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

ОТЧЁТ

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

Студент гр. 6304 _____ Корытов П.В.
Преподаватель _____ Племянников А.К.

Санкт-Петербург
2019

Цель 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 хэш- функциями и атакой на них, в том числе и в программном продукте Cryptool 1 и 2.

1. Исследование лавинного эффекта MD5, SHA-1, SHA-256, SHA-512

1.1. Описание алгоритмов

1.1.1. MD5

MD5 перерабатывает сообщение произвольной длины в сообщение длины 128 бит.

Сообщение дополняется, чтобы длина делилась на 512

- Добавляется бит 1
- Дописываются нули, чтобы осталось 64 бита
- В последние 64 бита записывается длина сообщения по модулю 2^{64}

После чего сообщение делится на блоки по 512 бит. Для каждого блока выполняет 4 раунда по 16 операций.

Структура одной операции MD5 представлена на рис. 1

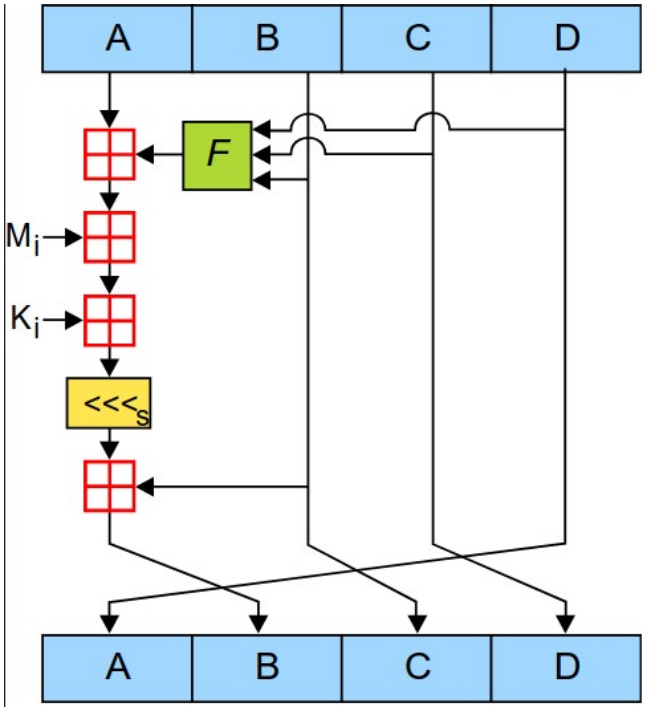

Рисунок 1. Одна операция MD5

MD5 оперирует на 128-битном состоянии, поделённом на 4 слова по 32

бит.

- Инициализация для 1-го блока:
  - $A := 0x67452301$
  - $B := 0xefcdab89$
  - $C := 0x98badcfe$
  - $D := 0x10325476$
- F — нелинейная функция:

$$F(B,C,D) = \begin{cases} (B \wedge C) \vee (\neg B \wedge D), & \text{если } i \in [0,15] \\ (B \wedge D) \vee (C \wedge \neg D), & \text{если } i \in [16,31] \\ B \oplus C \oplus D, & \text{если } i \in [32,47] \\ C \oplus (B \vee \neg D) & \text{если } i \in [48,63] \end{cases} \tag{1.1}$$

- M_i — часть входного блока размером 32 бит. Выбор номера блока:

$$\begin{cases} i, & \text{если } i \in [0,15] \\ (5i+1) \bmod 16, & \text{если } i \in [16,31] \\ (3i+5) \bmod 16 & \text{если } i \in [32,47] \\ 7i \bmod 16 & \text{если } i \end{cases} \tag{1.2}$$

- K_i — константа такого же размера, разная для каждой операции

$$K_i = \text{floor}(2^{32} \bullet |\sin(i+1)|) \tag{1.3}$$

- $<<<_s$ — сдвиг влево на s бит
  - $s[0,15] = \{7,12,17,22,7,12,17,22,7,12,17,22,7,12,17,22\}$
  - $s[16,31] = \{5,9,14,20,5,9,14,20,5,9,14,20,5,9,14,20\}$
  - $s[32,47] = \{4,11,16,23,4,11,16,23,4,11,16,23,4,11,16,23\}$
  - $s[48,63] = \{6,10,15,21,6,10,15,21,6,10,15,21,6,10,15,21\}$
- $\boxplus$ — сложение по модулю 2^{32}

1.1.2. SHA-1

Длина хэша SHA-1 — 160 бит. Размер блока — 512 бит.

Размер блоков и правила дополнения совпадают с таковыми для MD5.

Вид одной операции представлен на рис. 2

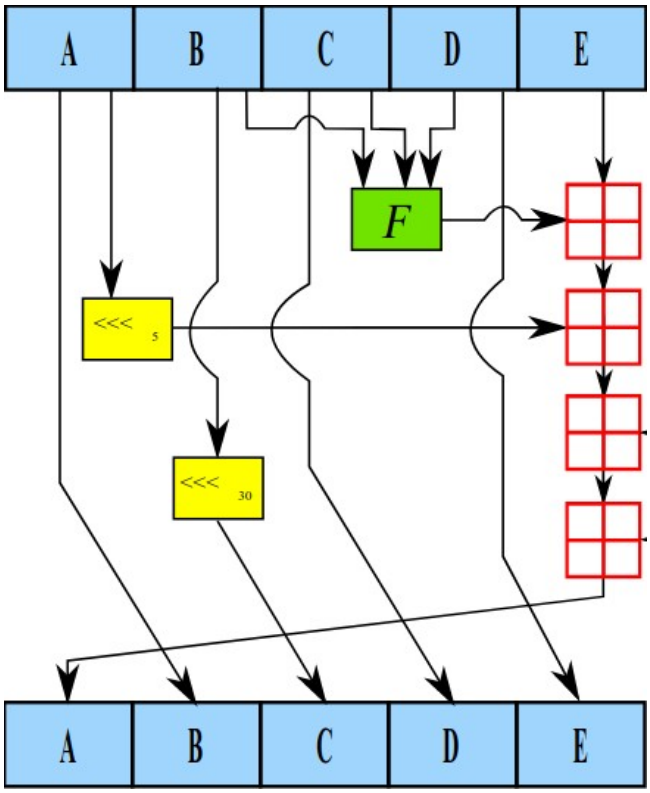

Рисунок 2. Одна итерация SHA-1

- Инициализация:
  - $h0 = \texttt{0x67452301}$
  - $h1 = \texttt{0xefcdab89}$
  - $h2 = \texttt{0x98badcfe}$
  - $h3 = \texttt{0x10325476}$
  - $h4 = \texttt{0xc3d2e1f0}$
- W_i — расширение 16 32-х слов входного блока до 80. Для $i > 15$:

$$W_i = (W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \lll 1 \tag{1.4}$$

- F такая же, как в MD5, но с промежутками по 20 вместо 16
- K_i — константа:

$$K = \begin{cases} \texttt{0x5A827999}, & \text{если } i \in [0, 19] \\ \texttt{0x6ED9EBA1}, & \text{если } i \in [20, 39] \\ \texttt{0x8F1BBCDC}, & \text{если } i \in [40, 59] \\ \texttt{0xCA62C1D6}, & \text{если } i \in [60, 79] \end{cases} \tag{1.5}$$

1.2. Формулировка задания

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

1.3. Ход работы

1. Выбран и модифицирован исходный текст длиной 1686 знаков.

