КиберХакерс

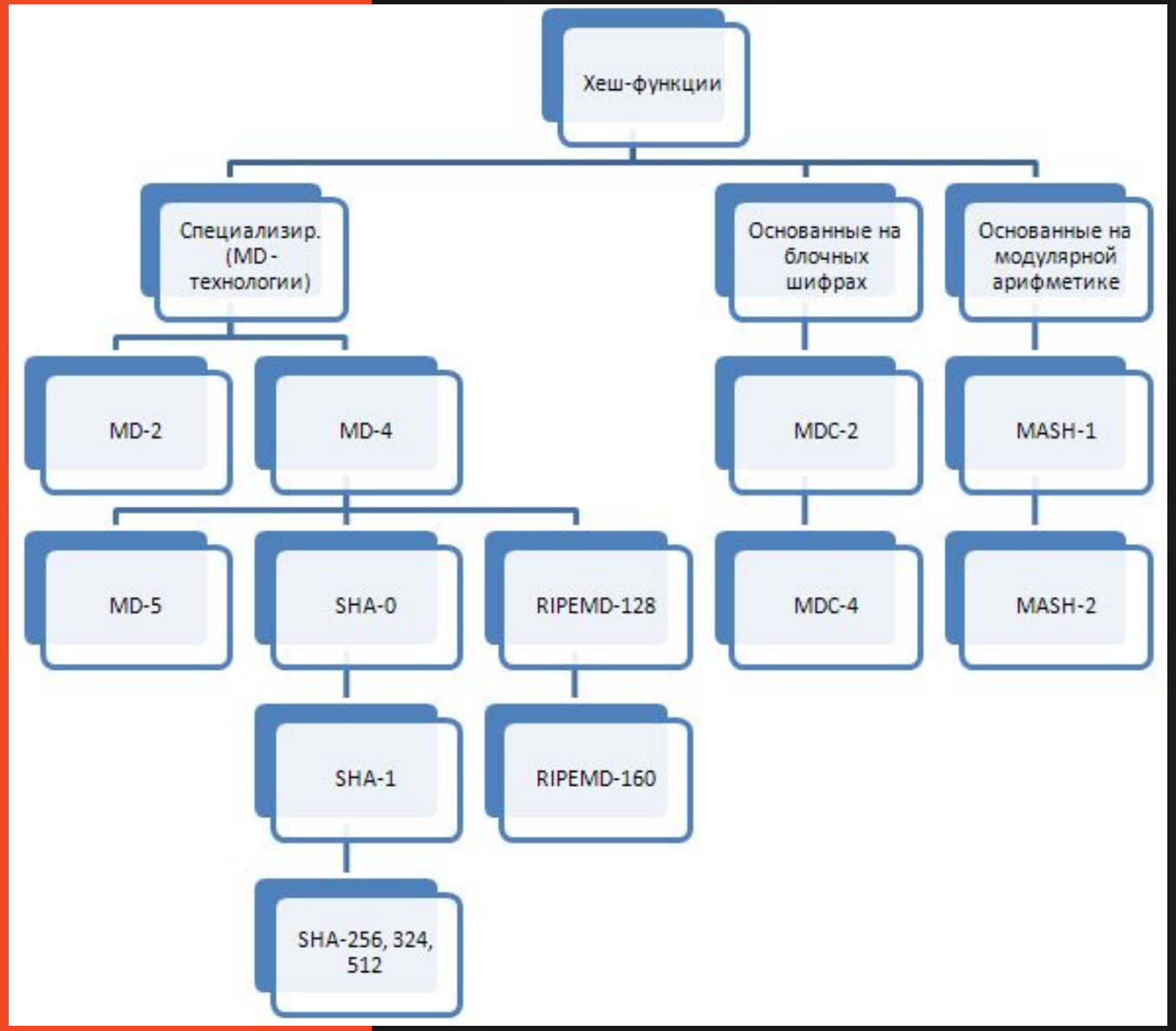
Хэш функции

Лектор: Комогоров Кирилл

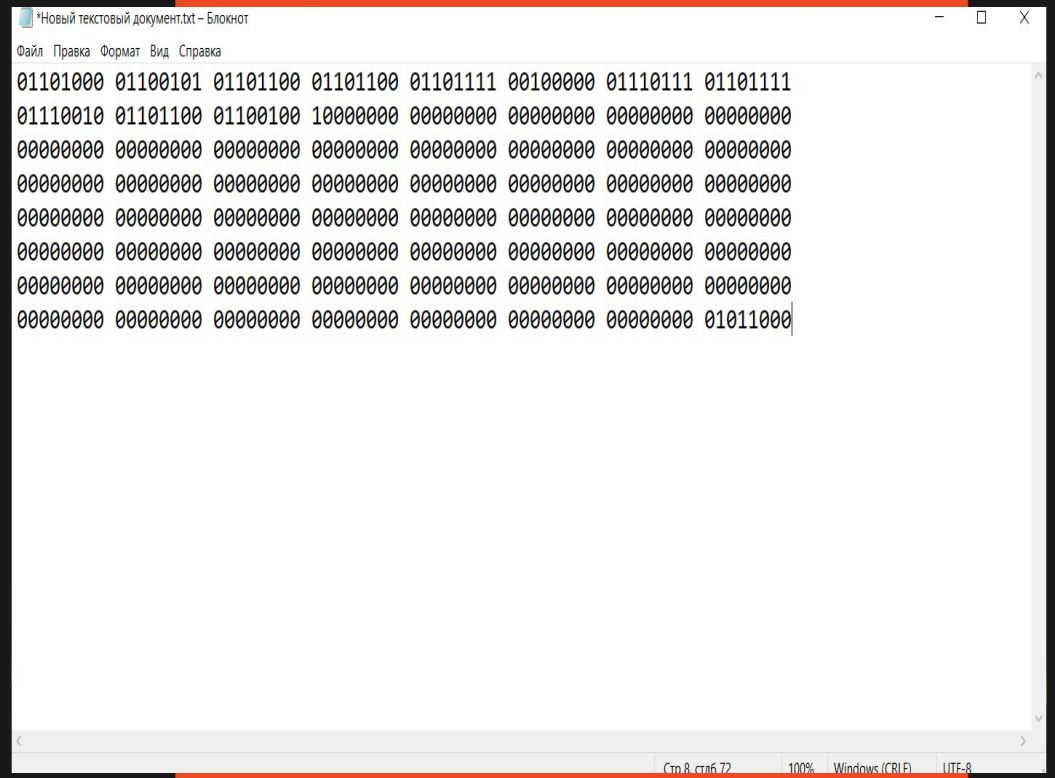
Что же такое Хэш – функция?

Хэш – функция с точки зрения компьютерных наук – функция, осуществляющая преобразование массива входных данных произвольной длины в выходную битовую строку установленной длины, выполняемое определённым алгоритмом. С точки зрения алгебры хэш – функция является необратимым отображением множества произвольной длины во множество с фиксированным числом значений.

Какие же существуют алгоритмы?



SHA – 256



Шаг 1 - Пролог

Возмём строку “hello world”, переведём её в двоичный формат. Затем к данной записи допишем 1 справа и затем дополним нулями так, что бы общий размер в битах составлял 448 бит (512 бит – 64 бита (размер сообщения))

Затем добавьте к данному блоку 64 бита в формате Big – Endian. Это будет колличество бит наших исходных данных

Шаг 2 – Инициализируем значения хэша (h)

Теперь мы создаем 8 хэш-значений. Это жестко запрограммированные константы, которые представляют собой первые 32 бита дробных частей квадратных корней из первых восьми простых чисел: 2, 3, 5, 7, 11, 13, 17, 19

