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

ОТЧЕТ

по практической работе №3
по дисциплине «Криптографические методы защиты информации»
Тема: «Изучение Хеш-функций»

Студент гр. 9361 _____ Кисляков Н.

Преподаватель _____ Племянников А.К.

Санкт-Петербург
2023

Цель работы

Исследование хеш-функций MD5, SHA-256, SHA-512, SHA-3, кода аутентификации HMAC для контроля целостности и анализ атак дополнительной коллизии на хеш-функцию. Получить практические навыки работы с хеш-функциями и алгоритмом атаки дополнительной коллизии, в том числе с использованием приложения CrypTool 1 и 2.

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

1.1 Задание

1. Открыть текст не менее 1000 знаков. Добавить ваши ФИО последней строкой. Перейти к утилите Indiv. Procedures -> Hash -> Hash Demonstration.

2. Задать хеш-функцию, подлежащую исследованию: MD5, SHA-1, SHA-256, SHA-512.

3. Для каждой хеш-функции повторить следующие действия:

a) изменить (добавлением, заменой, удалением символа) исходный файл;

b) зафиксировать количество измененных битов в дайджесте модифицированного сообщения;

c) вернуть сообщение в исходное состояние.

4. Выполнить процедуру 3 раза (добавлением, заменой, удалением символа) и подсчитать среднее количество измененных бит дайджеста. Зафиксировать результаты в таблице.

1.2 Основные параметры и обобщенная схема хеш-функций MD5, SHA-1

Хэш-функции MD5:

На рисунке 1 представлена обобщённая схема хэш-функции MD5. К основным параметрам хэш-функции MD5 относится:

- Длина сообщения кратна 512 битам, в последний блок добавляются недостающие биты (в виде 1000…000) и 64-битное представление длины исходного сообщения $L_0 = L_0 mod(2_{64})$;

- Длина хэша – 128 бит;
- Число этапов – 4, в каждом этапе 16 раундов, то есть общее число итераций – 64;
- MD-буфер – 4 регистра (A, B, C, D) по 32 бит (в сумме 128 бит)

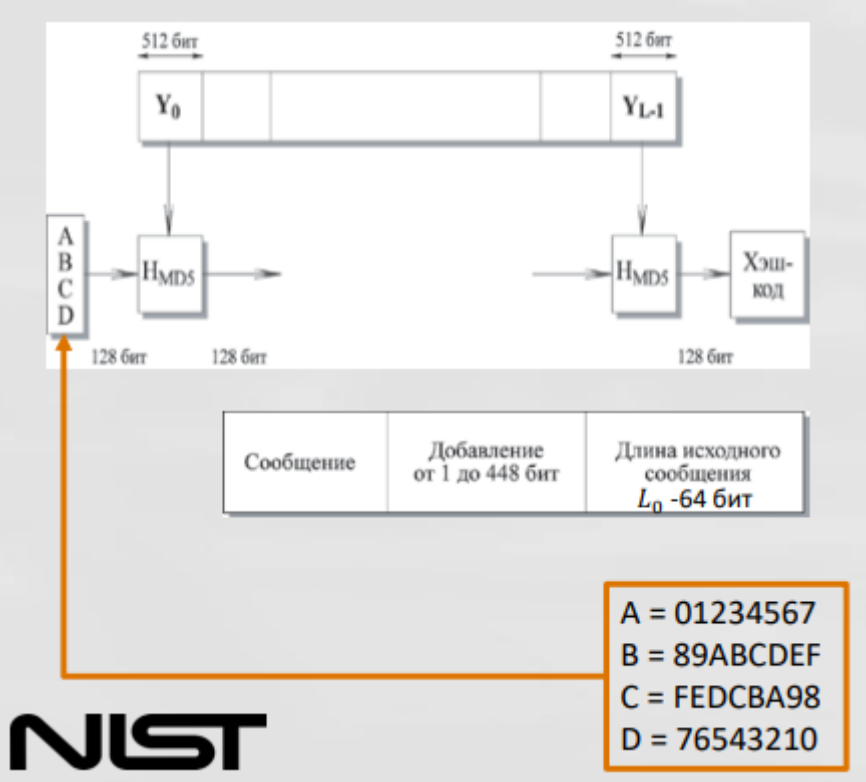

Рисунок 1 – Схема хэш-функции MD5

Функция сжатия H_{MD5}:

На рисунке 2 представлена обобщённая схема функции сжатия H_{MD5}. К основным параметрам функции сжатия H_{MD5} относится:

- Каждый цикл переопределяет значение MD-буфера (A, B, C, D);
- T – массив вычисляемых величин (по 32 бита):

$T_i = int\left(2^{32} \cdot abc\left(sin(i)\right)\right), i = \overline{1,64}$, где $i$nt() – целая часть числа, $abc()$ – абсолютное значение; sin() – синус.

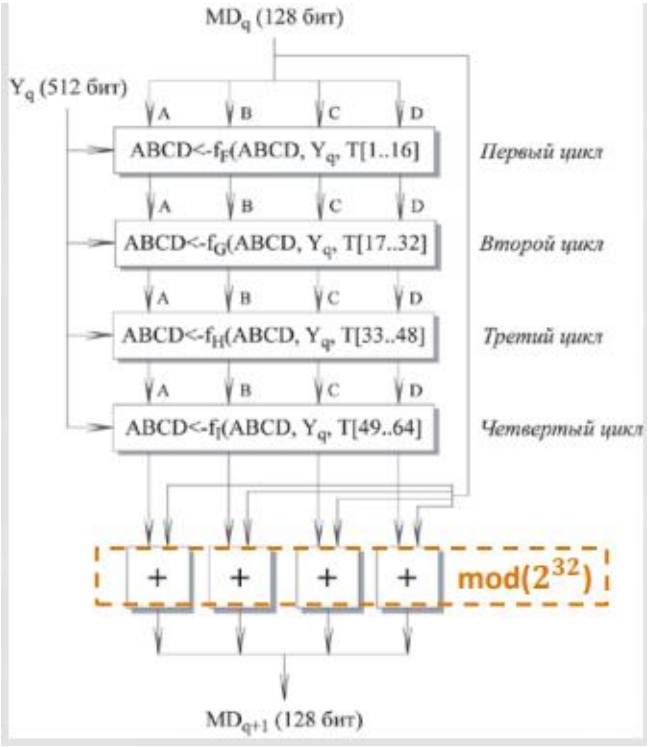

Рисунок 2 – Схема функции сжатия H_{MD5}

Цикл сжатия H_{MD5}:

На рисунке 3 представлена обобщённая схема цикла сжатия H_{MD5}. К основным параметрам цикла сжатия H_{MD5} относится:

Каждый цикл состоит из 6 шагов;

- f – одна из элементарных функций:
  a) $f_F = (B \wedge C) \vee (not\ B \wedge D)$;
  b) $f_G = (B \wedge d) \vee (C \wedge not\ D)$;
  c) $f_H = B \oplus C \oplus D$;
  d) $f_I = C \oplus (B \wedge not\ D)$;
- Сложение выполняется по модулю 2^{32};
- CLS_5 циклический сдвиг влево на 5 разрядов.

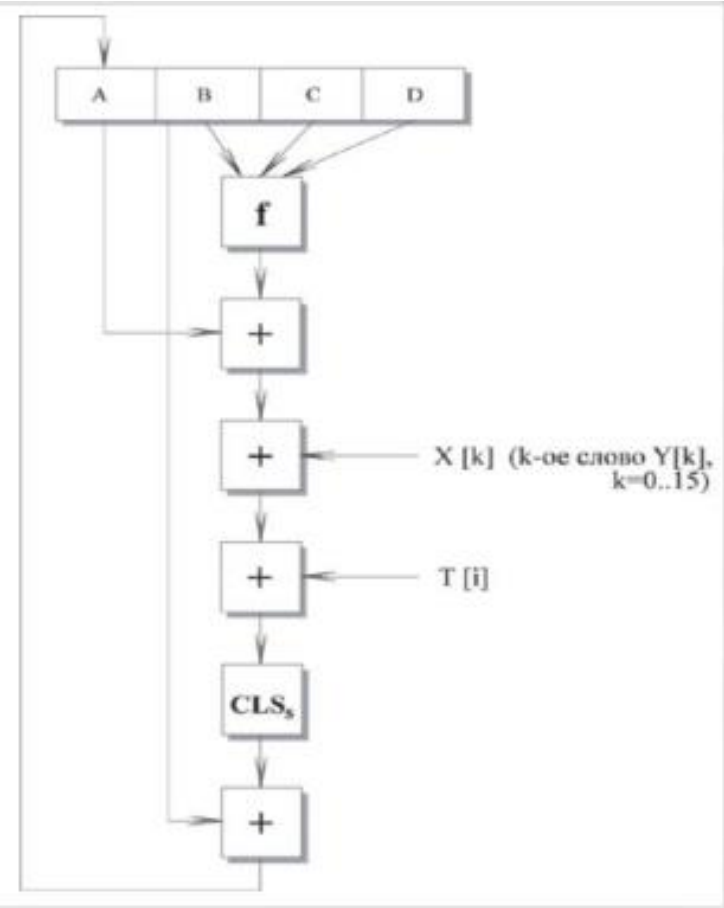

Рисунок 3 – Схема цикла сжатия H_{MD5}

Хэш-функции SHA-1:

На рисунке 4 представлена обобщённая схема хэш-функции SHA-1. К основным параметрам хэш-функции SHA-1 относится:

- Длина сообщения кратна 512 битам, в последний блок добавляются недостающие биты (в виде 1000…000) и 64-битное длина исходного сообщения L_0;
- Длина хэша – 160 бит;
- Число итераций – 80;
- MD-буфер – 5 регистров (A, B, C, D, E) по 32 бит (в сумме 160 бит).

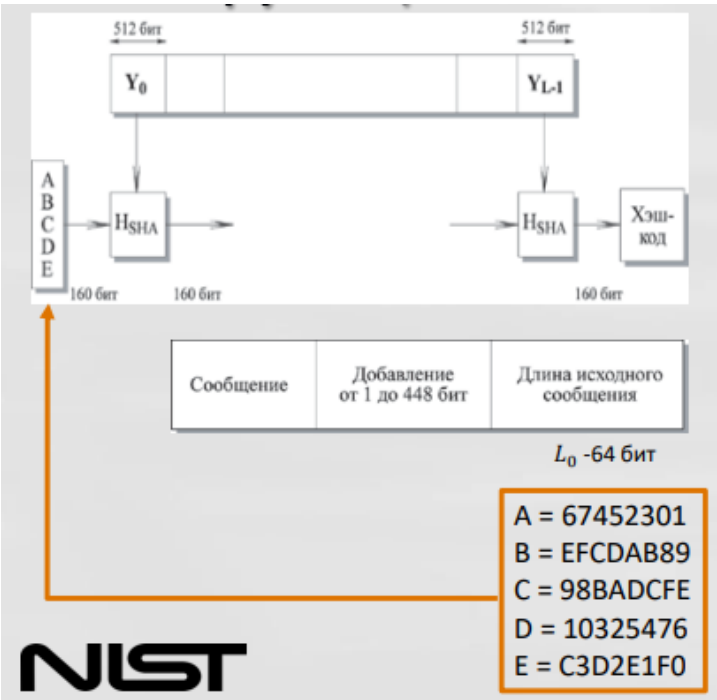

Рисунок 4 – Схема хеш-функции SHA-1

Функция сжатия H_{SHA-1}:

На рисунке 6 представлена обобщённая схема функции сжатия H_{SHA-1}. К основным параметрам функции сжатия H_{SHA-1} относится:

- Состоит из 80 циклов обработок;
- Все 80 циклов обработок имеют одинаковую структуру;
- Каждый цикл переопределяет 160-битное значение MD-буфера;
- В каждом цикле используется дополнительная константа K_t, принимающая 4 различных значения;
- Сложение выполняется по модулю 2^32 .

