Московский Государственный Университет

имени М. В. Ломоносова

Механико-математический факультет

Кафедра математических и компьютерных методов анализа

Дипломная работа

студента 601 группы

Медведева Егора Михайловича

Сравнительный анализ хеш-функций
Comparative analysis of hash functions

Научный руководитель

профессор, д.ф.-м.н. Чубариков В. Н.

Рецензент

ФИО РЕЦЕНЗЕНТА И ТД

Москва, 2018 год.

Содержание

1 Введение

2 Постановка задачи и формулировка основных результатов

Применение.

3 Основные и вспомогательные определения

В данной главе введем основные определения, которые будут использоваться в этой работе. Начнем с формального определения хеш-функции:

Пусть
$\{0,1\}^n$ - множество строк длины m, состоящих из битов 0 или 1.
$\{0,1\}^*$ - множество всех строк конечной длины, состоящих из битов 0 или 1.

Определение 1. Криптографической хеш-функцией h называется преобразование вида

$$h : \{0,1\}^* \to \{0,1\}^n$$

обладающее следующими свойствами:

1. Значение $h(M)$(где $M \in \{0,1\}^*$) должно "вычисляться легко"
2. При изменении всего лишь одного бита входного сообщения значение $h(M)$ меняет хотя бы половину своих битов
3. Прообраз для заданного $h(M)$ должен "вычисляться сложно" (pre-image resistance)
4. Второй прообраз $M^{'} \neq M$, такой что $h(M^{'}) = h(M)$, должен "вычисляться сложно" (second pre-image resistance)
5. Нахождение M и $M^{'} \neq M$, таких что $h(M) = h(M^{'})$, "вычисляется сложно" (collision resistance)

Замечание 1. Далее под "хеш-функцией" для удобства имеем в виду "криптографическую хеш-функцию".

Входную строку M будем называть "сообщением". Значение хеш-функции $h(M)$ будем называть "хешем", "хеш-кодом" или "хеш-суммой".

Под фразами "вычисляется сложно" или "вычислительно неразрешима" далее подразумеваем, что задача не решается за разумное время на современной вычислительной технике (очевидно, что задачи из свойств 2), 3) и 4) можно решить, например, полным перебором).

Пример 1. Пусть h - хеш-функция, определенная российским стандартом ГОСТ Р 34.11-2012 256 (ее описание см. далее). Ниже представлены результаты хеширования некоторых строк (результат представлен в шестнадцатеричной системе счисления):

1. $M =$ "Московский Государственный Университет имени М. В. Ломоносова" $\Rightarrow$ $h(M) =$ "2609a10022385596400318f6b959b9d449edbf7820ec188c7d8ddbc06a09ab0b"

2. $M =$ "МГУ им. М. В. Ломоносова" $\Rightarrow$
   $h(M) =$ "15414d11b2cbd98c858870463ed42189023845521f1bcb914817897c9f312d43"

3. $M =$ "МГУ им М. В. Ломоносова" $\Rightarrow$
   $h(M) =$ "4b54a14ab2320e4ed25e542410e424aad19ccc04fcd4debf46430efa6d972326"

Замечание 2. Отличие двух последних примеров состоит в присутствии и отсутствии знака точки: "им." и "им". Но, как следует из свойства 2) определения 1, хеши различаются очень сильно.

Следующее определение играет важную роль при построении хеш-функций.

Определение 2. Блочным шифром называются алгоритмы шифрования и расшифрования с одинаковым ключом $K \in \{0,1\}^m$ (то есть так называемый симметричный шифр), представляемые в виде функций E_K и D_K и являющимися для каждого фиксированного ключа биективным отображением на множестве $\{0,1\}^n$:

$$E_K(M) := E(K,M) : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^n,$$
$$D_K(C) := D(K,C) : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^n,$$

причем $D = E^{-1}$, то есть $\forall K$:

$$D_K(E_K(M)) = M \text{ и}$$
$$E_K(D_K(C)) = C.$$

4 Атака дней рождений

5 Принципы построения хеш-функций

В общем случае в основе построения хеш-функций лежит итеративная последовательная схема, когда на вход каждой итерации поступает блок исходного текста и результат предыдущей итерации. Ядром каждой итерации служит функция сжатия f, принимающая на вход блок определенной длины m и результат предыдущей итерации длины n, то есть:

$$f : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^n$$

Данная конструкция называется Структурой Меркла-Дамгарда, которая была придумана независимо Ральфом Мерклом и Иваном Дамгардом. Ими же было установлено, что если функция сжатия устойчива к коллизиям, то и хеш-функция будет также устойчива к коллизиям.

