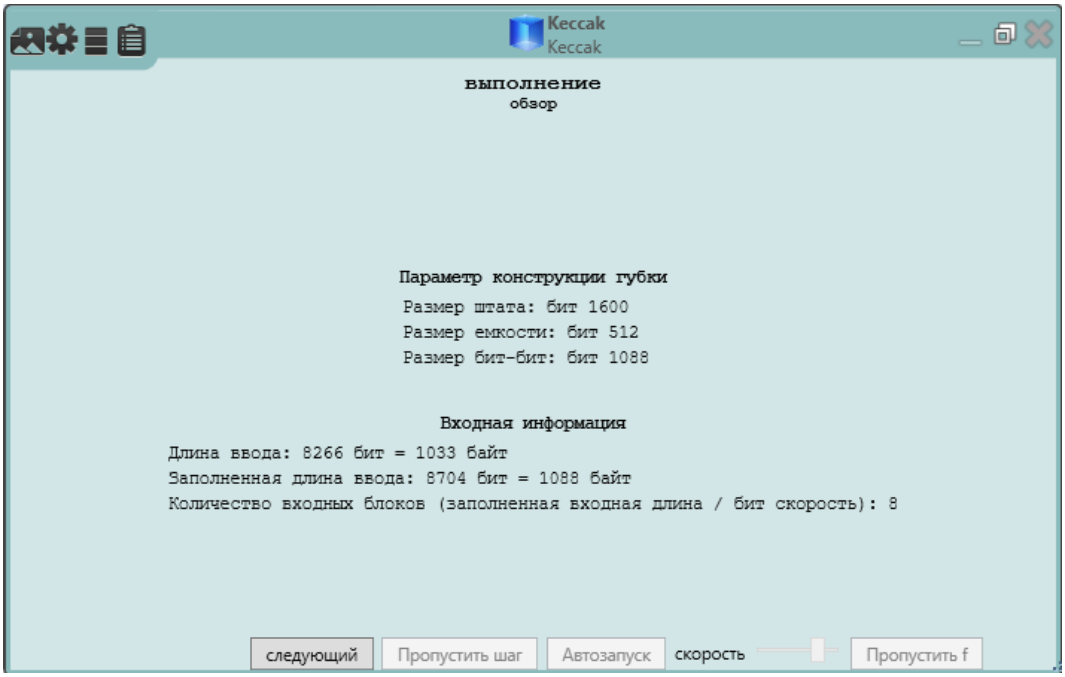

Рисунок 4 – Исходная информация о тексте

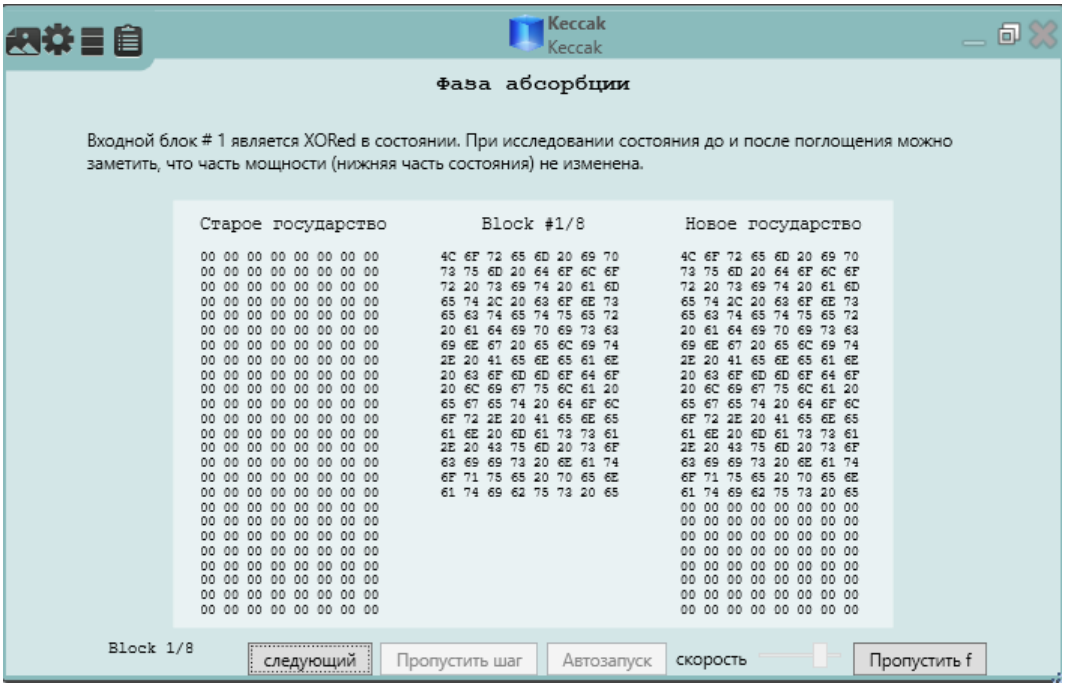

Рисунок 5 – Построение первого блока

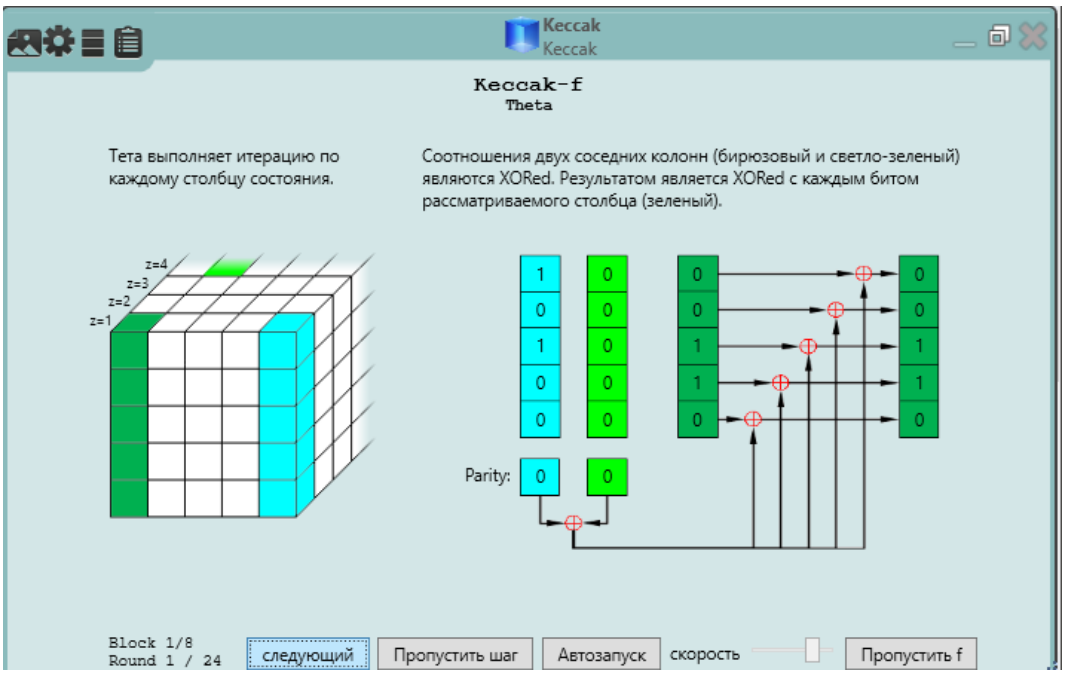

Рисунок 6 – Начальное состояние стадии Theta

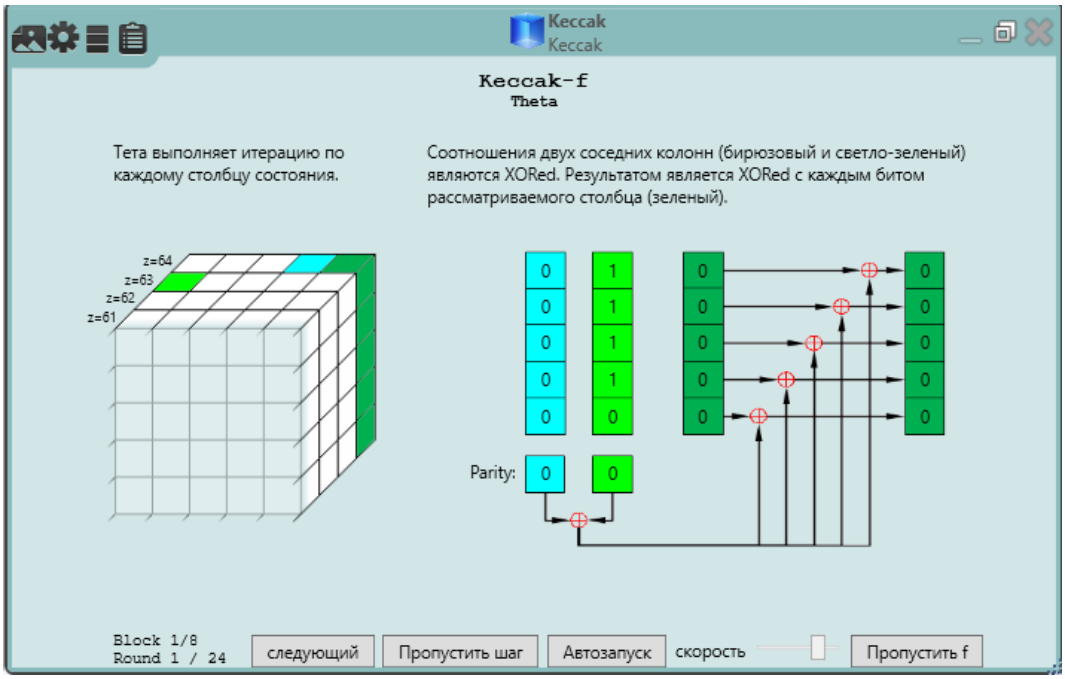

Рисунок 7 – Конечное состояние стадии Theta

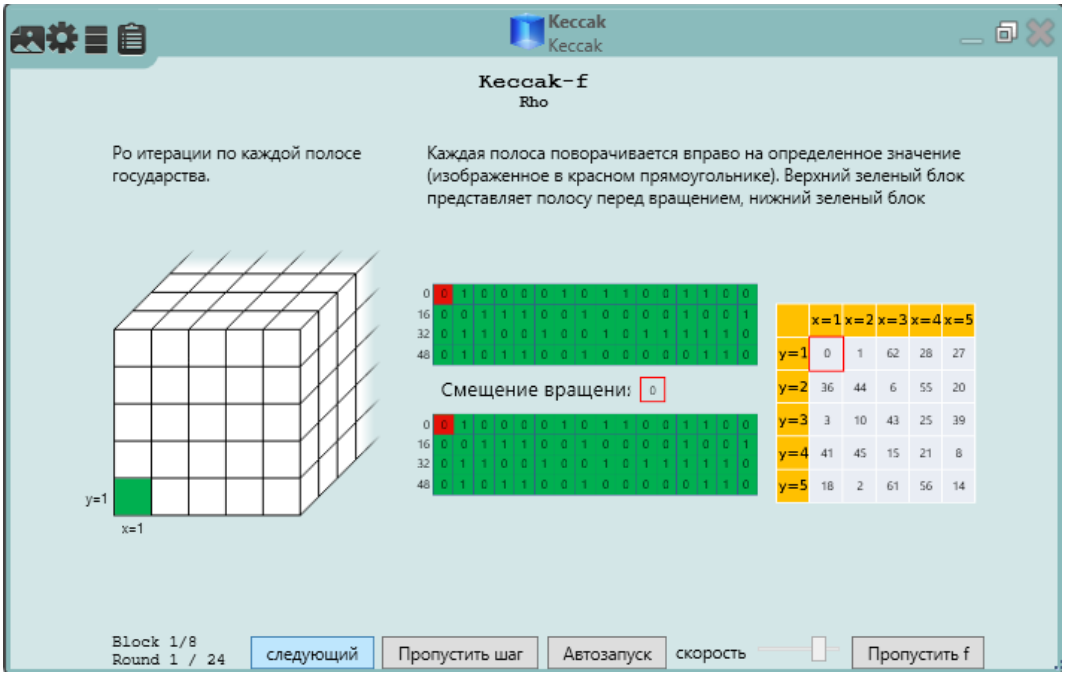

Рисунок 8 – Начальное состояние стадии Rho

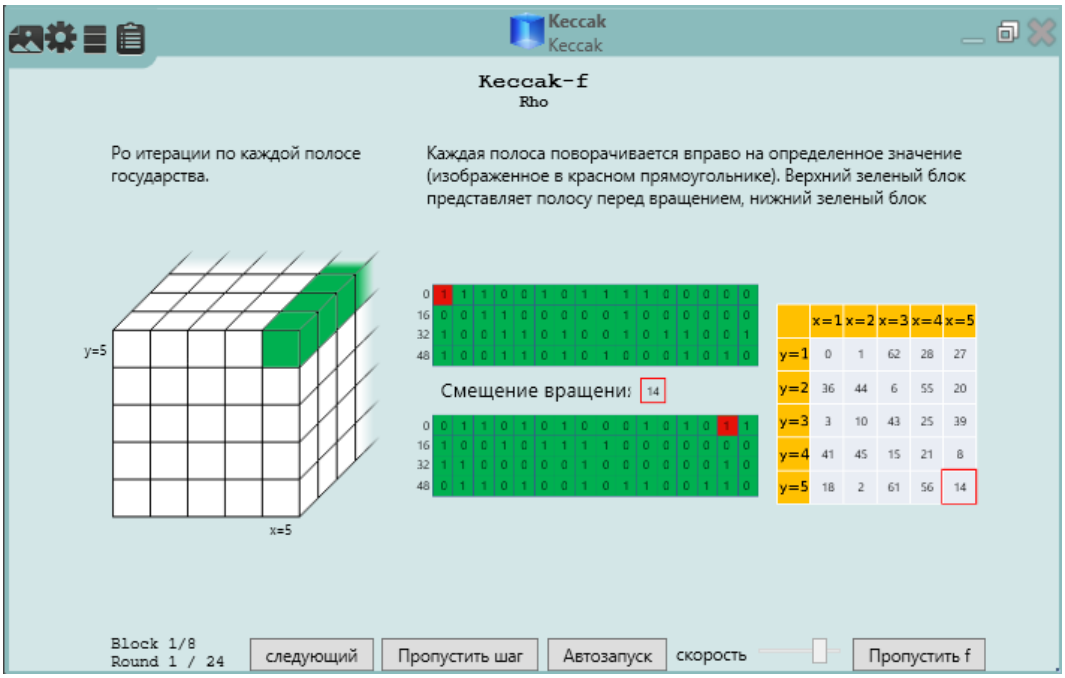

Рисунок 9 – Конечное состояние стадии Rho

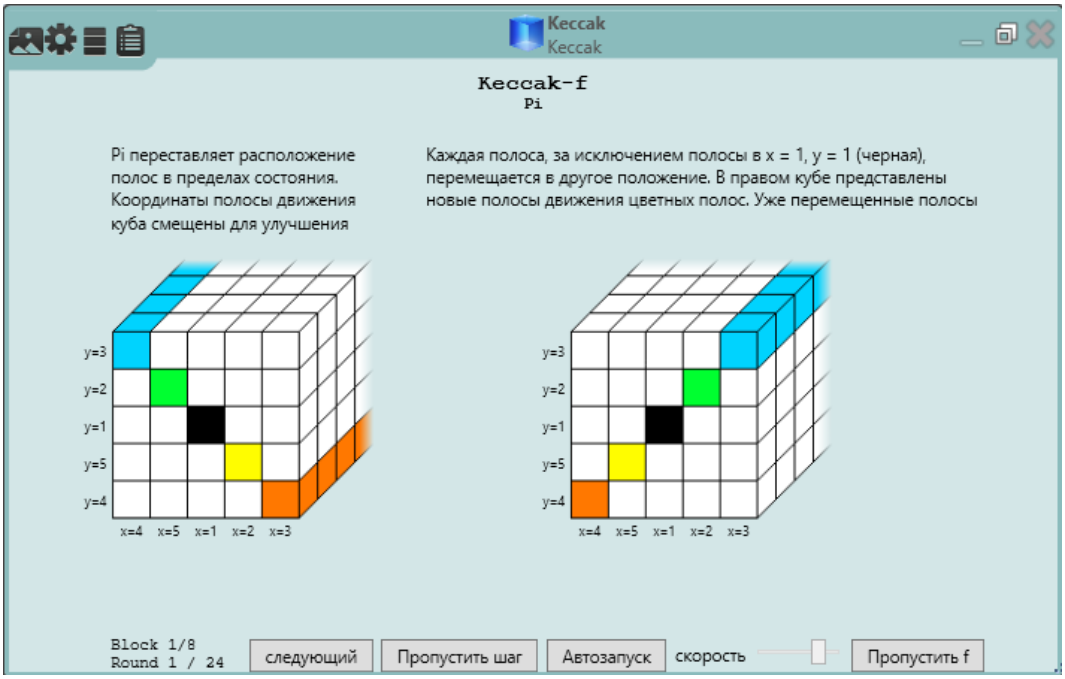

Рисунок 10 – Начальное состояние стадии Pi

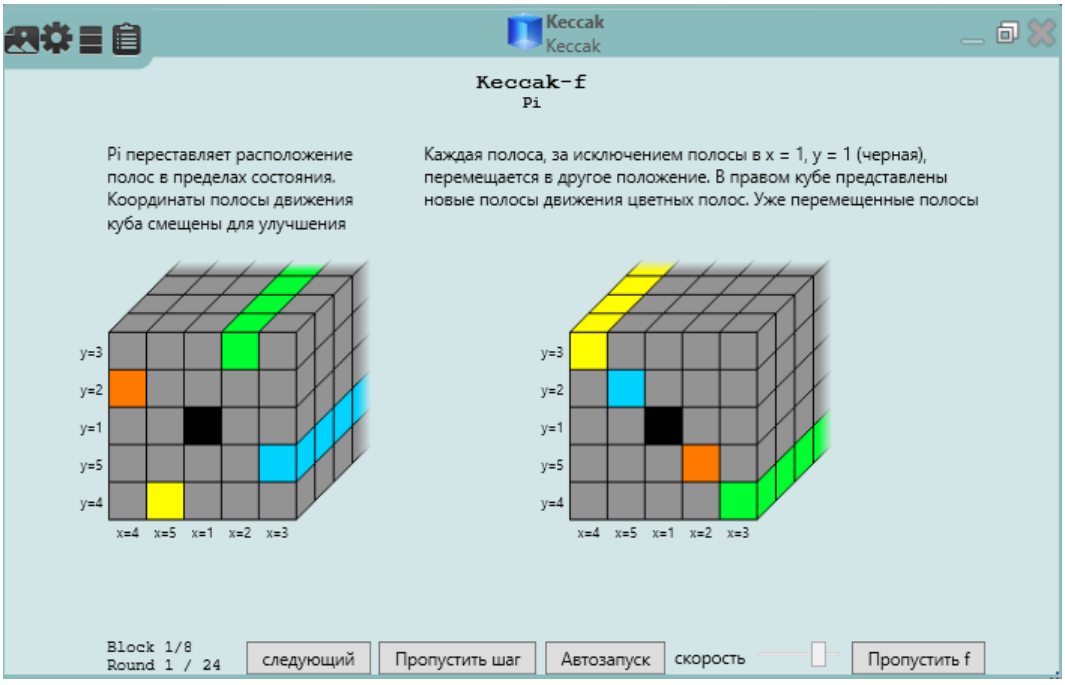

Рисунок 11 – Конечное состояние стадии Pi

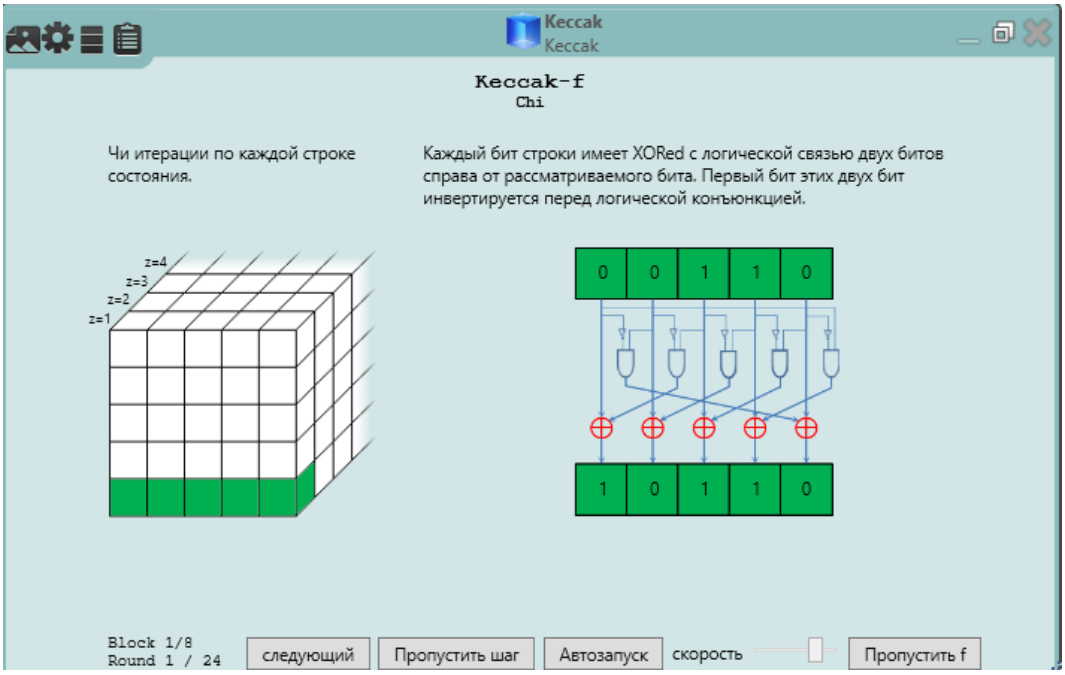

Рисунок 12 – Начальное состояние стадии Chi

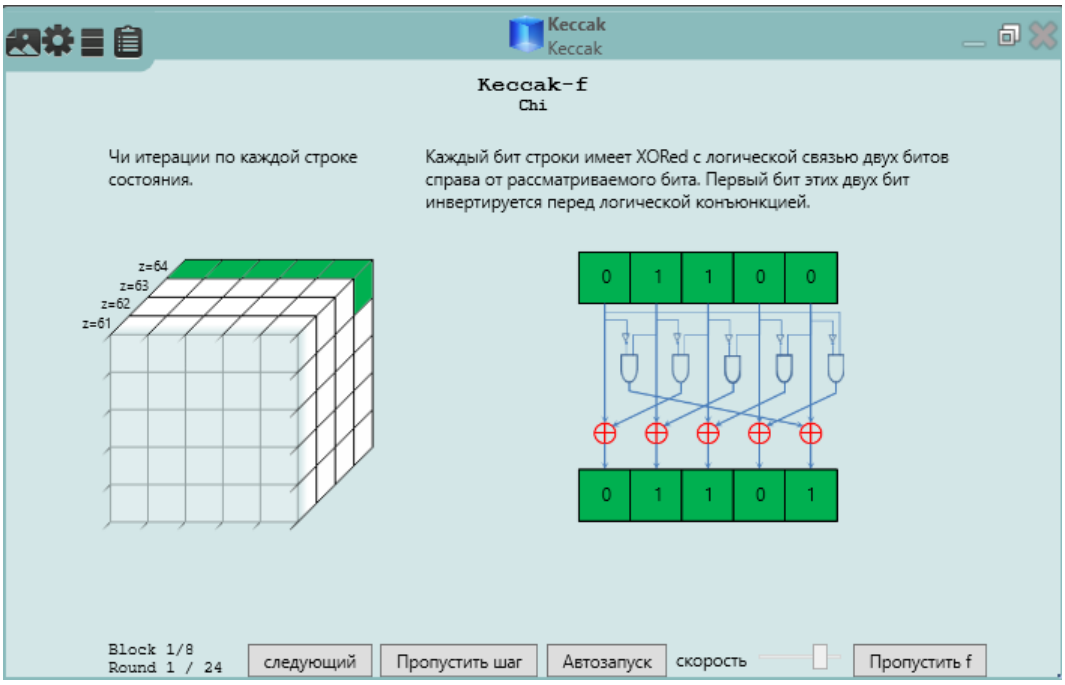

Рисунок 13 – Конечное состояние стадии Chi

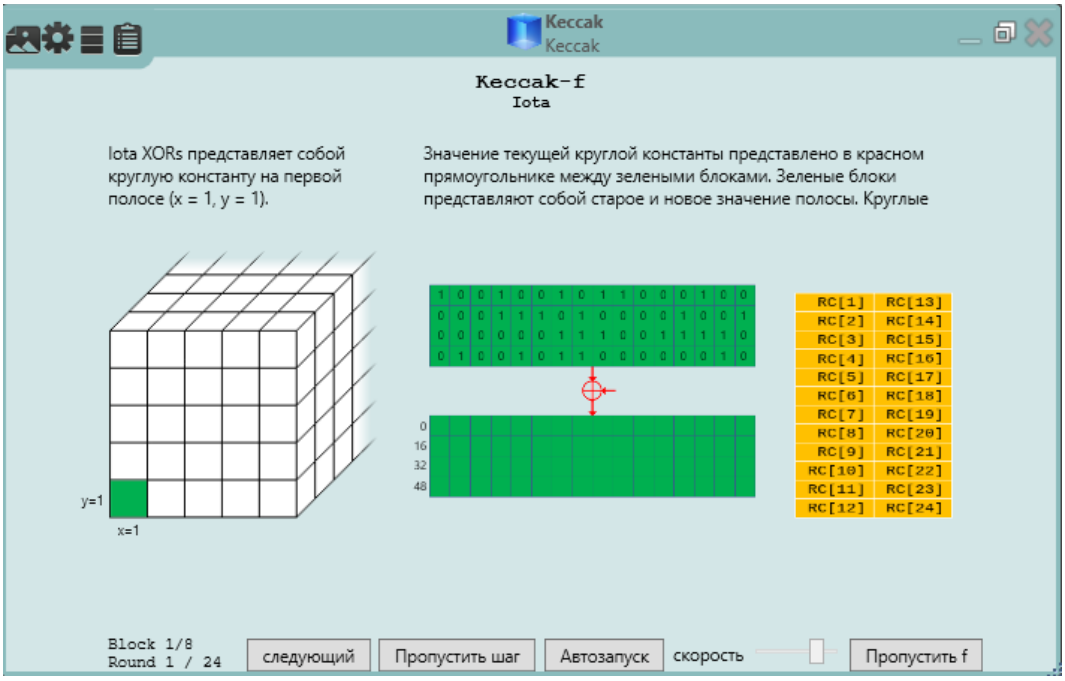