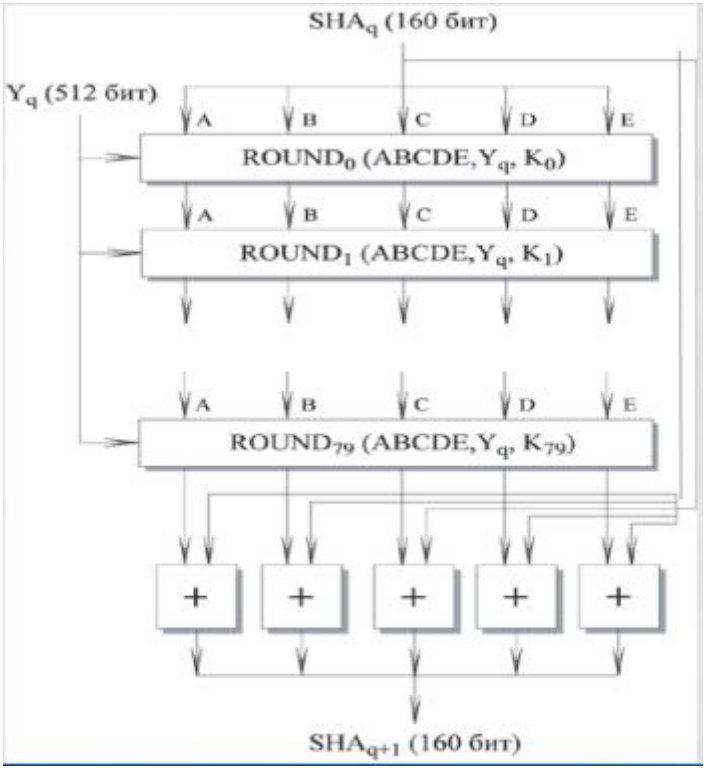

Рисунок 5 – Функция сжатия H_{SHA-1}

Цикл сжатия H_{SHA-1}:

На рисунке 6 представлена обобщённая схема цикла сжатия H_{SHA-1}.

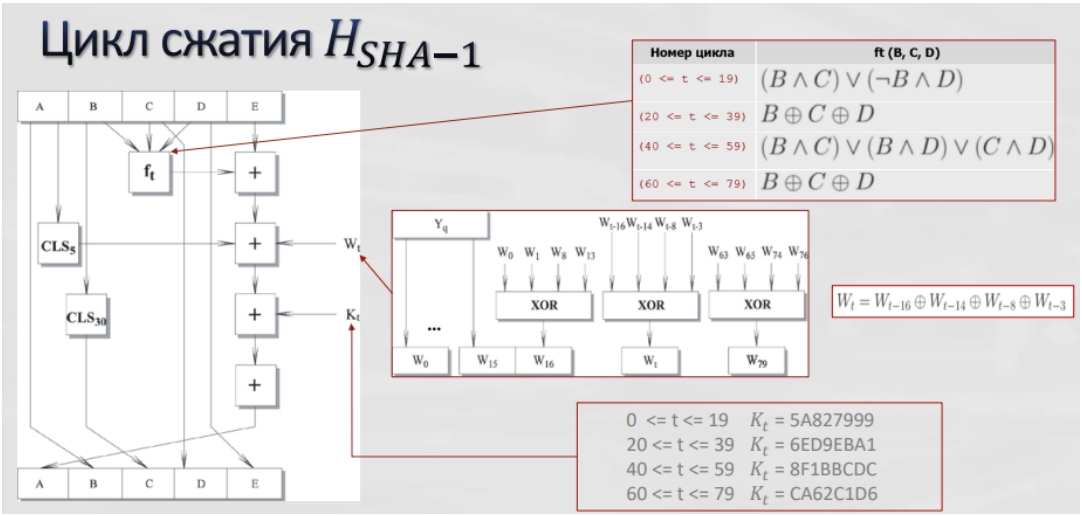

Рисунок 6 – Цикл сжатия H_{SHA-1}

1.3 Таблица с фактическими и усредненными параметрами лавинного эффекта для исследованных хеш-функций

Заранее был подготовлен текст, который состоит из не менее 1000 символов и в конце дописаны ФИО на рисунке 7.

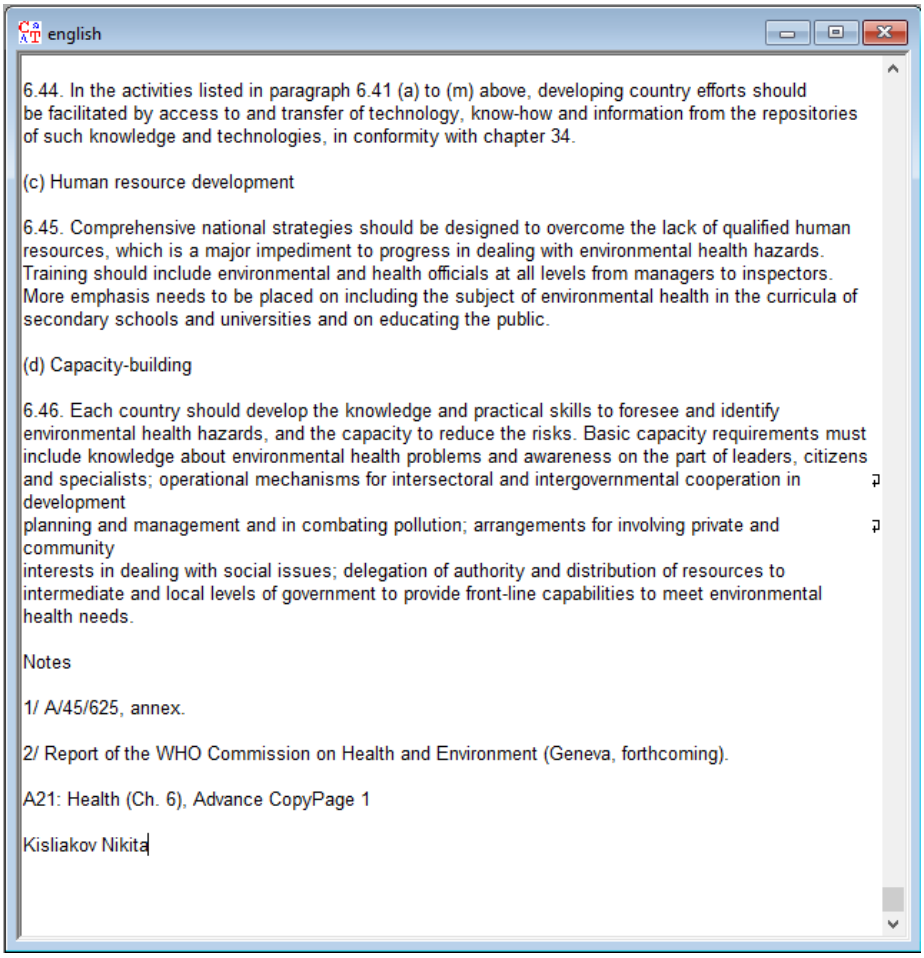

Рисунок 7 – Подготовленный текст

В CrypTool 1, с помощью утилиты Hash Demonstration на выбранном тексте выбирались хэш-функции MD5, SHA-1, SHA-256, SHA-512.

Для каждой хеш-функции применяли добавление, изменение и удаления символа, также фиксировались количество измененных бит хэша и была вычислена среднее количество измененных бит.

Как видно из таблицы 1, при добавлении, изменении или удалении одного символа открытого текста среднее количество измененных бит хэша составляет 51% от общего количества бит хэша.

Таблица 1

Хэш-функция	Изменение	Количество измененных бит хэша	Среднее количество измененных бит
MD5	Добавление	49.22% (63 из 128)	49.22% (63 из 128)
	Изменение	53.91% (69 из 128)	
	Удаление	49.22% (63 из 128)	
SHA-1	Добавление	53.75% (86 из 160)	51.25% (82 из 160)
	Изменение	51.25% (82 из 160)	
	Удаление	47.50% (76 из 160)	
SHA-256	Добавление	51.95% (133 из 256)	50.78% (130 из 256)
	Изменение	50.78% (130 из 256)	
	Удаление	50.39% (129 из 256)	
SHA-512	Добавление	51.76% (265 из 512)	47.85% (245 из 512)
	Изменение	47.46% (243 из 512)	
	Удаление	47.85% (245 из 512)	

2. Хеш-функция SHA-3

2.1 Задание

1. Открыть шаблон Keccak Hash (SHA-3) в CrypTool 2.

2. В модуле Keccak сделать следующие настройки:

а) Adjust manually=ON;

б) Keccak version= SHA3-512.

3. Загрузить файл из предыдущего задания

4. Запустить проигрывание шаблона в режиме ручного управления:

а) сохранить скриншоты преобразований первого раунда;

б) сохранить скриншот заключительной фазы;

в) сохранить значение дайджеста.

5. Вычислить значения дайджеста для модифицированных текстов из предыдущего задания.

6. Подсчитать лавинный эффект с помощью самостоятельно разработанной автоматизированной процедуры.

2.2 Основные параметры, обобщенная схема и перестановки хеш-функции Keccak (SHA-3), на основе изученной презентации

Хэш-функция Keccak:

- В основе Keccak лежит конструкция под названием Sponge (губка);
- Алгоритм состоит из двух этапов:

a) Absorbing (впитывание). На каждом шаге очередной блок сообщения p_i длиной r подмешивается к части внутреннего состояния S, которая затем целиком модифицируется функцией f – многораундовой бесключевой псевдослучайной перестановкой.

b) Squeezing (отжатие). Чтобы получить хэш, функция f многократно применяется к состоянию, и на каждом шаге извлекается и сохраняется кусок размера r до тех пор, пока не получим выход Z необходимой длины (путем конкатенации).

Обобщенная схема хэш-функции Keccak продемонстрирована на рисунке 8/

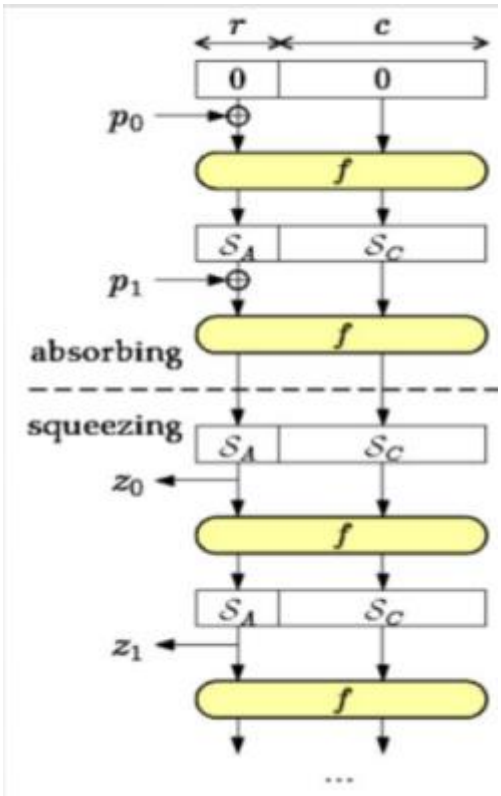

Рисунок 8 – Обобщенная схема хэш-функции Keccak

Внутреннее состояние хэш-функции Keccak:

- Текущее внутренне состояние представлено в виде набора битов, сгруппированных в виде виртуального объекта в трехмерном пространстве (трехмерного массива);
- Объект можно разбить на плоскости или точнее слои вдоль трех осей координат, а элементы каждого слоя – на фрагменты в виде столбцов или строк;
- Трехмерное представление текущего внутреннего состояния помогает применить набор простейших логических операций (XOR, AND, NOT) оптимальным образом для реализации псевдослучайных перестановок.

Внутреннее состояние хэш-функции Keccak показано на рисунке 9.