h0 := 0x6a09e667

h1 := 0xbb67ae85

h2 := 0x3c6ef372

h3 := 0xa54ff53a

h4 := 0x510e527f

h5 := 0x9b05688c

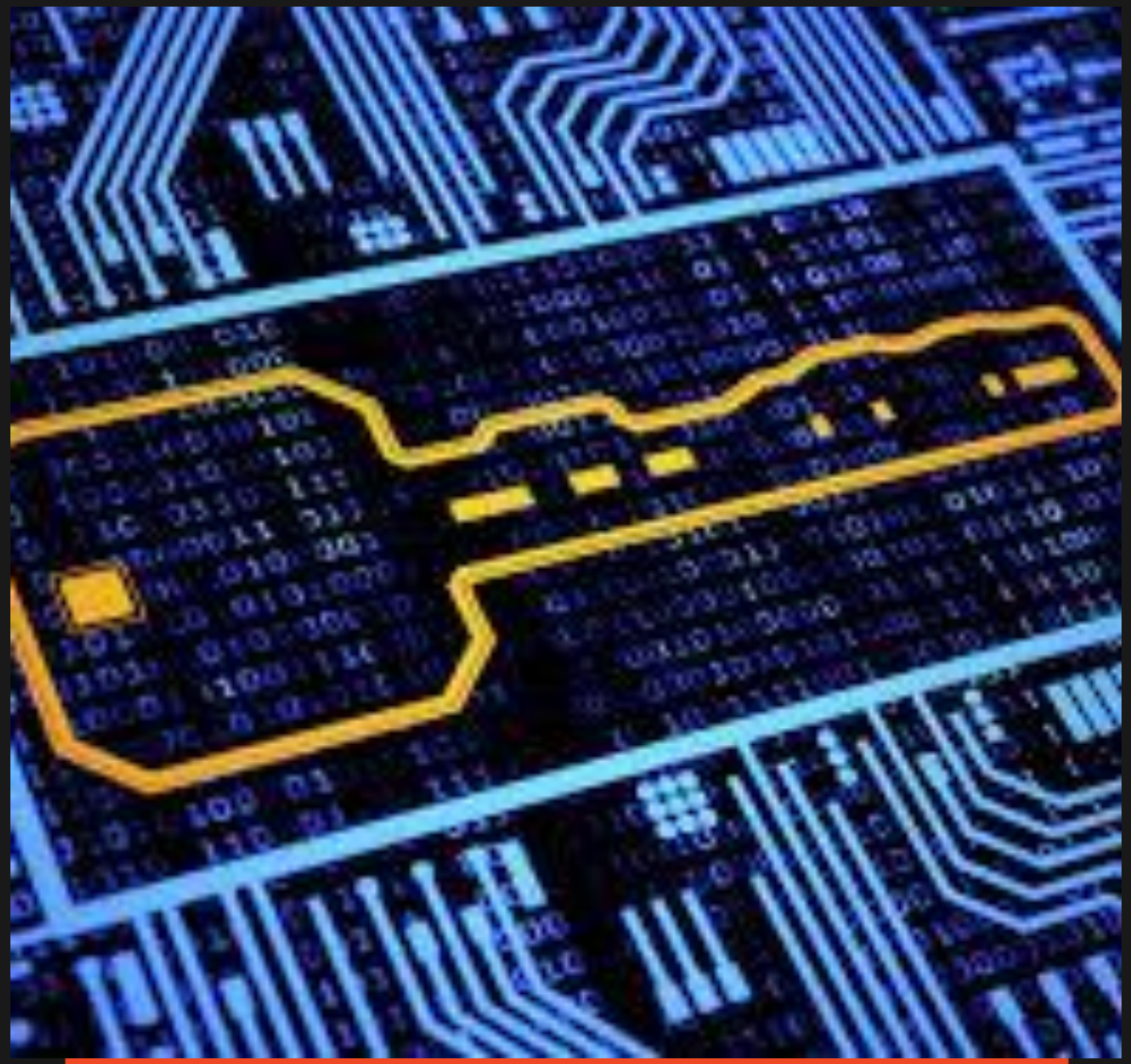
h6 := 0x1f83d9ab

h7 := 0x5be0cd19

Шаг 3 - Инициализируем округленные константы (k)