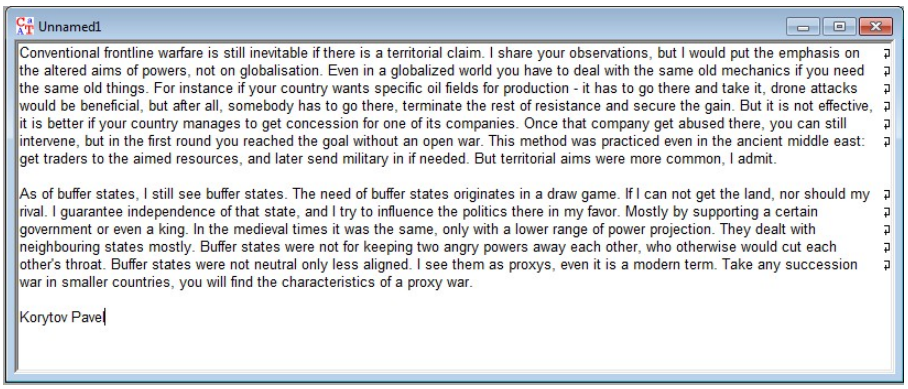

Рисунок 3. Исходный текст

2. С помощью указанной утилиты произведено исследование изменения дайджеста для указанных хэш-функций

	MD5	**SHA-1**	**SHA-256**	**SHA-512**
Добавление	53.91	50.61	51.95	49.22
Удаление	53.13	46.25	54.30	47.46
Изменение	55.47	43.13	48.83	52.93
Среднее	54.17	46.63	51.69	49.87

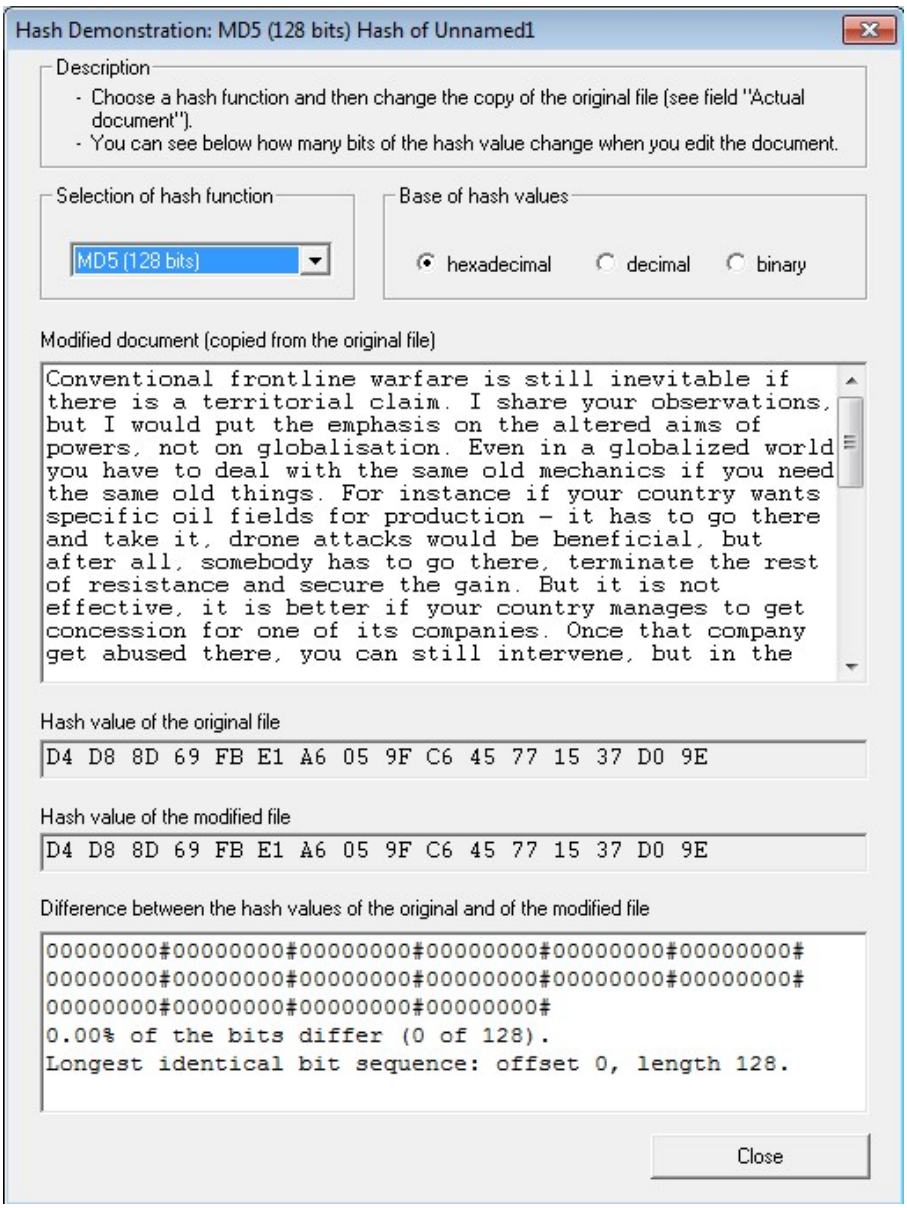

Рисунок 4. Hash Demonstration

2. Хэш-функция SHA-3

2.1. Описание алгоритма

SHA-3 построен на основе криптографической губки. Вид этой структуры представлен на рис. 5

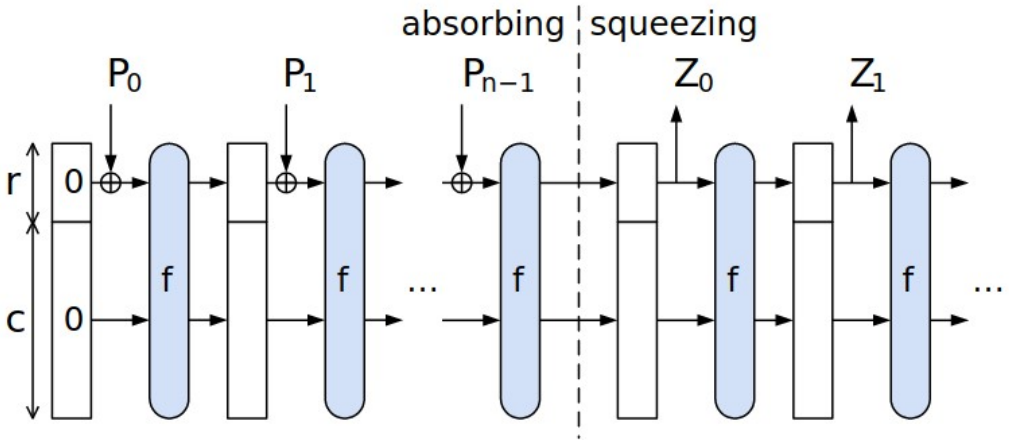

Рисунок 5. Губка

2.1.1. Дополнение

Сообщение должно быть поделено на блоки по r бит. Для этого к сообщению добавляется блок вида $10 \ldots 01$.

