Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

«Исследование криптографических хеш-функций»

Студент: Белицкий В.Д.

ФИТ 3 курс 5 группа

Преподаватель:

Савельева Маргарита Геннадьевна

Минск 2023

1. **Описание приложения**

Приложение написано на языке программирования C# и позволяет выполнить 2 задачи:

* хешировать входное сообщение двумя алгоритмами MD и SHA 256;
* оценивать скорость выполнения алгоритма;

1. **Методика выполнения поставленных задач**

Алгоритм хеширования MD5 состоит из 4 этапов, которые выполняются по 16 раз каждый. В каждом этапе происходит применение нелинейной функции к трем переменным: *a*, *b*, *c* и *d.* Результат функции затем добавляется к четвертой переменной, подблоку текста m и константе *t*. Далее полученный результат циклически сдвигается вправо на переменное число бит (*s*) и добавляется к одной из переменных *a, b, c* и *d*. В конце выполнения алгоритма, результат заменяет одну из указанных переменных. Конечный результат хеширования состоит из конкатенации последних значений этих четырех переменных, что образует 128-битный хеш. Результат работы приложения для данного алгоритма хеширования будет представлять собой полученный хеш. MD5 представлен на рисунке 2.1

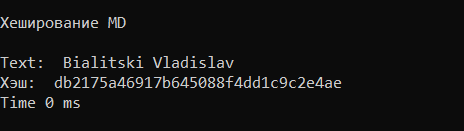


Рис. 2.1 – Реализация генерации тайного ключа

Были использованы библиотеки System.Security.Cryptography. Для хеширования были использованы функции изображены 2.2.

public static string ToHex(byte[] ba)

{

StringBuilder hex = new StringBuilder(ba.Length \* 2);

foreach (byte b in ba)

{

hex.AppendFormat("{0:x2}", b);

}

return hex.ToString();

}

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();

Рис. 2.2 – Реализация вычисления открытого ключа

SHA256 - это хеш-функция, входящая в семейство алгоритмов SHA-2. Она предназначена для создания дайджестов сообщений произвольной длины, а длина самого дайджеста составляет 256 бит. Процесс хеширования включает несколько этапов. Сначала исходное сообщение дополняется до достаточной длины, а затем разбивается на блоки, обычно по 512 бит. Каждый блок состоит из 16 слов.

Затем каждый блок проходит через 64 итерации, где на каждой итерации два слова подвергаются специальному преобразованию, а остальные слова определяют функцию преобразования. Результаты обработки каждого блока накапливаются и используются для формирования окончательного значения хеш-функции. Важно отметить, что из-за зависимости от предыдущего блока внутреннего состояния, обработка блоков не может выполняться параллельно.

Результат выполнения алгоритма SHA256 представлен на рисунке 2.3.

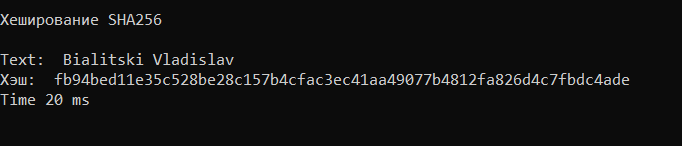


Рис. 2.3 – Результат алгоритма SHA256

Функция для хеширования с помощью алгоритма SHA256 представлена на рисунке 2.4.

public static string CreateSalt(int size)

{

var rng = new System.Security.Cryptography.RNGCryptoServiceProvider();

var buff = new byte[size];

rng.GetBytes(buff);

return Convert.ToBase64String(buff);

}

public static string GenerateSHA256(string input, string salt)

{

byte[] bytes = System.Text.Encoding.UTF8.GetBytes(input + salt);

System.Security.Cryptography.SHA256Managed sha256hashstring = new System.Security.Cryptography.SHA256Managed();

byte[] hash = sha256hashstring.ComputeHash(bytes);

return ToHex(hash);

Рисунок 2.4 – Функция хеширования SHA256

В алгоритмах хеширования, одним из распространенных концептов является использование соли. Соль представляет собой последовательность данных, которая передается вместе с исходными данными в хеш-функцию для вычисления хеша. Она применяется с целью усложнить процесс обнаружения прообраза хеш-функции через перебор возможных значений, включая такие атаки, как радужные атаки. Использование соли позволяет скрыть использование одинаковых прообразов при применении различных значений соли.

