**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра ИБ**

**отчет**

**по лабораторной работе №6**

**по дисциплине «Криптографические методы защиты информации»**

**Тема:** **Изучение хэш-функций**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 5381 |  | Кобылянский А.В. |
| Преподаватель |  | Племянников А.К. |

Санкт-Петербург

2018

**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 Формулировка задания.**

Задание

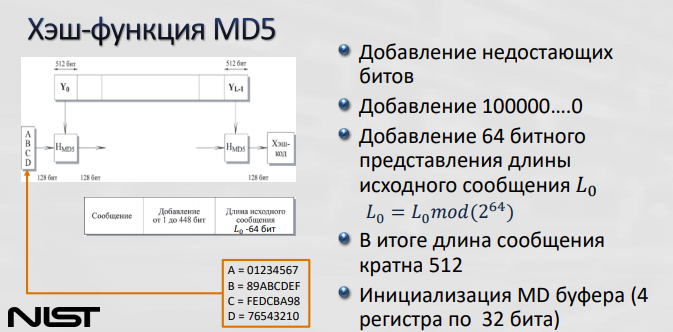
1. Открыть текст не менее 1000 знаков. Добавить свое ФИО последней строкой. Перейти к утилите Indiv.Procedures->Hash->Hash Demonstration..
2. Задать хэш-функцию, подлежащую исследованию: MD5, SHA-1, SHA-256, SHA-512.?,
3. Для каждой хэш-функции повторить следующие действия:
4. Измените (добавлением,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

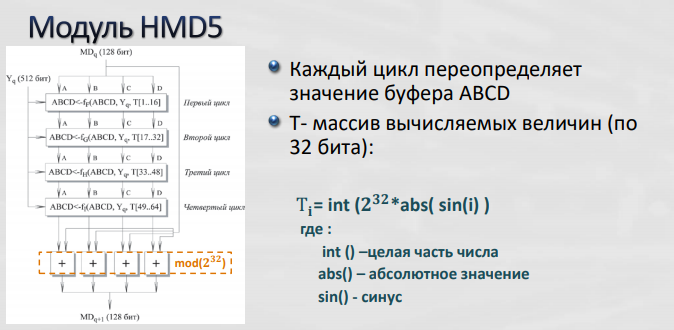
Содержание раздела отчета

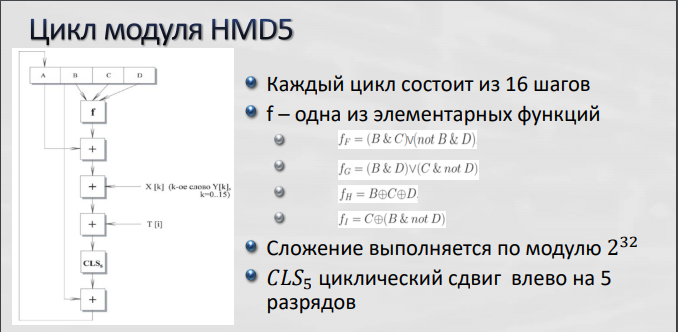
1. Формулировка задания
2. Основные параметры и обобщенная схема хэш-функций MD5, SHA-1 (см. лекцию)
3. Таблица с фактическими и усредненными параметрами лавинного эффекта для исследованных хэш-функций
4. Выводы

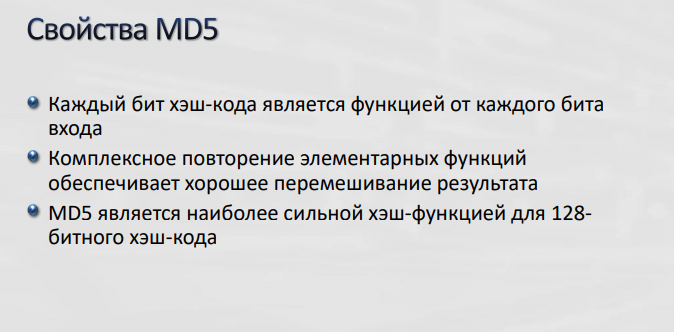
**1.2 Основные параметры и обобщенная схема хэш-функций MD5, SHA-1**

