**1 Криптографические хеш-функции**

**Криптографические хеш-функции** — это выделенный класс криптографически стойких хеш-функций, который имеет определённые свойства, делающие его пригодным для использования в криптографии.

Хэш-функции, долгое время использующиеся в компьютерных науках, представляют собой функции, математические или иные, которые получают на вход данные переменной длины (называемые прообразом) и преобразуют их в данные фиксированной, обычно меньшей, длины (называемые значением хэш-функции).

**Однонаправленная хэш-**функция – это хэш-функция, которая работает только в одном направлении: легко вычислить значение хэш-функции по прообразу, но трудно или невозможно создать прообраз, значение хэш-функции которого равно заданной величине. Хэш-функция является **открытой**, тайны ее расчета не существует. Безопасность однонаправленной хэш-функции заключается именно в ее однонаправленности - у выхода нет видимой зависимости от входа. Изменение одного бита прообраза приводит к изменению (в среднем) половины битов значения хэш-функции, что известно также как **лавинный эффект**.

Для того, чтобы хэш-функция **H** считалась криптографически стойкой, она должна удовлетворять трем основным требованиям, на которых основано большинство применений хэш-функций в криптографии:

* **Необратимость или стойкость к восстановлению первого прообраза**: для заданного значения хэш-функции m должно быть вычислительно невозможно найти блок данных X, для которого H(X)=m;
* **Стойкость к коллизиям первого рода или восстановлению второго прообраза:** для заданного сообщения M должно быть вычислительно невозможно подобрать не равное ему сообщение N, для которого H(N) = H(M);
* **Стойкость к коллизиям второго рода:** должно быть вычислительно невозможно подобрать пару различных сообщений (M, M'), имеющих одинаковый хэш. Так как количество возможных открытых текстов больше числа возможных значений свёртки, то для некоторой свёртки найдётся много прообразов, а следовательно, коллизии для хеш-функций обязательно существуют. Стойкость хеш-функции к коллизиям означает, что нет эффективного полиномиального алгоритма, позволяющего находить коллизии.

Данные свойства зависят друг от друга следующим образом:

* Обратимая функция неустойчива к восстановлению второго прообраза и коллизиям;
* Функция, нестойкая к восстановлению второго прообраза, нестойка к коллизиям; обратное неверно;
* Функция, устойчивая к коллизиям, устойчива к нахождению второго прообраза;
* Устойчивая к коллизиям хеш-функция не обязательно является односторонней.

**2 Принципы построения**

**2.1 Итеративная последовательная схема**

В общем случае в основе построения хеш-функции лежит итеративная последовательная схема. Ядром алгоритма является **сжимающая функция** — преобразование k входных в n выходных бит, где n — разрядность хеш-функции, а k — произвольное число, большее n. При этом сжимающая функция должна **удовлетворять всем условиям криптостойкости.**

Входной поток разбивается на блоки по (k − n) бит. Алгоритм использует временную переменную размером в n бит, в качестве начального значения которой берётся некое общеизвестное число. Каждый следующий блок данных объединяется с выходным значением сжимающей функции на предыдущей итерации. Значением хеш-функции являются выходные n бит последней итерации. Каждый бит выходного значения хеш-функции зависит от всего входного потока данных и начального значения. Таким образом достигается лавинный эффект.

Размер входного потока данных должен быть кратен (k − n). Как правило, перед началом алгоритма данные расширяются неким, заранее известным, способом.

**2.2 Сжимающая функция на основе симметричного блочного алгоритма**

В качестве сжимающей функции можно использовать симметричный блочный алгоритм шифрования. Для обеспечения большей безопасности можно использовать в качестве ключа блок данных, предназначенный к хешированию на данной итерации, а результат предыдущей сжимающей функции — в качестве входа. Тогда результатом последней итерации будет выход алгоритма. В таком случае безопасность хеш-функции базируется на безопасности используемого алгоритма.