Шляхтина Е. А.

Практическое задание №1
Нахождение прообраза функции сжатия MD2

1. **Описание алгоритма MD2.**

   Особенности реализации: 2-битный "байт" , $S = [1, 3, 0, 2]$, длина блока = 16 байт. Дано сообщение $m = m_0...m_{b-1}$, где m_i - i-ый байт сообщения.

   1) **Padding**: сообщение m дополняется num байтами со значением (num mod 4) до длины, кратной 16. N - новая длина сообщения

   2) **Контрольная сумма**: к концу сообщения добавляется 16 байт - контрольная сумма C по следующему алгоритму:

```
For i = 0 to 15 do              /* Обнуляется буффер контрольной суммы */
    Set C[i] to 0.
end /* of loop on i */
Set L to 0.
For i = 0 to N/16-1 do          /* Для каждого блока выполняется */
    For j = 0 to 15 do          /* Подсчет контрольной суммы блока */
        Set c to M[i*16+j].
        Set C[j] to C[j] xor S[c xor L].
        Set L to C[j].
     end /* of loop on j */
end /* of loop on i */
```

   Новая длина сообщения $N' = N + 16$

   3) Для вычисления хэша используется буффер X размера 48 байт, который инициализируется нулями. Основной цикл вычисления MD2:

```
For i = 0 to N'/16 - 1 do       /* Для каждого блока выполняется */
    block = m[i * 16 : (i + 1) * 16]
    X = F(block, X)
end /* of loop on i */
```

   MD2 = X[0..15], где функция F:

```
function F(msgBlock, X):
    Set newX to X.
    For i = 0 to 15 do
        Set b to msgBlock[i].
        Set newX[16 + i] to b.
        Set newX[2 * 16 + i] to b xor newX[i].
    end /* of loop on i */
    Set t to 0.
    For i = 0 to 17 do
        For j = 0 to 47 do
            Set newX[j] to newX[j] xor S[t].
            Set t to newX[j].
        Set t to (t + i) mod 4.
        end /* of loop on j */
    end /* of loop on i */
    return newX.
```

2. **Описание атаки.**
Определим функцию сжатия:

```
function compress(H, M):
    X = [0..47]
    For i = 0 to 15 do                  /* Обнуляется буффер X */
        Set X[i] to 0.
    end /* of loop on i */
    For i = 0 to 15 do                  /* Первая треть буффера X присваивается H*/
        X[i] = H[i]
    end /* of loop on i */
    return F(M,X)[0..15]
```

Задача: зная H_i и H_{i+1}, найти M: $H_{i+1} = compress(H_i, M)$.
Работу функции *compress* описывает Рис.1. Элементы матрицы A вычисляются по формуле $A_i^t = A_i^{t-1} \oplus S(A_{i-1}^t)$, где t - номер строки, i - номер столбца. Элементы матриц B и C вычисляются по аналогичным формулам.

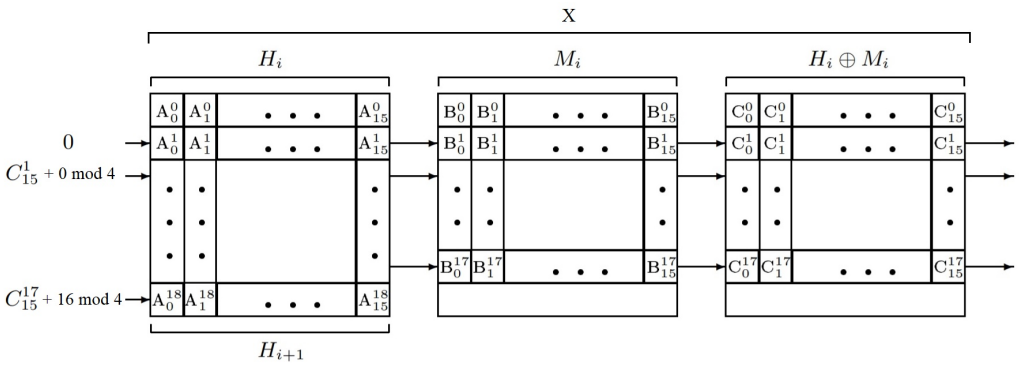

Рис. 1. Промежуточные результаты работы *compress*

Шаг 1: Получение всей необходимой информации

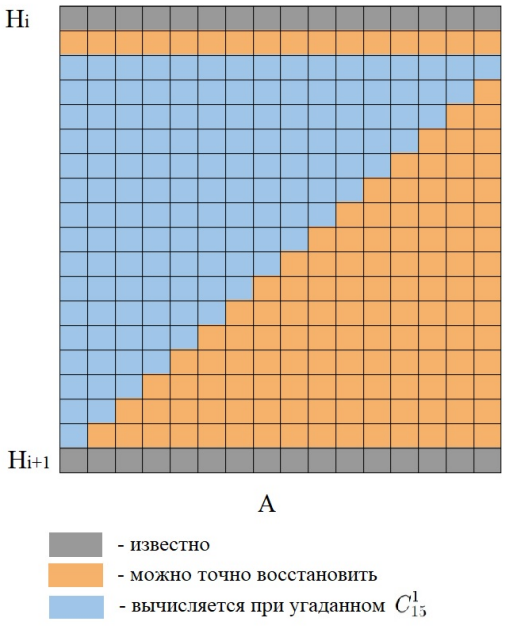

Рис. 2. Процесс вычисления матрицы A

1) Зная H_i и то, что по алгоритму для вычисления первой строки всегда подается 0, можем вычислить первую строку.
2) С помощью H_{i+1} по формуле $A_i^{t-1} = A_i^t \oplus S(A_{i-1}^t)$ вычисляется правый нижний треугольник матрицы A.
3) Угадав байт C_{15}^1, мы можем до конца восстановить матрицу A и узнать C_{15}^i, $t = \overline{2, 17}$.