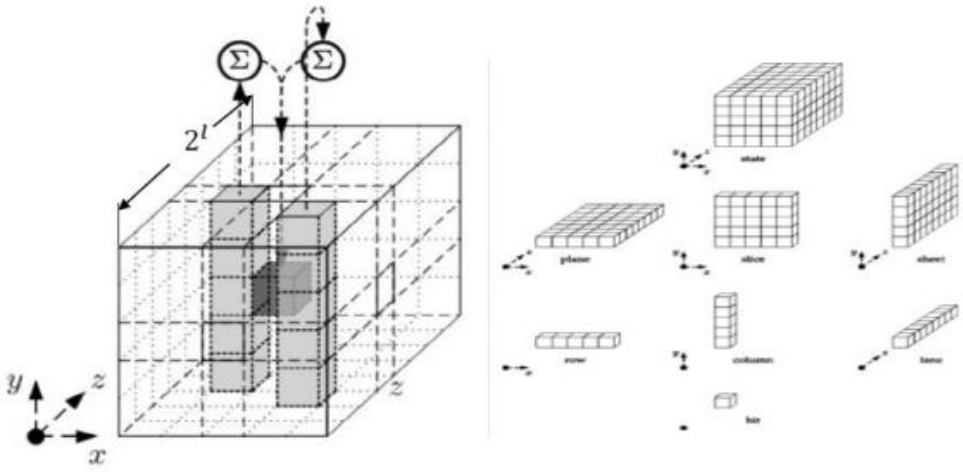

Рисунок 9 – Внутреннее состояние хэш-функции Keccak

Бесключевая псевдослучайная перестановка f хэш-функции Keccak:

- Раундовые функции выполняют 5-ти шаговую обработку внутреннего состояния;
- Количество раундов $12 + 2 * l$ для состояния размера 25×2^l бит ($0 \leq l \leq 6$).

Бесключевая псевдослучайная перестановка f хэш-функции Keccak представлена на рисунке 10.

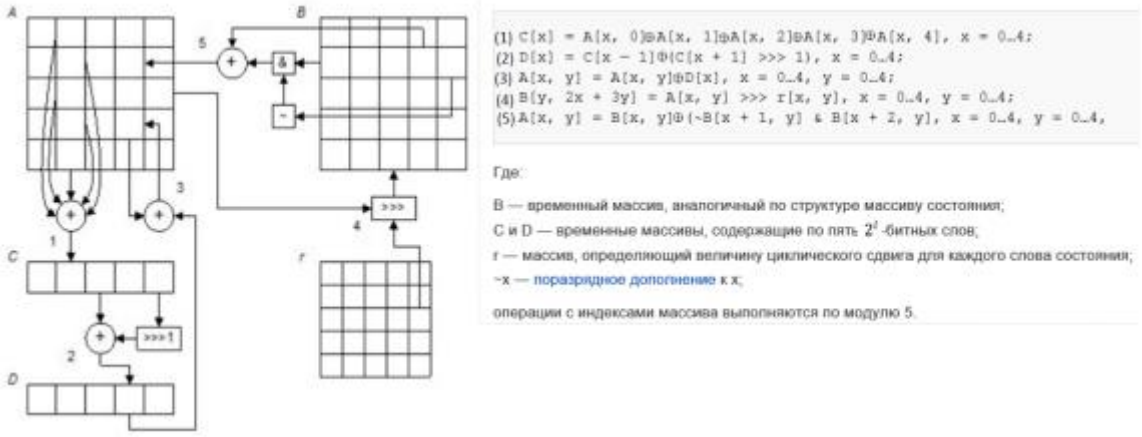

Рисунок 10 – Бесключевая псевдослучайная перестановка f

Оценка криптостойкости:

- Авторы Keccak доказали, что стойкость такой конструкции неотличима от стойкости идеальной хэш-функции (random oracle model) с размером дайджеста, равным c/2 (все состояние имеет размер $c + r$) при условии, что f – идеальная функция перестановки (без повторений внутреннего состояния);
- Сложность проведения атак гарантируется только до необходимой величины. После увеличения длины выхода хэш-функции за пределы расчетного, стойкость функции перестает зависеть от размера выхода.

Масштабируемость:

- Хэш-функция Keccak реализована таким образом, что функцию перестановки f пользователь может выбирать самостоятельно из набора предопределенных функций $\{f - 25, f - 50, f - 100, f - 200, f - 400, f - 800, f - 1600\}$;
- Каждая функция из набора обрабатывает внутренне состояние определенного размера $|S| = 25 \times 2^l, 0 \leq l \leq 6$;
- Для того, чтобы в реализации использовалась функция f обрабатывающая состояния размера $|S|$, необходимо, чтобы $r + c = |S|$;
- Количество раундов nn применения функции f вычисляется как $n = 12 + 2 * l, l = \log_2 |S/25|$.

2.3 Описание средства оценивания лавинного эффекта

Сначала откроем шаблон Keccak Hash (SHA-3) в CrypTool 2 с настройками Adjust manually=ON и Keccak version=SHA-512. В качестве исходного текста возьмем текст из пункта 1.3. Запустим проигрывание шаблона в режиме ручного управления. Скриншоты преобразований первого раунда продемонстрированы на рисунках 11, 12, 13, 14, 15, 16, 17. На рисунке 18 представлена заключительная фаза.