**МD5**

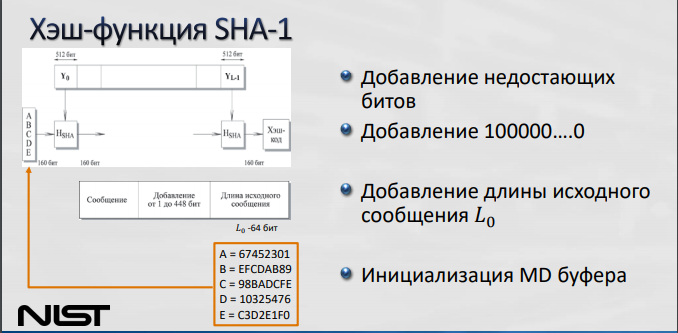


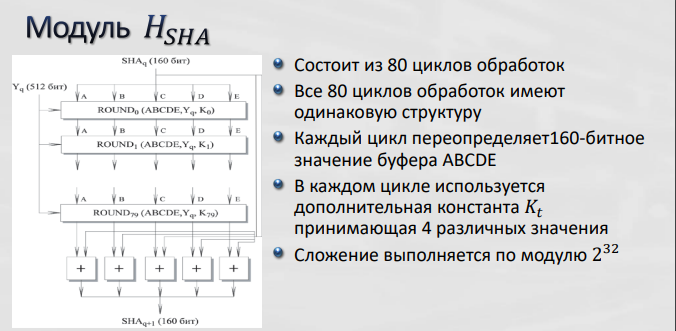


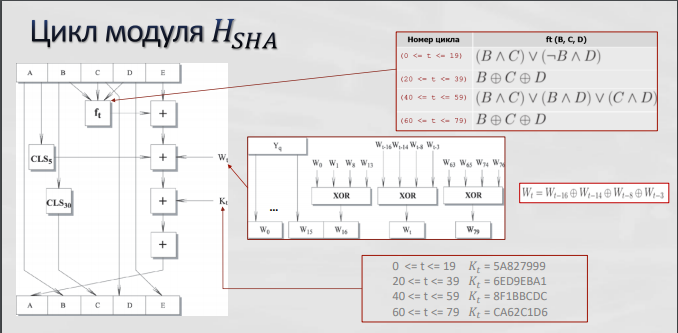


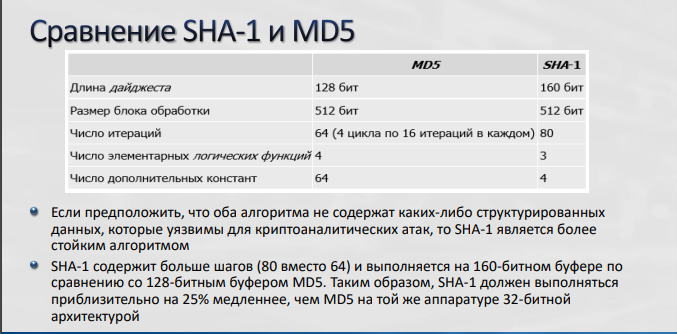


**SHA-1**









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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | MD5 | SHA-1 | SHA-256 | SHA-512 |
| 1 | 55 | 73 | 135 | 267 |
| 2 | 57 | 74 | 126 | 235 |
| 3 | 62 | 72 | 156 | 252 |
| Среднее | 58/128 | 73/160 | 139/256 | 251.(3)/512 |

**1.4 Выводы**

При незначительном изменении текста, данные хэш-функции меняют половину дайджеста. Наиболее эффективно показал себя алгоритм SHA-256.

