Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

“Криптография и информационная безопасность”

Индивидуальная работа №2

SHA 1

Преподаватель: O.Cerbu

Выполнил: Маруневич Николай группа I1902

Кишинев 2022

Алгоритм SHA-1 называется безопасным, так как на его основе невозможно вычислить сообщение, которому соответствует дайджест, или найти другое сообщение, которое может соответствовать полученному дайджесту. Любое изменение сообщения при передаче приведет с большой вероятностью к существенному изменению дайджеста, что исключит успешную сверку цифровой подписи.

В разделе 2 определены терминология и функции, использованные при построении формы SHA-1.

**2. Определение битовых строк и целых**

В описании используются обозначения для битовых последователей и целых чисел:

1. Шестнацатеричные цифры - это {0, 1, ... , 9, A, ... , F}. Шестнацатеричные числа A представляются в виде 4-битовых строк. Примеры: 7 = 0111, A = 1010.
2. Слово соответствует 32-битной строке, которая может быть представлена последовательностью из 8 hex цифр. Чтобы преобразовать слово в 8 hex-цифр, каждая 4-битная строка преобразуется в ее hex-эквивалент, как это описано выше в (a). Пример:  
   1010 0001 0000 0011 1111 1110 0010 0011 = A103FE23.
3. Целое между 0 и 232 - 1 включительно может рассматриваться как слово. Младшие четыре бита этого целого представляют собой самую правую hex-цифру слова. Например: целое 291 = 28+25+21+20 = 256+32+2+1 представляется hex-словом, 00000123.

Если z является целым, 0 <= z < 264, тогда z = (232)x + y где 0 <= x < 232 и 0 <= y < 232. Так как x и y могут быть представлены как слова X и Y, соответственно, z может быть представлен парой слов (X,Y).

1. Блок = 512-битовая строка. Блок (напр., B) может быть представлен в виде последовательности из 16 слов.

**3. Операция над словами**

Со словами будут производиться следующие операции:

1. Побитовые логические операции над словами

X AND Y = побитовая логическая "and" X и Y.

X OR Y = побитовая логическая "inclusive-or" X и Y.

X XOR Y = побитовая логическая "exclusive-or" X и Y.

NOT X = побитовая логическая "complement" X.

Пример:

01101100101110011101001001111011

XOR 01100101110000010110100110110111

--------------------------------

= 00001001011110001011101111001100

1. Операция X + Y определяется следующим образом: слова X и Y представляют собой целые x и y, где 0 <= x < 232 и 0 <= y < 232. Для положительных целых n и m, пусть n mod m будет остатком от деления n на m.

Вычислим z = (x + y) mod 232.

Затем 0 <= z < 232. Преобразуем z в слово, Z, и определяем Z = X + Y.

1. Операция циклического сдвига влево Sn(X), где X является словом и n - целое 0 <= n < 32, определяется как

Sn(X) = (X << n) OR (X >> 32-n).

В выше приведенном примере X << n был получен следующим образом: удаляются левые n бит X и затем результат дополняется n нулями справа (результат будет иметь 32 бита). X >> n получается удалением правых n бит X и затем дополнением результата нулями слева. Таким образом Sn(X) циклическому сдвигу X на n позиций влево.

**4. Дополнение сообщений**

SHA-1 используется для вычисления дайджеста сообщения или информационного файла, который поступает на вход. Сообщение или информационный файл должны рассматриваться как битовая строка. Длина сообщения является числом бит в сообщении (пустое сообщение имеет длину 0). Если число бит в сообщении кратно 8, ради компактности будем представлять сообщение в hex-формате. Целью дополнения сообщения является желание сделать полную длину сообщения кратной 512. В процессе расчета дайджеста сообщения SHA-1 последовательно обрабатывает блоки последовательность 512 бит. Далее специфицируется, как осуществляется дополнение исходного сообщения. Добавляется 1, за которой следует m "0" и 64-битное целое, чтобы получить сообщение с длиной, кратной 512. 64-битное целое характеризует длину исходного сообщения. Дополненное сообщение затем обрабатывается SHA-1 как n 512-битных блоков.

Предположим, что сообщение имеет длину l < 264. Прежде чем оно будет подано на вход SHA-1, сообщение дополняется справа следующим образом:

* 1. Добавляем к сообщению "1". Например: если исходное сообщение "01010000", то после дополнения получаем "010100001".
  2. Добавляются "0". Число нулей зависит от длины исходного сообщения. Последние 64 бита последнего 512-битного блока зарезервированы для длины l исходного сообщения.

Пример: Предположим, что оригинальное сообщение является строкой бит

01100001 01100010 01100011 01100100 01100101.

После шага (a) это дает

01100001 01100010 01100011 01100100 01100101 1.