Если последний блок имеет длину $r - 1$, то он дополняется единицей, следующий блок состоит из $r - 1$ нулей и 1.

Если длина последнего блока r, то все равно добавляется блок описанного вида.

2.1.2. Устройство губки

На входе:

- P разбивается на блоки $P_0, \ldots, P_{n-1}$ по r бит
- Инициализация S нулевым вектором
- Впитывание (Absorbing):
  - P_i дополняется c нулями до длины блока b
  - Это побитово складывается с S
  - Состояние модифицируется функцией f
- Отжатие (Squeezing)

  На каждом шаге от S сохраняются первые r байт, после чего применяется f. Так повторяется, пока не получится выход Z новой длины

2.2. Формулировка задания

- Открыть шаблон Keccak Hash (SHA-3) в Cryptool 2
- В модуле Keccak сделать следующие настройки:
  - Adjust manually=ON
  - Keccak version= SHA3–512
- Загрузить файл из предыдущего задания

- Запустить проигрывание шаблона в режиме ручного управления:
  - Сохранить скриншоты преобразований первого раунда
  - Сохранить скриншот заключительной фазы
  - Сохранить значение дайджеста
- Вычислить значения дайджеста для модифицированных текстов из предыдущего задания
- Подсчитать лавинный эффект с помощью самостоятельно разработанной автоматизированной процедуры

2.3. Ход работы

1. Открыт и настроен указанный шаблон с указанным файл
2. Проведены преобразования первого раунда