**2. Хэш-функция SHA-3**

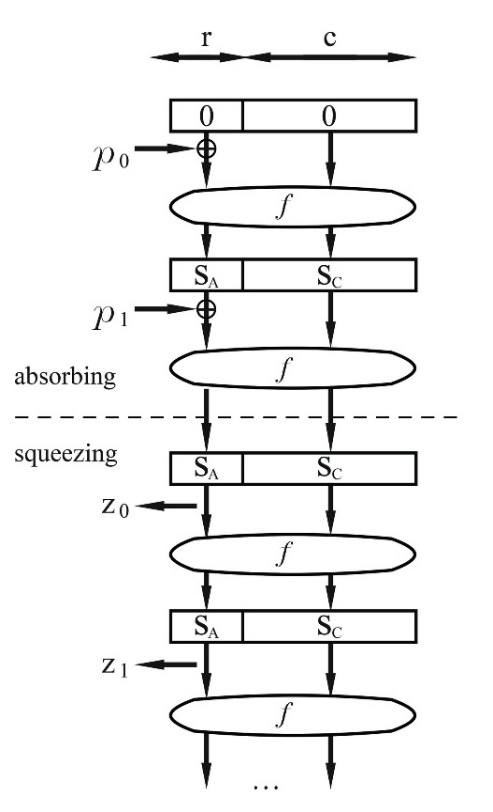
**2.1. Формулировка задания.**

Задание

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

Содержание раздела отчета

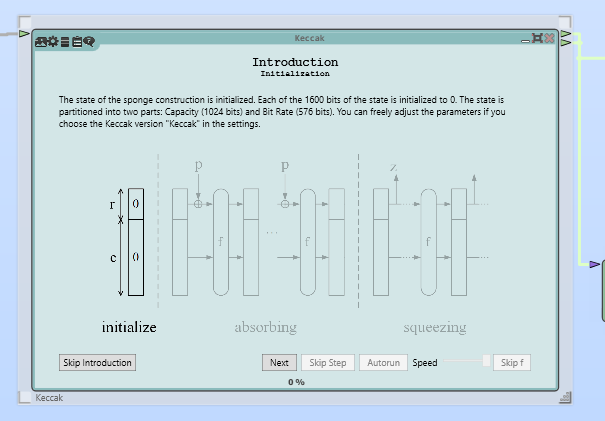
1. Формулировка задания
2. Основные параметры и обобщенная схема хэш-функции Keccak Hash, (SHA-3) на основе изученной презентации
3. Описание средства оценивания лавинного эффекта
4. Таблица с фактическими параметрами лавинного эффекта
5. Выводы
   1. **Основные параметры и обобщенная схема хэш-функции Keccak Hash, (SHA-3) на основе изученной презентации**

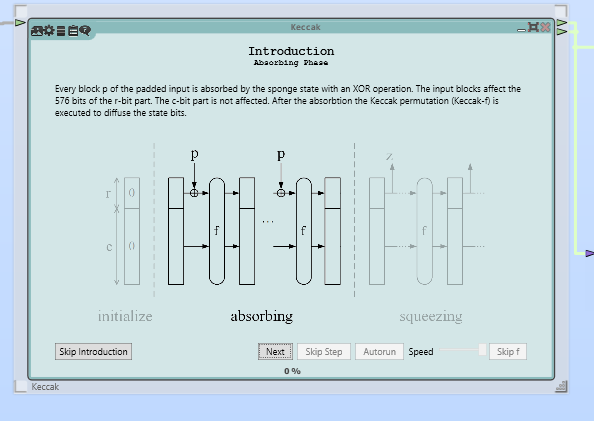
В основе Keccak (SHA-3) лежит конструкция под названием Sponge – губка. Сам алгоритм состоит из 2-х этапов:

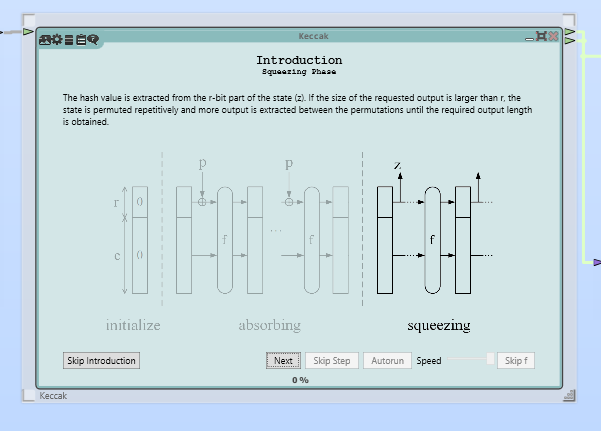
1. *Впитывание – Absording.* На каждом шаге очередной блок сообщения *pi* длиной r подмешивается к части внутреннего состояния *S*, которая затем целиком модифицируется функцией *f*-многораундовой бесключевой псевдослучайной перестановкой.
2. *Отжатие – Squeezing.* Чтобы получить хэш, функция *f* многократно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

