5.2. Простые хэш-функции

Все *хэш-функции* выполняются следующим образом. Входное значение (сообщение, файл и т.п.) рассматривается как последовательность n-битных блоков. Входное значение обрабатывается последовательно блок за блоком, и создается m-битное значение *хэш-кода*.

Одним из простейших примеров *хэш-функции* является побитный XOR каждого блока:

$C_i = b_{i1} \oplus b_{i2} \oplus ... \oplus b_{ik}$

Где

C_i - i-ый бит *хэш-кода*, $1 \le i \le n$.

k - число n-битных блоков входа.

b_{ij} - i-ый бит в j-ом блоке.

$\oplus$ - операция XOR.

В результате получается *хэш-код* длины n, известный как продольный избыточный контроль. Это эффективно при случайных сбоях для проверки целостности данных.

Часто при использовании подобного продольного избыточного контроля для каждого блока выполняется однобитный циклический сдвиг после вычисления *хэш-кода*. Это можно описать следующим образом.

- Установить n-битный *хэш-код* в ноль.
- Для каждого n-битного блока данных выполнить следующие операции:
  - сдвинуть циклически текущий *хэш-код* влево на один бит;
  - выполнить операцию XOR для очередного блока и *хэш-кода*.

Это даст эффект "случайности" входа и уничтожит любую регулярность, которая присутствует во входных значениях.

Хотя второй вариант считается более предпочтительным для обеспечения целостности данных и предохранения от случайных сбоев, он не может использоваться для обнаружения преднамеренных модификаций передаваемых сообщений. Зная сообщение, атакующий легко может создать новое сообщение, которое имеет тот же самый *хэш-код*. Для этого следует подготовить альтернативное сообщение и затем присоединить n-битный блок, который является *хэш-кодом* нового сообщения, и блок, который является *хэш-кодом* старого сообщения.

Хотя простого XOR или ротационного XOR (RXOR) недостаточно, если целостность обеспечивается только зашифрованным *хэш-кодом*, а само сообщение не шифруется, подобная простая функция может использоваться, когда все сообщение и присоединенный к нему *хэш-код* шифруются. Но и в этом случае следует помнить о том, что подобная *хэш-функция* не может проследить за тем, чтобы при передаче последовательность блоков не изменилась. Это происходит в силу того, что данная *хэш-функция*

определяется следующим образом: для сообщения, состоящего из последовательности 64-битных блоков X_1, X_2,..., X_N, определяется *хэш-код* C как поблочный XOR всех блоков, который присоединяется в качестве последнего блока:

$C = X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N$

Затем все сообщение шифруется, включая *хэш-код*, в режиме CBC для создания зашифрованных блоков Y_1, Y_2, ..., Y_{N+1}. По определению CBC имеем:

$X_1 = IV \oplus D_K [Y_1]$

$X_i = Y_{i-1} \oplus D_K [Y_i]$

$X_{N+1} = Y_N \oplus D_K [Y_{N+1}]$

Но X_{N+1} является *хэш-кодом*:

$X_{N+1} = X_1 \oplus X_2 \oplus ... \oplus X_N = (IV \oplus D_K [Y_1]) \oplus (Y_1 \oplus D_K [Y_2]) \oplus ... \oplus (Y_{N-1} \oplus D_K [Y_N])$

Так как сомножители в предыдущем равенстве могут вычисляться в любом порядке, следовательно, *хэш-код* не будет изменен, если зашифрованные блоки будут переставлены.