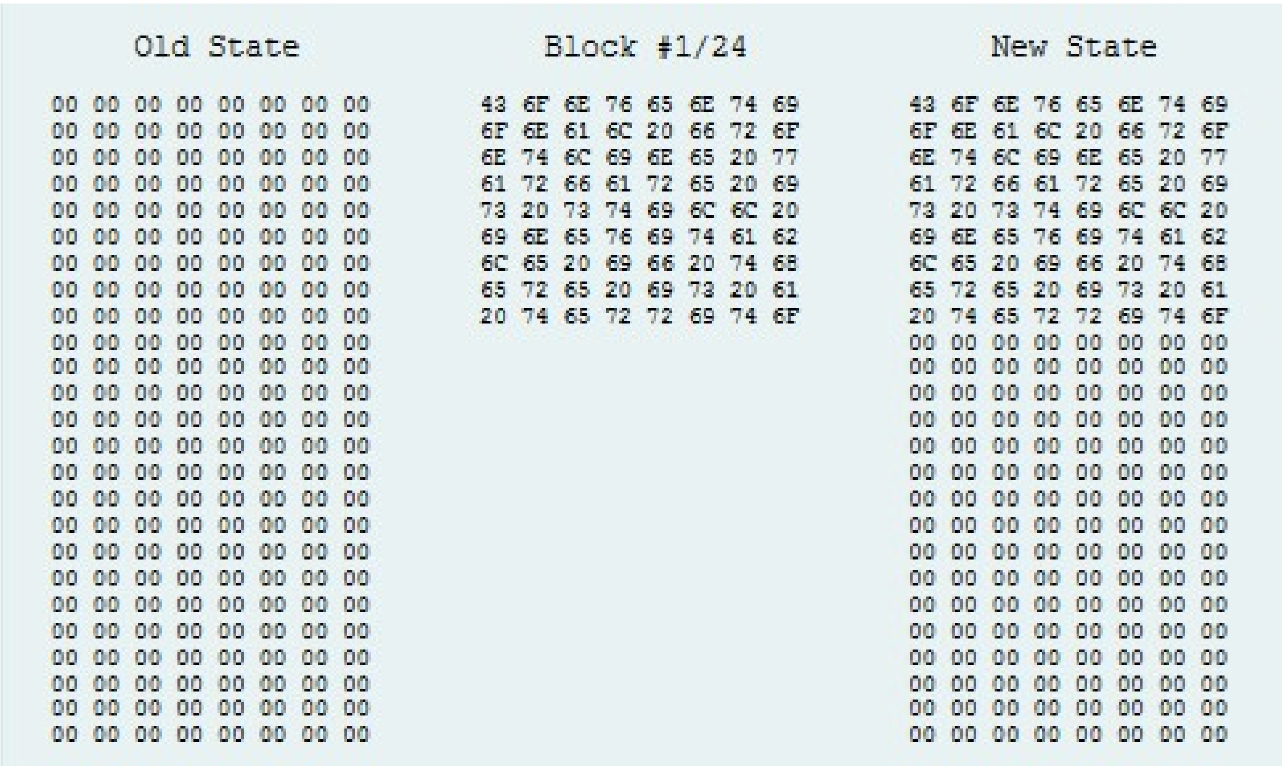

Рисунок 6. Первое подмешивание

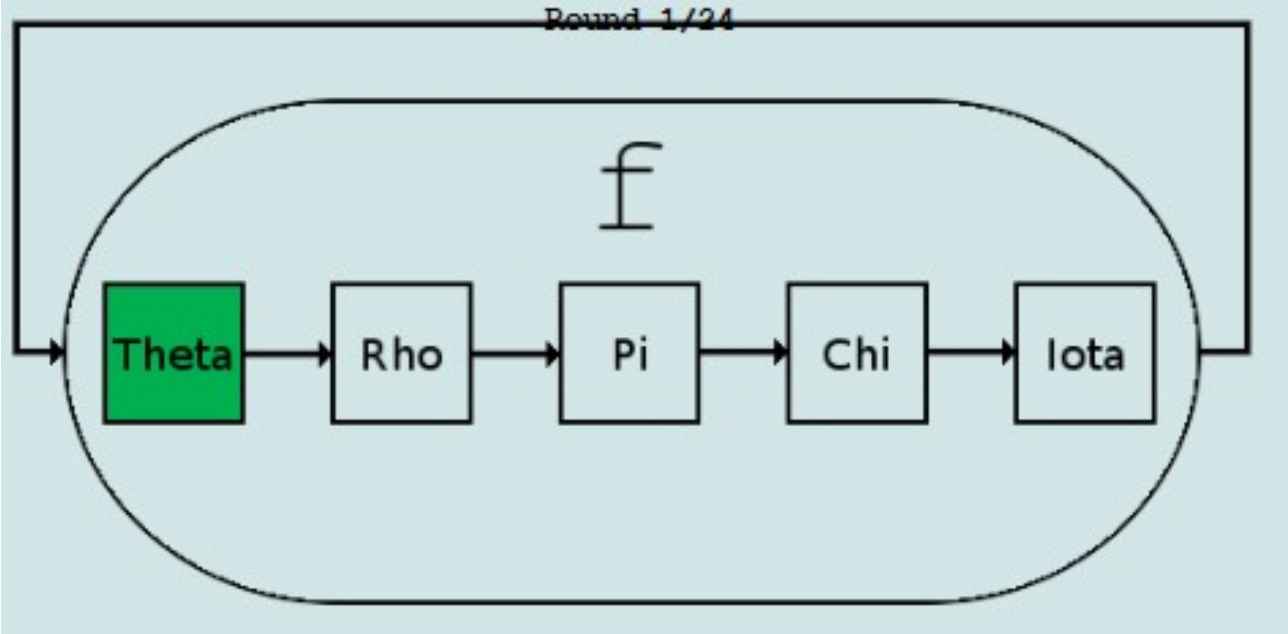

Рисунок 7. Структура f

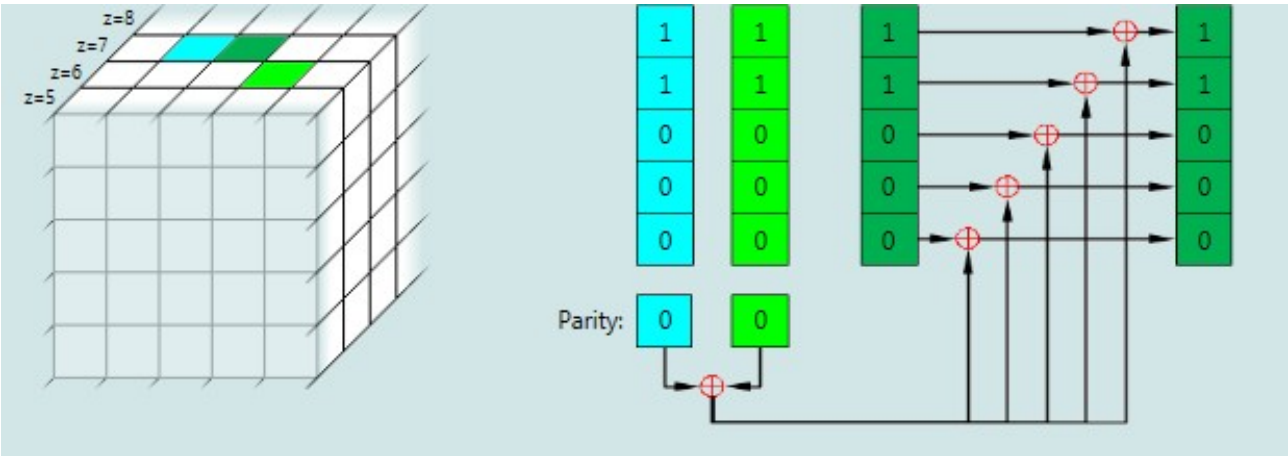

Рисунок 8. θ

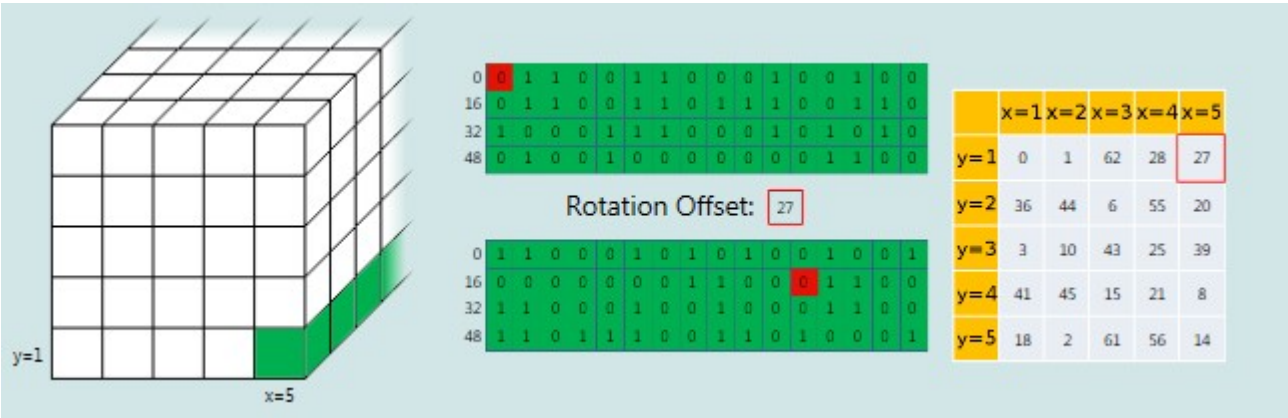

Рисунок 9. ρ

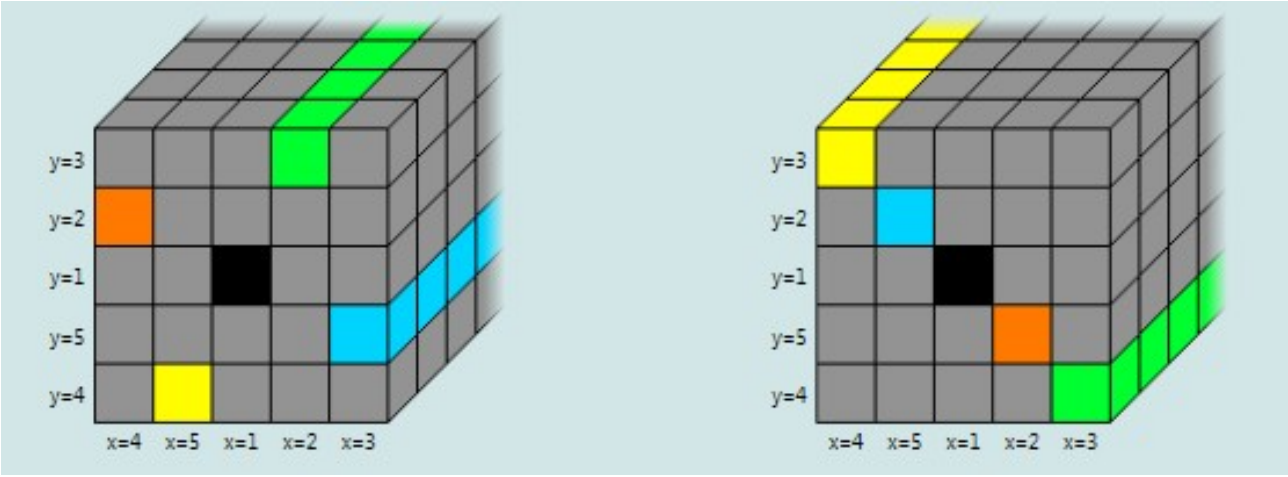