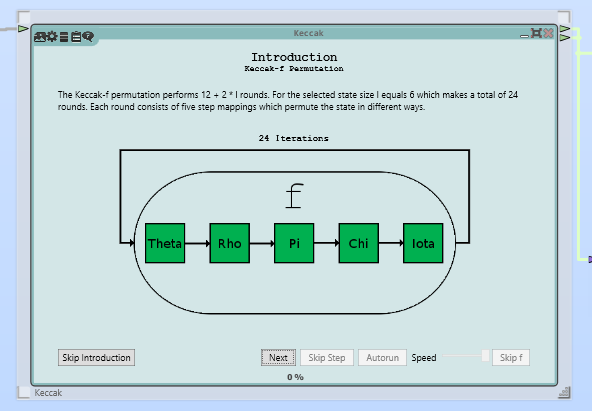
Рисунок 6.1

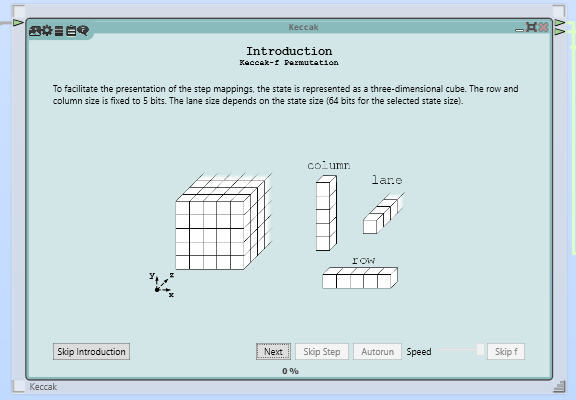
Обобщенная схема работы алгоритма представлена на рисунке 6.1.