Опишем подробнее все шаги схемы:

Пусть $M \in \{0,1\}^*$ - исходное сообщение для хеширования.

1. Разобьем сообщение M на блоки $M_1, \ldots, M_s$ длины m.
2. Дополним входное сообщение заранее определенным образом (например, нулями), если длина M не кратна m.
3. Определить начальное значение H_0
4. i-й шаг итерации ($i = 1, \ldots, s$) заключается в вычислении значения $H_i = f(M_i, H_{i-1})$
5. Значение H_s (возможно, после некоторых дополнительных преобразований) и является конечным хешем сообщения M.

Определение 3. Структурой Меркла-Дамгарда называется приведенный выше алгоритм вычисления хеша (рис. 1).

картинка
картинка
картинка
картинка
картинка
картинка
картинка
картинка

Также существуют улучшенные схемы, основанные на структуре Меркла-Дамгарда:

- Структура Девиса-Мейера
- Структура Матиса-Мейера-Осеаса
- Структура Миагучи-Пренеля

Во всех этих схемах в качестве функции сжатия f используется блочный шифр E. Например, могут использоваться следующие популярные стандарты шифрования - DES, AES или ГОСТ 28147-89.
Ниже приведем более подробное описание каждой из схем.

5.1 Структура Девиса-Мейера

Как сказано выше, данная структура использует блочный шифр E. В качестве ключа шифр использует входной блок, а на вход подается результат предыдущей итерации (для первой итерации - это некоторое начальное значение). Затем для полученного результата выполняем операцию побитового сложения с значением предыдущей итерации (см. рис 2). Результат сложения и будет финальным значением итерации. Математически все это записывает так:

$$H_i = E_{M_i}(H_{i-1}) \oplus H_{i-1}$$

картинка
картинка
картинка
картинка
картинка

5.2 Структура Матиса-Мейера-Осеаса

Отличие этой структуры от предыдущей заключается в том, что входной блок и значение предыдущей итерации меняются местами. Однако получается несоответствие длин, так как длина ключа и длина выходного значения шифра имеют разные значения. В связи с этим надо иметь какое-то дополнительное преобразование g:

$$g : \{0,1\}^n \to \{0,1\}^m$$

Тогда перед выполнением шифра E применяем функцию g к результату предыдущей итерации и уже полученное значение используем в качестве ключа шифра.
Далее к результату шифра применяем операцию побитового сложения с входным блоком (см. рис. 3). Математически данные преобразования выглядят так:

$$H_i = E_{g(H_{i-1})}(M_i) \oplus M_i$$

картинка
картинка
картинка
картинка
картинка

5.3 Структура Миагучи-Пренеля

Эта структура считается самой популярной и надеждной из представленных здесь схем. Она аналогична предыдущей структуре за исключением того, что для операции побитового сложения добавляется еще одно слагаемое - результат предыдущей итерации. Таким образом, математически весь алгоритм записывается так:

$$H_i = E_{g(H_{i-1})}(M_i) \oplus M_i \oplus H_{i-1}$$

картинка
картинка
картинка
картинка
картинка

6 Описание алгоритмов конкретных хеш-функций

В данной главе приведено подробное описание следующих хеш-функций:

1. MD5 (сокращение от Message Digest 5)
2. ГОСТ Р 34.11-2012 (выходное значение - 512 бит)

6.1 Алгоритм MD5 вычисления хеш значения

1. Входное сообщение $M \in \{0,1\}^l (l \geq 0)$ представляется в виде

$$m_0 m_1 \dots m_{l-1}$$

   и дополняется битами до общей длины, сравнимой с 448 по модулю 512, следующим образом: добавляется бит "1", а затем остальные биты "0". Входное сообщение должно обязательно дополняться, даже если его длина кратна 448 по модулю 512.

