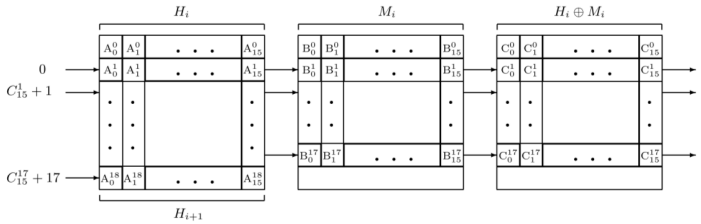
Отчет по первому практическому заданию «Нахождение прообраза функции сжатия MD2»

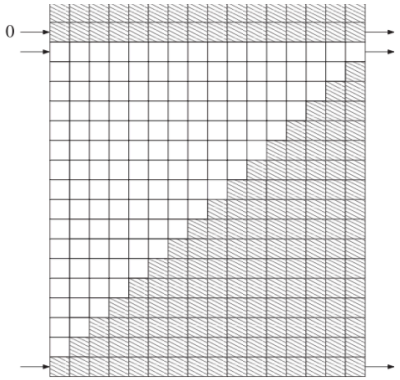
a) Задача: найти хэш некоторого сообщения M из n байт. Оно выполняется в 4 шага:
   1. Добавляем биты, чтобы дополнить длину нашего сообщения M до длины кратной 16. Если длина сообщения уже кратна 16, то мы все равно добавляем еще 16 бит. В задании мы дополняли сообщение i байтами, которые равнялись i mod 4.
   2. Добавляем контрольную сумму. Она высчитывается для каждого 16-байтного блока по следующим формулам: (L = 0 – вспомогательная переменная)
      c = M[i*16+j] – вспомогательная переменная, j – номер цикла, i – номер блока
      C[j] = Sbox[c xor L] xor C[j] – контрольная сумма
      L = C[j]
      Контрольная сумма добавляется к концу нашего сообщения (то есть длина сообщения увеличивается еще на 16 байтов)
   3. Будем работать с массивом X размера 48 байт. Для начала заполним его 0, а затем по следующим формулам для каждого блока сообщения:
      X[16+j] = M[i*16+j]
      X[32+j] = (X[16+j] xor X[j])
      Затем 18 раундов выполняется следующее: (t = 0)
      X[k] = (X[k] xor S[t])
      t = X[k]
      Так делаем для каждого k от 0 до 47. В каждом новом раунде t = (t+j) mod4 (j от 0 до 17)
   4. Вычисляется хэш, как первые 16 байтов массива X.

   После шага контрольной суммы для получения хэша может использовать функция сжатия F, которое действует итеративно. H(i+1) = F(Hi,Mi). Значение IV = H0 и по дефолту равно 0. Функция сжатия вычисляется с помощью трех матриц A, B, C:



   Как видно из них, каждая строчка является результатом вычисления предыдущих строк. Чтобы найти H(i+1) нужно заполнить последнюю строку матрицы A. Они заполняются с помощью формул: A[i][j] = A[i-1][j] xor Sbox[A[i][j-1]]. Боковым же слева подаются на вход значения из предыдущей матрицы (на картинке понятно, что именно).

b) Алгоритм атаки. Нам известны Hi и результат сжатия – H(i+1). Нужно вычислить Mi. Для начала мы ищем все что можно в матрице A. Нам будут известны первая и последняя строки. Из них мы можем найти правую половину матрицы и вторую строку (так как знаем, что на вход второй строки слева подается 0)



   Теперь если мы узнаем значение байта A[2][0], то мы сможем восстановить всю матрицу. Поэтому подберем его (всего 4 варианта подбора). Это первый шаг. Во втором шаге мы перебираем значения B[1][15]…B[4][[15]. Далее подбираем первую половину нашего сообщения – B[0][0]…B[0][7]. Зная эти данные мы можем найти C[1][7]…C[4][7] и B[1][7]…B[4][7]. (просто идем по нашим формулам слева направо в таблицах B и C. Как раз дойдем до 7го столбца) Записываем полученные данные в отдельную таблицу. Аналогично ищем все для второй половины сообщения. Теперь, чтобы найти C[1][7]…C[4][7] и B[1][7]…B[4][7] нужно идти справа налево и искать обратную функцию от Sbox. В итоге эти значения запишем во вторую таблицу. Отсортируем две таблицы и найдем отличия в них. Уберем эти строки – теперь искать одинаковые элементы в ней будет легче. Тем самым мы получим всевозможные Mi, которые у нас могут быть. Из них выбираем наиболее подходящий.

c) Третье задание у меня не получилось, потому что я не смогла построить оптимально таблицу, ждала 4 часа для получения какого-то ответа, в итоге обнаружила у себя ошибку. Я не придумала как сделать так, чтобы программа работала быстрее, сами таблицы A, B, C она быстро считает, а поиск одинаковых по таблице – долго. Я пыталась ручками найти нужные мне значения, которые я подбираю – в итоге получилось, что тест неправильный, потому что при подсчете я зашла в тупик. В общем, я запуталась и поэтому не смогла сделать последний режим программы.