Рисунок 14 – Начальное состояние стадии Iota

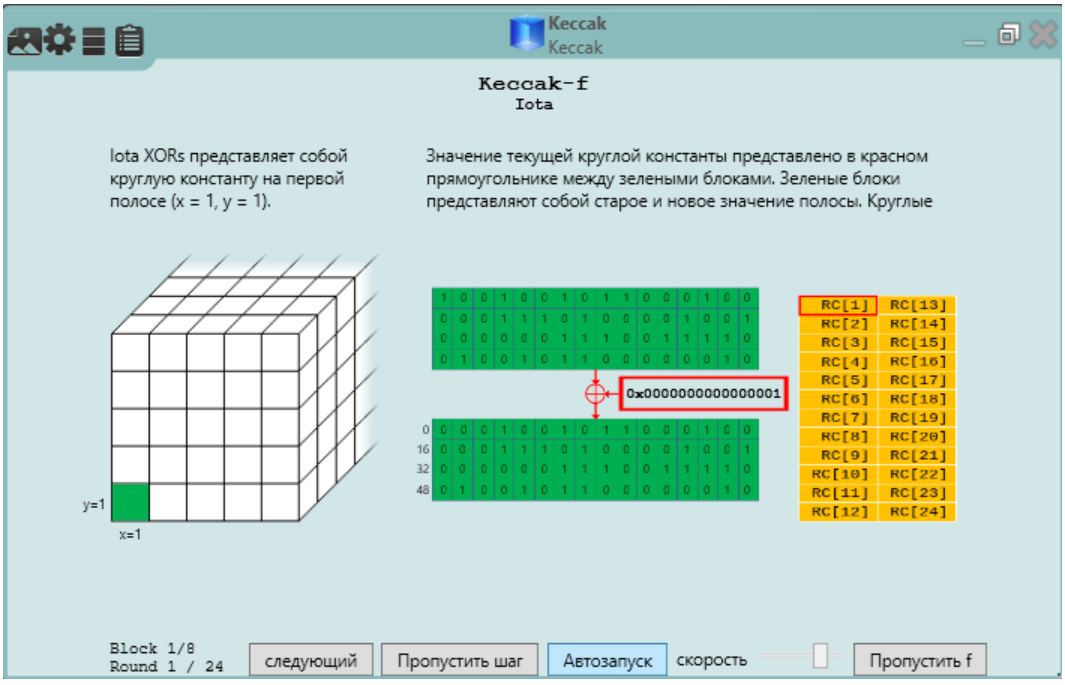

Рисунок 15 – Конечное состояние стадии Iota

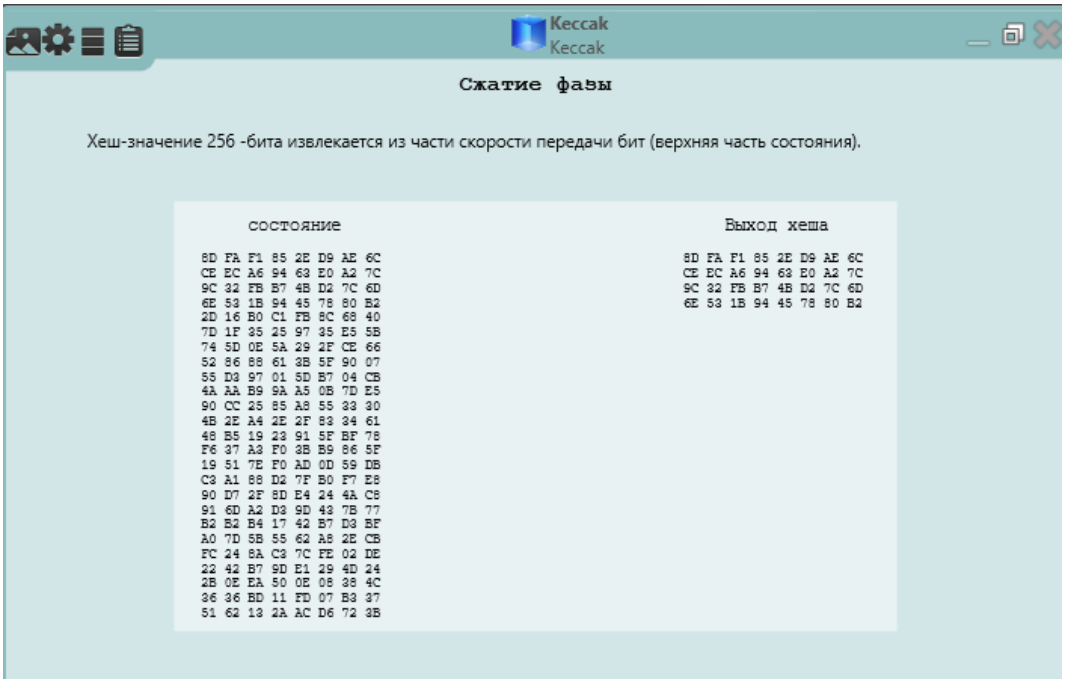

Рисунок 16 – Скриншот заключительной фазы

Аналогично первому разделу проведем сравнение лавинного эффекта для операций добавления, изменения и удаления символа. Дайджест исходного текста: D4 76 A9 F6 B6 29 A8 31 37 17 DC A8 07 7F 75 15 4C D3 49 0F 95 C4 29 ED 5C 99 DD 70 9B 32 B3 16 D4 64 A1 02 02 CA 83 C4 72 51 A7 2D 4E 37 2C 4E 9A B8 AD BB 71 43 8A D2 CB 9A 9F CD D6 FC 1A 6D.

Дайджест после добавления восклицательного знака: 7C 6F F6 94 01 3D E4 D6 37 E3 C1 AF 04 90 48 05 6C F7 C8 0B 79 A7 C1 F8 E1 6B 2E 20 24 0C 60 DC 74 A5 FA 8D 81 AE 39 19 47 EF 71 37 BD CB 7D 1B 58 8B A9 94 74 BC 0E D0 FE 26 43 AA 3D A7 2B C5.