2. Далее сообщение дополняется 64-мя битами - длиной исходного сообщения в битовом представлении. Если его длина сообщения больше, чем 2^{64}, то берутся первые 64 бита (меньшей разрядности). После двух первых шагов должны получить сообщение с длиной N, кратной 512.

3. Основная часть алгоритма использует так называемый md-буфер - четыре 32-х битных слова (A, B, C, D) со следующими начальными значениями (представлены в 16-тиричной системе счисления):

$$\begin{gathered} \text{A} = 01\ 23\ 45\ 67, \\ \text{B} = 89\ \text{AB}\ \text{CD}\ \text{EF}, \\ \text{C} = \text{FE}\ \text{DC}\ \text{BA}\ 98, \\ \text{D} = 76\ 54\ 32\ 10 \end{gathered}$$

4. Определим четыре функции:

$$F, G, H \text{ и } I(X,Y,Z) : \{0,1\}^{32} \times \{0,1\}^{32} \times \{0,1\}^{32} \to \{0,1\}^{32}$$

$$\begin{gathered} F(X,Y,Z) = X \wedge Y \vee \neg X \wedge Z \\ G(X,Y,Z) = X \wedge Z \vee Y \wedge \neg Z \\ H(X,Y,Z) = X \oplus Y \oplus Z \\ I(X,Y,Z) = Y \oplus (X \vee \neg Z) \end{gathered}$$

5. Инициализируем вспомогательный массив T[i] $, 0 \leq i \leq 63$:

$$T[i] = int(2^{32} * \left|sin(i)\right|)$$

6. Далее следует 4 раунда, описанные в псевдокоде ниже:

```
For i = 0 to N/16 - 1 do
    For j = 0 to 15 do
        X[j] := M[16*i + j]

    AA = A;      BB = B;      CC = C;      DD = D;

    // Раунд 1
    // Пусть [abcd k s i] обозначает следующее преобразование:
    // a = b + ((a + F(b, c, d) + X[k] + T[i]) < < < s)

    [ABCD  0  7  1]  [DABC  1  12   2]  [CDAB  2  17  3]  [BCDA  3  22  4]
    [ABCD  4  7  5]  [DABC  5  12   6]  [CDAB  6  17  7]  [BCDA  7  22  8]
    [ABCD  8  7  9]  [DABC  9  12 10]  [CDAB 10 17 11]  [BCDA 11 22 12]
    [ABCD 12 7 13]  [DABC 13 12 14]  [CDAB 14 17 15]  [BCDA 15 22 16]

    // Раунд 2
    // Пусть [abcd k s i] обозначает следующее преобразование:
    // a = b + ((a + G(b, c, d) + X[k] + T[i]) < < < s)

    [ABCD  1  5 17]  [DABC  6  9 18]  [CDAB  11  14  19]  [BCDA  0  20  20]
    [ABCD  5  5 21]  [DABC 10 9 22]  [CDAB  15  14  23]  [BCDA  4  20  24]
    [ABCD  9  5 25]  [DABC 14 9 26]  [CDAB 3 14 27]  [BCDA 8 20 28]
    [ABCD 13 5 29]  [DABC  2  9 30]  [CDAB 7 14 31]  [BCDA 12 20 32]

    // Раунд 3
    // Пусть [abcd k s i] обозначает следующее преобразование:
    // a = b + ((a + H(b, c, d) + X[k] + T[i]) < < < s)

    [ABCD  5  4  33]  [DABC  8  11   34]  [CDAB  11  16  35]  [BCDA  14  23  36]
    [ABCD  1  4  37]  [DABC  4  11   38]  [CDAB  7  16  39]  [BCDA  10  23  40]
    [ABCD  13  4  41]  [DABC  0  11 42]  [CDAB 3 16 43]  [BCDA 6 23 44]
    [ABCD 9 4 45]  [DABC 12 11 46]  [CDAB 15 16 47]  [BCDA 2 23 48]

    // Раунд 4
    // Пусть [abcd k s i] обозначает следующее преобразование:
    // a = b + ((a + I(b, c, d) + X[k] + T[i]) < < < s)

    [ABCD  0  6  49]  [DABC  7  10   50]  [CDAB  14  15  51]  [BCDA  5  21  52]
    [ABCD  12  6  53]  [DABC  3  10   54]  [CDAB  10  15  55]  [BCDA  1  21  56]
    [ABCD  8  6  57]  [DABC  15  10 58]  [CDAB 6 15 59]  [BCDA 13 21 60]
    [ABCD 4 6 61]  [DABC 11 10 62]  [CDAB 2 15 63]  [BCDA 9 21 64]

    A += AA;  B += BB;  C += CC;  D += DD;
```

