МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Институт: №8 «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Курсовой проект
по курсу «Криптография»

Группа: М8О-306Б-21

Студент: Н. И. Лохматов

Преподаватель: А. В. Борисов

Оценка:

Дата: 06.04.2024

Москва, 2024

Оглавление

1 Тема....3
2 Задание....3
3 Теория....4
4 Ход лабораторной работы....5
5 Выводы....5

1 Тема

Применение дифференциального криптоанализа к различным алгоритмам хеширования для исследования для исследования различия отдельных бит при изменении количества раундов.

2 Задание

№0. Строку, в которой записано своё ФИО, подать на вход в хеш-функцию ГОСТ Р 34.11-2012 (Стрибог). Младшие 4 бита выхода интерпретировать как 16-тиричное число, которое в дальнейшем будет номером варианта.

№1. Программно реализовать один из алгоритмов функции хеширования в соответствии с номером варианта. Алгоритм содержит в себе несколько раундов.

№2. Модифицировать оригинальный алгоритм таким образом, чтобы количество раундов было настраиваемым параметром программы. в этом случае новый алгоритм не будет являться стандартом, но будет интересен для исследования.

№3. Применить подходы дифференциального криптоанализа к полученным алгоритмам с разным числом раундов.

№4. Построить график зависимости количества раундов и возможности различения отдельных бит при количестве раундов 1,2,3,4,5,…

№5. Сделать выводы.

3 Теория

Мой вариант 7 – Keccak.

Keccak основан на конструкции sponge construction, которая характеризуется своей универсальностью и позволяет получить из функции не только хеш-значение фиксированной длины, но и шифрование, генерацию псевдослучайных чисел и другие криптографические операции. Алгоритм работает путём поглощения входных данных (сообщений) блоками и последующего "отжима" хеш-значения или другого выходного сообщения нужной длины.

Keccak имеет несколько параметров, таких как длина блока (r), длина хеша и количество раундов перестановок. Их выбор определяет безопасность и производительность конкретной реализации Keccak.

В сердце Keccak лежит функция перестановок, которая обрабатывает внутреннее состояние фиксированной длины через серию операций. Количество раундов и их сложность играют ключевую роль в обеспечении криптографической стойкости.

Keccak-256 является одной из наиболее часто используемых версий Keccak, генерирующей хеш длиной в 256 бит. Это делает её особенно подходящей для систем, где требуется высокая стойкость к коллизиям, например, в блокчейн-технологиях и криптовалютах, включая Ethereum.

Дифференциальный криптоанализ — это метод анализа криптографических алгоритмов, основанный на изучении, как различия во входных данных алгоритма могут влиять на различия в выходных данных. Основная идея состоит в том, чтобы рассмотреть два сообщения, которые отличаются фиксированным образом, и анализировать, как это влияет на выходные данные после применения шифра. Дифференциал — это разница между двумя входами и соответствующими выходами.

4 Ход лабораторной работы

Я решил не искать готовую реализацию алгоритма, а написать её самому.

Основная идея алгоритма

За основу я взял объяснение алгоритма с сайта BitcoinWiki (указан в списке литературы).

Keccak основан на конструкции Sponge. Это означает, что для получения хеша нужно проделать следующие незамысловатые действия: взять исходное сообщение M и дополнить его до длины кратной r. В виде формулы их можно изобразить следующим образом: M=M||0x01||0x00||…||0x00||0x80. То есть к сообщению дописывается единичный байт, необходимое количество нулей и завершается байт со значением 0x80.

Однако в случае, если необходимо дополнить всего один байт, то достаточно добавить лишь 0x81.

Затем для каждого блока Mi длиной r бит выполняем:

- Сложение по модулю 2 с первыми r-битами набора начальных состояний S. Перед началом работы функции все элементы S будут равны нулю.
- N раз применяем к полученным в результате данным функцию f. Набором начальных состояний S для блока Mi+1 будет результат последнего раунда блока Mi.
- После того как все блоки Mi закончатся взять итоговый результат и вернуть его в качестве хеш-значения.

В начале алгоритма внутреннее состояние S инициализируется нулями. Это состояние имеет фиксированный размер и состоит из двух частей: активной части (r) и пассивной части (c). Суммарный размер состояния S равен b = r + c, где для Keccak-256 b = 1600 бит.

Функция перестановки (keccak-f): Эта функция является сердцем алгоритма. Она принимает внутреннее состояние S и преобразует его через серию операций, описанных ниже. Количество раундов обычно фиксировано (для Keccak-256 это 24 раунда).

Theta (θ): Выравнивает влияние каждого бита на другие биты в его столбце и на биты в других столбцах.

Rho (ρ) и Pi (π): Изменяют позиции битов внутри состояния для разрушения структуры входных данных.

Chi (χ): Применяет нелинейное преобразование к каждому ряду состояния, усиливая криптографическую стойкость.

Iota (ι): Вводит раундовую константу в состояние для избежания симметрии и гарантии различных преобразований на каждом раунде.

После поглощения всех блоков сообщения, хеш выжимается из активной части состояния S. Если требуемая длина хеша больше, чем размер r, функция перестановки применяется снова, позволяя генерировать выходные данные неограниченной длины.