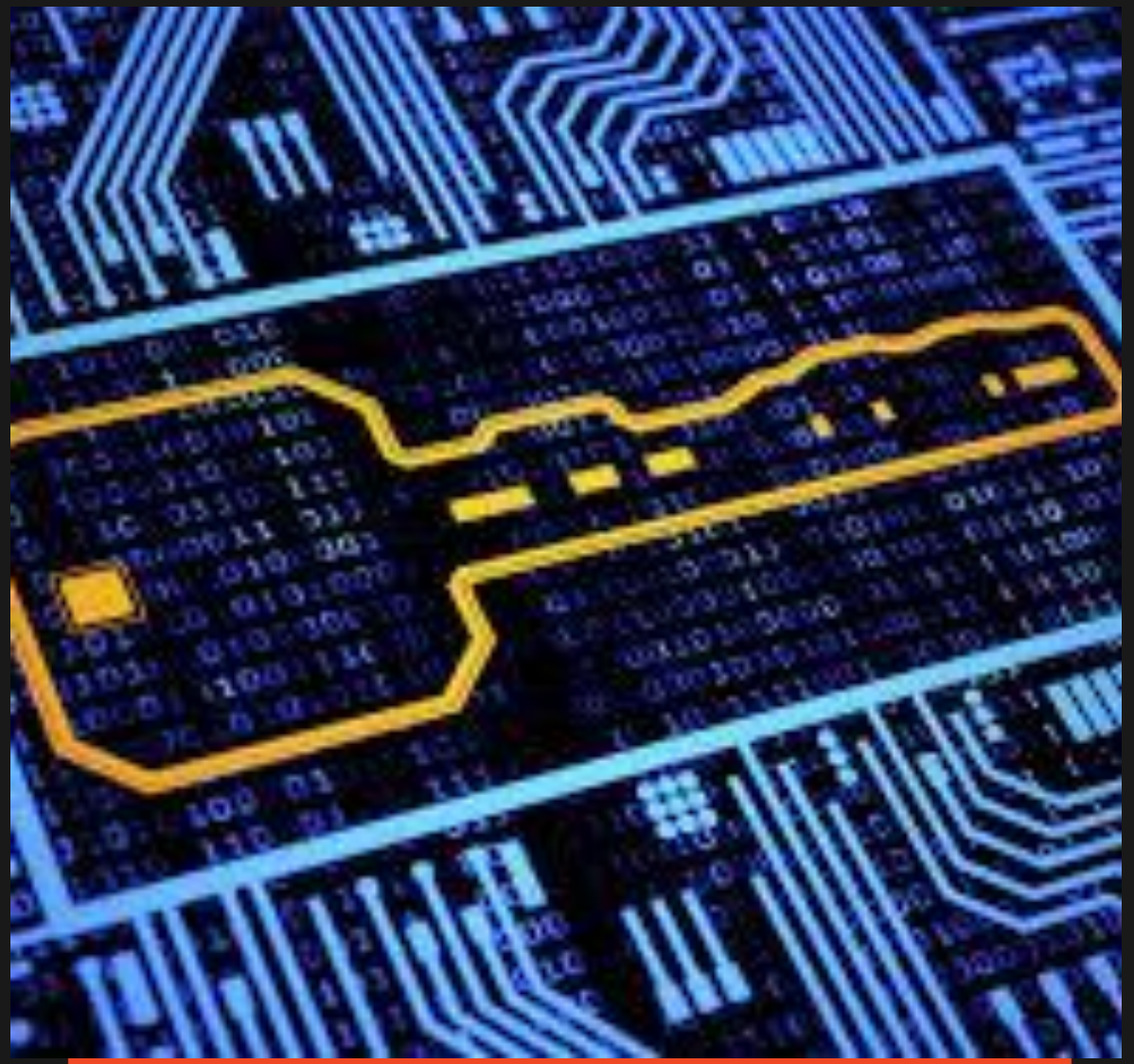
Как и в предыдущем шаге, мы создадим еще несколько констант. На этот раз их будет 64. Каждое значение (0—63) представляет собой первые 32 бита дробных частей кубических корней первых 64 простых чисел (2—311).

0x428a2f98 0x71374491 0xb5c0fbcf 0xe9b5dba5
0x3956c25b 0x59f111f1 0x923f82a4 0xab1c5ed5
0xd807aa98 0x12835b01 0x243185be 0x550c7dc3
0x72be5d74 0x80deb1fe 0x9bdc06a7 0xc19bf174
0xe49b69c1 0xefbe4786 0x0fc19dc6 0x240ca1cc
0x2de92c6f 0x4a7484aa 0x5cb0a9dc 0x76f988da
0x983e5152 0xa831c66d 0xb00327c8 0xbf597fc7
0xc6e00bf3 0xd5a79147 0x06ca6351 0x14292967
0x27b70a85 0x2e1b2138 0x4d2c6dfc 0x53380d13
0x650a7354 0x766a0abb 0x81c2c92e 0x92722c85
0xa2bfe8a1 0xa81a664b 0xc24b8b70 0xc76c51a3
0xd192e819 0xd6990624 0xf40e3585 0x106aa070
0x19a4c116 0x1e376c08 0x2748774c 0x34b0bcb5
0x391c0cb3 0x4ed8aa4a 0x5b9cca4f 0x682e6ff3
0x748f82ee 0x78a5636f 0x84c87814 0x8cc70208
0x90befffa 0xa4506ceb 0xbef9a3f7 0xc67178f2



Шаг 4 — Цикл фрагментов

Следующие шаги будут выполняться для каждого 512-битного «фрагмента» из наших входных данных. Поскольку фаза «hello world» короткая, у нас есть только один фрагмент. В каждой итерации цикла мы будем изменять хэш - значения h0 - h7, что приведет нас к конечному результату.



Шаг 5 — Созданием расписание сообщений (w)

Скопируем входные данные из шага 1 в новый массив, где каждая запись представляет собой 32-битное слово.