7. Финальным результатом будет являться строка $A_1B_1C_1D_1$, где X_1 - это X начиная с младших бит. $X \in \{A, B, C, D\}$

Замечание 3. Операция $<<<$ обозначает циклический сдвиг битов влево.

6.2 Алгоритм ГОСТ Р 34.11-2012 вычисления хеш значения

1. Инициализируем вспомогательные переменные для алгоритма:

$$h := 0^{512} \in \{0,1\}^{512}$$
$$N := 0^{512} \in \{0,1\}^{512}$$
$$EPSILON := 0^{512} \in \{0,1\}^{512}$$

2. While $|M| \geq 512$:
   * Разложим сообщение M в виде M' || m, где m $\in \{0,1\}^{512}$ *
   h := round(h, m, N)
   N := N + 512 (mod 2^{512})
   EPSILON := EPSILON + 512 (mod 2^{512})
   M := M'

3. Дополним сообщение M, полученное на втором шаге, следующим образом: добавляется бит "1", а затем остальные биты "0" так, чтобы получилось сообщение длины 512. Сохраним полученное сообщение в m.
   h := round(h, m, N)
   N := N + $|M|\,(mod 2^{512})$
   EPSILON := EPSILON + $|m|\,(mod 2^{512})$
   h := round(h, N, 0)
   h := round(h, EPSILON, 0)

4. Полученное значение h и есть финальный результат хеш-функции.

Обозначения: Далее используются следующие обозначения:
$a \in \{0,1\}^n$ и $b \in \{0,1\}^n \Rightarrow$ a || b $\in \{0,1\}^{2n}$- их конкатенация.
$LP(x)$ - композиция преобразований L и P. P применяется первым.

Для определения функции round(h, m, N), которая используется в предыдущем алгоритме, нам понадобятся дополнительные преобразования:

- Преобразование X: побитовое исключающее ИЛИ (XOR) двух векторов одинаковой длины.
- Преобразование S:

$$S : \{0,1\}^{512} \to \{0,1\}^{512}$$
$$S(a) = S(a_{63}||\dots||a_0) = Pi(a_{63})||\dots||Pi(a_0)$$

- Преобразование P:

$$P : \{0,1\}^{512} \to \{0,1\}^{512}$$
$$P(a) = P(a_{63}||\dots||a_0) = a_{Tau(63)}||\dots||a_{Tau(0)}$$
Tau представлен в **Приложении А**

- Преобразование L:

$$L : \{0,1\}^{512} \to \{0,1\}^{512}$$
$$L(a) = L(a_7||\dots||a_0) = l(a_7)||\dots||l(a_0)$$

- Преобразование Pi:

$$Pi : \{0,1\}^{8} \to \{0,1\}^{8}$$

Входное битовое сообщение переводим в десятичную систему счисления, затем из массива Pi' (представлен в **Приложении А**) берем соответствующий элемент и переводим его обратно в двоичное представление.

- Преобразование l:

$$l : \{0,1\}^{64} \to \{0,1\}^{64}$$

Выбираем строки матрицы A (представлена в **Приложении А**) с такими номерами, что соответствующие этим номерам биты входящего сообщения имеют "1", и ко всем этим строкам применяем преобразование X.