Так как l = 40, число бит в строке вше равно 41 и добавляется 407 "0", доводя длину сообщения до 448. Это дает в шестнацатеричном представлении

61626364 65800000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000.

* 1. Получаем 2-словное представление l, числа бит в исходном сообщении. Если l < 232, тогда первое слово будет содержать нули. Добавляем эти два слова к дополненному сообщению.

Пример: Предположим, что мы имеем исходное сообщение (b). Тогда l = 40 (заметим, что l вычисляется до выполнения дополнений). Представление двух слов 40 в hex-формате 00000000 00000028. Следовательно дополненное сообщение в окончательном hex-формате будет иметь вид

61626364 65800000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000028.

Дополненное сообщение будет содержать 16 \* n слов для некоторого n > 0. Дополненное сообщение рассматривается как последовательность из n блоков M(1), M(2), первые символы (или биты) сообщения.

**5. Используемые функции и константы**

В SHA-1 используется последовательность логических функций f(0), f(1),..., f(79). Каждая f(t), 0 <= t <= 79, выполняет операции над тремя 32-битовыми словами B, C, D и выдает на выходе 32-битовое слово. f(t;B,C,D) определяется следующим образом: для слов B, C, D,

f(t;B,C,D) = (B AND C) OR ((NOT B) AND D) ( 0 <= t <= 19)

f(t;B,C,D) = B XOR C XOR D (20 <= t <= 39)

f(t;B,C,D) = (B AND C) OR (B AND D) OR (C AND D) (40 <= t <= 59)

f(t;B,C,D) = B XOR C XOR D (60 <= t <= 79).

В SHA-1 используется последовательность слов K(0), K(1), ... , K(79). В hex-формате получается

K(t) = 5A827999 ( 0 <= t <= 19)

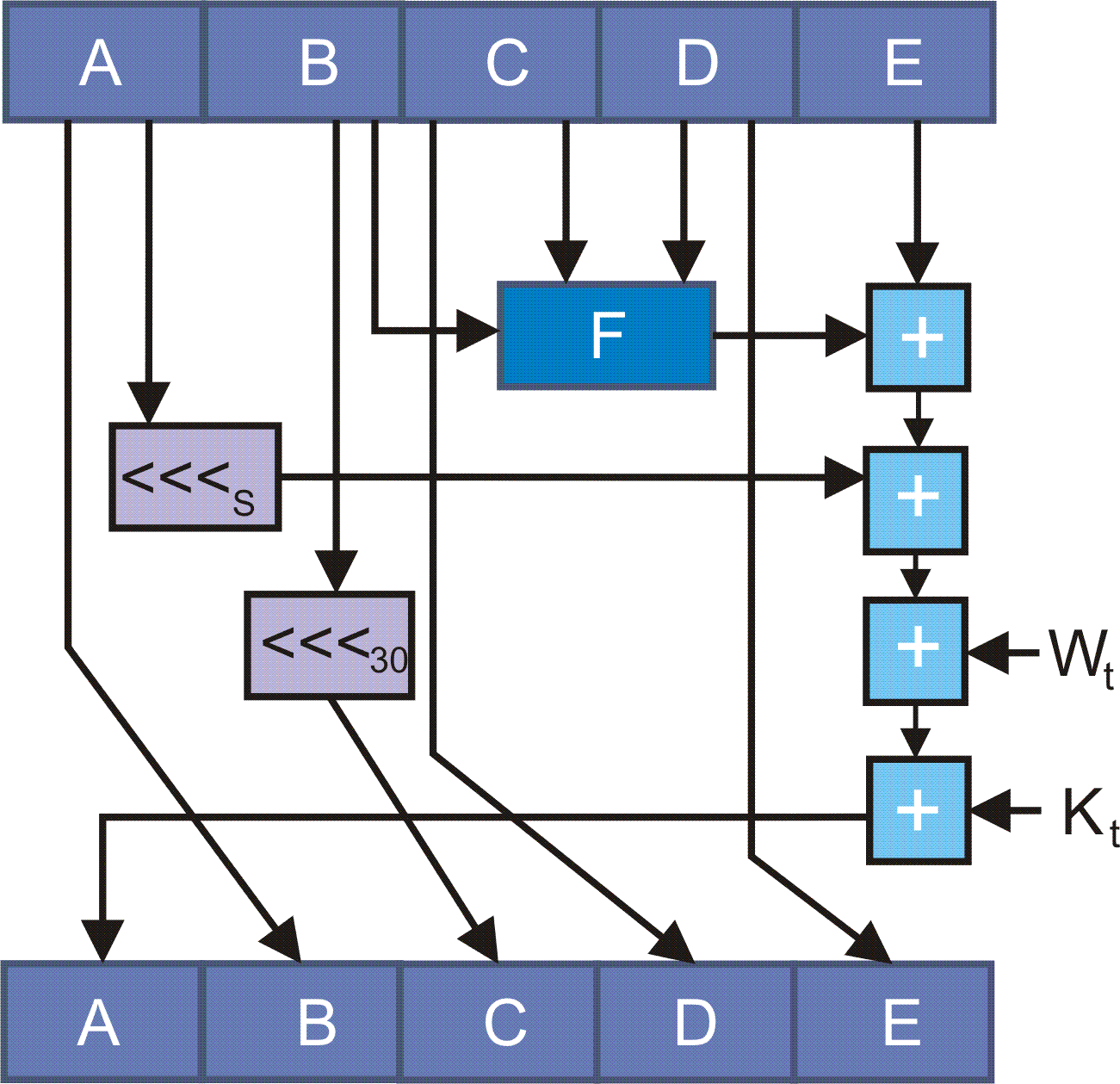
K(t) = 6ED9EBA1 (20 <= t <= 39)