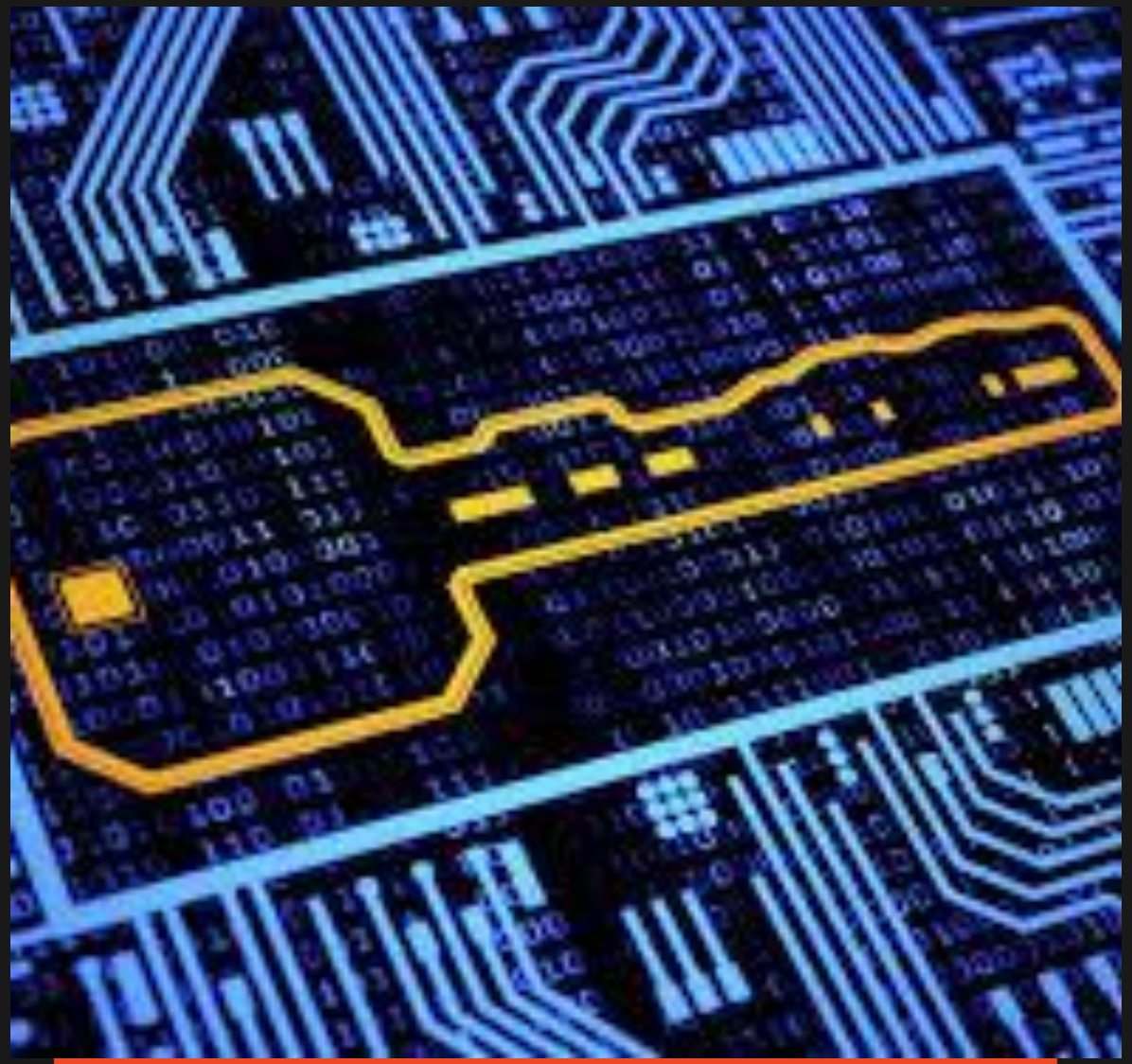
Добавим еще 48 слов, инициализированных нулем, чтобы у нас получился массив w [0 .. 63]

Изменим обнуленные индексы в конце массива, используя следующий алгоритм:
Для **i** из w[16…63]:

s0 = (w[i-15] rightrotate 7) xor (w[i-15] rightrotate 18) xor (w[i-15] rightshift 3)

s1 = (w[i- 2] rightrotate 17) xor (w[i- 2] rightrotate 19) xor (w[i-2] rightshift 10)

w[i] = w[i-16] + s0 + w[i-7] + s1



Шаг 6 — Сжатие

Инициализируем переменные **a, b, c, d, e, f, g, h** и установим их равными текущим значениям хэш-функции соответственно **h0, h1, h2, h3, h4, h5, h6, h7**.