Шаг 2: Метод встречи посередине

1) Угадываем $B_{15}^1..B_{15}^4$
2) Для всех возможных значений $B_0^0..B_7^0$ найдем

- $B_7^1..B_7^4$ (т.к. известны $A_{15}^1..A_{15}^4$ и $B_0^0..B_7^0$);
- $C_7^1..C_7^4$ (т.к. известны $B_{15}^1..B_{15}^4$ и $C_0^0..C_7^0$)

В таблицу T внесем запись: $(B_7^1, .., B_7^4, C_7^1, .., C_7^4) : (B_0^0..B_7^0, \quad)$
3) Для всех возможных значений $B_8^0..B_{15}^0$ найдем

- $B_7^1..B_7^4$ (т.к. известны $B_{15}^1..B_{15}^4$ и $B_8^0..B_{15}^0$);
- $C_7^1..C_7^4$ (т.к. известны $C_{15}^1..C_{15}^4$ и $C_8^0..C_{15}^0$)

В таблицу T внесем запись или добавим к уже существующей:
$(B_7^1, .., B_7^4, C_7^1, .., C_7^4) : (\quad, B_8^0..B_{15}^0)$
4) В полученной таблице T ищем такие записи, где присутстыуют и левые, и правые части сообщения. Из них формируются всевозможные кандидаты в искомое сообщение и проверяются с помощью функции *compress*.

Таким образом, с помощью этого перебора будет найдет корректный прообраз M: $H_{i+1} = compress(H_i, M)$.

3. **Теоритическая оценка времени работы и требуемой памяти. Реальное время работы.**
1) **Память**:
Требуется хранить в памяти матрицу A ($19 \cdot 16 = 304$ байт); части матриц B и C, необходимые для расчетов ($2 \cdot 5 \cdot 16 = 160$ байт); Sbox (4 байта); таблицу с записями вида $(B_7^1, .., B_7^4, C_7^1, .., C_7^4) : ([B_0^0..B_7^0, ...], [B_8^0..B_{15}^0, ...])$ (в среднем 9320 байт - найдено опытным путем). T.e всего 9788 байт.

2) **Время работы**:
Для расчета времени работы будем учитывать число арифметических операций.

- Работа с матрицами:
  Зафиксировав пятерку параметров $(C_{15}^1, B_{15}^4, B_{15}^3, B_{15}^2, B_{15}^1)$, нужно выполнить для матрицы A: 136 операций XOR не зависящих от C_{15}^1 и 136 операций XOR зависящих от C_{15}^1;
  для матрицы B: для всевозможных значений $B_0^0, .., B_{15}^0$ $4 \cdot 16 = 64$ операции XOR, т.е. $64 \cdot 4^{16} = 2^{38}$ операций;
  для матрицы C: аналогично матрице B 2^{38} операций;
  T.e. $136 + 4 \cdot 136 + 1024 \cdot 2^{38} * 2 \approx 2^{49}$ операций
- Работа с таблицей:
  Для каждой пятерки параметров $(C_{15}^1, B_{15}^4, B_{15}^3, B_{15}^2, B_{15}^1)$ строится таблица вида $(B_7^1, .., B_7^4, C_7^1, .., C_7^4) : ([B_0^0..B_7^0, ...], [B_8^0..B_{15}^0, ...])$. Опытным путем найдено, что в ней в среднем 256 записей. Для каждой записи нужно проверить левые и правые половинки предполагаемого сообщения. Предполагая, что 4^{16} всевозможных вариантов сообщений равномерно распределены на пространстве ключей таблицы

(т.е. на одну запись приходится $2^{32-8} = 2^{24}$ сообщений), получим что для проверки одной записи таблицы нужно 2^{24} раз вызывать функцию *check*, которая включает в себя 50 операций сложения, 18 операций взятия остатка от деления, 304 операции XOR и 1 операцию сравнения. Таким образом, имеем еще $256 \cdot 2^{24} \cdot 373 \approx 2^{40}$ арифметических операций.

Если взять время выполнение одной арифметической операции за 2^{-31} сек, то получим $(2^{40} + 2^{49}) \cdot 2^{-31} \approx 73$ч

3) **Реальное время работы**:

Реальное время работы программы бутет больше ввиду операций чтения, обращения к памяти... Однако на тестах, предоставленных в задании, программа работает достаточно быстро, т.к. первый подходящий прообраз находится при параметрах, находящихся в самом начале списка проверки.

Например, для **варианта 16**:

$H_i = 3\ 2\ 2\ 2\ 2\ 0\ 0\ 2\ 0\ 3\ 2\ 0\ 1\ 0\ 3\ 1$

$H_{i+1} = 0\ 0\ 3\ 3\ 0\ 2\ 1\ 1\ 0\ 3\ 3\ 0\ 2\ 2\ 1\ 1$

Ответ: $M_i = 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 3\ 2\ 2\ 2$

Время работы составило 8.47 сек.