Рисунок 10. π

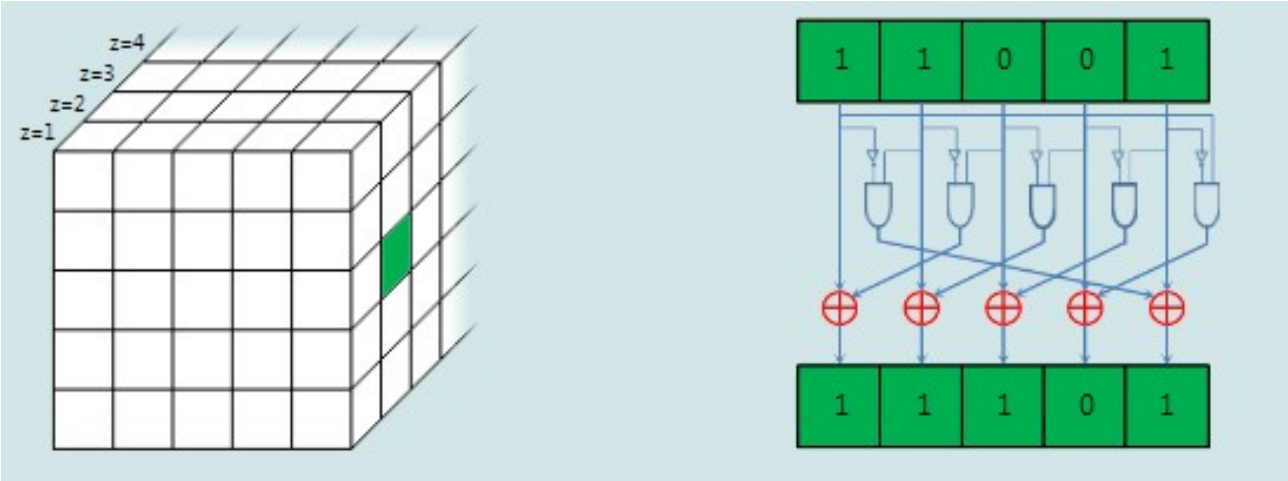

Рисунок 11. χ

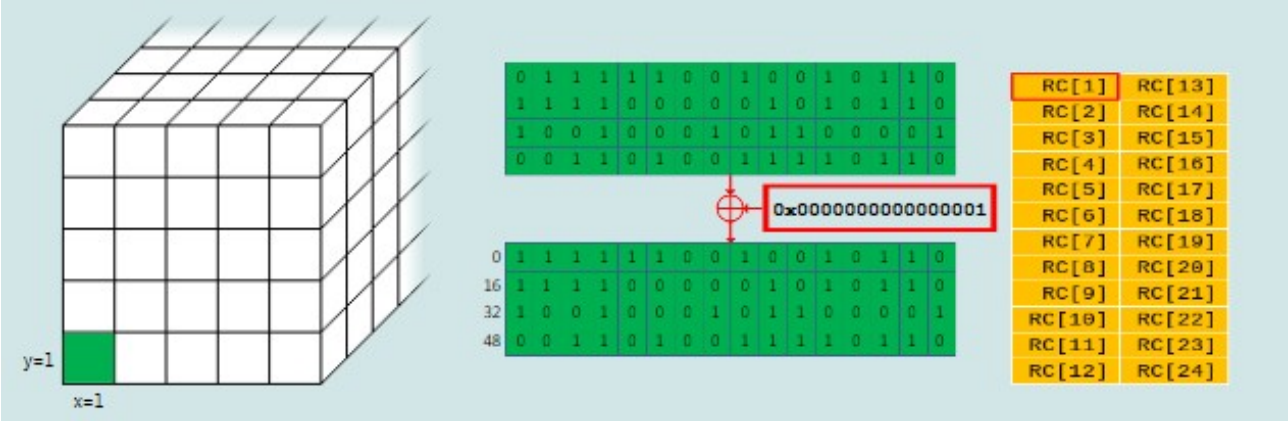

Рисунок 12. ι

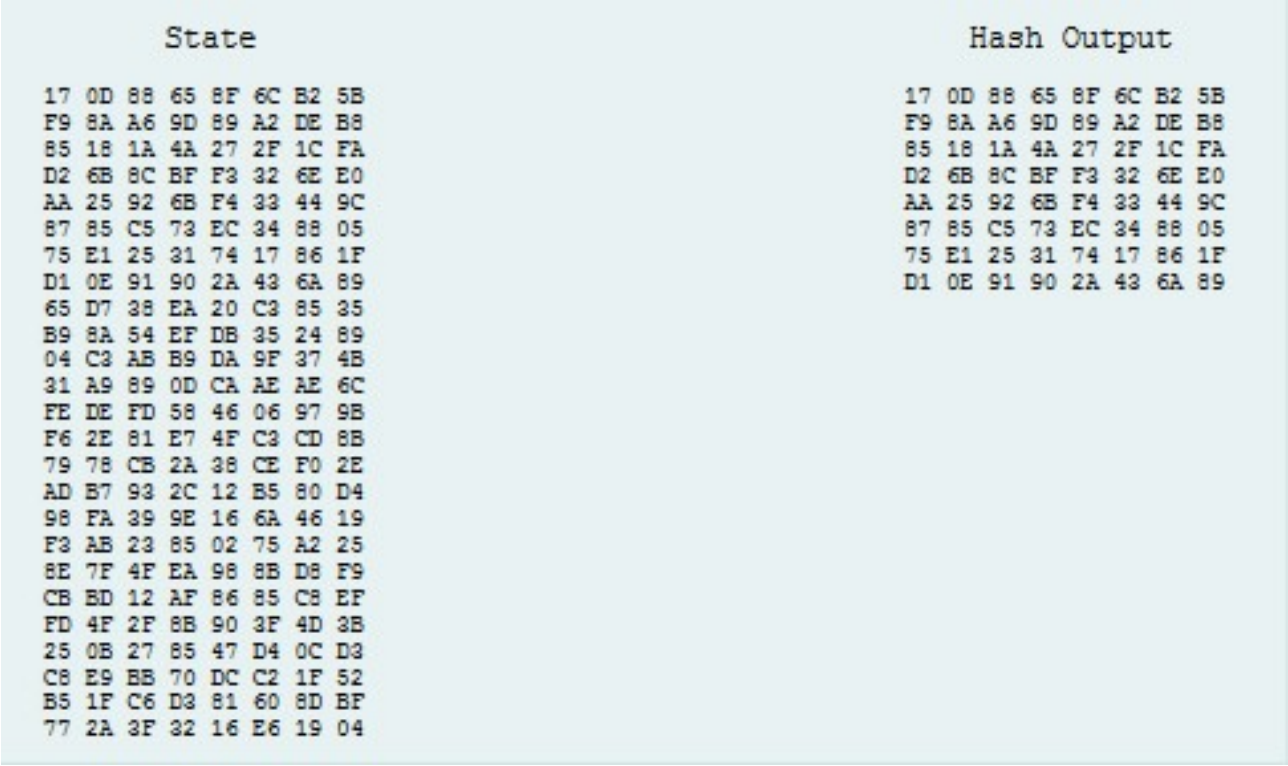

Рисунок 13. Результаты отжатия

3. Создана автоматизированная процедура для оценки лавинного эффекта. Она представлена на рис. 14