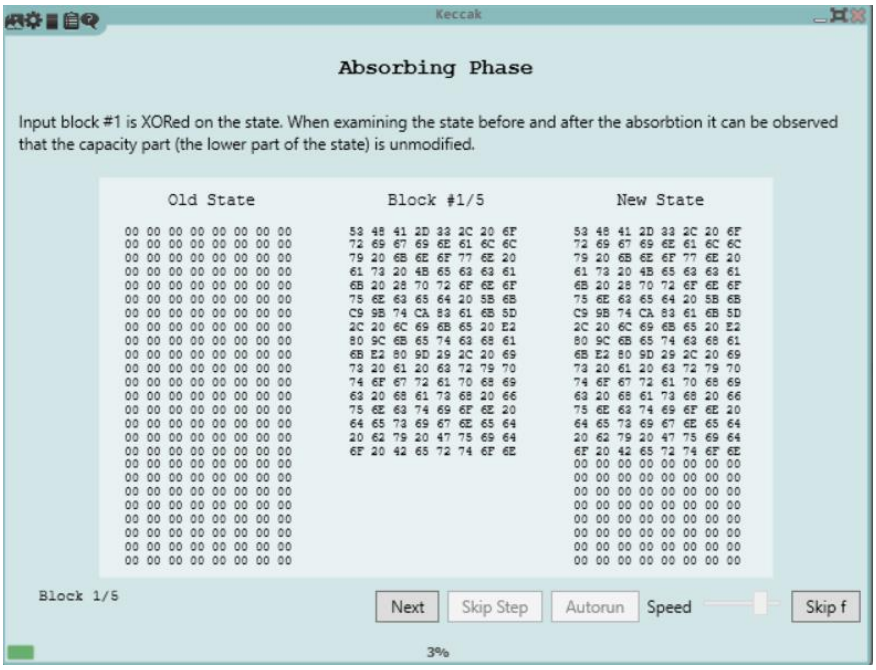

Рисунок 11 – Фаза впитывания

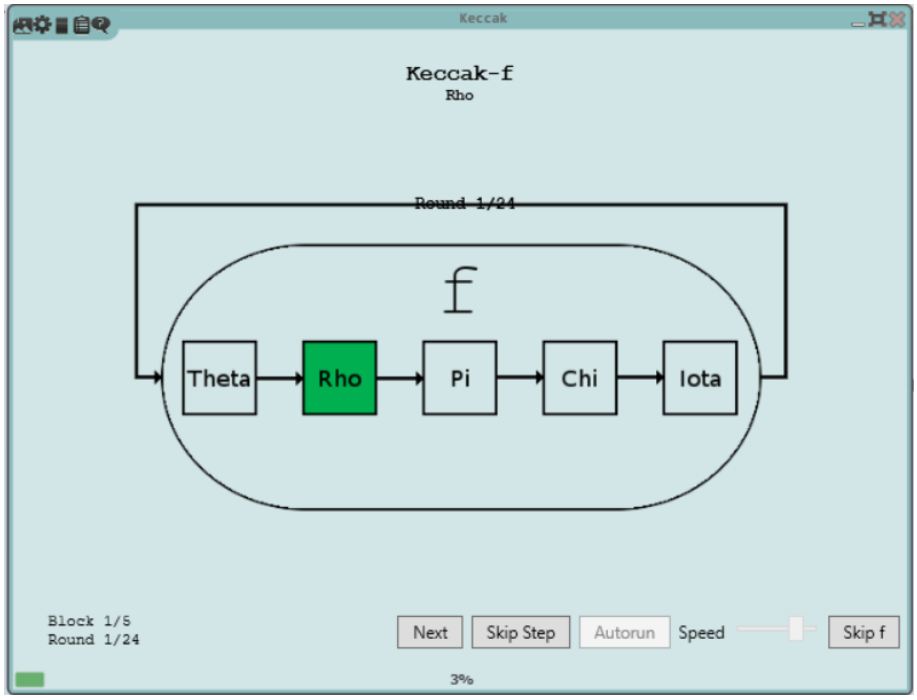

Рисунок 12 – Функция перестановки f

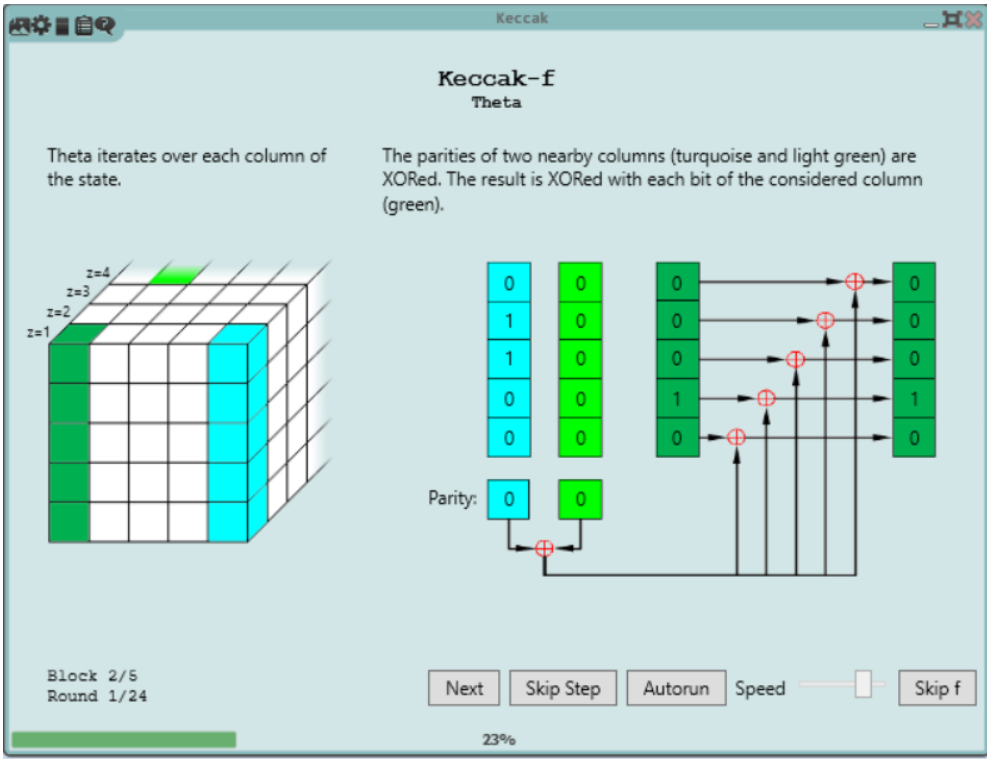

Рисунок 13 – Шаг «Theta» первого раунда функции перестановки

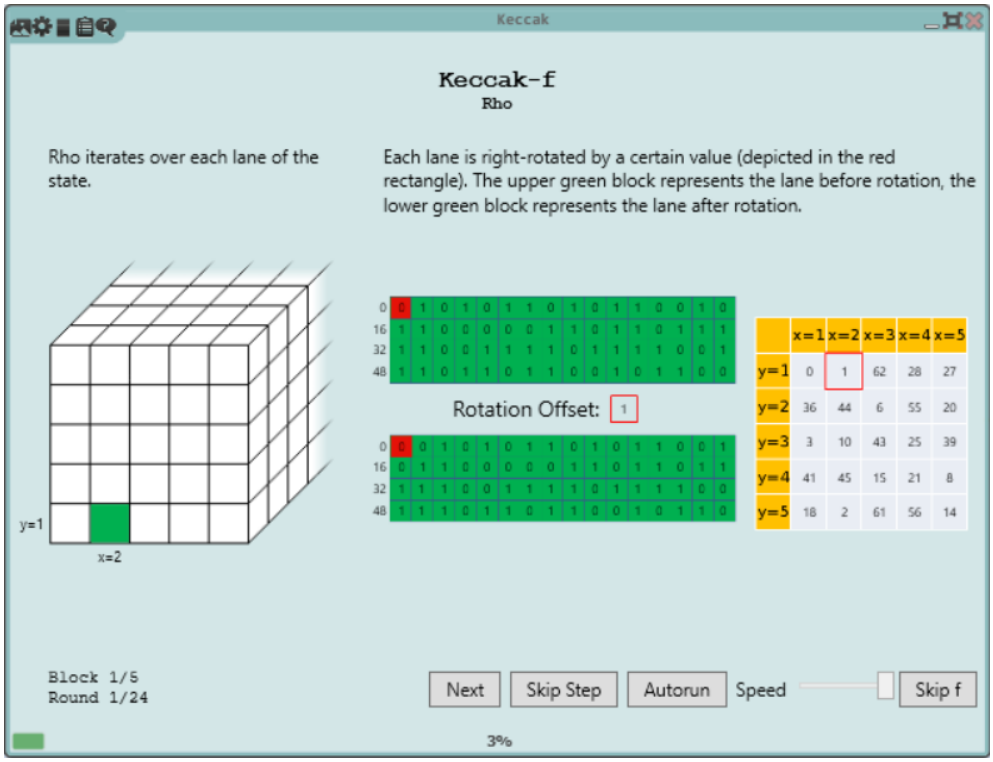

Рисунок 14 – Шаг «Rho» первого раунда функции перестановки

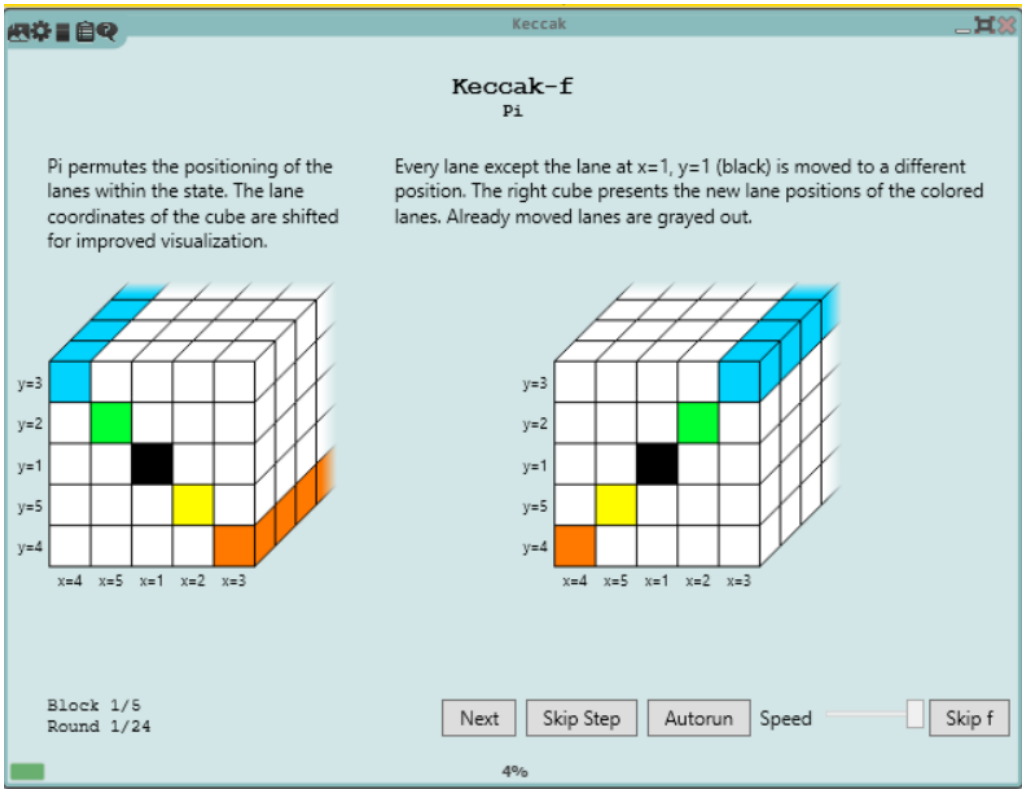

Рисунок 15 – Шаг «Pi» первого раунда функции перестановки

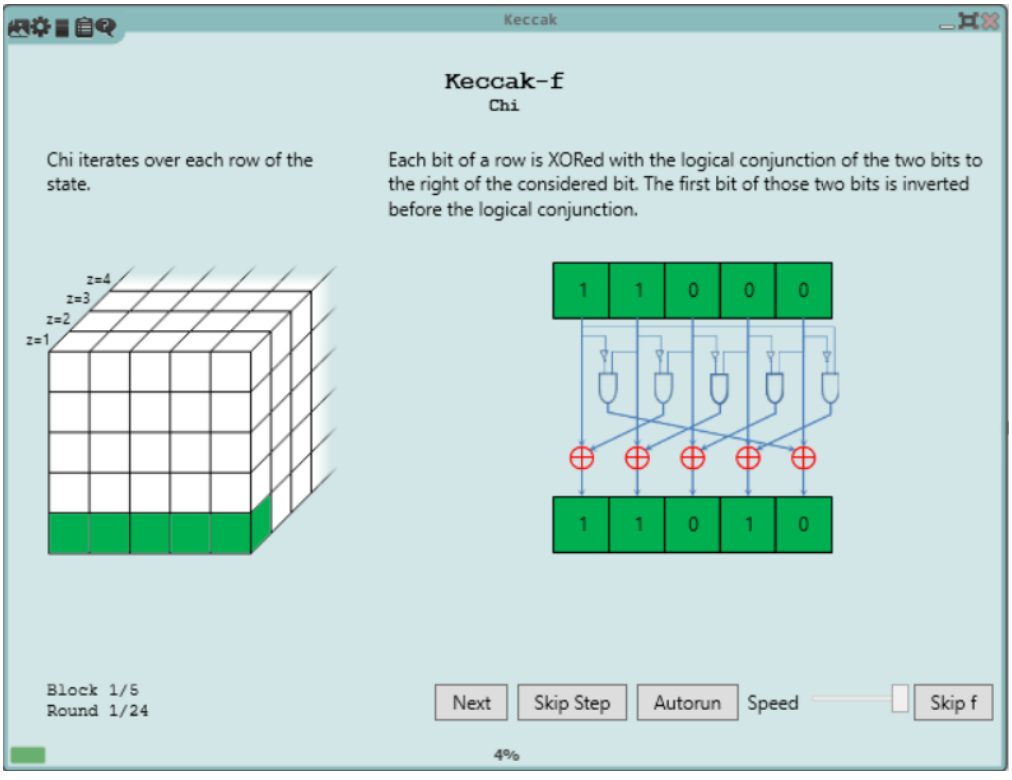

Рисунок 16 – Шаг «Chi» первого раунда функции перестановки

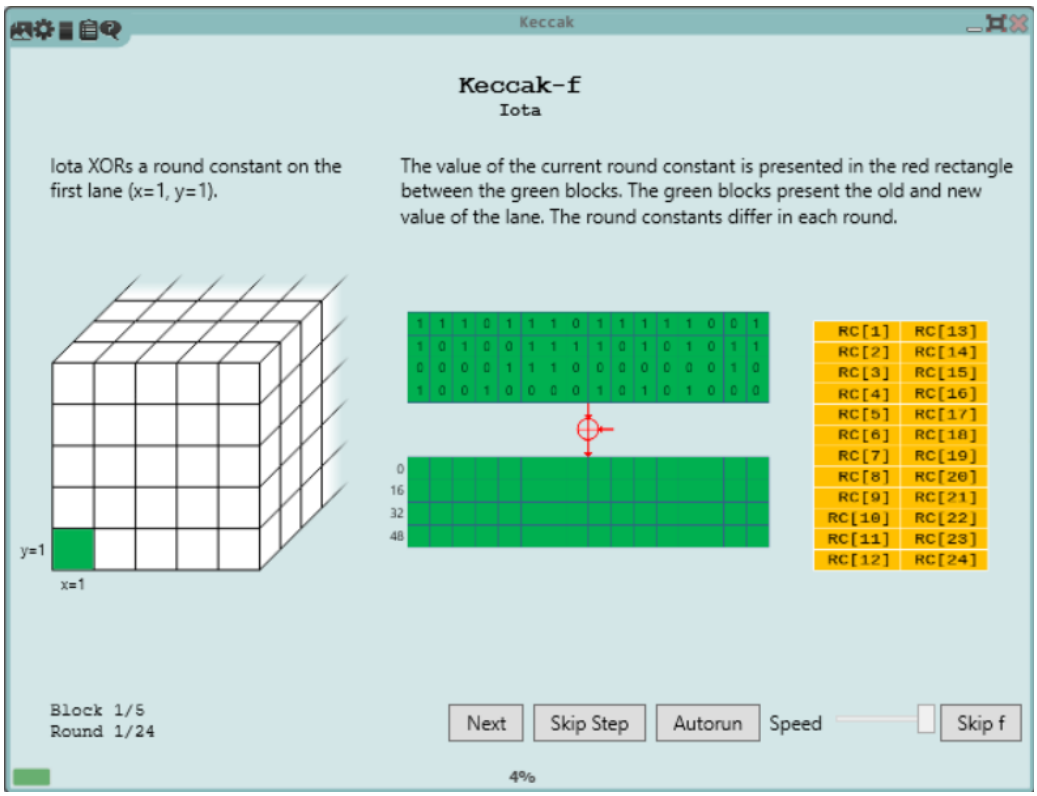

Рисунок 17 – Шаг «Iota» первого раунда функции перестановки

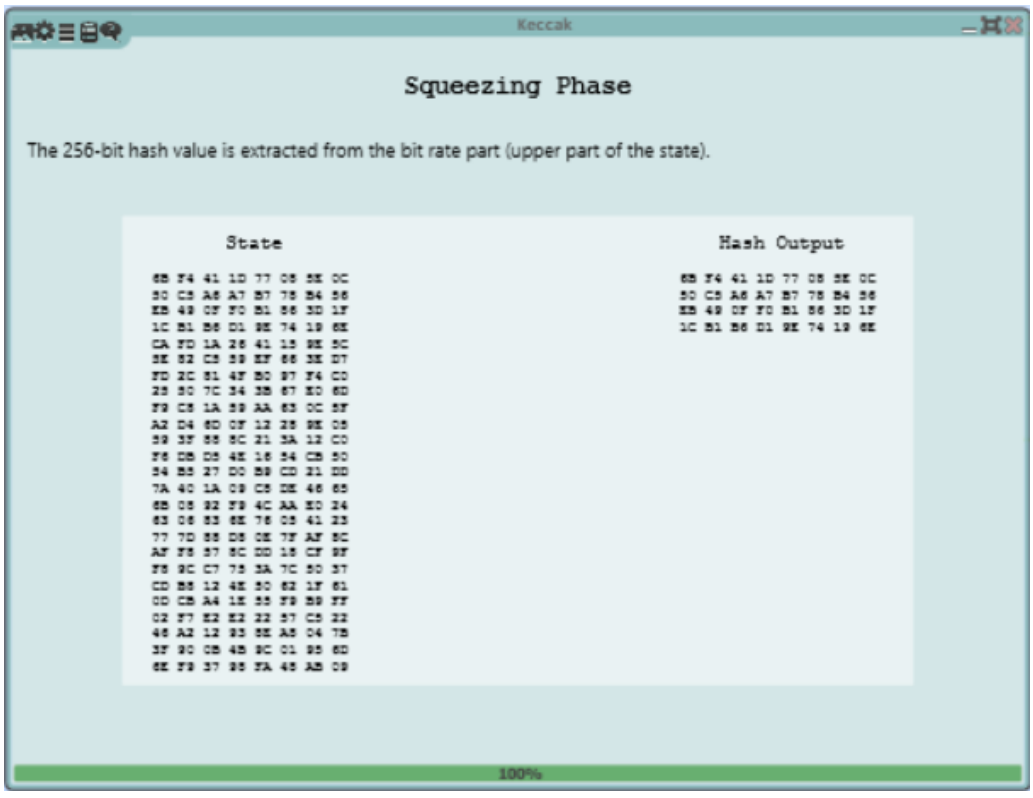

Рисунок 18 – Фаза «отжатия»

Полученное значение хэша: « 6B F4 41 1D 77 08 5E 0C 50 C5 A6 A7 B7 78 B4 56 EB 49 0F F0 B1 86 3D 1F 1C B1 B6 D1 9E 74 19 6E».

