Лабораторна №1

Гешування

Мета: дослідити принципи роботи гешування

Завдання:

Дослідити існуючі механізми гешування. Реалізувати алгоритм гешування SHA (будь-якої версії). Довести коректність роботи реалізованого алгоритму шляхом порівняння результатів з існуючими реалізаціями.

Хід роботи:

1. Терміни які наведені у роботі:
    1. H[5] - хеш значення, початкові дані залежать від вибраного SHA алгоритму. З ітераціями змінюється за алгоритмом.
    2. M[N] - блоки у 512 бітів (або 1024) розраховується залежно від заданої строки для гешування. N - кількість блоків, теж залежить від довжини заданої строки.
    3. W[80] - слова, 32-бітні значення, перші 16 слів відповідають першим 16 підстрокам *M* розмірами в 32 біта.
    4. K[] - константи

```
private final int[] K = new int[]{
        0x5a827999,
        0x6ed9eba1,
        0x8f1bbcdc,
        0xca62c1d6
};
```

2. Логічні операції, які використовуються
    1. & - and
    2. | - or
    3. ~ - not
    4. ^ - xor
    5. Integer.rotateLeft() - циклічний сдвиг вліво

3. Для початку переводимо дану строку у бінарний вигляд, та розбиваємо на блоки:

    1. Кожну літеру переводимо у бінарний вигляд. Наприклад: "a" = 97 = 01100001. Не забуваємо додати нулі зліва, щоб довжина строки була = 8
    2. Отримані бінарні числа складаємо в одну довгу строку. Наприклад "abc" = 01100001 01100010 01100011.
    3. За алгоритмом до отриманої строки додаємо в кінець "1".
    4. Далі потрібно додати нулі і довжину утвореної бінарної строки так, щоб строка була рівна 512-бітам, щоб потім розбити на блоки. Для цього
        1. Довжина утвореної строки зберігається в кінці та займає 64 біта.
        2. Кількість нулів = 512 - 64 - 25. **512** - блок, **64** - стільки бітів займає довжина строки, **25** - утворена бінарна строка з доданою 1 в кінець.
        3. Отримаємо: 423 нуля потрібно додати
        4. Отримаємо M = 01100001 01100010 01100011 **1** 000...000 00..011000
        5. Дане повідомлення M = 512 бітів, за алгоритмом повідомлення повинно бути довжиною % 512 = 0. Тоді треба утворену M розбити на блоки M[N] де кількість блоків = M.len() / 512.

4. Підготуємо хеш значення. Для алгоритму SHA-1 вони мають константні значення.

```
public void setH() {
    H = new int[5];

    H[0] = 0x67452301;
    H[1] = 0xefcdab89;
    H[2] = 0x98badcfe;
    H[3] = 0x10325476;
    H[4] = 0xc3d2e1f0;
}
```

5. У циклі **i to M[N].len**. Тобто проходимось по блокам.
6. Йде розрахунок слів **W[80]**
    1. Створюємо цикл **t = 0 to 80**
    2. Якщо **t** менше 16, то беремо 32-бітне значення з **M[i]**
    3. Якщо **t** то 16, то використовуємо формулу *W[t - 3] ^ W[t - 8] ^ W[t - 14] ^ W[t - 16]*. Отримане значення циклічно сдвигаємо вліво на 1.

```
public void calculateW(int i, int t) {
    if (t < 16) {
        String m = Ms[i].substring(t * Integer.SIZE, (t * Integer.SIZE) + Integer.SIZE);

        try {
            W[t] = Integer.parseInt(m, 2);
        } catch (Exception e) {
            W[t] = Integer.parseInt("-" + m, 2);

        }
        System.out.println(String.format("W[%d]:\t%8s", t, Integer.toHexString(W[t])).replace(' ',
    } else {
        W[t] = Integer.rotateLeft(W[t - 3] ^ W[t - 8] ^ W[t - 14] ^ W[t - 16], 1);
    }
}
```

6. **H[0..4]** значення переписуємо у змінні **a, b, c, d, e**.
7. Далі у циклі **t = 0 to 80** творимо магію

Де ROTL - циклічний сдвиг вліво, на скільки написано над буквою L. Також є невідома функція *f(b, c, d)*. Вона виконує наступне залежно від **t**:

```
    public int ft(int x, int y, int z, int t) {
        if (t < 20) {
            return (x & y) ^ (~x & z);
        } else if (t < 40) {
            return x ^ y ^ z;
        } else if (t < 60) {
            return (x & y) ^ (x & z) ^ (y & z);
        } else {
            return x ^ y ^ z;
        }
    }
```