Модификация алгоритма и дифференциальный криптоанализ

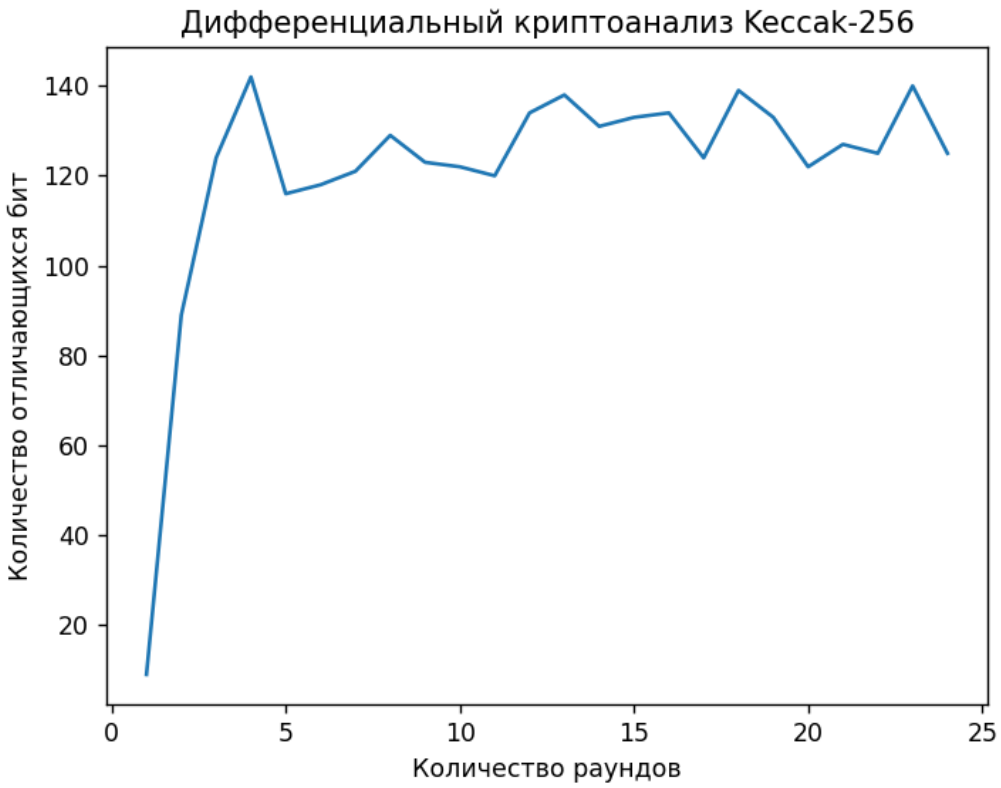
Файл kp.py:

```
import matplotlib.pyplot as plt
from keccak import keccak


def differential_analysis(message1, message2, r, round, output_length):
    hash1 = keccak(message1, r, round, output_length)
    hash2 = keccak(message2, r, round, output_length)

    diff_bits = sum(bin(b1 ^ b2).count('1') for b1, b2 in zip(hash1, hash2))
    return diff_bits

def main():
    r = 1088
    output_length = 32

    message1 = b"Hello, world!"
    message2 = b"Hello, world?"

    rounds = range(1, 25)
    differences = [differential_analysis(message1, message2, r, round,
output_length) for round in rounds]
```

```
    plt.plot(rounds, differences)
    plt.xlabel('Количество раундов')
    plt.ylabel('Количество отличающихся бит')
    plt.title('Дифференциальный криптоанализ Keccak-256')
    plt.show()

main()
```

Как видно, функция хеширования принимает также количество раундов на вход. Также я написал функцию, которая принимает два сообщения с небольшим отличием в битах, количество раундов для Keccak и возвращает разницу в выходных хэшах. Теперь применим эту функцию к различным количествам раундов (от 1 до 24), используя два входных сообщения с однобитовым отличием. Для каждого количества раундов запишем количество различных бит в результатах хэшей двух сообщений. На основе собранных данных построим график, показывающий зависимость количества отличающихся бит в выходных хэшах от количества раундов в Keccak.

График:



Все выводы касательно проведённого анализа я написал в блоке «Выводы».

5 Выводы

После окончания работы над курсовым проектом и анализом получившегося результата я выделил ряд особенностей:

- В начальных раундах количество отличающихся бит быстро увеличивается. Это происходит вследствие того, что уже после нескольких раундов вносятся существенные изменения в состояние хеша, что хорошо для криптостойкости, так как затрудняет анализ отношений между исходным сообщением и хешем.
- После начального увеличения количество изменяющихся бит достигает некоторого уровня и колеблется около него. Процесс достиг стабильности и внесение дополнительных раундов не приводит к значительному увеличению стойкости хеша.
- Существует некоторая неопределенность в количестве отличающихся бит в зависимости от раунда, что говорит о сложности и непредсказуемости влияния каждого раунда на хеш. Для криптографического алгоритма это вполне логичный исход, поскольку такое поведение затрудняет атаки.
- На графике видно, что после определённого количества раундов (в данном случае, начиная с 4-5 раундов) изменения становятся менее заметными, а значит, добавление дополнительных раундов после достижения определенного порога уже не приносит значительного увеличения безопасности.
- Можно сделать вывод о том, что определённое количество раундов (возможно, где-то в районе 12-14 раундов) может быть оптимальным балансом между безопасностью и производительностью для Keccak-256.

6 Список используемой литературы

Keccak – SHA-3 – Алгоритм хэширования – BitcoinWiki – https://bitcoinwiki.org/ru/wiki/sha-3

The Keccak reference – https://csrc.nist.gov/csrc/media/Projects/hash-functions/documents/Keccak-reference-3.0.pdf