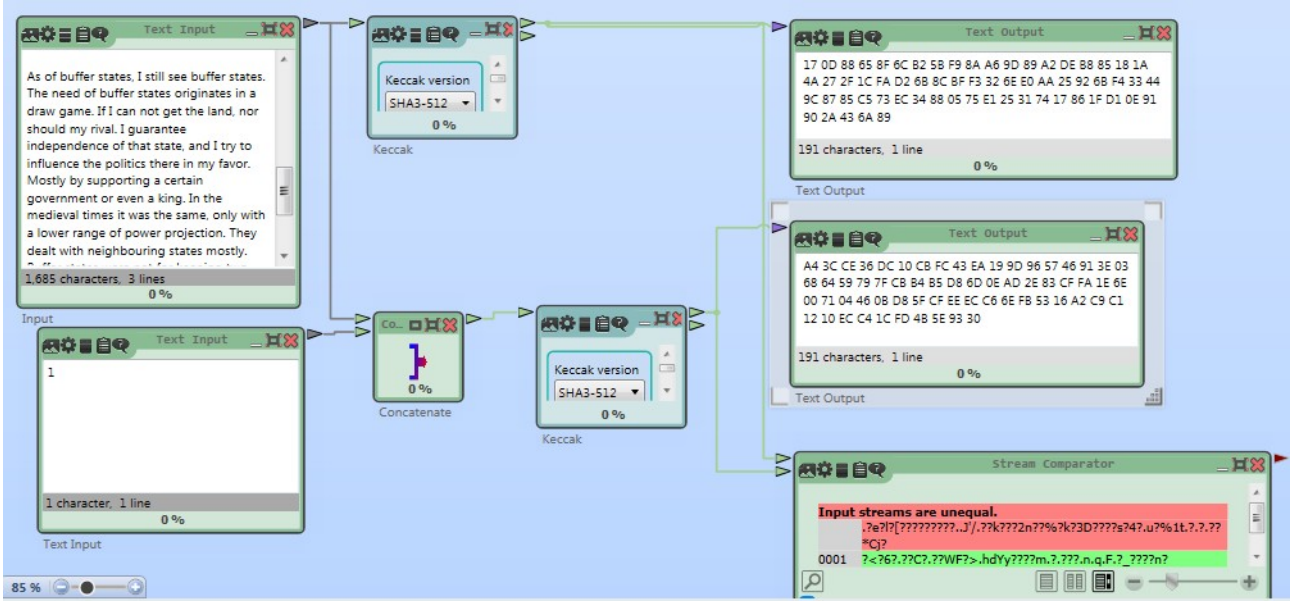

Рисунок 14. Процедура для оценки лавинного эффекта

Результаты:

	SHA-3
Добавление	99.8
Удаление	53.13
Изменение	100
Среднее	99.8

3. Контроль целостности по коду HMAC

3.1. Описание механизма

HMAC позволяет контролировать целостность данных, передаваемых в ненадежной среде. Два клиента разделяют общий секретный ключ.

$$HMAC_k(\texttt{text}) = H((K \oplus \texttt{opad})||H((K \oplus \texttt{ipad})||\texttt{text})), \qquad (3.1)$$

где:

- $||$ — конкатенация;
- K — секретный ключ;
- `ipad` — блок, где байт `0x36` повторяется b раз;
- `opad` — блок, где байт `0x5c` повторяется b раз;
- H — хэш-функция.

3.2. Формулировка задания

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.

3.3. Ход работы

1. Использован тот же самый текст
2. Сгенерирован секретный ключ. Результаты на рис. 15