Дайджест после замены последней буквы h на H: A0 C1 BB 90 00 23 A6 70 B2 A3 B0 06 14 55 F0 E5 4D 06 19 26 CB 07 53 98 4A 2D 9B D9 E6 12 B1 82 89 3C CC 28 2E A5 E6 FC 46 F9 E6 FB C4 43 B5 51 6F 96 7A 08 9A D4 89 C2 A0 43 0B 79 B4 50 3B FE.

Дайджест после удаления последней буквы h: 92 3B 70 73 70 24 21 1F

32 F5 60 A4 B0 89 29 28 ED 06 56 2F E8 3B F1 39 89 ED BD 36 B0 9D F6 23 93 BA E1 F6 EE AF 46 A0 F3 D4 63 52 FF F3 88 44 78 EA 7E F1 C4 6A DA E8 DA 45 F2 0A 2E FE 6A D0.

После добавления символа разница 256 битов, после замены 238, после удаления 252. В среднем, получилось 248.7 битов или 48.6%.

Вывод.

С помощью программы Cryptool 2 была изучена визуализация работы алгоритма Keccak (SHA3-512), а также исследован лавинный эффект операций добавления, замены и удаления символа. В среднем лавинный эффект алгоритма Keccak оказался равным около 50%, что примерно равно эффекту рассмотренных ранее алгоритмов хэширования.

Контроль целостности по коду HMAC.

Задание.

