Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Исследование криптографических хеш-функций

Студент: Коктыш Е. С.

ФИТ 3 курс 6 группа

Преподаватель: Нистюк О. А.

Минск 2024

1. **Теоретические сведения**

Хеш-функция – математическая или иная функция, h = H(М), которая принимает на входе строку символов М, называемую также прообразом, переменной длины n и преобразует ее в выходную строку фиксированной (обычно – меньшей) длины, l.

Хеширование (или хэширование, англ. hashing ) – это преобразование входного массива данных определенного типа и произвольной длины (практически) в выходную битовую строку фиксированной длины.

Преобразования называются хеш-функциями или функциями свертки, а их результаты называют хешем, хеш-кодом, хеш-таблицей или дайджестом сообщения .

Все существующие функции хеширования можно разделить на два больших класса:

* бесключевые хеш-функции, зависящие только от сообщения,
* хеш-функции с секретным ключом, зависящие как от сообщения, так и от секретного ключа.

Определение 3. Криптографическая хеш-функция – это специальный класс хеш-функций, который имеет различные свойства, необходимые для решения задач в области криптографии.

Основные задачи, решаемые с помощью хеш-функций:

* аутентификация (хранение паролей),
* проверка целостности данных,
* защита файлов,
* обнаружение зловредного ПО,
* криптовалютные технологии.

К основным свойствам хеш-функций можно отнести следющие.

Свойство 1. Детерминированность: независимо от того, сколько раз вычисляется H(M), M – const, при использовании одинакового алгоритма код хеш-преобразования h всегда должен быть одинаковым.

Свойство 2. Скорость вычисления хеша h: если процесс вычисления h не достаточно быстрый, система просто не будет эффективной.

Свойство 3: Сложность обратного вычисления: для известного H (М) невозможно (практически) определить М. Это важнейшее свойство хеш-функции для криптографических применений – односторонности преобразования.

Большинству даже простых пользователей известно, что наиболее распространенная область применения хеширования – хранение паролей. К примеру, если пользователь забыл пароль и пытается воспользоваться доступными функциями-сервисами восстановления пароля, то в этом случае, такой сервис может выдать пароль, как правило, в двух случаях:

* информационная система, в которую входит пользователь с помощью пароля, на самом деле, не хеширует этот пароль,
* используемая система восстановления пароля использует некоторую базу данных, содержащую простые, наиболее часто используемые пароли (например, 123456 или qwerty); примером такой системы восстановления пароля является Online Reverse Hash Lookup.

В плане односторонности хешей на основе блочных шифров отметим одно обстоятельство. Блочный шифр необратим по ключу шифрования, и, если в качестве такого ключа на текущем шаге преобразования используется выход предыдущего шага, а в качестве шифруемого сообщения – очередной блок сообщения (или наоборот), то можно получить хеш-функцию с хорошими криптографическими характеристиками с точки зрения односторонности.

Такой подход использовался, например, в российском стандарте хеширования – ГОСТ Р 34.11-94.

Основным недостатком хеш-функций на основе блочных шифров является сравнительно невысокая производительность.

Свойство 4. Даже минимальные изменения в хешируемых данных (М ≠ М') должны изменять хеш: Н(M) ≠ Н(М').

Определение 4. Коллизией хеш-функции Н называют ситуацию, при которой различным входам (в общем случае – х и у или М ≠ М') соответствует одинаковый хеш-код: H(x) = H(y) или H(М) = H(М').

Свойство 5. Коллизионная устойчивость (стойкость).

Зная М, трудно найти такое М' (М ≠ М'), для которого H(М) = H(М').

Если последнее равенство выполняется, то говорят о коллизии 1- го рода.

Если случайным образом выбраны два сообщения (М и М’), для которых H(М) = H(М'), говорят о коллизии 2-го рода.

Мерой криптостойкости хеш-функции считается вычислительная сложность нахождения коллизии.

Для хеш-функций одним из основных средств поиска коллизий является метод, основанный на известной статистической задаче – «парадоксе дня рождения».

В более общем случае: для того, чтобы хеш-функция H(M) считалась криптографически стойкой, она должна удовлетворять трем основным требованиям: необратимостью вычислений (свойство 3), устойчивостью к коллизиям первого рода и устойчивостью к коллизиям второго рода (свойство 5).

Хеш-функция – это функция, выполняющая отображение из множества М в число, находящееся в интервале в интервале [0, m–1]: h: M → [0, m–1].