Запустим цикл сжатия, который изменит значения **a .. h**. Выглядит он следующим образом:

Для **i** от 0 до 63
S1 = (e rightrotate 6) xor (e rightrotate 11) xor (e rightrotate 25)

ch = (e and f) xor ((not e) and g)

temp1 = h + S1 + ch + k[i] + w[i]

S0 = (a rightrotate 2) xor (a rightrotate 13) xor (a rightrotate 22)

maj = (a and b) xor (a and c) xor (b and c)

temp2 := S0 + maj

h = g

g = f

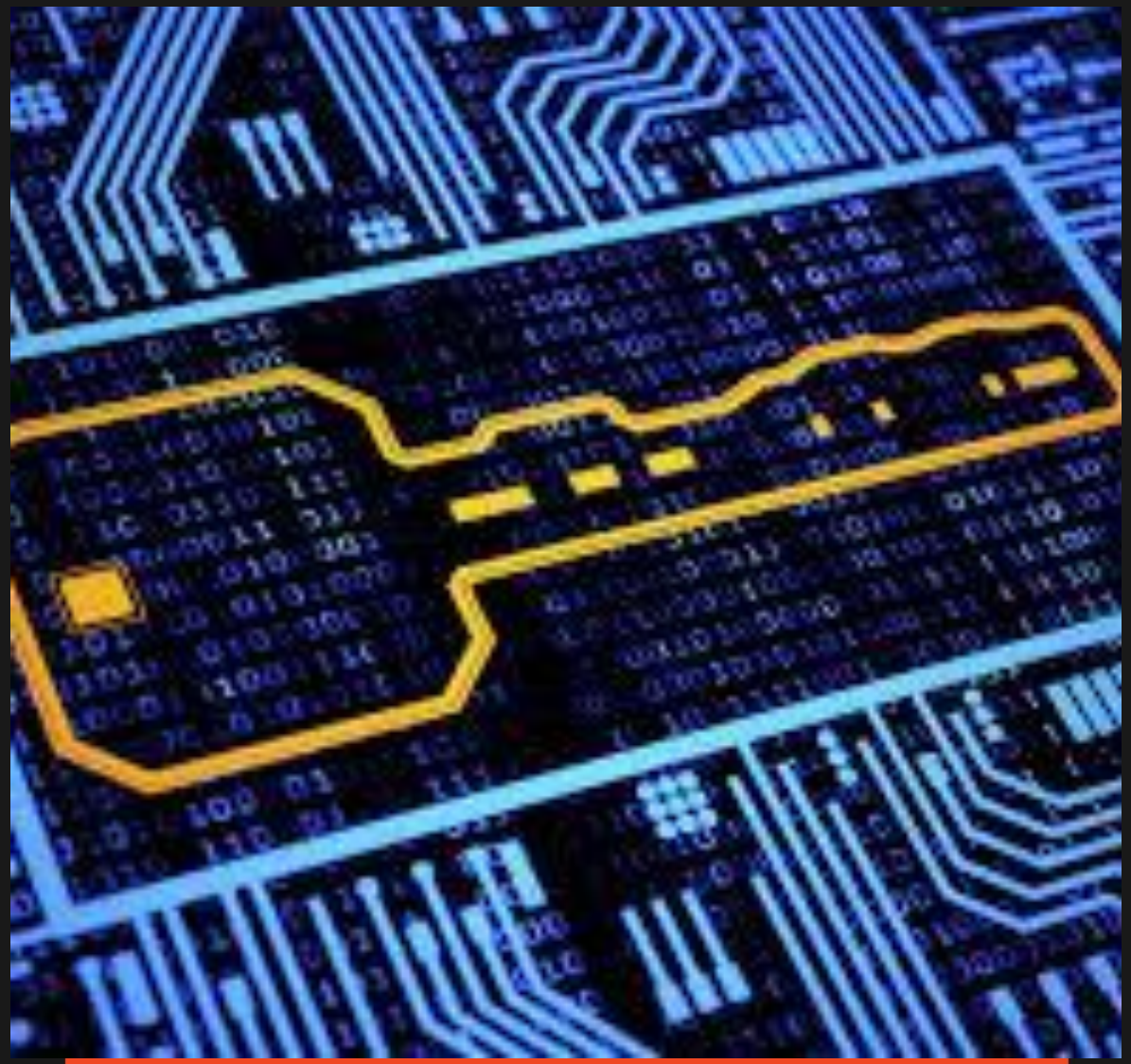
e = d + temp1

d = c

c = b

b = a

a = temp1 + temp2



Шаг 7 — Изменим окончательные значения

После цикла сжатия, во время цикла фрагментов, мы изменяем хеш - значения, добавляя к ним соответствующие переменные a - h. Как и ранее, все сложение производится по модулю 2 ^ 32:

h0 = h0 + a = 10111001010011010010011110111001

h1 = h1 + b = 10010011010011010011111000001000

h2 = h2 + c = 10100101001011100101001011010111

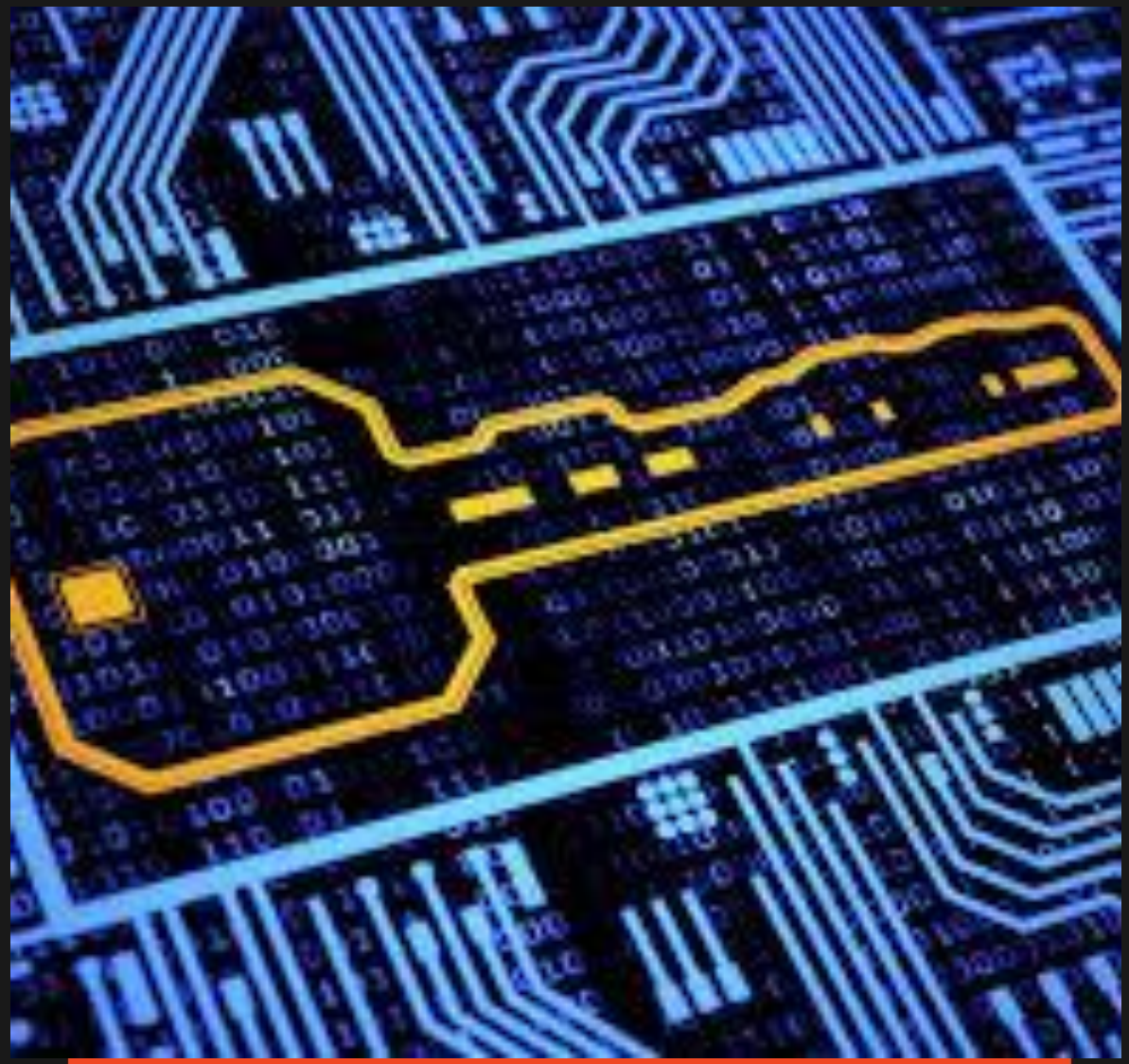
h3 = h3 + d = 11011010011111011010101111111010