1. Выбрать текст на английском языке (не менее 1000 знаков), добавить собственное ФИО и сохранить в файле формата .TXT
2. Придумать пароль и сгенерировать секретный ключ утилитой Indiv.Procedures->Hash-> Key Generation из Cryptool 1. Сохранить ключ в файле формата .TXT. Прочитать Help к этой утилите.
3. Сгенерировать HMAC для имеющегося текста и ключа с помощью утилиты Indiv.Procedures->Hash-> Generation of HMACs. Сохранить HMAC в файле формата .TXT. Прочитать Help к этой утилите.
4. Передать пароль, HMAC (и его характеристики), исходный текст и модифицированный текст коллеге, не раскрывая, какой текст является корректным. Попросите коллегу определить это самостоятельно

Основные теоретические положения.

HMAC - один из механизмов проверки целостности информации, позволяющий гарантировать то, что данные, передаваемые или хранящиеся в ненадёжной среде, не были изменены посторонними лицами. Механизм HMAC использует MAC — стандарт, описывающий способ обмена данными и способ проверки целостности передаваемых данных с использованием секретного ключа. Два клиента, использующие HMAC, как правило, разделяют общий секретный ключ. HMAC — надстройка над MAC.

Алгоритм HMAC можно записать в виде одной формулы:

$$HMAC_K(text) = H((K \oplus opad)||H((K \oplus ipad)||text))$$

где $\oplus$ — операция xor, ||- конкатенация, K — секретный ключ, ipad — блок вида (0x36 0x36 0x36 …0x36), где байт 0x36 повторяется b раз, H — хэшфункция, opad — блок вида (0x5c 0x5c 0x5c …0x5c), где байт 0x5c повторяется b раз.

Использование HMAC в Cryptool 1.

С помощью Cryptool 1 был сгенерирован ключ 7D 63 B9 FB A3 E5 17 35 71 09 AD 97 CD F5 F5 9B (см. рис. ниже).

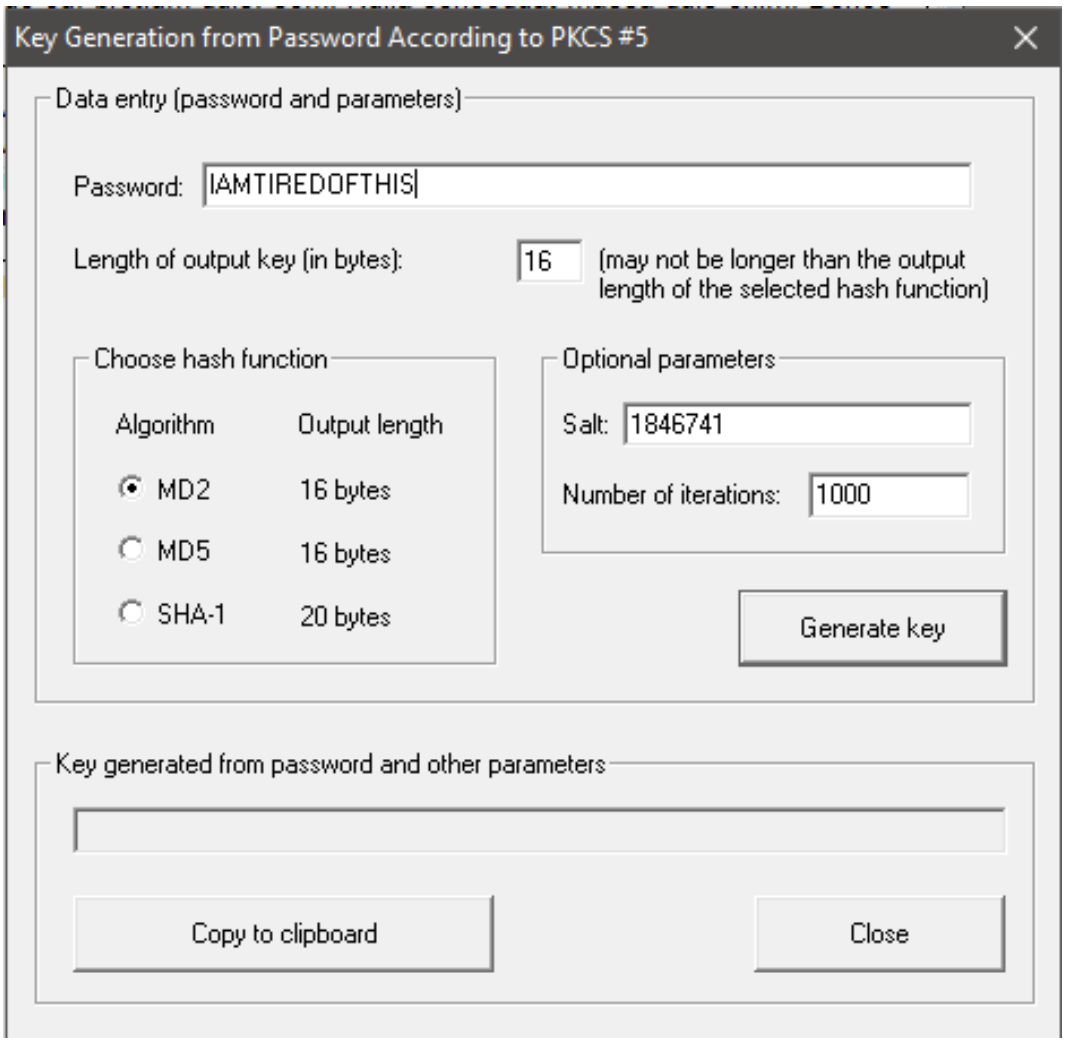

Рисунок 17 – Сгенерированный ключ

Вычислим HMAC с помощью встроенной утилиты (см. рис. 18).