Тогда функцию round(h, m, N)

$$\text{round(h, m, N)}: \{0,1\}^{512} \times \{0,1\}^{512} \times \{0,1\}^{512} \to \{0,1\}^{512}$$

определим следующим образом:

round(h, m, N) = X(X(E(LPSX(h, N), m), h), m), где
E(K, m) = $X(K_{13}, LPSX(K_{12}, \dots (K_2, LPSX(K_1, m))))$ и
$K_1 = K$,
$K_i = LPSX(K_{i-1}, C[i-1]), i = 2, \dots, 13$

7 Приложение А

Приведенные ниже константы/массивы используются в пункте 2 главы 6

- Tau = (0, 8, 16, 24, 32, 40, 48, 56, 1, 9, 17, 25, 33, 41, 49, 57, 2, 10, 18, 26, 34, 42, 50, 58, 3, 11, 19, 27, 35, 43, 51, 59, 4, 12, 20, 28, 36, 44, 52, 60, 5, 13, 21, 29, 37, 45, 53, 61, 6, 14, 22, 30, 38, 46, 54, 62, 7, 15, 23, 31, 39, 47, 55, 63)

- Pi' = (252, 238, 221, 17, 207, 110, 49, 22, 251, 196, 250, 218, 35, 197, 4, 77, 233, 119, 240, 219, 147, 46, 153, 186, 23, 54, 241, 187, 20, 205, 95, 193, 249, 24, 101, 90, 226, 92, 239, 33, 129, 28, 60, 66, 139, 1, 142, 79, 5, 132, 2, 174, 227, 106, 143, 160, 6, 11, 237, 152, 127, 212, 211, 31, 235, 52, 44, 81, 234, 200, 72, 171, 242, 42, 104, 162, 253, 58, 206, 204, 181, 112, 14, 86, 8, 12, 118, 18, 191, 114, 19, 71, 156, 183, 93, 135, 21, 161, 150, 41, 16, 123, 154, 199, 243, 145, 120, 111, 157, 158, 178, 177, 50, 117, 25, 61, 255, 53, 138, 126, 109, 84, 198, 128, 195, 189, 13, 87, 223, 245, 36, 169, 62, 168, 67, 201, 215, 121, 214, 246, 124, 34, 185, 3, 224, 15, 236, 222, 122, 148, 176, 188, 220, 232, 40, 80, 78, 51, 10, 74, 167, 151, 96, 115, 30, 0, 98, 68, 26, 184, 56, 130, 100, 159, 38, 65, 173, 69, 70, 146, 39, 94, 85, 47, 140, 163, 165, 125, 105, 213, 149, 59, 7, 88, 179, 64, 134, 172, 29, 247, 48, 55, 107, 228, 136, 217, 231, 137, 225, 27, 131, 73, 76, 63, 248, 254, 141, 83, 170, 144, 202, 216, 133, 97, 32, 113, 103, 164, 45, 43, 9, 91, 203, 155, 37, 208, 190, 229, 108, 82, 89, 166, 116, 210, 230, 244, 180, 192, 209, 102, 175, 194, 57, 75, 99, 182)

