Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет по лабораторной работе №11

«Исследование криптографических хеш-функций»

Студентка: Пунько А.А,

ФИТ 3 курс 5 группа

Преподаватель: Берников В. О.

Минск 2020

Цель: изучение алгоритмов хеширования и приобретение практических навыков их реализации и использования в криптографии

Задачи:

1.Закрепить теоретические знания по алгебраическому описанию и алгоритмам реализации операций вычисления однонаправленных хеш-функций.

2. Освоить методику оценки криптостойкости хэш-преобразований на основе «парадокса дня рождения».

3. Разработать приложение для реализации заданного алгоритма хеширования (из семейств MD и SHA).

4. Оценить скорость вычисления кодов хеш-функций.

5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

# 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. 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 (анг. message digest).

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:

1. аутентификация (хранение паролей),
2. проверка целостности данных,
3. защита файлов,
4. обнаружение зловредного ПО,
5. криптовалютные технологии.

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'). Мерой криптостойкости хеш-функции считается вычислительная сложность нахождения коллизии.

Основной постулат парадокса «дней рождения» гласит: в группе минимум из 23 человек с вероятностью более 0,5 день рождения у 2-х лиц одинаков. Парадоксом является высокая (как кажется на первый взгляд) вероятность наступления указанного события. При этом предполагается, что

* в этой группе нет близнецов;
* люди рождаются независимо друг от друга, т. е. дата (день) рождения любого человека не влияет на дату рождения другого;
* люди рождаются равномерно и случайно, т. е. люди с равной вероятностью могут рождаться в любой день года; с формальной точки зрения это означает, что вероятность р1 рождения отдельно выбранного члена группы (как и любого человека) в любой выбранный день равна р1 =1/365 (хотя известно, что в реальности рождение людей не совсем соответствует такому предположению).

Алгоритмы семейства MD-x (2/4/5/6) являются творениями Р. Ривеста; MD – Message Digest. Алгоритм MD6, в отличие от предыдущих версий алгоритма этого семейства, не стандартизован. Алгоритмы семейства SHA (SHA – Secure Hash Algorithm) являются в настоящее время широко распространенными. По существу во многих случаях завершился переход от SHA-1 к стандартам версии SHA-2. SHA-2 – собирательное название алгоритмов SHA-224, SHA256, SHA-384 и SHA-512. SHA-224 и SHA-384 являются, по сути, аналогами SHA-256 и SHA-512 соответственно. Известен также алгоритм хеширования, долгое время использовавшийся в качестве национального стандарта (ГОСТ 34.11-94) России.

С 5 августа 2015 г. утвержден и опубликован в качестве действующего стандарта (FIPS 202) алгоритм SHA-3, одной из отличительных особенностей которого является использование конструкции «криптографической губки». В этой конструкции реализован итеративный подход для создания функции с произвольной длиной на входе и произвольной длиной на выходе на основе определенного преобразования. На основе технологии «губки» построен ныне действующий в РБ стандарт хеширования.

Алгоритмы семейства MD входные сообщения максимальной длины 264-1 бит (в общем случае – L бит) преобразуют в хеш длиной l = 128 бит. Исключением является последняя – 6 – из версий алгоритма, где длина результирующего хеша может изменяться от 1 до 512 бит. Максимальный объем хешируемых сообщений для алгоритмов SHA-1, SHA-256, SHA-224 такой же, как и для алгоритмов MD. Однако длина хешей разная: в SHA-1 – 160 бит; во алгоритмах, относящихся к семейству SHA-2 – соответствует числу, дополняющему через дефис название алгоритма. Максимальная же длина входных сообщений в алгоритмах SHA-512, SHA-384, SHA-512/256, SHA-512/224 составляет 2128-1 бит. Базовые алгоритмы обоих рассматриваемых семейств (MD и SHA) условно можно разделить на 5 стадий:

1. расширение входного сообщения;
2. разбивка расширенного сообщения на блоки;
3. инициализация начальных констант;
4. обработка сообщения поблочно (основная процедура алгоритма хеширования);
5. вывод результата.

Входное сообщение «дополняется» (расширяется) так, чтобы его длина (в битах) была конгруэнтной к 448 по модулю 512. Это значит, что сообщение начальной длиной L бит расширяется так, что остаются незаполненными всего лишь 64 бита, чтобы итоговая длина L' была кратной 512. В указанные 64 бита записывается двоичная длина L. Расширение происходит всегда, даже если длина сообщения уже соответствует 448, по модулю 512. Эта операция выполняется следующим образом: один бит «1» добавляется к сообщению, а затем добавляются биты «0», так что длина в битах дополненного сообщения стала конгруэнтной 448 по модулю 512. Добавляется не менее одного бита, но не более 448 бит.

Основой рассматриваемых базовых алгоритмов является модуль, состоящий из циклических преобразований каждого 512-битного блока, который делится на подблоки длиной 32 либо 64 (в алгоритмах SHA-512, SHA-384, SHA-512/256, SHA-512/224) бита. При длине подблока в 16 бит каждый 512-битный блок должен состоять из 32 подблоков.