8. Нижче наведено кожний крок *магії*

```
            a               b               c               d               e
t = 0:   116fc33    67452301    7bf36ae2    98badcfe    10325476
t = 1:  8990536d     116fc33    59d148c0    7bf36ae2    98badcfe
t = 2:  a1390f08    8990536d    c045bf0c    59d148c0    7bf36ae2
t = 3:  cdd8e11b    a1390f08    626414db    c045bf0c    59d148c0
t = 4:  cfd499de    cdd8e11b    284e43c2    626414db    c045bf0c
t = 5:  3fc7ca40    cfd499de    f3763846    284e43c2    626414db
t = 6:  993e30c1    3fc7ca40    b3f52677    f3763846    284e43c2
t = 7:  9e8c07d4    993e30c1     ff1f290    b3f52677    f3763846
t = 8:  4b6ae328    9e8c07d4    664f8c30     ff1f290    b3f52677
t = 9:  8351f929    4b6ae328    27a301f5    664f8c30     ff1f290
t = 10: fbda9e89    8351f929    12dab8ca    27a301f5    664f8c30
t = 11: 63188fe4    fbda9e89    60d47e4a    12dab8ca    27a301f5
t = 12: 4607b664    63188fe4    7ef6a7a2    60d47e4a    12dab8ca
t = 13: 9128f695    4607b664    18c623f9    7ef6a7a2    60d47e4a
t = 14: 196bee77    9128f695    1181ed99    18c623f9    7ef6a7a2
t = 15: 20bdd62f    196bee77    644a3da5    1181ed99    18c623f9
t = 16: 4e925823    20bdd62f    c65afb9d    644a3da5    1181ed99
t = 17: 82aa6728    4e925823    c82f758b    c65afb9d    644a3da5
t = 18: dc64901d    82aa6728    d3a49608    c82f758b    c65afb9d
t = 19: fd9e1d7d    dc64901d    20aa99ca    d3a49608    c82f758b
t = 20: 1a37b0ca    fd9e1d7d    77192407    20aa99ca    d3a49608
t = 21: 33a23bfc    1a37b0ca    7f67875f    77192407    20aa99ca
t = 22: 21283486    33a23bfc    868dec32    7f67875f    77192407
t = 23: d541f12d    21283486     ce88eff    868dec32    7f67875f
t = 24: c7567dc6    d541f12d    884a0d21     ce88eff    868dec32
t = 25: 48413ba4    c7567dc6    75507c4b    884a0d21     ce88eff
t = 26: be35fbd5    48413ba4    b1d59f71    75507c4b    884a0d21
t = 27: 4aa84d97    be35fbd5    12104ee9    b1d59f71    75507c4b
t = 28: 8370b52e    4aa84d97    6f8d7ef5    12104ee9    b1d59f71
t = 29: c5fbaf5d    8370b52e    d2aa1365    6f8d7ef5    12104ee9
t = 30: 1267b407    c5fbaf5d    a0dc2d4b    d2aa1365    6f8d7ef5
t = 31: 3b845d33    1267b407    717eebd7    a0dc2d4b    d2aa1365
t = 32:  46faa0a    3b845d33    c499ed01    717eebd7    a0dc2d4b
t = 33: 2c0ebc11     46faa0a    cee1174c    c499ed01    717eebd7
t = 34: 21796ad4    2c0ebc11    811bea82    cee1174c    c499ed01
t = 35: dcbbb0cb    21796ad4    4b03af04    811bea82    cee1174c
```

```
t = 35: dcbbb0cb    21796ad4    4b03af04    811bea82    cee1174c
t = 36:  f511fd8    dcbbb0cb     85e5ab5    4b03af04    811bea82
t = 37: dc63973f     f511fd8    f72eec32     85e5ab5    4b03af04
t = 38: 4c986405    dc63973f     3d447f6    f72eec32     85e5ab5
t = 39: 32de1cba    4c986405    f718e5cf     3d447f6    f72eec32
t = 40: fc87dedf    32de1cba    53261901    f718e5cf     3d447f6
t = 41: 970a0d5c    fc87dedf    8cb7872e    53261901    f718e5cf
t = 42: 7f193dc5    970a0d5c    ff21f7b7    8cb7872e    53261901
t = 43: ee1b1aaf    7f193dc5    25c28357    ff21f7b7    8cb7872e
t = 44: 40f28e09    ee1b1aaf    5fc64f71    25c28357    ff21f7b7
t = 45: 1c51e1f2    40f28e09    fb86c6ab    5fc64f71    25c28357
t = 46: a01b846c    1c51e1f2    503ca382    fb86c6ab    5fc64f71
t = 47: bead02ca    a01b846c    8714787c    503ca382    fb86c6ab
t = 48: baf39337    bead02ca    2806e11b    8714787c    503ca382
t = 49: 120731c5    baf39337    afab40b2    2806e11b    8714787c
t = 50: 641db2ce    120731c5    eebce4cd    afab40b2    2806e11b
t = 51: 3847ad66    641db2ce    4481cc71    eebce4cd    afab40b2
t = 52: e490436d    3847ad66    99076cb3    4481cc71    eebce4cd
t = 53: 27e9f1d8    e490436d    8e11eb59    99076cb3    4481cc71
t = 54: 7b71f76d    27e9f1d8    792410db    8e11eb59    99076cb3
t = 55: 5e6456af    7b71f76d     9fa7c76    792410db    8e11eb59
t = 56: c846093f    5e6456af    5edc7ddb     9fa7c76    792410db
t = 57: d262ff50    c846093f    d79915ab    5edc7ddb     9fa7c76
t = 58:  9d785fd    d262ff50    f211824f    d79915ab    5edc7ddb
t = 59: 3f52de5a     9d785fd    3498bfd4    f211824f    d79915ab
t = 60: d756c147    3f52de5a    4275e17f    3498bfd4    f211824f
t = 61: 548c9cb2    d756c147    8fd4b796    4275e17f    3498bfd4
t = 62: b66c020b    548c9cb2    f5d5b051    8fd4b796    4275e17f
t = 63: 6b61c9e1    b66c020b    9523272c    f5d5b051    8fd4b796
t = 64: 19dfa7ac    6b61c9e1    ed9b0082    9523272c    f5d5b051
t = 65: 101655f9    19dfa7ac    5ad87278    ed9b0082    9523272c
t = 66:  c3df2b4    101655f9     677e9eb    5ad87278    ed9b0082
t = 67: 78dd4d2b     c3df2b4    4405957e     677e9eb    5ad87278
t = 68: 497093c0    78dd4d2b     30f7cad    4405957e     677e9eb
t = 69: 3f2588c2    497093c0    de37534a     30f7cad    4405957e
t = 70: c199f8c7    3f2588c2    125c24f0    de37534a     30f7cad
t = 71: 39859de7    c199f8c7    8fc96230    125c24f0    de37534a
t = 72: edb42de4    39859de7    f0667e31    8fc96230    125c24f0
t = 73: 11793f6f    edb42de4    ce616779    f0667e31    8fc96230
t = 74: 5ee76897    11793f6f    3b6d0b79    ce616779    f0667e31
t = 75: 63f7dab7    5ee76897    c45e4fdb    3b6d0b79    ce616779
t = 76: a079b7d9    63f7dab7    d7b9da25    c45e4fdb    3b6d0b79
t = 77: 860d21cc    a079b7d9    d8fdf6ad    d7b9da25    c45e4fdb
t = 78: 5738d5e1    860d21cc    681e6df6    d8fdf6ad    d7b9da25
t = 79: 42541b35    5738d5e1    21834873    681e6df6    d8fdf6ad
```