2.4 Таблица с фактическими параметрами лавинного эффекта

После для исходного текста были выполнены те же модификации, которые использовались в пункте 1.3 для исследования лавинного эффекта 17 хэш-функций MD5, SHA-1, SHA-256, SHA-512. Для модифицированных текстов были найдены дайджесты. Результаты исследований хэш-функции SHA-3 были записаны в таблицу 2.

Таблица 2

Модификация	Дайджест
Без изменения	6B F4 41 1D 77 08 5E 0C 50 C5 A6 A7 B7 78 B4 56 EB 49 0F F0 B1 86 3D 1F 1C B1 B6 D1 9E 74 19 6E
Добавление	43 F1 C0 F2 3E CF AC 65 16 22 31 F5 6D 4F 12 19 2E 6E 1C 8A 7F 17 72 A5 9E BD 6F D8 75 DD 4A 31
Изменение	C5 12 A0 F2 16 E1 4B D1 D8 E5 D1 1E EA B4 3A 2D C7 28 0C 8B AC C5 63 6B 36 01 DC D1 17 22 69 D6
Удаление	F6 4E A9 F9 46 39 6C 8C 1E DE B6 53 58 16 88 F5 37 B0 73 AD 32 69 C7 1E 70 1B 96 04 72 99 26 90

Теперь найдем количество измененных бит в дайджестах при добавлении, изменении и удалении одного символа исходного текста, а также вычислим среднее количество измененных бит хэша при помощи шаблона «Avalanche (Hash Functions)». Найденное количество измененных бит представлено на рисунках 19, 20, 21.

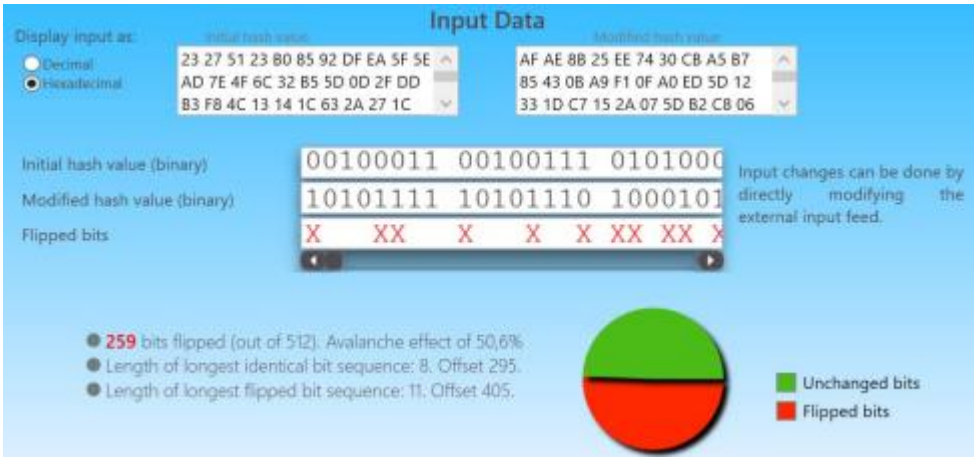

Рисунок 19 – Добавление

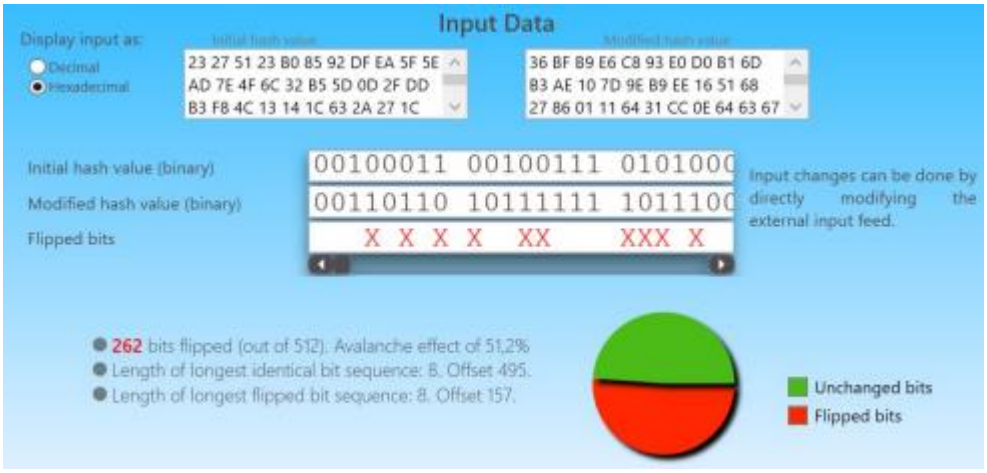

Рисунок 20 – Изменение

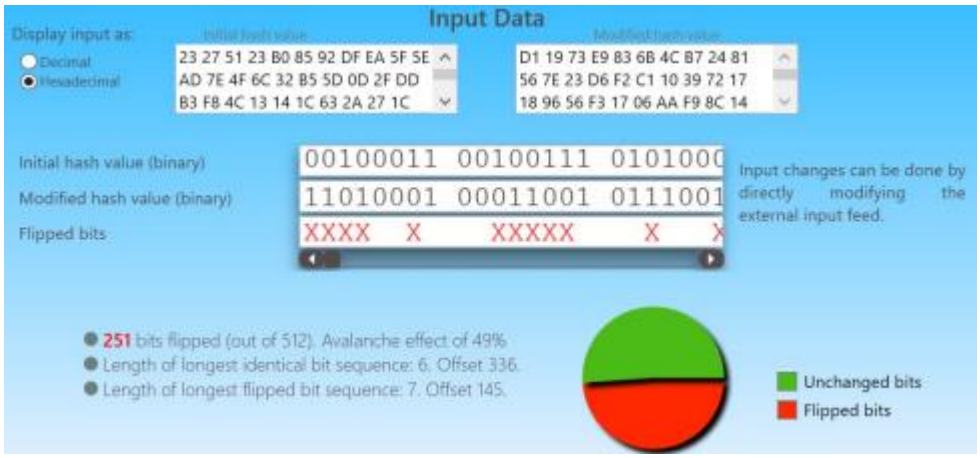

Рисунок 21 – Удаление

Результаты исследования количества измененных бит в дайджестах при добавлении, изменении и удалении одного символа исходного текста занесем в таблицу 3.

Таблица 3

Хэш-функция	Изменение	Количество измененных бит хэша	Среднее количество измененных бит
SHA-3	Добавление	50.6% (259 из 512)	50.2% (257 из 512)
	Изменение	51.2% (262 из 512)	
	Удаление	49% (251 из 512)	

Как видно из результатов, при добавлении, изменении и удалении одного символа исходного текста среднее количество измененных бит составляет около 50% от общего количества бит дайджеста.

3. Контроль целостности по коду HMAC

3.1 Задание

1. Выбрать текст на английском языке (не менее 1000 знаков), добавить ваши 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 и модифицированный тексты коллеге, не раскрывая, какой текст корректен. Попросить коллегу определить это самостоятельно.

3.2 Выбранная схема генерации ключа и ее параметры

На рисунке 22 приставлен текст, размер, которого составляет не менее 1000 символов и в конце добавили ФИО.

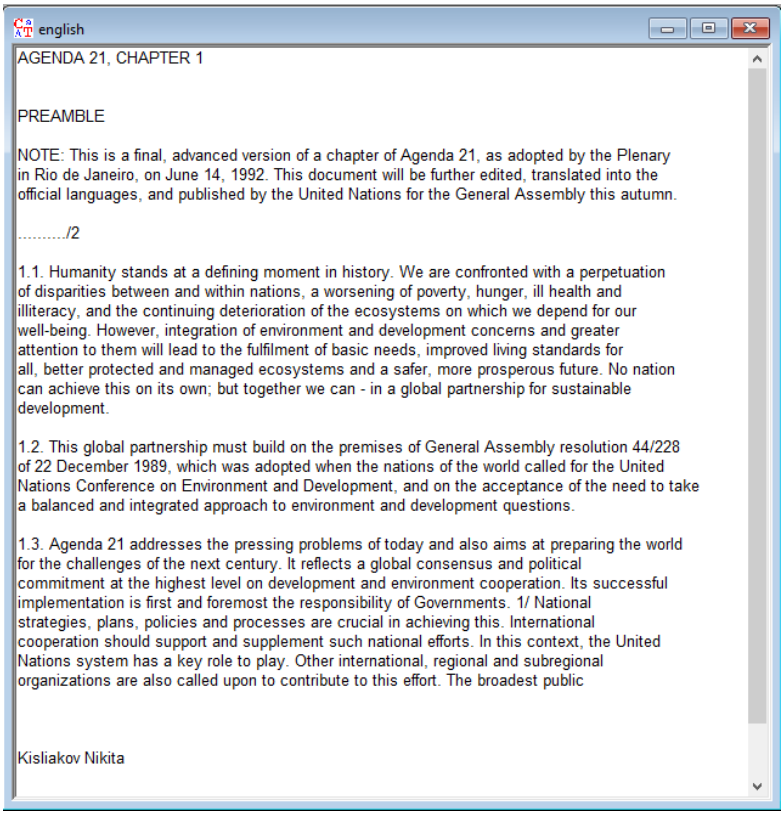

Рисунок 22 – Исходный текст

В CrypTool 1, с помощью утилиты Key Generation, сгенерируем секретный ключ по паролю «ab12cd34ef56gh78ig» на рисунке 23.

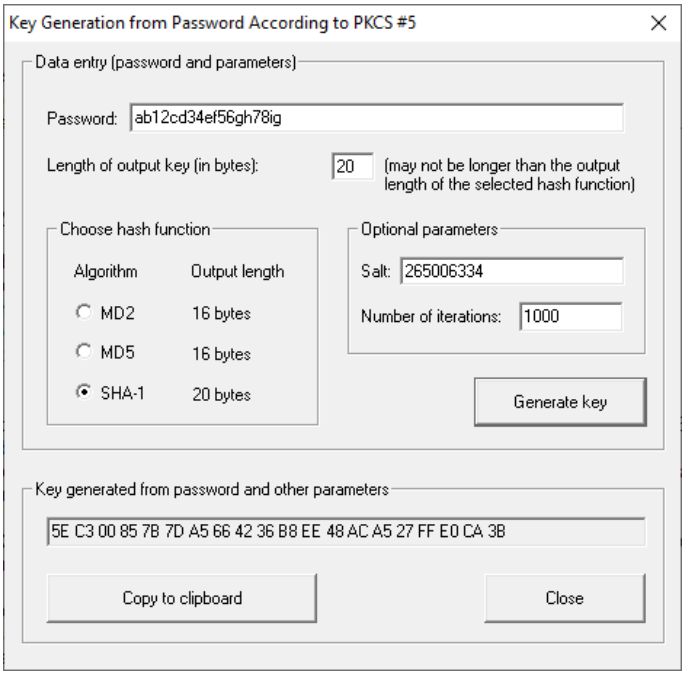

Рисунок 23 – Генерация секретного ключа

В итоге получили селящий сгенерированный секретный ключ: «5E C3 00 85 7B 7D A5 66 42 36 B8 EE 48 AC A5 27 FF E0 CA 3B».