Как было отмечено выше, основная операция заключается в циклической (пораундовой или поэтапной) обработке 512-битных блоков. Таких циклов может быть 3 (как в MD-4), или 4 (как в MD-5), или более. В каждом цикле используется своя нелинейная функция (обычно обозначаемая по порядку F, G, H,…), зависящая от текущего состояния 4 (в MD), 5 (в SHA-1), 8 (SHA-256) и т. д. переменных, начальные состояния которых известны, а текущие – зависят от выполненных операций над хешируемым сообщением.

Главный модуль состоит из четырех похожих этапов (у MD-4 было только три этапа). На каждом этапе 16 раз используются различные операции. Каждая операция представляет собой нелинейную функцию над тремя из a, b, c и d. Затем она добавляет этот результат к четвертой переменной, подблоку текста Мj и константе ti. Далее результат циклически сдвигается вправо на переменное число s бит и добавляет результат к одной из переменных a, b, c и d. Наконец результат заменяет одну из этих переменных. Результатом хеширования, h, является конкатенация последних значений указанных переменных, т. е. 32\*4 = 128 бит.

Для SHA-1 цикл состоит из четырех этапов по 20 операций в каждом (в MD5 – 4 этапа по 16 операций в каждом). Каждая операция представляет собой нелинейную функцию над тремя из 5: a, b, c, d, e. Сдвиг и сложение – аналогично MD5.

Как мы заметили, в алгоритмах MD-5, SHA-1результат текущего действия прибавляется к результату предыдущего. Это направлено на усиление лавинного эффекта. Этой же цели служит то обстоятельство, что значения циклического сдвига влево на каждом этапе были приближенно оптимизированы: четыре сдвига, используемые на каждом этапе, отличаются от значений, используемых на других этапах.

# Практическая часть

В рамках данной лабораторной работе было разработано оконное приложение на языке C#, реализующее хэширование текста с использованием алгоритмов MD5 и SHA-256. Стартовое окно приложения показано на рисунке 1.

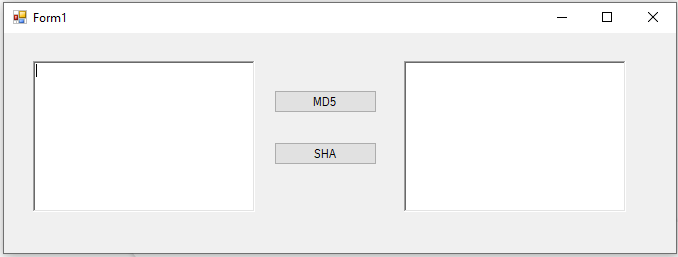


Рисунок 1 – Стартовое окно приложения

Результат хэширования ФИО с использованием алгоритма MD5 приведен на рисунке 2.

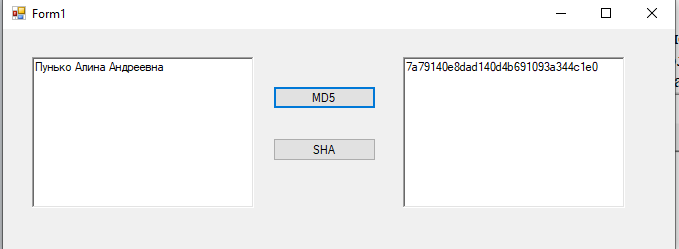


Рисунок 2 – Хэширование с использованием алгоритма MD5

Результат хэширования ФИО с использованием алгоритма SHA256 приведен на рисунке 3.

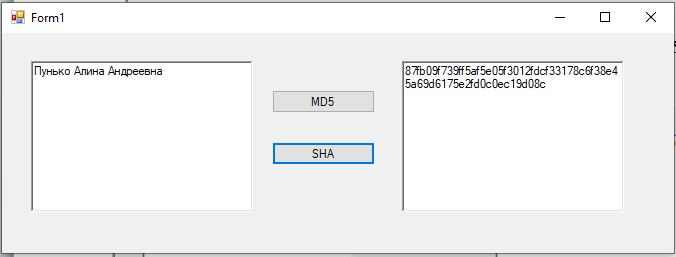


Рисунок 3 – Хэширование с использованием алгоритма SHA256

При разработке приложения использовалась встроенная библиотека System.Security.Cryptography, которая предоставляет готовые реализации многих алгоритмов шифрования и хэширования.

Реализация хэширования алгоритмом MD5 представлена в листинге 1.

static string GetMd5Hash(MD5 md5Hash, string input)

{

byte[] data = md5Hash.ComputeHash(Encoding.UTF8.GetBytes(input));

StringBuilder sBuilder = new StringBuilder();

foreach (var t in data)

{

sBuilder.Append(t.ToString("x2"));

}

return sBuilder.ToString();

}

Листинг 1 – Хэширование с использованием алгоритма MD5

Реализация хэширования алгоритмом SHA256 представлена в листинге 2.

static string ComputeSha256Hash(string rawData)

{

using (SHA256 sha256Hash = SHA256.Create())

{

byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));

StringBuilder builder = new StringBuilder();

for (int i = 0; i < bytes.Length; i++)

{

builder.Append(bytes[i].ToString("x2"));

}

return builder.ToString();

}

}

Листинг 2 – Хэширование с использованием алгоритма SHA256

# Вывод

В результате выполнения данной лабораторной работы были получены теоретические сведения о классификации и работе различных алгоритмов хэширования, а также было создано приложение, реализующее хэширование с использованием алгоритмов MD5 и SHA256.