9. **a, b, c, d, e** складаємо з **H[0..4]** та записуємо у **H[0..4]**.
10. І нарешті цифри пройшовши такий довгий шлях можуть поєднатися в одну цілу загешовану строку.

```
H[0] || H[1] || H[2] || H[3] || H[4]
```

Результат

```
abc => a9993e364706816aba3e25717850c26c9cd0d89d
```

Перевірка з онлайн калькулятором. http://www.sha1-online.com/

1. Спробуємо строку "abc"
2. Спробуємо строку "abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq"
3. Спробуємо строку "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum luctus hendrerit ligula porta maximus. Sed."
4. Що до трохи більше символів? "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer luctus ante magna, id maximus libero vestibulum eget. Curabitur ullamcorper arcu quis felis fringilla congue. Morbi vitae semper odio. Sed egestas mauris iaculis orci consectetur commodo. Quisque vulputate, velit nec semper congue, elit urna hendrerit dui, pulvinar iaculis ligula dui id ligula. Cras malesuada tristique mauris, id convallis purus varius et. Phasellus dignissim ante auctor, ullamcorper quam sit amet, sagittis nulla. Donec sed lacinia nisi. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce fringilla tortor eu velit malesuada mattis. Nam quis velit consequat, convallis urna et, lobortis ante. Nam a augue eu metus ullamcorper lacinia. Quisque sit amet ligula felis. Nullam sagittis arcu ut molestie fringilla. In ac ex porta, dictum purus a."