- A = 8e20faa72ba0b470 47107ddd9b505a38 ad08b0e0c3282d1c d8045870ef14980e 6c022c38f90a4c07 3601161cf205268d 1b8e0b0e798c13c8 83478b07b2468764 a011d380818e8f40 5086e740ce47c920 2843fd2067adea10 14aff010bdd87508 0ad97808d06cb404 05e23c0468365a02 8c711e02341b2d01 46b60f011a83988e 90dab52a387ae76f 486dd4151c3dfdb9 24b86a840e90f0d2 125c354207487869 092e94218d243cba 8a174a9ec8121e5d 4585254f64090fa0 accc9ca9328a8950 9d4df05d5f661451 c0a878a0a1330aa6 60543c50de970553 302a1e286fc58ca7 18150f14b9ec46dd 0c84890ad27623e0 0642ca05693b9f70 0321658cba93c138 86275df09ce8aaa8 439da0784e745554 afc0503c273aa42a d960281e9d1d5215 e230140fc0802984 71180a8960409a42 b60c05ca30204d21 5b068c651810a89e 456c34887a3805b9 ac361a443d1c8cd2 561b0d22900e4669 2b838811480723ba 9bcf4486248d9f5d c3e9224312c8c1a0 effa11af0964ee50 f97d86d98a327728 e4fa2054a80b329c 727d102a548b194e 39b008152acb8227 9258048415eb419d 492c024284fbaec0 aa16012142f35760 550b8e9e21f7a530 a48b474f9ef5dc18 70a6a56e2440598e 3853dc371220a247 1ca76e95091051ad 0edd37c48a08a6d8 07e095624504536c 8d70c431ac02a736 c83862965601dd1b 641c314b2b8ee083

  Матрица А состоит из 64-х строк и 16-ти столбцов. Для сокращения места, в одной текстовой строке представлены четыре строки матрицы А, которые разделены пробелами.

- Итерационные константы C[i]:

  C[1] = b1085bda1ecadae9ebcb2f81c0657c1f 2f6a76432e45d016714eb88d7585c4fc
  4b7ce09192676901a2422a08a460d315 05767436cc744d23dd806559f2a64507

  C[2] = 6fa3b58aa99d2f1a4fe39d460f70b5d7 f3feea720a232b9861d55e0f16b50131
  9ab5176b12d699585cb561c2db0aa7ca 55dda21bd7cbcd56e679047021b19bb7

  C[3] = f574dcac2bce2fc70a39fc286a3d8435 06f15e5f529c1f8bf2ea7514b1297b7b
  d3e20fe490359eb1c1c93a376062db09 c2b6f443867adb31991e96f50aba0ab2

  C[4] = ef1fdfb3e81566d2f948e1a05d71e4dd 488e857e335c3c7d9d721cad685e353f
  a9d72c82ed03d675d8b71333935203be 3453eaa193e837f1220cbebc84e3d12e

  C[5] = 4bea6bacad4747999a3f410c6ca92363 7f151c1f1686104a359e35d7800fffbd
  bfcd1747253af5a3dfff00b723271a16 7a56a27ea9ea63f5601758fd7c6cfe57

  C[6] = ae4faeae1d3ad3d96fa4c33b7a3039c0 2d66c4f95142a46c187f9ab49af08ec6
  cffaa6b71c9ab7b40af21f66c2bec6b6 bf71c57236904f35fa68407a46647d6e

  C[7] = f4c70e16eeaac5ec51ac86febf240954 399ec6c7e6bf87c9d3473e33197a93c9
  0992abc52d822c3706476983284a0504 3517454ca23c4af38886564d3a14d493

  C[8] = 9b1f5b424d93c9a703e7aa020c6e4141 4eb7f8719c36de1e89b4443b4ddbc49a
  f4892bcb929b069069d18d2bd1a5c42f 36acc2355951a8d9a47f0dd4bf02e71e

  C[9] = 378f5a541631229b944c9ad8ec165fde 3a7d3a1b258942243cd955b7e00d0984
  800a440bdbb2ceb17b2b8a9aa6079c54 0e38dc92cb1f2a607261445183235adb

  C[10] = abbedea680056f52382ae548b2e4f3f3 8941e71cff8a78db1fffe18a1b336103
  9fe76702af69334b7a1e6c303b7652f4 3698fad1153bb6c374b4c7fb98459ced

  C[11] = 7bcd9ed0efc889fb3002c6cd635afe94 d8fa6bbbebab07612001802114846679
  8a1d71efea48b9caefbacd1d7d476e98 dea2594ac06fd85d6bcaa4cd81f32d1b

  C[12] = 378ee767f11631bad21380b00449b17a cda43c32bcdf1d77f82012d430219f9b
  5d80ef9d1891cc86e71da4aa88e12852 faf417d5d9b21b9948bc924af11bd720