Таким образом, в ходе выполнения поставленных задач были успешно реализованы все требуемые цели. Были изучены исследованы асимметричные шифры, а также применение соли в хеш-функциях.

1. **Оценить быстродействие выбранного алгоритма хеширования.**

Рассмотрим рисунок 2.5, где несколько раз произведены функции хеширования. Сразу же видно, что SHA256 будет намного быстрее и эффективнее. Построим график.

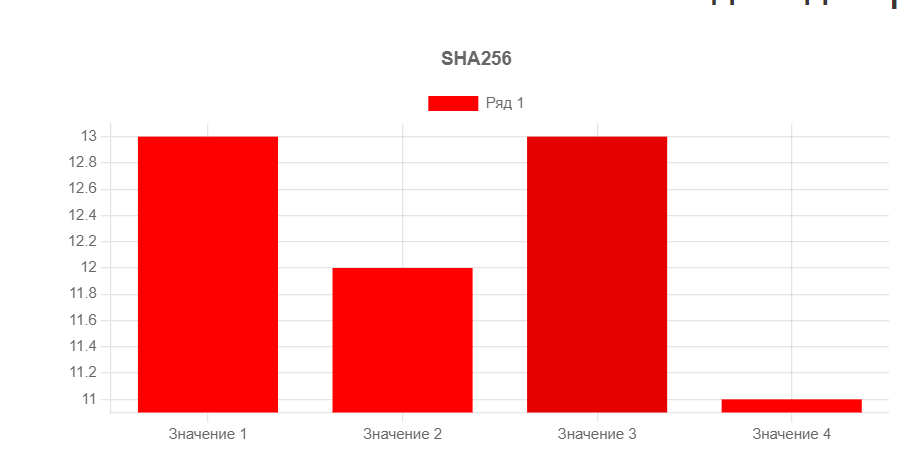


Рисунок 2.5 – График, описывающий скорость работы алгоритма SHA256

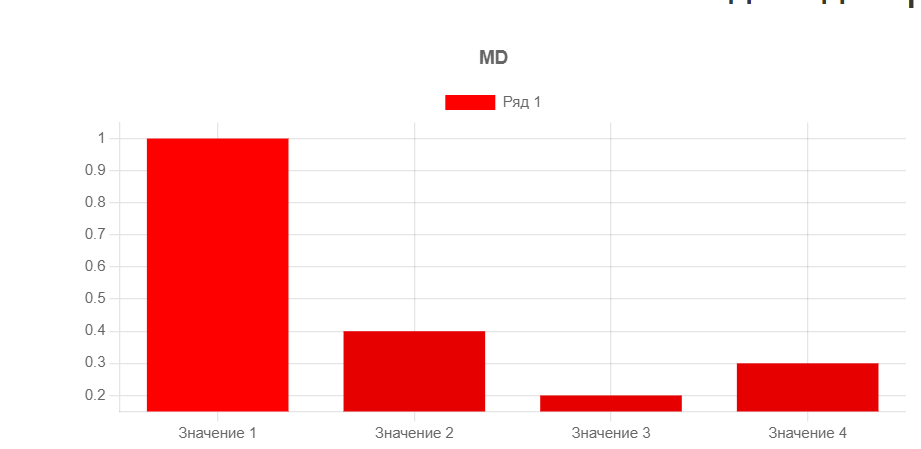


Рисунок 2.6 – График, описывающий скорость работы алгоритма MD

Алгоритм хеширования MD гораздо быстрее во всех испытаниях, по сравнения с SHA256. Однако SHA-256 обеспечивает более высокий уровень безопасности и стойкости к различным атакам, таким как коллизии и восстановление исходных данных. Это достигается за счет более сложных вычислительных операций, большей длины хеш-значения (256 бит) и лучших криптографических свойств.

**Вывод**

В ходе лабораторной работы были приобретены навыки хеширования используя алгоритмы MD5 и SHA256. Были изучены основные принципы работы хеширования.

Также было разработано приложение, на языке программирования C#, для реализации задач, связанных с хешированием данных используя алгоритмы SHA256 и MD.