3.3 Выбранная схема создания HMAC

Воспользуемся утилитой Generation of HMACs сгенерируем HMAC для исходного текста и секретного ключа на рисунке 24.

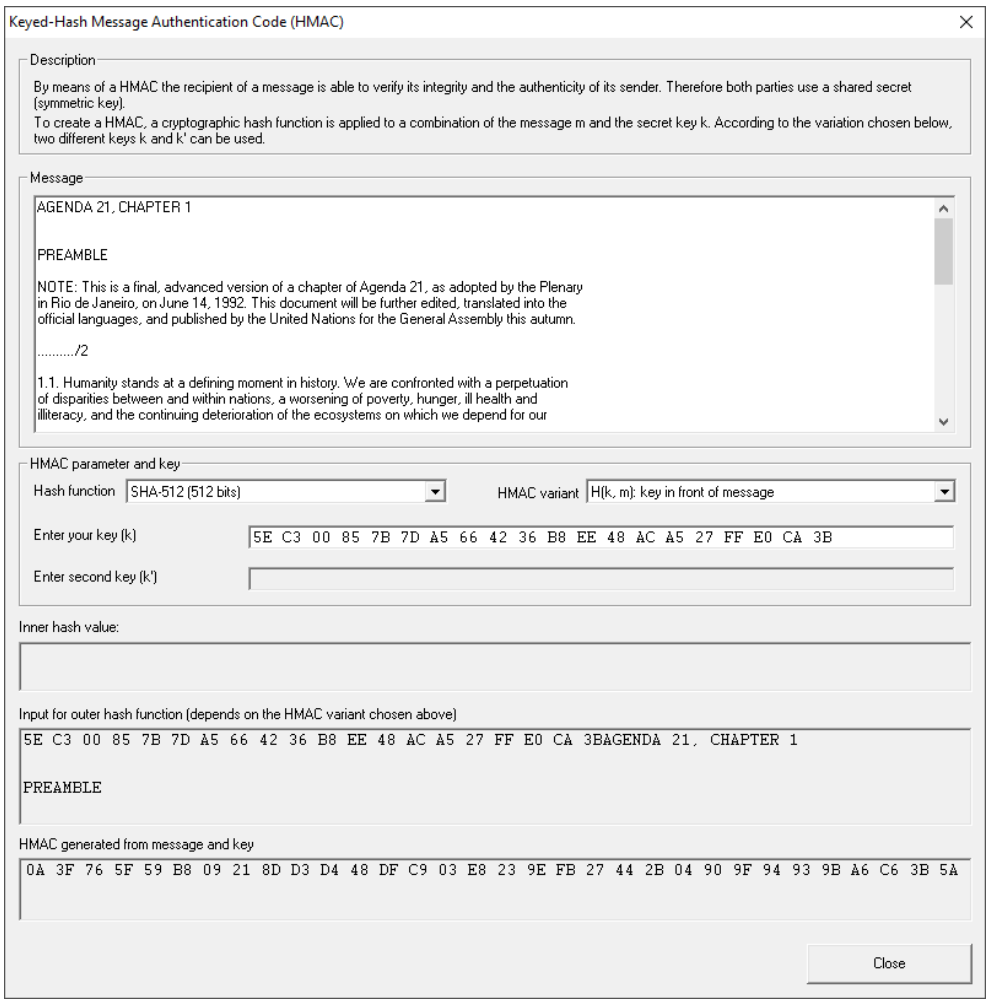

Рисунок 24 – Генерация HMAC

Сгенерированный HMAC: «79 BD 38 50 0A 0E 3A 87 46 38 62 C2 DC CA DB 4E 65 41 AD BB 33 5F 1A 66 CB 61 93 E3 09 58 41 C4 0A 3F 76 5F 59 B8 09 21 8D D3 D4 48 DF C9 03 E8 23 9E FB 27 44 2B 04 90 9F 94 93 9B A6 C6 3B 5A».

3.4 Описание действий передающий стороны на примере выполненного задания

На рисунке 25 продемонстрирована генерация секретного ключа. К исходным данным относится пароль «123456nikita». С помощью пароля генерируется секретный ключ со следущими параметрами (Salt: 1682723281, количество итераций: 1000, хеш-функция: SHA-1). В итоге мы получили секретный ключ «C5 AE 95 DB AA 76 A1 71 19 6D 0C 6F 1E 66 3A 9D 61 26 DE 87».

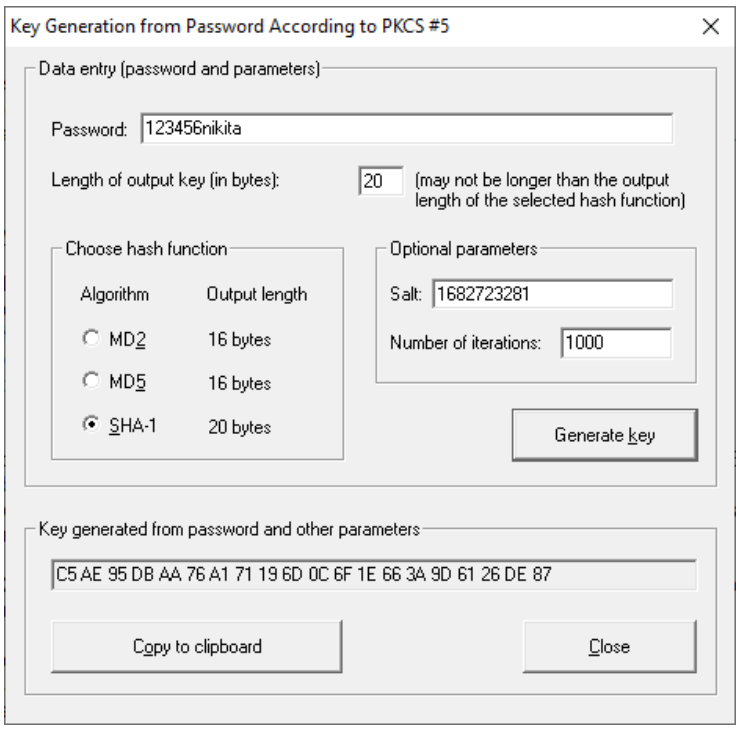

Рисунок 25 – Генерация секретного ключа

Затем генерируем HMAC секретным ключом и с заданными параметрами (хеш-функция: SHA-512, тип: H (k, m)), как показано на рисунке 26.

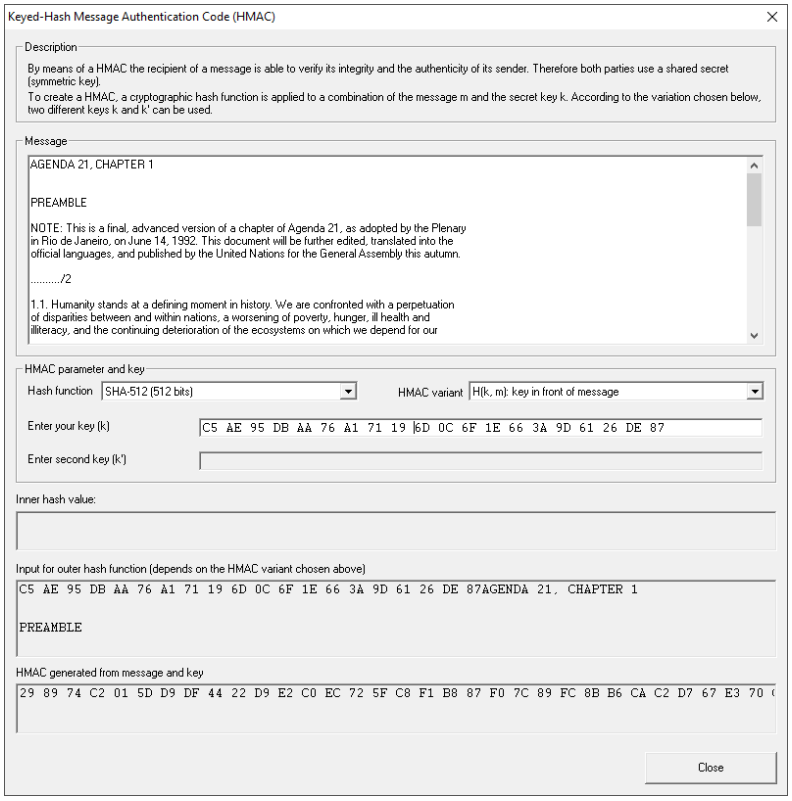

Рисунок 26 – Генерация HMAC с заданным секретным ключом и Параметрами

Теперь отправляем принимающей стороне придуманный пароль; параметры секретного ключа; HMAC; параметры HMAC; два текста (первый – модифицированный, второй – исходный).

3.5 Описание действий принимающей стороны на примере выполненного задания

От передающей стороны была получена следующая информация: пароль («123456nikita»); параметры генерации ключа (Salt: 1682723281, количество итераций: 1000, хеш-функция: SHA-1); HMAC («29 89 74 C2 01 5D D9 DF 44 22 D9 E2 C0 EC 72 5F C8 F1 B8 87 F0 7C 89 FC 8B B6 CA C2 D7 67 E3 70 C4 4C 4C 55 CE A2 3A E5 1A A7 60 6D 2B A1 20 D0 80 DE F0 14 54 CA 21 C1 80 CA 4A 37 B3 09 AB 1A»); параметры HMAC (хеш-функция: SHA-512, тип: H (k, m)); два текста (неизвестно который из них модифицированный). Для обоих текстов по паролю и характеристиками HMAC вычислим HMAC. Результаты вычислений представлены на рисунках 27, 28.