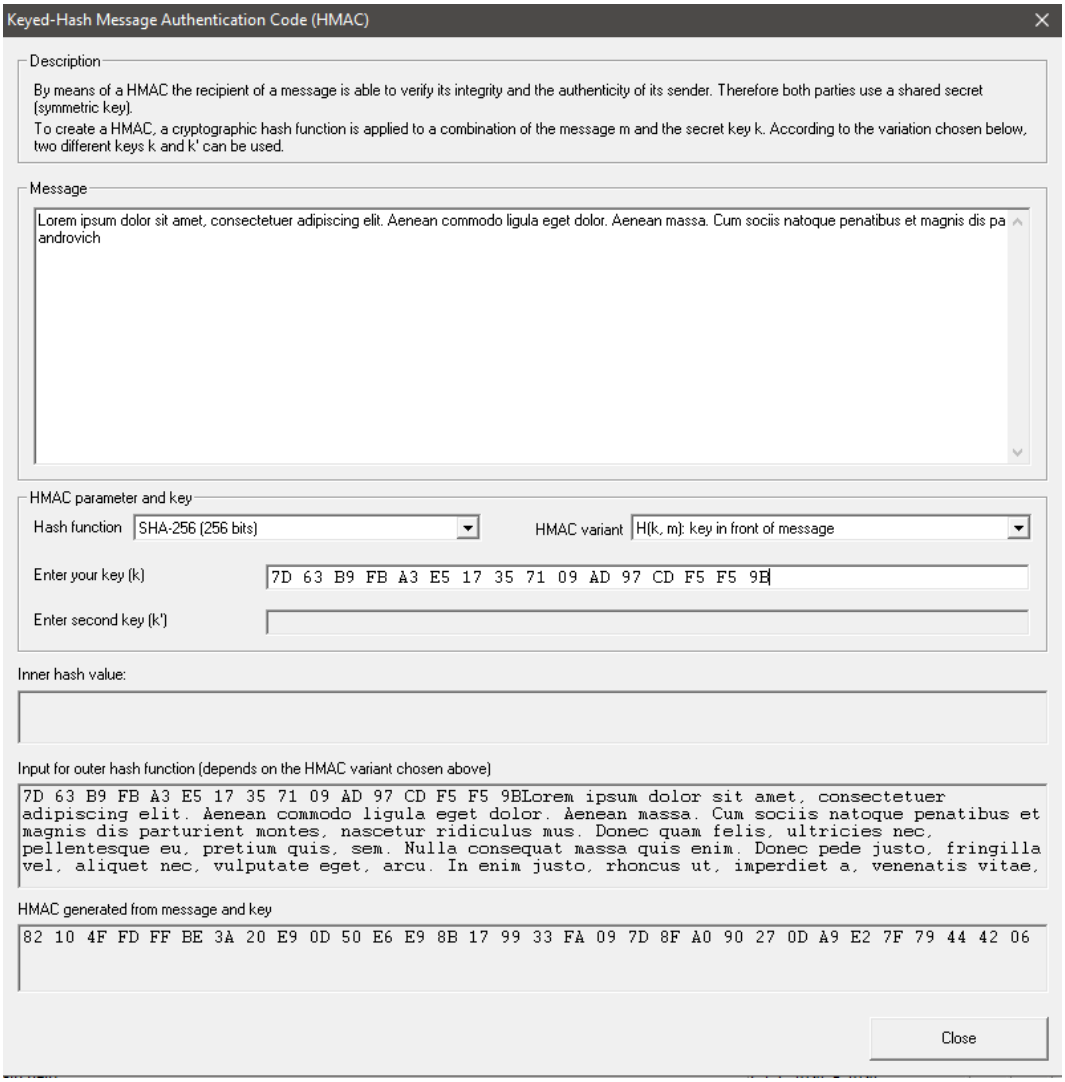

Рисунок 18 – Вычисление HMAC

Значение HMAC равно: 82 10 4F FD FF BE 3A 20 E9 0D 50 E6 E9 8B 17 99 33 FA 09 7D 8F A0 90 27 0D A9 E2 7F 79 44 42 06

Передадим коллеге параметры хэширования HMAC и секретный ключ. Передадим 2 текста, исходный и модифицированный, в который была добавлена в конец фраза «hello world!».

У измененного текста значение HMAC отличается (см. рис. 19), а значит это не подлинный текст.