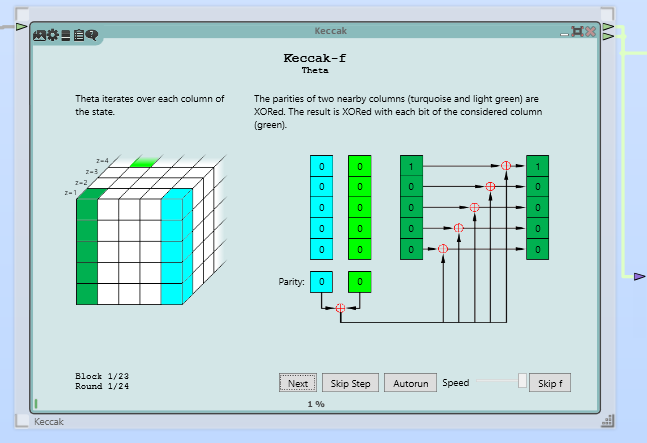


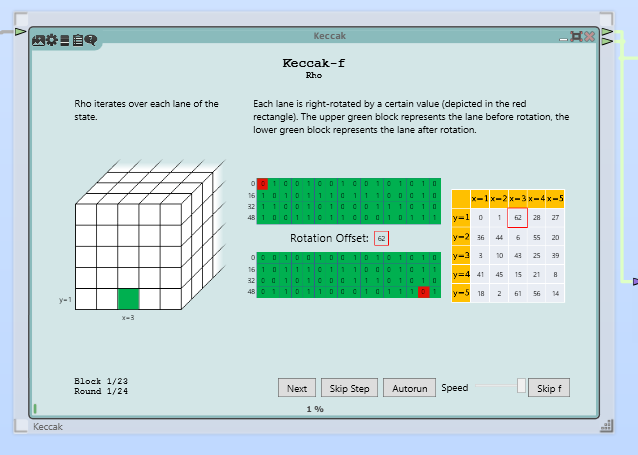


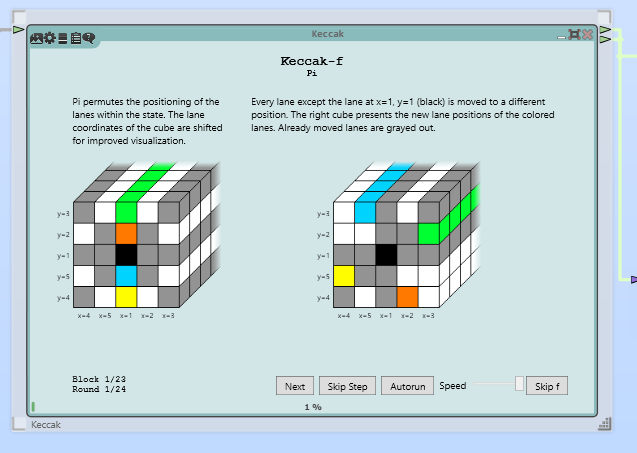


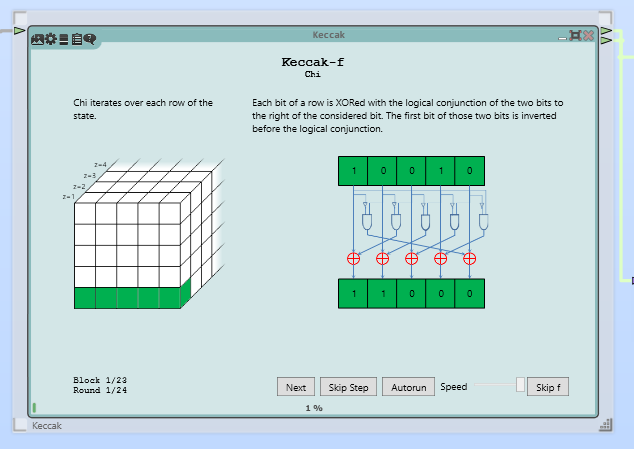


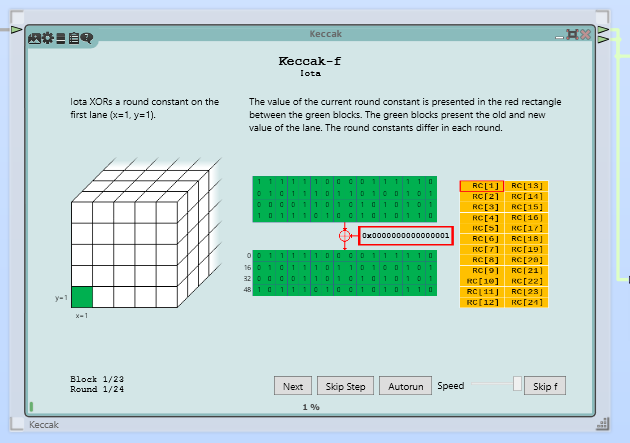


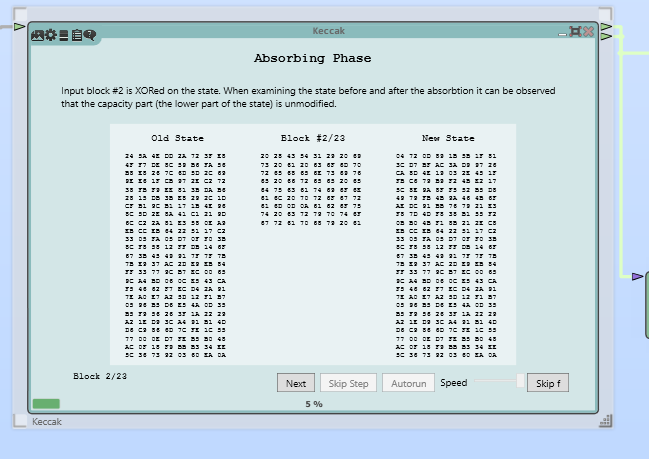












* 1. **Описание средства оценивания лавинного эффекта**

Считаем хэш от исходного текста, сохраняем его. Считаем хэш от текста с замененным символом, с удаленным символом, с добавленным символом. С помощью программы на питоне находим, сколько бит меняется при данных изменениях. Затем вычисляем среднее значение количества измененных бит.

* 1. **Таблица с фактическими параметрами лавинного эффекта**

|  |  |
| --- | --- |
|  | Keccak(512) |
| Добавление символа | 272 |
| Замена символа | 260 |
| Удаление символа | 276 |
| Среднее | 269.(3) / 512 |

* 1. **Выводы**

Алгоритм хэширования SHA-3 работает по схеме Sponge, чтобы захэшировать строку, нужно сначала разбить её на куски определенного размера и на каждом раунде ксорить(подмешивать) их не ко всему 1600 битному состоянию, а только к его началу размера r. То есть, на каждом раунде очередной кусок строки подмешивается только к части состояния, тогда как псевдослучайная перестановка f обрабатывает всё состояние целиком, размазывая таким образом строку по состоянию и делая его зависимым от всей строки. Чтобы получить собственно хэш, мы продолжаем применять функцию перестановки f к состоянию, и на каждом этапе копируем из него лишь кусок размера r до тех пор, пока не получим хэш необходимой длины (эти куски мы конкатенируем). Это т.н. «отжатие» губки.