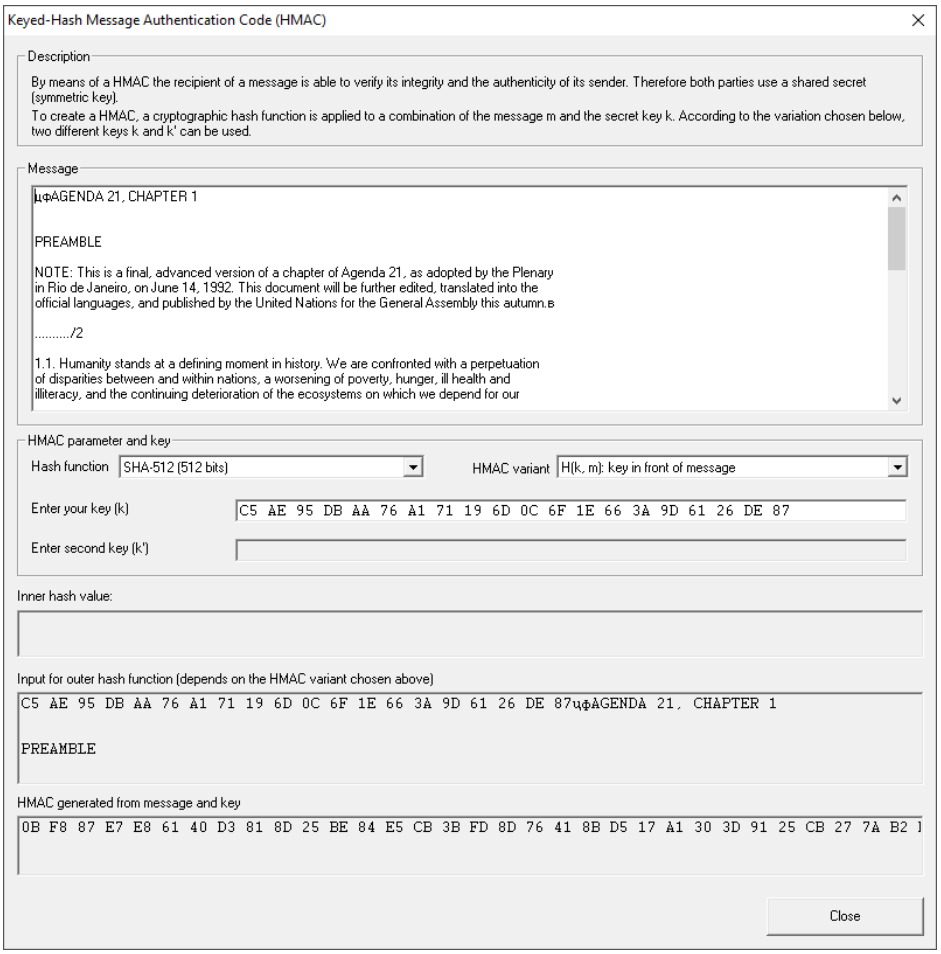

Рисунок 27 –HMAC первого текста

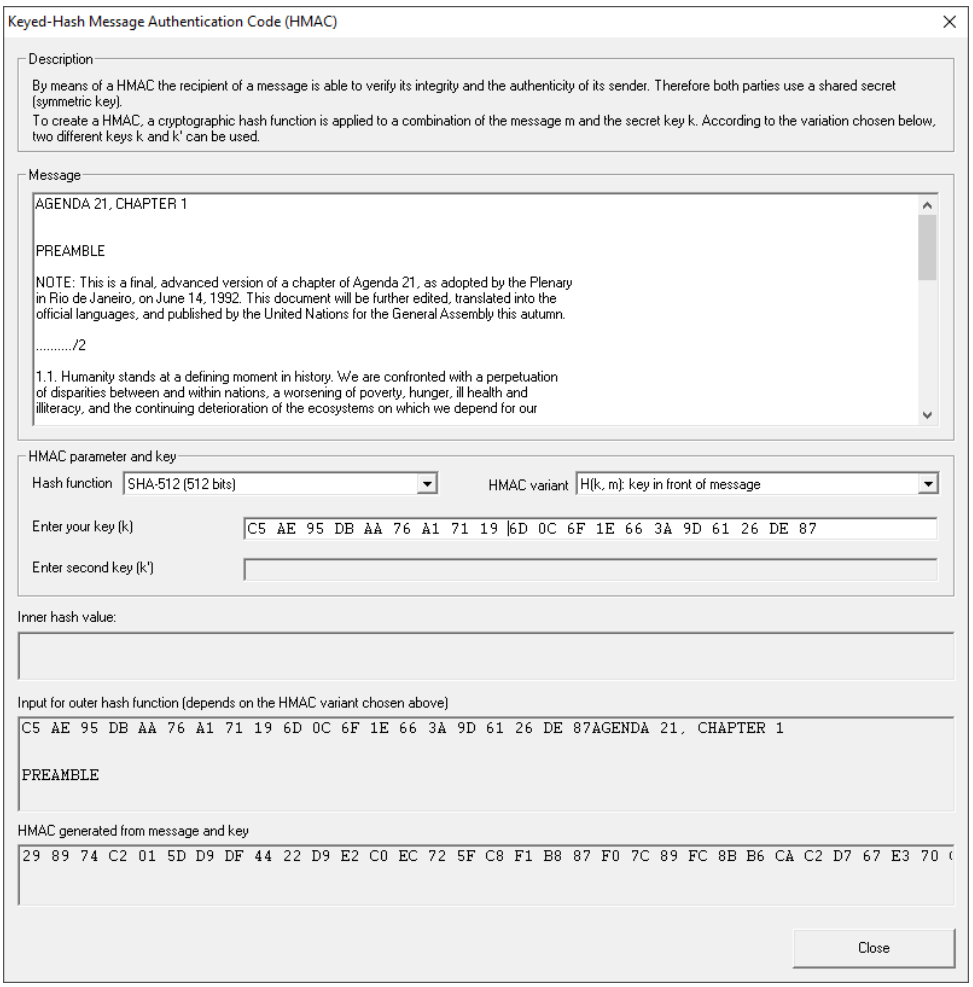

Рисунок 28 – HMAC второго текста

HMAC первого текста: «EF 8E F9 61 E0 2F 2B 26 17 A2 5F 99 BA 00 5A 04 65 9A 66 C0 C5 E8 3A BB 8E 2E AF 22 83 4B AE 04 1E 01 02 09 A1 11 D1 18 D9 DE 54 B7 EB D4 CD ED A6 A1 06 F6 08 33 C3 35 1B 44 86 B2 36 61 AB A7». Как видно, он не совпадает с полученным от передающей стороны, следовательно это модифицированный текст.

HMAC второго текста: «29 89 74 C2 01 5D D9 DF 44 22 D9 E2 C0 EC 72 5F C8 F1 B8 87 F0 7C 89 FC 8B B6 CA C2 D7 67 E3 70 C4 4C 4C 55 CE A2 3A E5 1A A7 60 6D 2B A1 20 D0 80 DE F0 14 54 CA 21 C1 80 CA 4A 37 B3 09 AB 1A». Видно, что он совпадает с полученным от передающей стороны, следовательно это исходный текст.

4. Атака дополнительной коллизии на хеш-функцию

4.1 Задание

1. Сформировать два текста на английском языке – истинный и фальсифицированный. Сохранить тексты в файлах формата .txt.

2. Утилитой Analysis –> Attack on the hash value... модифицировать сообщения 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.

4.2 Описание атаки в терминах парадокса «дней рождения»

Модель атаки коллизии на хеш-функцию основана на одном из парадоксов дней рождений. Оказывается, что в группе, состоящей всего из 23 или более человек, вероятность совпадения дней рождения (по числу и месяцу) хотя бы у двоих из группы превышает 50 %. Применительно к хеш-функции это означает, что сложность атаки, цель которой состоит в поиске двух сообщений с одинаковыми значением хеш-функции, пропорциональна $\sqrt{2^N}$, где N – длина хеш-кода.

4.3 Представление результатов атаки: исходные и модифицированные тексты, статистика, дайджесты исходных и модифицированных сообщений

Подготовим два текста, один истинный, второй фальсифицированный. Данные текста представлены на рисунке 29,30.

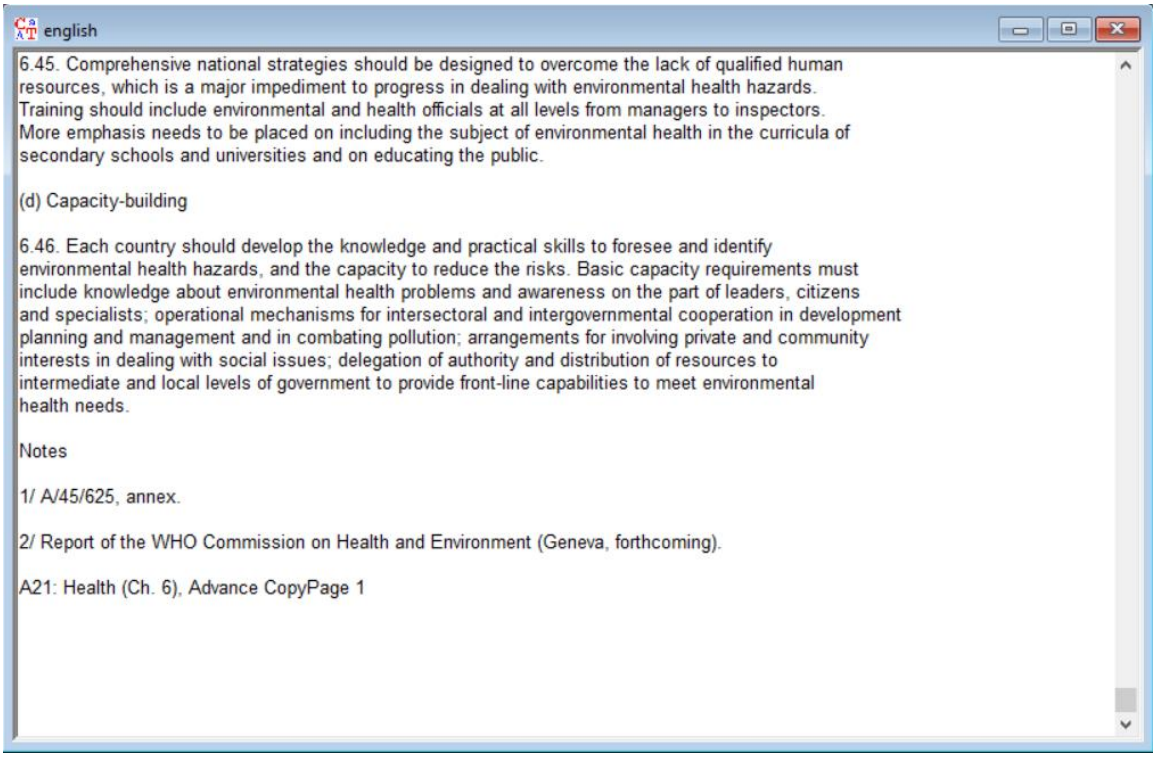

Рисунок 29 – Истинный текст

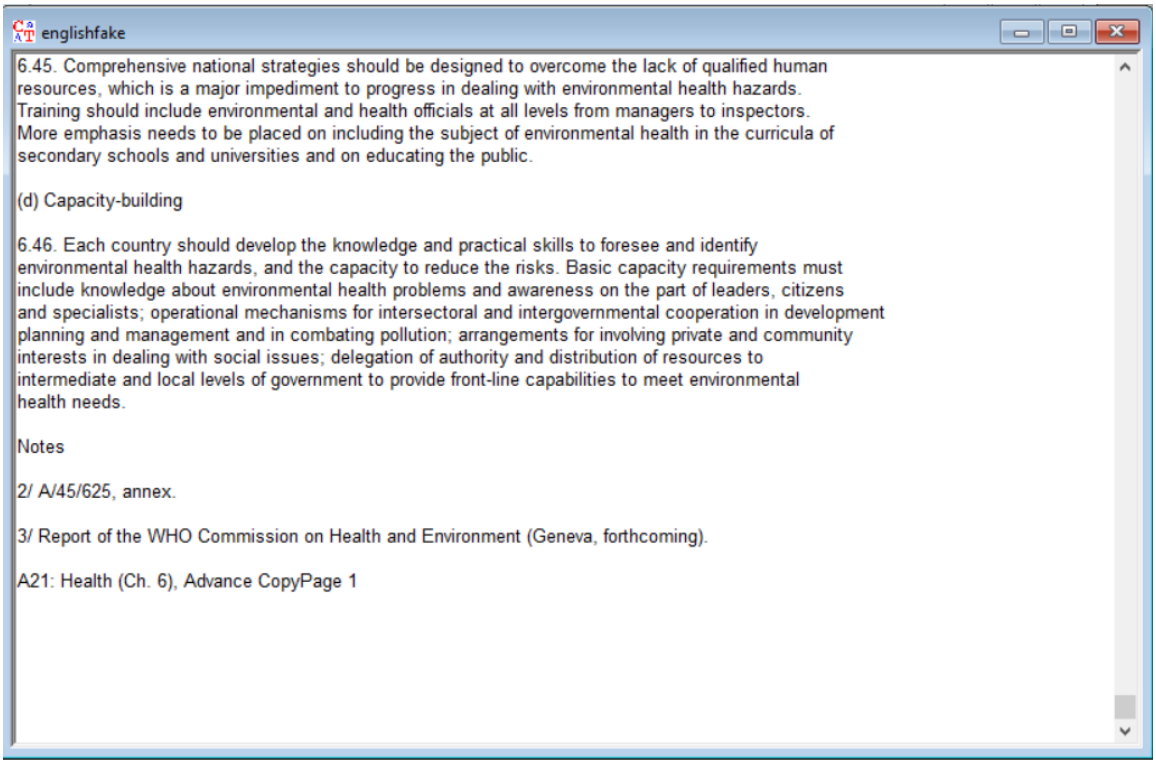

Рисунок 30 – Фальсифицированный текст

Хэш истинного текста: «92 5C 8C C7 87 B6 1D 7B 0E 00 A3 80 1E 5B 88 6B A6 33 BE 58».