K(t) = 8F1BBCDC (40 <= t <= 59)

K(t) = CA62C1D6 (60 <= t <= 79).

**6. Вычисление дайджеста сообщения**

На рисунке представлена схема реализации алгоритма SHA1. Символ "+" на рисунке указывает на операцию сложения по модулю 232, F - нелинейная функция, <<<n - обозначает сдвиг кода влево на n разрядов, Wt и Kt - константы, а A,B,C,D,E - 32-битовые слова.



Методы, представленные в разделах 6.1 и 6.2, выдают идентичные дайджесты сообщения. Хотя, использование метод 2 позволяет сэкономить 64 32-битовых слов в памяти, он вероятно удлинит время исполнения из-за усложнения вычисления адресов для { Wа[t] } на шаге (c). Существуют другие методы вычисления, которые дают идентичные результаты.

### 6.1. Метод 1

Дайджест сообщения вычисляется с использованием дополненного сообщения, как это описано в разделе 4. Вычисление описано для случая двух буферов, каждый из которых содержит пять 32-битных слов, и последовательность из восьмидесяти 32-битовых слов. Слова первого 5-словного буфера помечены как A,B,C,D,E. Слова второго 5-словного буфера помечены как H0, H1, H2, H3, H4. Слова массива из 80 слов помечены, как W(0), W(1),..., W(79). Используется также буфер для одного слова TEMP.

Чтобы сгенерировать дайджест сообщения, блоки из 16-слов M(1), M(2),..., M(n), определенные в разделе 4, обрабатываются последовательно. Обработка каждого M(i) включает в себя 80 шагов.

Прежде чем обрабатывать какой-либо блок, все H инициализируются следующим образом: в hex-представлении:

H0 = 67452301

H1 = EFCDAB89

H2 = 98BADCFE

H3 = 10325476

H4 = C3D2E1F0.

Теперь обрабатываются M(1), M(2), ... , M(n). Чтобы обработать M(i), мы выполняем следующее:

* 1. Разделяем M(i) на 16 слов W(0), W(1), ... , W(15), где W(0) является самым левым словом.
  2. Для t = 16 до 79 пусть

W(t) = S^1(W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16)).

* 1. Пусть A = H0, B = H1, C = H2, D = H3, E = H4.
  2. Для t = 0 to 79 do

TEMP = S^5(A) + f(t;B,C,D) + E + W(t) + K(t);

E = D; D = C; C = S30(B); B = A; A = TEMP;

* 1. Пусть H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E.

После обработки M(n), дайджест сообщения является 160-битовой строкой, в виде 5 слов:

H0 H1 H2 H3 H4.

### 6.2 Метод 2

Предыдущий метод предполагает, что последовательность W(0), ... , W(79) представляет собой массив из 80 32-битных слов. Это эффективно с точки зрения минимизации времени вычисления, так как адреса W(t-3), ... ,W(t-16) на этапе (b) легче вычислять. Когда имеется дефицит памяти, можно для { W(t) } организовать кольцевой буфер, который может использовать массив из 16 32-битных слов W[0], ... W[15]. В этом случае в hex-формате получим: MASK = 0000000F. Затем обрабатываем M(i) следующим образом:

* 1. Делим M(i) на 16 слов W[0], ... , W[15], где W[0] - самое левое слово.
  2. Пусть A = H0, B = H1, C = H2, D = H3, E = H4.
  3. Для t = 0 до 79 выполнить

s = t AND MASK;

если (t >= 16) W[s] = S^1(W[(s + 13) AND MASK] XOR W[(s + 8) AND

MASK] XOR W[(s + 2) AND MASK] XOR W[s]);

TEMP = S^5(A) + f(t;B,C,D) + E + W[s] + K(t);

E = D; D = C; C = S^30(B); B = A; A = TEMP;

* 1. Пусть H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E.