h4 = h4 + e = 11000100100001001110111111100011

h5 = h5 + f = 01111010010100111000000011101110

h6 = h6 + g = 10010000100010001111011110101100

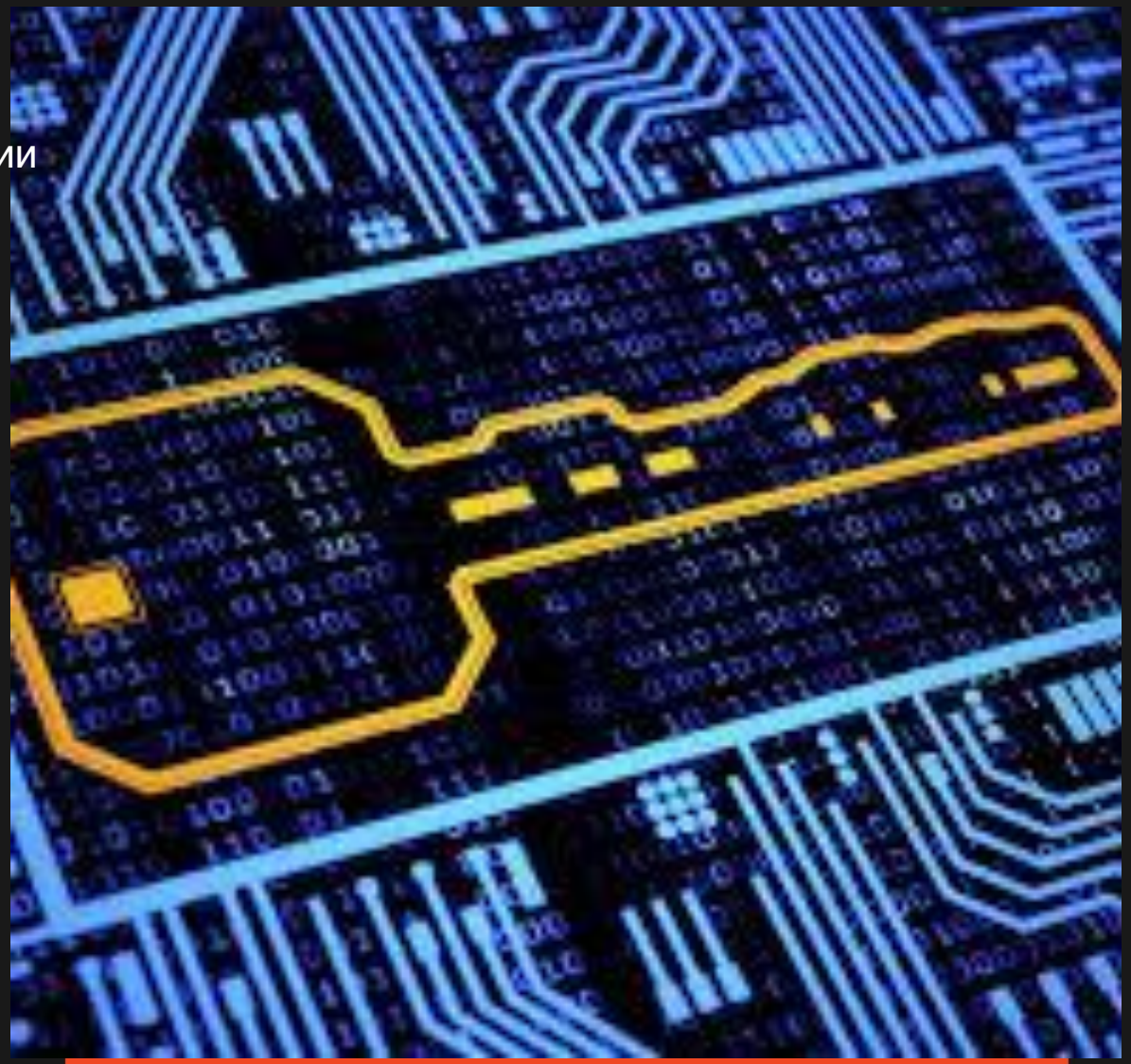
h7 = h7 + h = 11100010111011111100110111101001



Шаг 8 — Финальный хэш

Итоговое значение хэша у нас выходит из конкатонации

Значений переменных h0 – h7.



Методы взлома хэш - функций

-Поиск первого прообраза

-Поиск второго прообраза

-Поиск коллизии

Так же сильно помогают так называемые Rainbow Table

В качестве противодействия данным атакам используется технология солирования – это изменение сообщения посредством добавления какого то элемента в конкретные места исходного сообщения перед его хэшированием

Итог: Мы изучили вопросы касаемо хэш функций

Мы узнали что собой представляет хэш функция

Мы узнали алгоритм работы такой хэш функции как SHA256

Мы посмотрели на различные вектора атак на хэш функции