Хэш истинного текста: «5D B7 24 C8 D5 AE CF C7 29 07 4E F9 68 5A F4 0C 67 8D A1 23».

Воспользуемся утилитой CrypTool1 1 Attack on the hash value… модифицируем сообщения для получения одинакового дайджеста. В качестве метода модификации выбрать Attach characters -> Printable characters. Статистика атаки показана на рисунке 31.

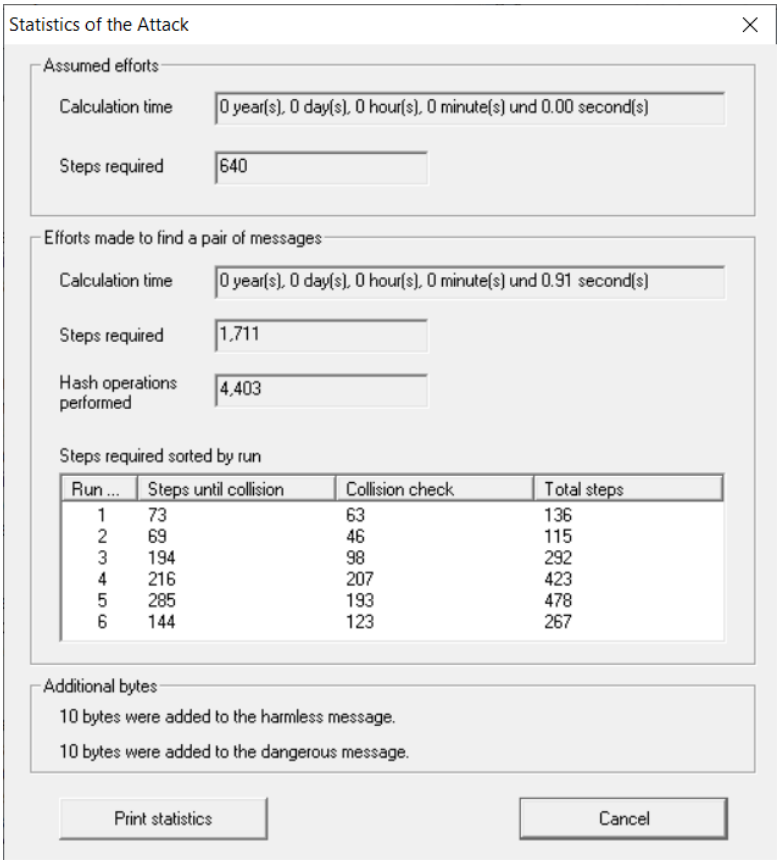

Рисунок 31 – Статистика атаки

Полученные тексты представлены на рисунках 32 и 33.

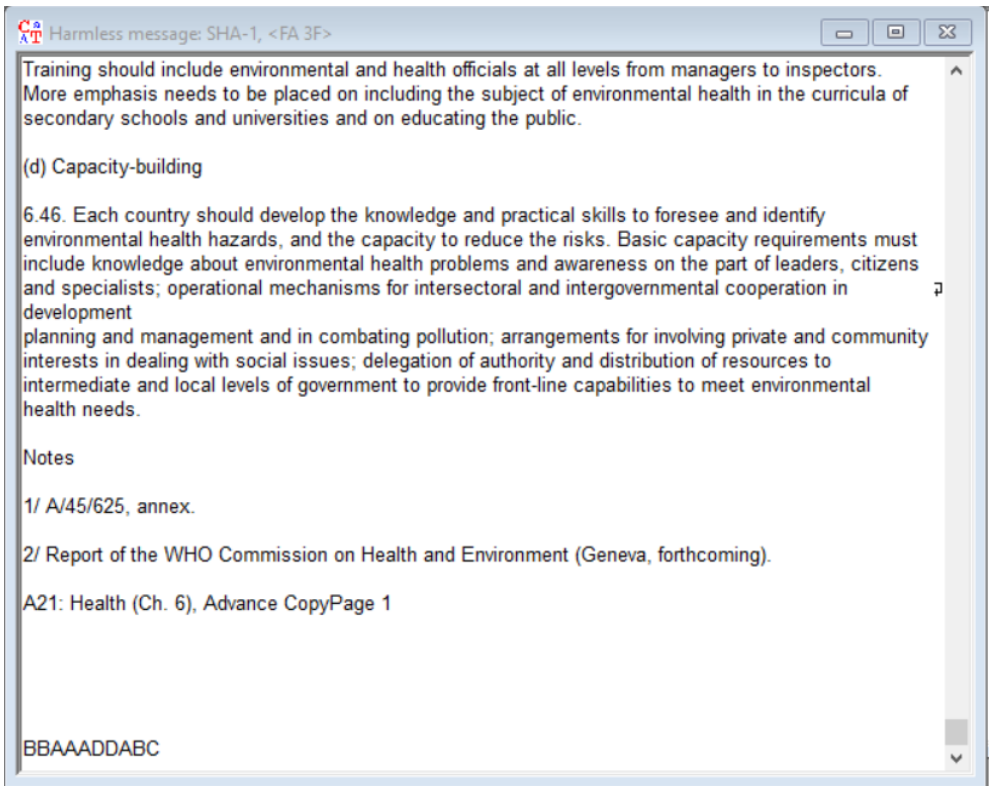

Рисунок 32 – Истинный текст

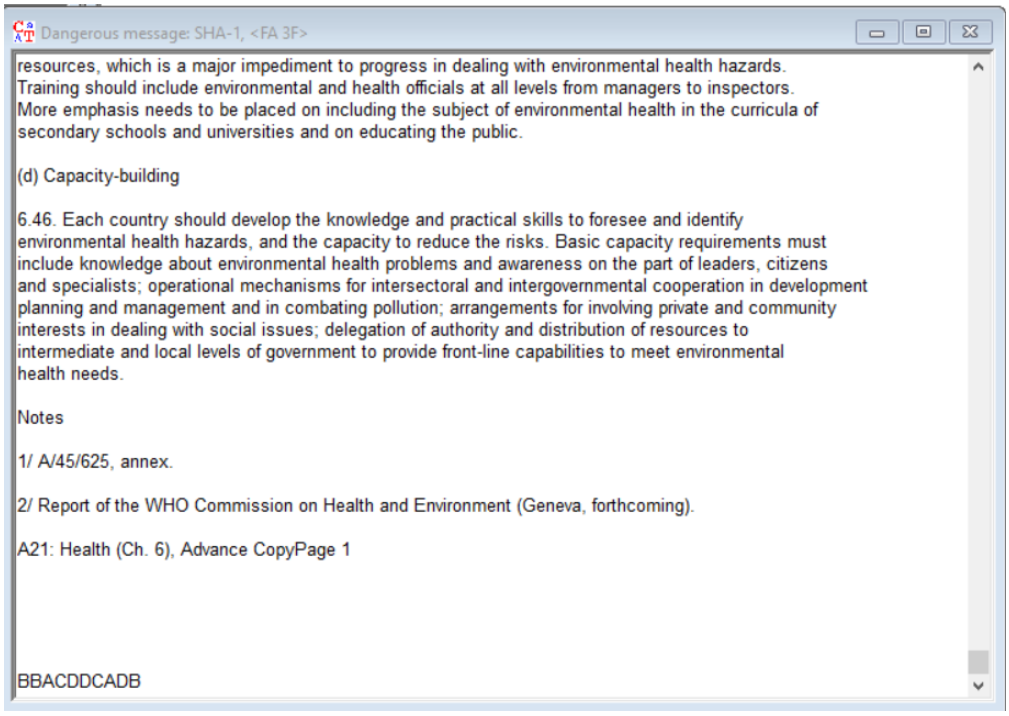

Рисунок 33 – Фальсифицированный текст

Хэш истинного текста после атаки: «FA 3F 16 1B 2C CB 40 68 14 B5 1B A1 4E 59 65 EE B0 B0 31 CF» (хэш-функция SHA-1).

Хэш фальсифицированного текста: «FA 3F 55 AE 94 F2 44 67 19 1B 40 62 9F 39 2B 51 A1 70 0B CE» (хэш-функция SHA-1).

Следовательно, атака прошла успешно, так как первые байты (16 бит) хэша совпадают с заданной точностью (16 бит).

4.4 Таблица с оценками временной сложности атаки

Теперь зафиксируем временную сложность атаки при различном количестве совпадающих частей хэша. Результаты запишем в таблицу 4.

Таблица 4

Количество совпадающих бит	Время атаки
8 бит	0 секунд
16 бит	0 секунд
24 бит	0,1 секунда
32 бит	1,72 секунды
40 бит	27,56 секунд
48 бит	7 минут 21,09
56 бит	~ 1 час 6 минут
64 бит	~ 16 часов
72 бит	~ 11 дней
80 бит	~ 180 дней
88 бит	~ 8 лет
96 бит	~ 1 век 30 лет
104 бит	~ 2 тысячелетия 3 века
112 бит	~ 35 тысячелетий
120 бит	~ 510 тысячелетий

Выводы

В данной лабораторной работе были изучен хэш-функции (MD5, SHA-1, SHA-256, SHA-512, SHA-3), код аутентификации HMAC, а также атака дополнительной коллизии на хэш-функцию. Также были получены практические навыки работы с рассматриваемыми хэш-функциями с использованием приложения CrypTool 1 и CrypTool 2.

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

В результате применения инструментария CrypTool 1 был исследован лавинный эффект для хэш-функций MD5, SHA-1 и Sha-256, а также SHA-512.

Был установлен факт того, что добавление, изменение или удаление одного символа в исходном тексте может изменить около 51% битов дайджеста для этих хэш-функций.

Отсюда следует, что рассмотренные хэш-функции обладают лавинными эффектом.

2. Хеш-функция SHA-3:

При помощи демонстрационного примера была изучена работа хэш-функции SHA-3. При этом было установлено, что в основе SHA — 3 лежит конструкция «Губка». На этапе «впитывания» очередные блоки сообщения подмешиваются к части внутреннего состояния, а на этапе «отжатия» извлекаются очередные части дайджеста. На каждой итерации применяется многораундовая бесключевая псевдослучайная перестановка, которая может выбирается пользователем из набора предопределенных функций. В рассматриваемом SHA-3 используется функция f-1600, размер дайджеста составляет 512 бит, размер блока сообщения составляет 576 бит, размер внутреннего состояния составляет 1600 бит, а количество раундов равно 24.

При помощи инструментария CrypTool 2 был изучен лавинный эффект для хэш функции SHA-3. Было определено, что добавление, изменение или удаление одного символа в исходном тексте приводит к изменению в среднем

50% битов дайджеста для этой хэш-функций. Отсюда следует, что рассмотренная хэш-функция обладает лавинным эффектом.

3. Контроль целостности по коду HMAC:

В данной главе был изыуен HMAC, это механизм целостости информции. С помощью него, позволяет генерировать то, что данные, передаваемые или хранящиеся в надежной среде, не блыли посторонними лицами.

При помощи известной информации о пароле, HMAC и методе генерации ключа и HMAC, была осуществлена проверка целостности двух сообщений, полученных от коллеги. В результате было выявлено, что одно из сообщений было модифицировано.

4. Атака дополнительной коллизии на хеш-функцию.

Была изучена ата дополнительной коллизии на SHA-1. Дайджест состяалял 160 бит. В зависиомти от размера дайджеста зависит время. Если дайджест будет не более 48 бит. То будет занямаить не более 10 мину, если больше 48 бит . то потребуется гораздо больше времени.