1. **Контроль целостности по коду HMAC**

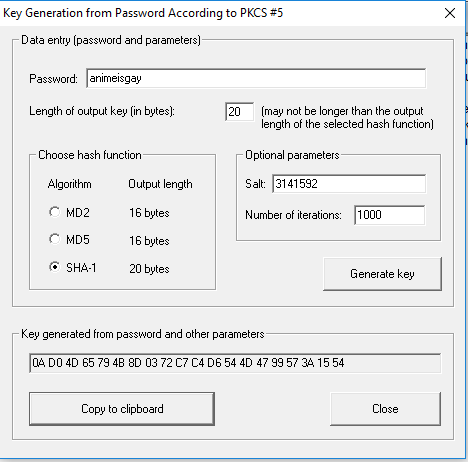
**3.1. Формулировка задания.**

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.

Содержание раздела отчета

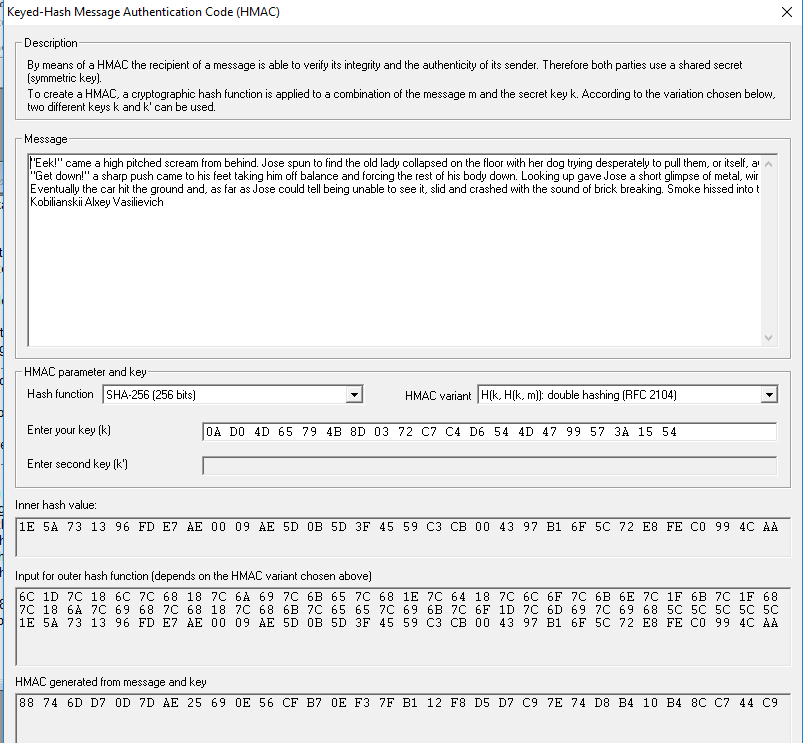
1. Формулировка задания.
2. Выбранная схема генерации ключа и ее параметры.
3. Выбранная схема создания HMAC.
4. Описание действий передающий стороны на примере выполненного задания.
5. Описание действий принимающей стороны на примере выполненного задания.
6. Выводы.
   1. **Выбранная схема генерации ключа и ее параметры.**



**Ключ:**

0A D0 4D 65 79 4B 8D 03 72 C7 C4 D6 54 4D 47 99 57 3A 15 54

* 1. **Выбранная схема создания HMAC.**



**3.4 Описание действий передающей стороны на примере выполненного задания.**

1. Сгенерировать секретный код из пароля и соли
2. Согласовать параметры алгоритма HMAC и секретный код с принимающей стороной;
3. Посчитать HMAC от передаваемого текста;
4. Изменить текст
5. Передать исходный текст, искаженный текст и HMAC.

**3.5 Описание действий принимающей стороны на примере выполненного задания.**

1. Согласовать параметры алгоритма HMAC и секретный код с передающей стороной;
2. Получить HMAC и 2 текста;
3. Посчитать HMAC от переданных текстов;
4. Сравнить свои HMAC и полученный. Текст, у которого посчитанный HMAC совпал с передаваемым и будет корректным.

**3.6 Выводы**

HMAC позволяет убедиться в том, что данные не изменялись каким бы то ни было способом с тех пор, как они были созданы, переданы или сохранены доверенным источником. Для такого рода проверки необходимо, чтобы, например, две доверяющие друг другу стороны заранее договорились об использовании секретного ключа, который известен только им. Тем самым гарантируется аутентичность источника и сообщения. Недостаток такого подхода очевиден — необходимо наличие двух доверяющих друг другу сторон.