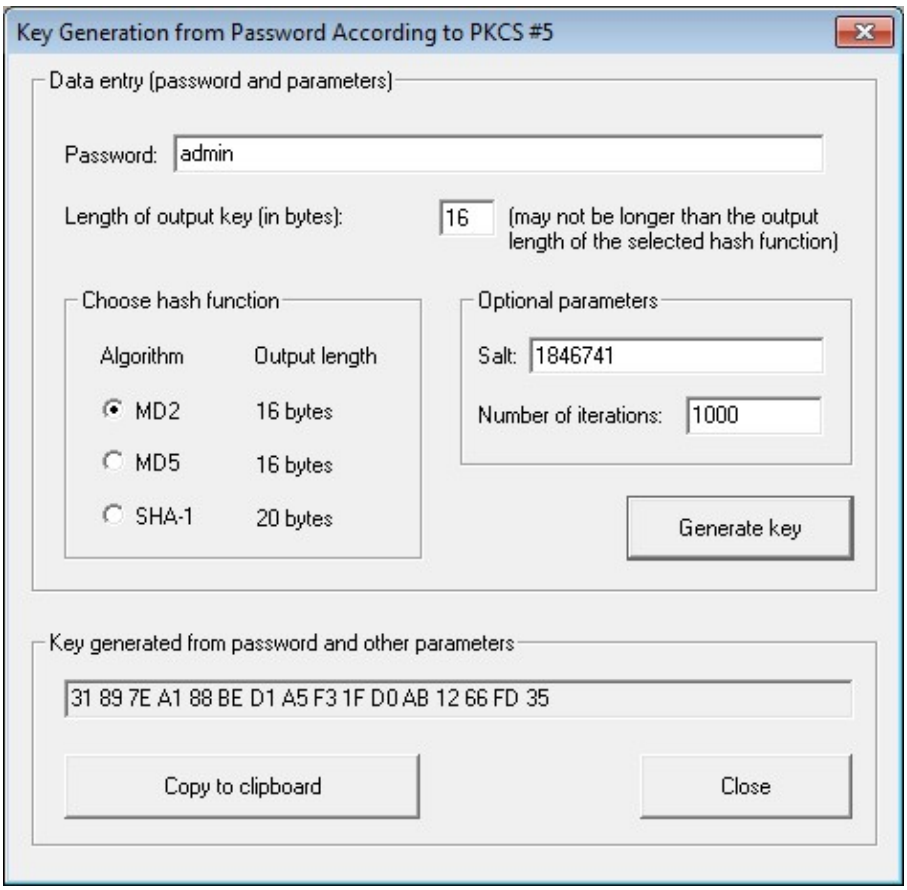

Рисунок 15. Генерация секретного ключа

3. Сгенерирован HMAC для сообщения

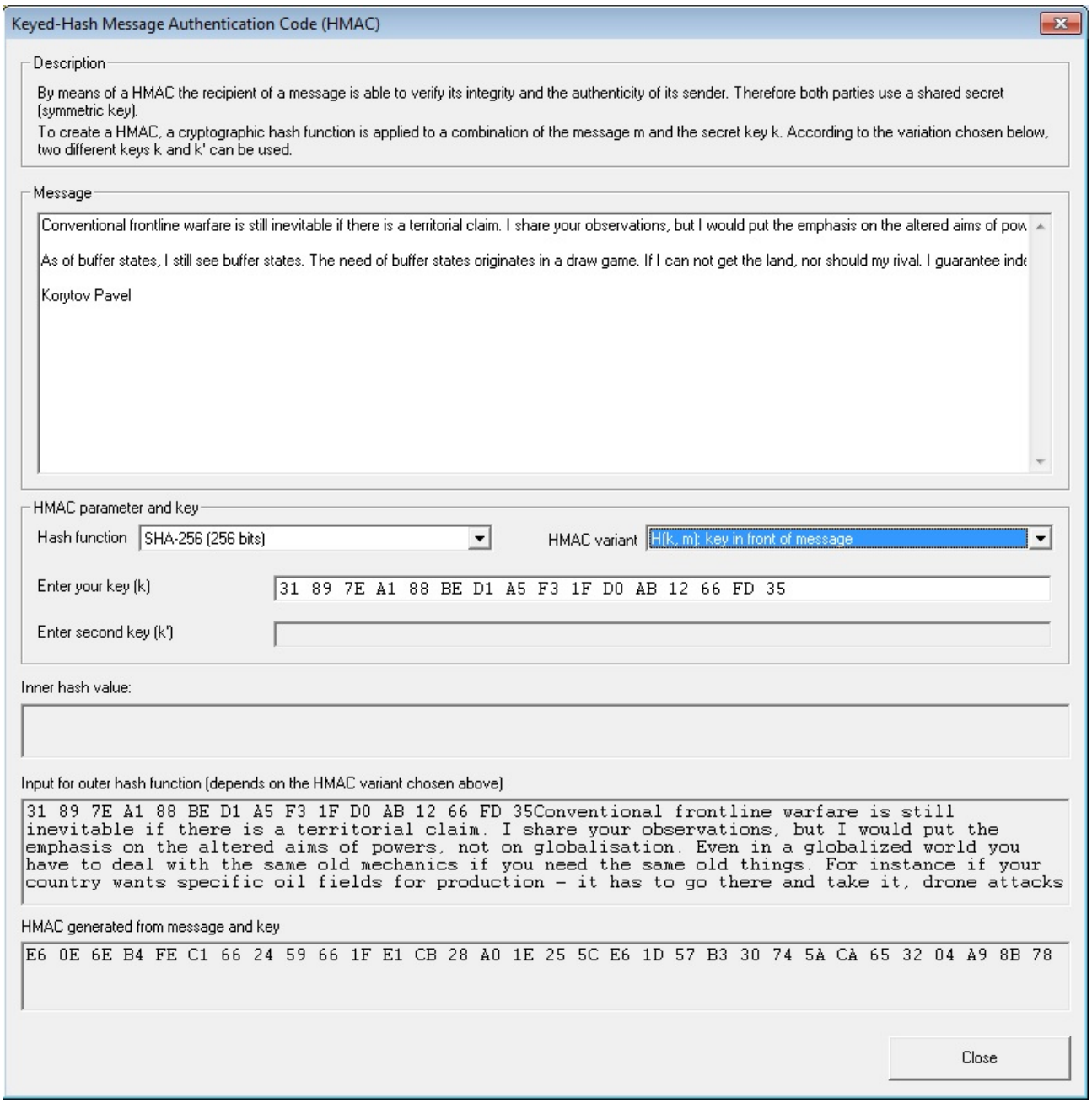

Рисунок 16. Генерация HMAC

4. Атака дополнительной коллизии на хэш-функции

4.1. Описание атаки

Основана на парадоксе дней рождения.

Пусть $P(n)$ — вероятность, что в группе из n человек хотя бы двое имеют одинаковый день рождения. Количество дней в году — N.

$$P(N) = 1 - 1 \bullet \left(1 - \frac{1}{N}\right) \bullet \left(1 - \frac{2}{N}\right) \bullet \ldots \bullet \left(1 - \frac{n-1}{N}\right) = \\ = 1 - \prod_{i=0}^{n-1} \left(1 - \frac{i}{N}\right) \tag{4.1}$$

Поскольку при $x \ll |x|$: $e^x \approx 1 + x$,

$$P(n) \approx 1 - \prod_{i=0}^{n-1} \exp(-i/N) = 1 - \exp \sum_{i=0}^{n-1} (-i/N) = 1 - e^{-n(n-1)/2N} \quad (4.2)$$

Попробуем найти такое n, чтобы $P(n) \geq 0.5$:

$$\frac{1}{2} \geq e^{-\frac{n(n-1)}{2N}} \geq e^{-\frac{n^2}{2N}} \Rightarrow n \geq \sqrt{2 \ln 2 \bullet N} \quad (4.3)$$

Для $N = 365$ такое n — 23.

Пусть h — хэш-функция. Нужно найти $x_1 \neq x_2$, такие, что $h(x_1) = h(x_2)$. Длина хэш-функции — L.

Сложность такой атаки оценивается как $o(2^{L/2})$

4.2. Формулировка задания

- Сформировать два текста на английском языке — один истинный, а другой фальсифицированный. Сохранить тексты в файлах формата *.txt
- Утилитой Analysis-> Attack on the hash value… произвести модификацию сообщений для получения одинакового дайджеста. В качестве метода модификации выбрать Attach characters-> Printable characters.
- Проверить, что дайджесты сообщений действительно совпадают с заданной точностью.
- 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.

4.3. Ход работы

1. Сформированы два текста на английском языке.