Мы ранее отмечали, что стойкость хеш-преобразования к коллизии означает, что трудно найти такие Мi и Мj (Мi, Мj М), при которых h(Мi) = h(Мj), i ≠ j, 1 ≤ i, j ≤ n

Для выполнения анализа атаки на основе парадокса «дней рождения» будем использовать те же принципы, которые мы применяли для вероятностной оценки дней рождения.

В атаке «дней рождения» m соответствует количеству календарных дней в году, а М – множеству людей, составляющих группу. Люди «хешируются» в их дни рождения, которые могут быть одним из значений m.

Допустим (переходя в информационную область), нам нужно найти коллизию с вероятностью 0,99 (Рс(Аn) = 0.99). Мы хотим определить наименьшее n, при котором хеш двух значений из Аn будет «одним днем рождения», что в интересующей нас плоскости означает, что два входных набора данных (Мi, Мj М) хешируются в одинаковое значение: h(Мi) = h(Мj). Допустим далее, что все входные данные хешируются в m выходных хеш-кодов.

1. **Практическая часть**

В данной лабораторной работе необходимо разработать оконное приложение, реализующее алгоритм хеширования MD5. При этом можно воспользоваться доступными готовыми библиотеками.

**2.1. Шифрование**

Для данной лабораторной работы был выбран алгоритм MD5, из-за его высокой криптостойкости.

Функция принимает в себя один параметр – исходный текст, хэш которого необходимо получить. Затем получаем хэш сообщения. В метод необходимо передать сообщение в двоичном виде. Код приложения представлен в листинге 2.1.

|  |
| --- |
| function md5(s) {      tеxt = '';      var n = s.length,      state = [1732584193, -271733879, -1732584194, 271733878], i;      for (i = 64; i <= s.length; i += 64) {          md5cycle(state, md5blk(s.substring(i - 64, i)));      }      s = s.substring(i - 64);      var tail = [0,0,0,0, 0,0,0,0, 0,0,0,0, 0,0,0,0];      for (i = 0; i < s.length; i++) {          tail[i >> 2] |= s.charCodeAt(i) << ((i % 4) << 3);      }      tail[i >> 2] |= 0x80 << ((i % 4) << 3);      if (i > 55) {          md5cycle(state, tail);          for (i = 0; i < 16; i++) {              tail[i] = 0;          }      }      tail[14] = n \* 8;      md5cycle(state, tail);      return state} |

Листинг 2.1 – Функция MD5

Результат работы приложения приведен на рисунке 2.1.

|  |
| --- |
|  |

Рисунок 2.1. – Результаты хэширования сообщений «Koktysh»

Основное правило хеширования гласит, что разные строки должны образовывать разный хеш, результат работы приложения, подтверждающий это правило представлен на рисунке 2.2:

|  |
| --- |
|  |

Рисунок 2.2. – Результаты хэширования сообщений «koktysh»

**2.2. Время выполнения**

Код для реализации скорости выполнения алгоритма MD5 в зависимости от количества символов представлена в листинге 3.1.

|  |
| --- |
| const start20 = process.hrtime();      md5("12345678912345678912");      const end20 = process.hrtime(start20);      res.write(`Hash of 20 symbols took ${end20[1] / 1000000} milliseconds\n`);      const start60 = process.hrtime();      md5("123456789123456789121234567891234567891212345678912345678912");      const end60 = process.hrtime(start60);      res.write(`Hash of 60 symbols took ${end60[1] / 1000000} milliseconds\n`);      const start100 = process.hrtime();      md5("1234567891234567891212345678912345678912123456789123456789121234567891234567891212345678912345678912");      const end100 = process.hrtime(start100);      res.write(`Hash of 100 symbols took ${end100[1] / 1000000} milliseconds\n`); |

Листинг 2.1 – Функция MD5

Результат работы приложения приведен на рисунке 3.1.

|  |
| --- |
|  |

Рисунок 2.1. – Время выполнения хэширования разного количества символов

На рисунке 3.2 продемонстрирован график скорости выполнения алгоритма MD5 в зависимости от количества символов в исходном сообщении.

|  |
| --- |
|  |

Рисунок 3.2. – График зависимости времени выполнения от количества символов

Как показано на графике, время, необходимое для вычисления хеш-значения, остается примерно постоянным независимо от длины входных данных.

# Вывод

В данной лабораторной работе были закрепилены теоретические знания по алгебраическому описанию, алгоритмам реализации операций вычисления однонаправленных хэш-функций, разработано приложение для реализации алгоритма хеширования MD5, а также оценена скорость вычисления кодов хеш-функций.