1. **Атака дополнительной коллизии на хэш-функцию**

**4.1 Формулировка задания.**

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.

Содержание раздела отчета

1. Формулировка задания.
2. Описание атаки в терминах парадокса «дня рождения» (см. лекцию).
3. Представление результатов атаки: исходные и модифицированные тексты, статистика, дайджесты исходных и модифицированных сообщений.
4. Таблица с оценками временной сложности атаки.
5. Выводы.
   1. **Описание атаки в терминах парадокса «дня рождения».**

Атака дополнительной коллизии основана на одной из проблем парадокса дня рождений. Он заключается в следующем: в группе, состоящей из 23=с\*√365 или более человек, [вероятность](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C) совпадения [дней рождения](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BD%D1%8C_%D1%80%D0%BE%D0%B6%D0%B4%D0%B5%D0%BD%D0%B8%D1%8F) (число и месяц) хотя бы у двух людей превышает 50 [%](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%BD%D1%82). Применительно к хэш-функции это означает, что сложность атаки, целью которой является поиск двух сообщений с одинаковыми значением хэш-функции, пропорционально , где N – длина хэш-кода.

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

Оригинальный harmless

Текст: We will never ask you for your account password. We may ask for your account number, but never for your password.

Хэш: DF 77 AC 0C D0 FF 21 43 2B C0 D5 6E 59 81 95 75 2E 28 DD 0F

Оригинальный dangerous

Текст: Please send me your password and nudes.

Хэш: CC C0 4F 56 91 C2 AC 11 2A B3 88 4C 78 89 15 BB 0B FB 04 AE

Модифицированный harmless

Текст: We will never ask you for your account password. We may ask for your account number, but never for your password.ACAACBAABC

Хэш: 3C 9D 6E A8 82 22 36 3E 87 8B E8 5B 90 F2 D1 BD 76 22 3C 17

Модифицированный dangerous текст.

Текст: Please send me your password and nudes.ACCBBBABCC

Хэш: 3C 9D 64 B2 C7 14 3C 09 62 A7 03 AB 1E 75 D6 04 B0 85 C8 AD

COMPUTING EFFORTS

Calculating time: 0 year(s), 0 day(s), 0 hour(s), 0 minute(s) und 0.00 second(s)

Steps required

Hash operations performed

RunNo. Steps until collision Check of the collision Total steps

01 273 258 531

02 332 49 381

03 324 167 491

TEXT MODIFICATION

10 bytes were added to the harmless message.

10 bytes were added to the dangerous message.

**4.4 Таблица с оценками временной сложности атаки.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8 | 16 | 32 | 40 | 48 |
| 0 с | 0 с | 0.18 с | 0.97 с | 51.18 с |

* 1. **Выводы**

Использование дополнительной коллизии на хеш-функцию прошло очень успешно и в очень короткие сроки. Что показывает эффективность использования этой уязвимости.

**Вывод**

* При незначительном изменении текста, хэш-функции MD5, SHA-1, SHA-256, SHA-512 меняют половину дайджеста при замене даже одного символа.
* HMAC позволяет убедиться в том, что данные не изменялись каким бы то ни было способом с тех пор, как они были созданы, переданы или сохранены доверенным источником. Для такого рода проверки необходимо, чтобы, например, две доверяющие друг другу стороны заранее договорились об использовании секретного ключа, который известен только им. Тем самым гарантируется аутентичность источника и сообщения. Недостаток такого подхода очевиден — необходимо наличие двух доверяющих друг другу сторон.
* Использование дополнительной коллизии на хеш-функцию прошло очень успешно и в очень короткие сроки. Что показывает эффективность использования этой уязвимости.
* Алгоритм хэширования SHA-3 работает по схеме Sponge, чтобы захэшировать строку, нужно сначала разбить её на куски определенного размера и на каждом раунде ксорить(подмешивать) их не ко всему 1600 битному состоянию, а только к его началу размера r. То есть, на каждом раунде очередной кусок строки подмешивается только к **части**состояния, тогда как псевдослучайная перестановка **f** обрабатывает всё состояние **целиком**, размазывая таким образом строку по состоянию и делая его зависимым от всей строки. Чтобы получить собственно хэш, мы продолжаем применять функцию перестановки f к состоянию, и на каждом этапе копируем из него лишь кусок размера r до тех пор, пока не получим хэш необходимой длины(эти куски мы конкатенируем). Это т.н. «отжатие» губки.