   Исходный: `Russia is still trying to expand its sphere of influence and acquire more buffer states. With the exception of the eastern frontier, where nothing can be done because of China, it has steadily supported a number of allied states like Kazakhstan or managed to force the local power to concede its foreign policy.`

   Модифицированный: `Russia is not trying to expand its sphere of influence and acquire more buffer states. With the exception of`

`the western frontier, where nothing can be done because of USA, it has steadily helped a number of peaceful states like Belarus or managed to persuade the local power to follow its foreign policy.`

2. Проведена атака на значение хэша для получения совпадения первых 16 бит

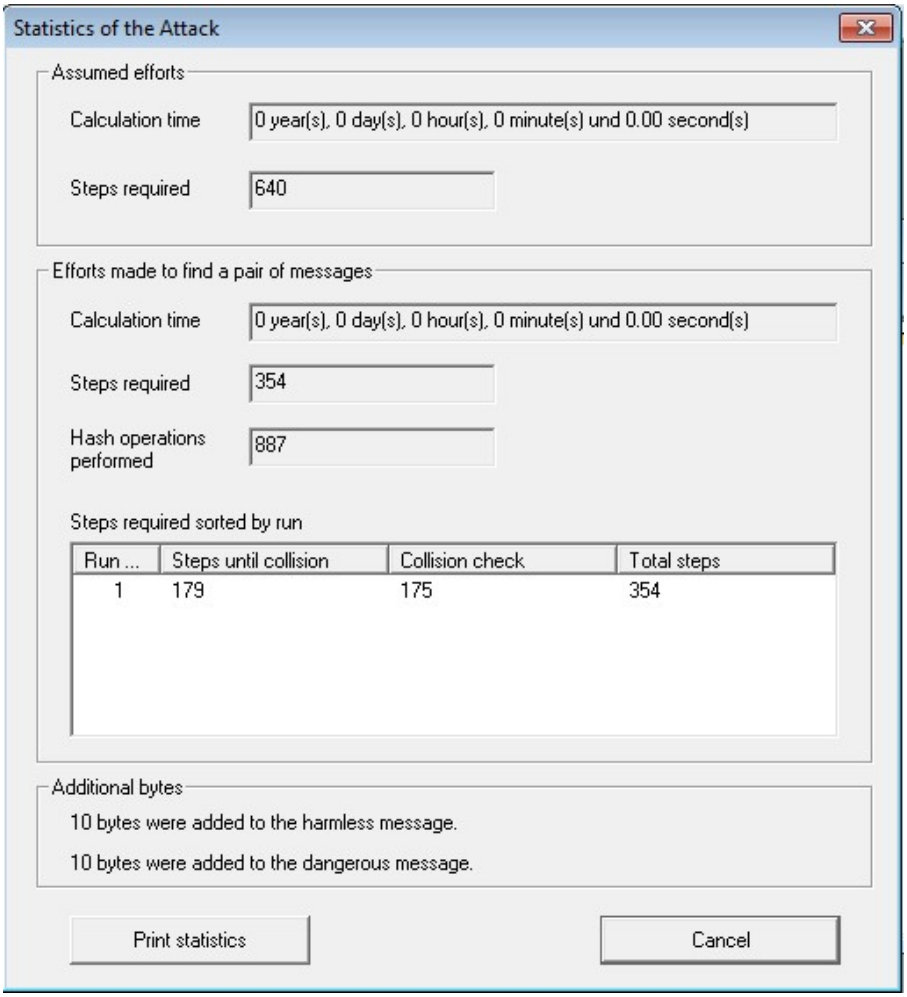

Рисунок 17. Статистика атаки

Значения первых двух байт функции MD5 действительно совпадают для обоих текстов.

3. Оценено время атаки для разного количества совпадающих бит дайджеста

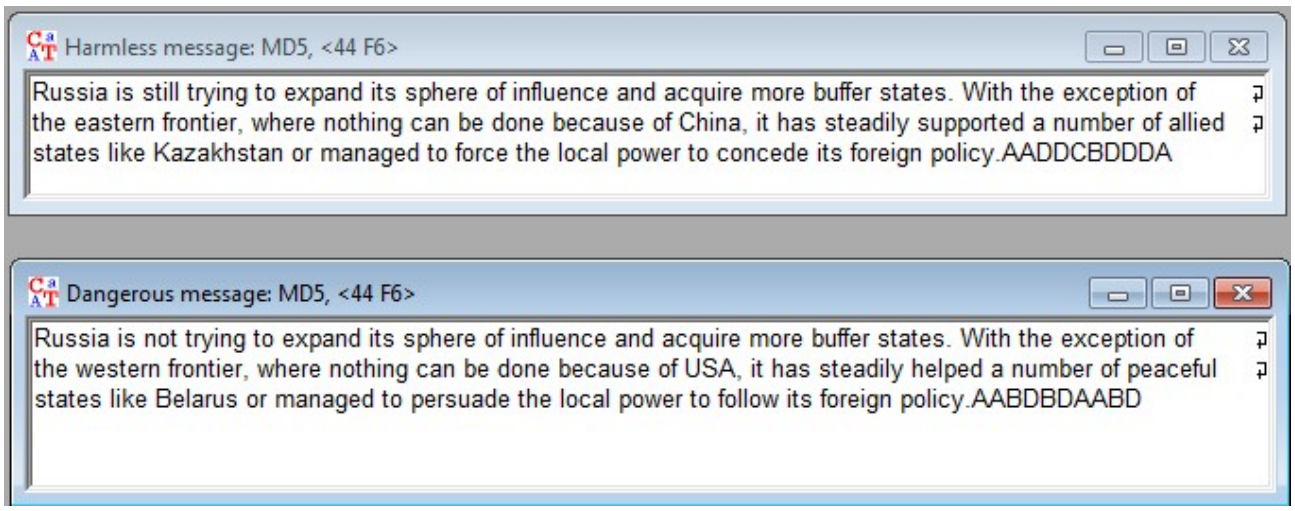

Рисунок 18. Модифицированные тексты

Совпадение дайджеста (бит)	**Время атаки**
8	0 с
16	0 с
24	0 с
32	1 с
40	7 с
48	2 мин
56	24 мин
64	6 часов
72	4.6 дня
80	71.2 дня
88	3.2 года
96	52 года
104	$8.2 \bullet 10^2$ лет
112	$1.3 \bullet 10^4$ лет
120	$2.1 \bullet 10^4$ лет
128	$7 \bullet 10^{93}$ лет

Выводы

Исследовано применение хэш-функций MD5, SHA-1, SHA-256, SHA-512, SHA-3, контроль целостности по коду HMAC и атака дополнительной коллизии.

Хэш-функция	**MD5**	**SHA-1**	**SHA-256**	**SHA-512**	**SHA-3-512**
Длина дайджеста	128	160	256	512	512
Размер блока обработки	128	160	256	512	1600
Размер блока	512	512	512	1024	576
Число итераций	64	80	64	80	24

Лавинный эффект функций MD5, SHA-1, SHA-256, SHA-512 выражается в том, что при изменении одного символа изменяется примерно 50% дайджеста.

Для функции SHA-3 лавинный эффект выражается в почти полном изменении дайджеста. Для размера выхода до 512 бит стойкость SHA-3 неотличима от идеальной хэш-функции.

Механизм HMAC может быть использован для контроля целостности данных при передаче по ненадежному каналу, но для этого клиентам нужен общий секретный ключ, который нужно передать по закрытому каналу.

Для хэш-функции с n-битным значением сложность атаки дополнительной коллизии — поиска двух разных значений с одинаковыми хэш-кодами — примерно равна $2^{n/2}$