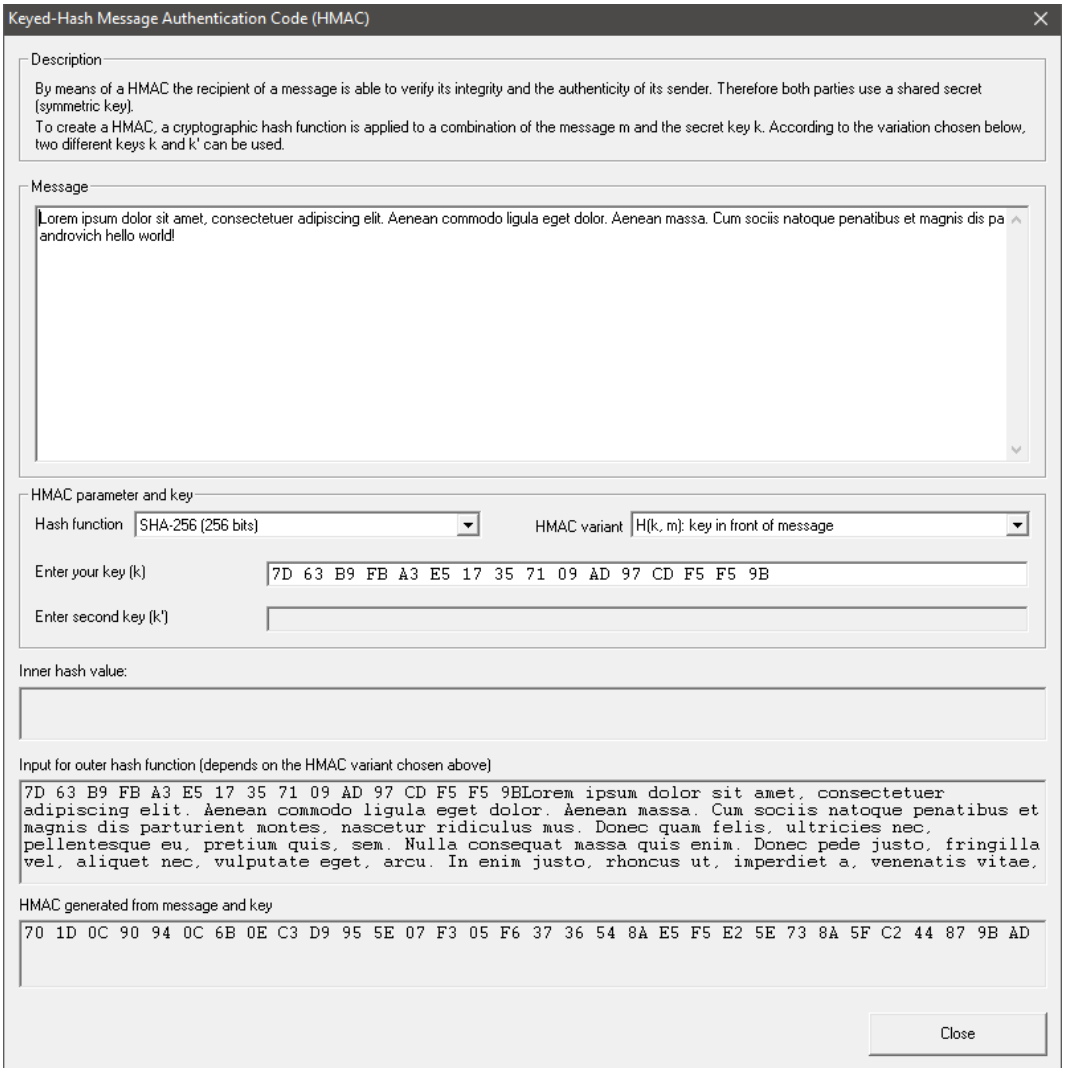

Рисунок 19 – Вычисление HMAC модифицированного текста

Выводы.

С помощью механизма HMAC была выполнена валидация полученного от коллеги сообщения на корректность. При наличии секретного ключа и HMAC дайджеста исходного текста можно определить, были ли внесены изменения в открытый текст. Для этого необходимо вычислить HMAC полученного текста и сравнить его с известным значением HMAC.

Атака дополнительной коллизии на хэш-функцию.

Задание.

1. Сформировать два текста на английском языке - один истинный, а другой фальсифицированный. Сохранить тексты в файлах формата *.txt
2. Утилитой Analysis-> Attack on the hash value…произвести модификацию сообщений для получения одинакового дайджеста. В качестве метода модификации выбрать Attach characters-> Printable characters.
3. Проверить, что дайджесты сообщений действительно совпадают с заданной точностью.
4. Сохранить исходные тексты, итоговые тексты и статистику атаки для отчета.
5. Зафиксировать временную сложность атаки для 8, 16, 32, 40, 48, …бит совпадающих частей дайджестов.

Основные теоретические положения.

Атака дополнительной коллизии основана на одной из проблем парадокса дня рождений. Он заключается в следующем: в группе, состоящей из 23 или более человек, вероятность совпадения дней рождения (число и месяц) хотя бы у двух людей превышает 50 %. Применительно к хэшфункции это означает, что сложность атаки, целью которой является поиск двух сообщений с одинаковыми значением хэш-функции, пропорционально $\sqrt{2^N}$, где N – длина хэш-кода.

Атака на дайджест в Cryptool 1.

Сформируем истинный и фальсифицированный тексты на английском языке (см. рис. 20).

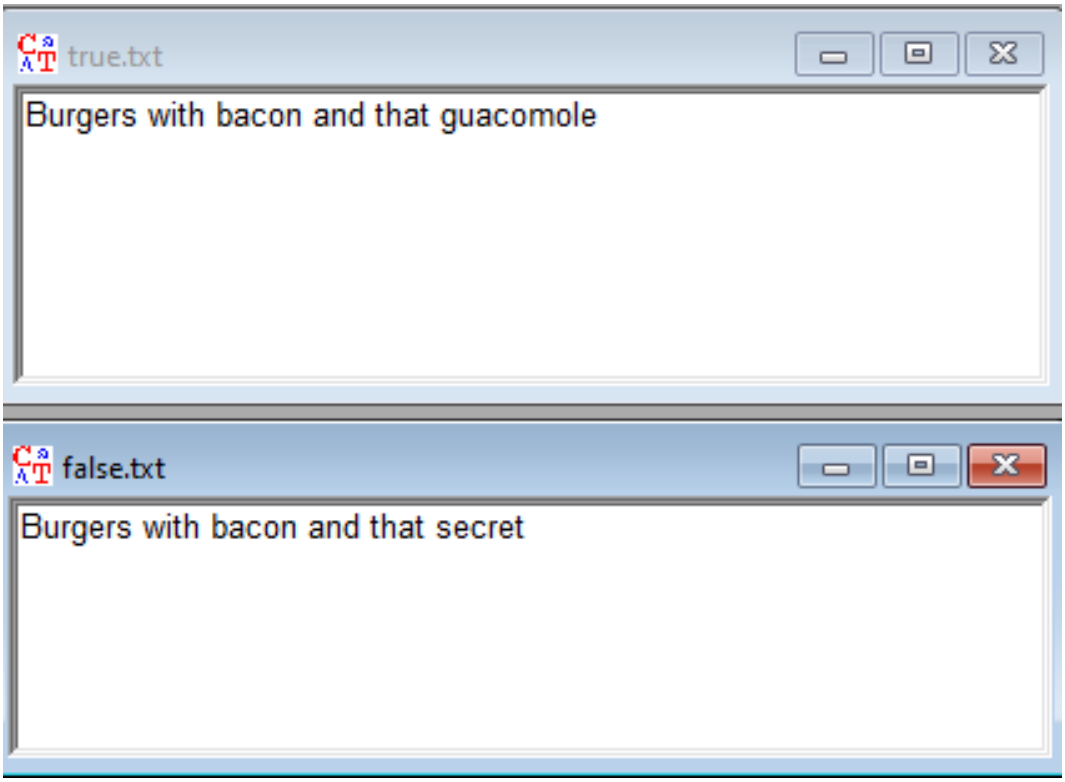

Рисунок 20 – Сформированные тексты

Интерфейс атаки средствами Cryptool с выбранными настройками приведен на рисунке 21.

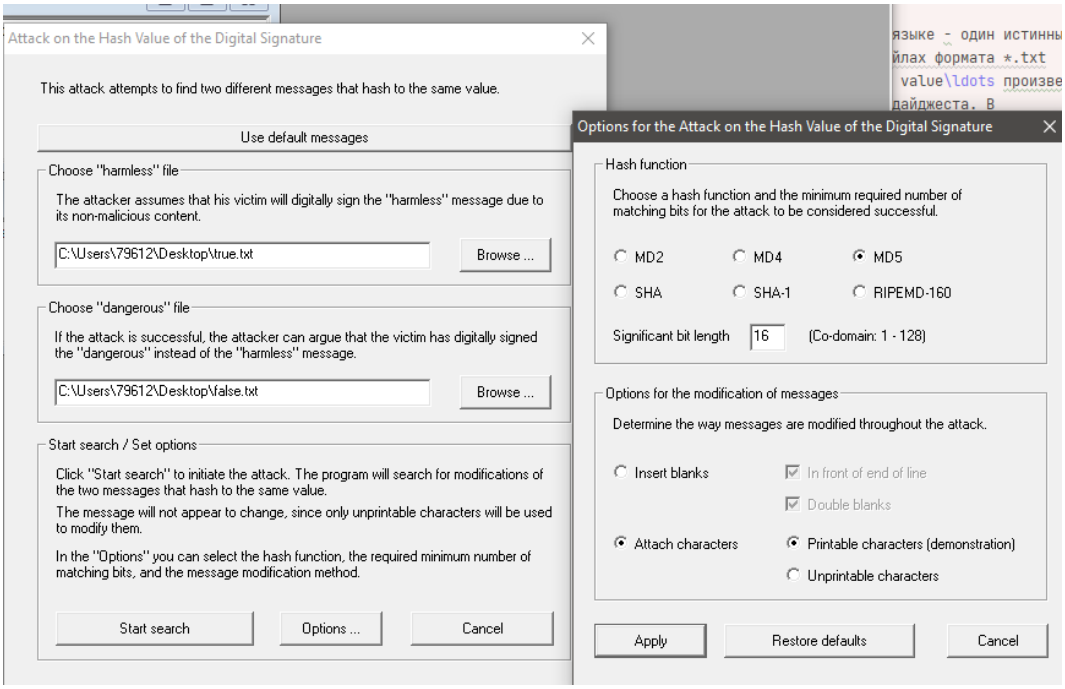

Рисунок 21 – Интерфейс атаки на дайджест Cryptool 1

Результат атаки приведен на рисунке 22.

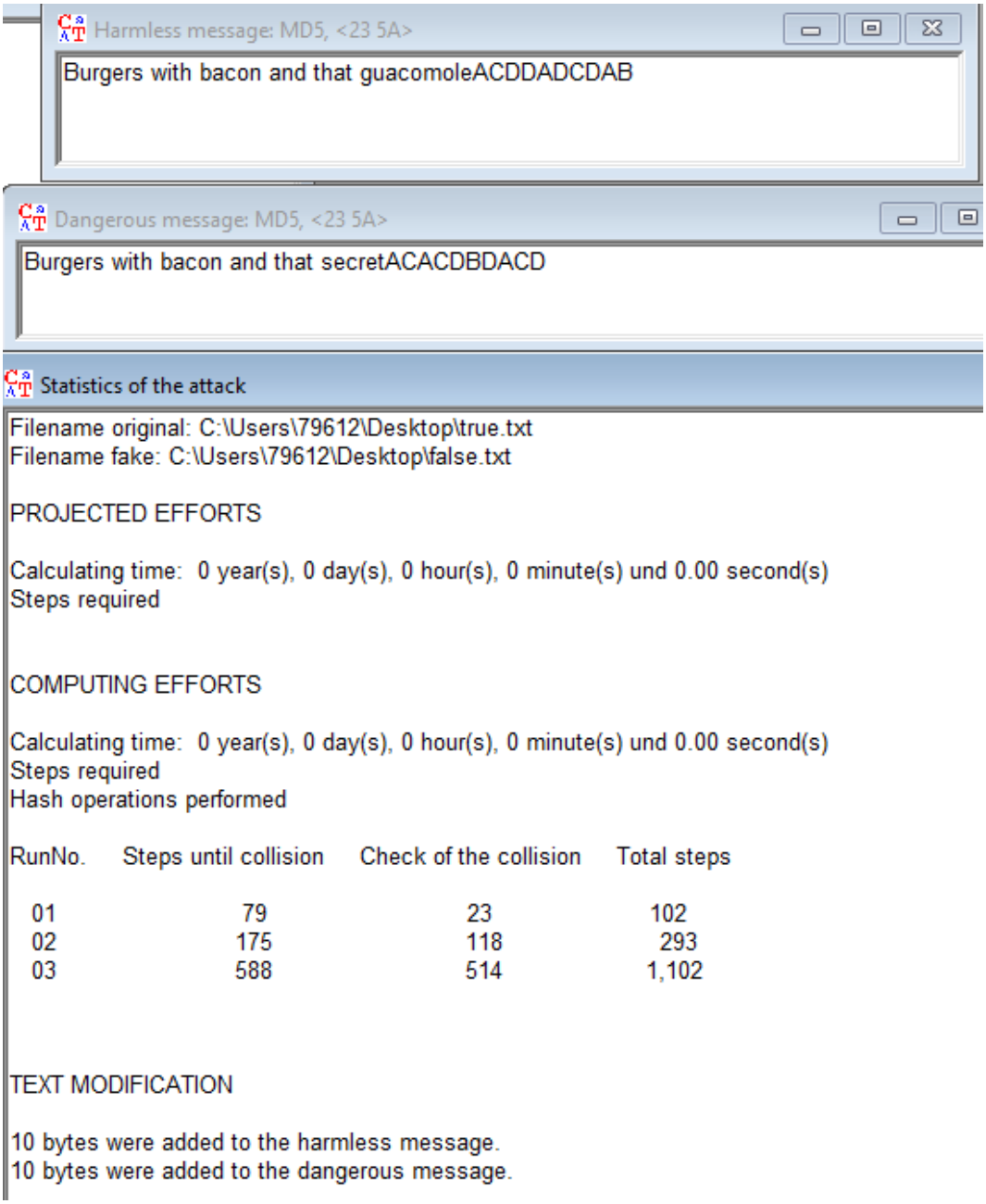

Рисунок 22 – Результат атаки для 16 бит совпадающих частей дайджеста

Исследуем временную сложность атаки в зависимости от количества бит совпадающих частей дайджеста (см. табл. 2).

Таблица 2 – Временная сложность атаки в зависимости от количества бит совпадающих частей дайджеста

Количество совпадающих битов	Время атаки
8	меньше секунды
16	меньше секунды
32	2 секунды
40	7 секунды
48	1 минута 17 секунд
64	3 часа 20 минут
96	около 30 лет
128	$2 \cdot 10^{92}$ лет

Вывод.

Была проведена атака на истинный и фальсифицированный текст средствами Cryptool 1. Эмпирически полученные результаты отражают экспоненциальный характер зависимости времени взлома от количества бит, что согласуется с теоретической оценкой.

Заключение.

1. Был изучен лавинный эффект алгоритмов MD5, SHA-1, SHA-256, SHA-512.

	MD5	SHA-1	SHA-256	SHA-512
Добавление (!)	65 of 128	76 of 160	136 of 256	260 of 512
Изменение (H)	52 of 128	85 of 160	124 of 256	261 of 512
Удаление (h)	56 of 128	89 of 160	122 of 256	264 of 512
Среднее в битах	57.7	83.3	127.3	261.7
Среднее в процентах	45%	52%	49.7%	51.1%

В среднем операции, изменяющие один символ (добавление, замена, удаление) исходного текста размером более 1000 символов приводят к изменению примерно половины битов дайджеста. Отклонение среднего в процентах от значения в 50% лежит в пределах статистической погрешности и может быть вызвано особенностями выбора добавляемого символа и символа замены.

2. С помощью Cryptool 2 была рассмотрена визуальная демонстрация работы алгоритма Keccak (SHA-3). Алгоритм по схеме Sponge, состоящей из двух этапов: впитывание (строка разбивается на куски определенной длины, на каждом шаге очередной блок сообщения подмешивается к части внутреннего состояния S, которая затем целиком модифицируется функцией f-многораундовой бесключевой псевдослучайной перестановкой) и отжатие (для получения хэша функция f многократно применяется к состоянию, и на каждом шаге сохраняется кусок определённого размера, пока не получится хэш определенной длины). В среднем лавинный эффект алгоритма Keccak, связанный с удалением/добавлением/изменением символа, оказался равным около 50%, что примерно равно эффекту рассмотренных ранее алгоритмов хэши-

рования.

3. Был рассмотрен механизм проверки целостности информации HMAC (Hash-based Message Authentication Code – код аутентификации сообщения на основе хэширования). Два клиента, использующие HMAC, разделяют общий секретный ключ. Для проверки корректности полученного сообщения вычисленное значение HMAC сравнивается с известным. С помощью Cryptool 1 была смоделирована модель взаимодействия отправителя и получателя, были проверены на корректность два сообщения, одно из которых было истинным, а второе – сфальсифицированным.

4. Была рассмотрена атака дополнительной коллизии средствами Cryptool 1. Полученная экспериментально зависимость времени взлома от количества совпадающих битов имеет экспоненциальный характер, что согласуется с известными теоретическими оценками. Зависимость времени взлома от количества совпадающих битов для алгоритма MD5 приведена ниже.

Количество совпадающих битов	Время атаки
8	меньше секунды
16	меньше секунды
32	2 секунды
40	7 секунды
48	1 минута 17 секунд
64	3 часа 20 минут
96	около 30 лет
128	$2 \cdot 10^{92}$ лет