# ТЕМА 3. Алгоритмы симметричного шифрования

## 3.1. Основные понятия и определения

Рассмотрим общую схему симметричного шифрования.


**Рисунок 3.1 – Схема симметричного шифрования**

**Симметричные алгоритмы** можно классифицировать на:

1. *Моно- и многоалфавитные подстаноки.*
2. *Перестановки*
3. *Гаммирование.*
4. *Блочные шифры.*

Алгоритмы симметричного шифрования различаются способом, которым обрабатывается исходный текст. Возможно шифрование блоками или шифрование потоком.

Все действия, производимые над данными блочным криптоалгоритмом, основаны на том факте, что преобразуемый блок может быть представлен в виде целого неотрицательного числа из диапазона, соответствующего его разрядности. Так, например, 32-битный блок данных можно интерпретировать как число из диапазона 0..4'294'967'295. Кроме того, блок, разрядность которого обычно является "степенью двойки", можно трактовать как несколько независимых неотрицательных чисел из меньшего диапазона (рассмотренный выше 32-битный блок можно также представить в виде 2 независимых чисел из диапазона 0..65535 или в виде 4 независимых чисел из диапазона 0..255).

Над этими числами блочным криптоалгоритмом и производятся по определенной схеме следующие действия:

- Биективные математические функции (сложение, исключающее ИЛИ, умножение по модулю $2^N+1$, умножение по модулю $2^N$);
- Битовые сдвиги (арифметические и циклические сдвиги влево и вправо);
- Табличные подстановки.

Описанные выше операции циклически повторяются в алгоритме, образуя так называемые *раунды*. Входом каждого *раунда* является выход предыдущего *раунда* и ключ, который получен по определенному алгоритму из ключа шифрования K. Ключ *раунда* называется *подключом*. Каждый алгоритм шифрования может быть представлен следующим образом:

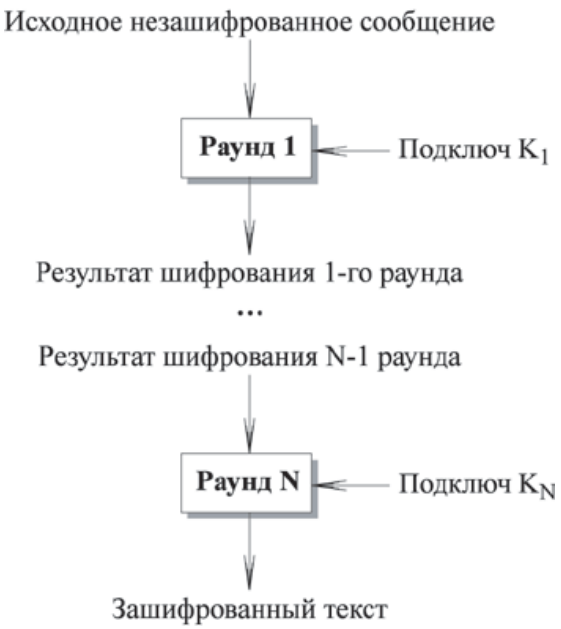


**Рисунок 3.2 – Структура алгоритма симметричного шифрования**

Характерным признаком блочных алгоритмов является многократное и косвенное использование материала ключа. Это диктуется в первую очередь требованием невозможности обратного декодирования в отношении ключа при известных исходном и зашифрованном текстах. Для решения этой задачи в приведенных выше преобразованиях чаще всего используется не само значение ключа или его части, а некоторая, иногда необратимая (небиективная) функция от материала ключа. Более того, в подобных преобразованиях один и тот же блок или элемент ключа используется многократно. Это позволяет при выполнении условия обратимости функции относительно величины X сделать функцию необратимой относительно ключа Key.

Поскольку операция зашифровки или расшифровки отдельного блока в процессе кодирования пакета информации выполняется многократно (иногда до сотен тысяч раз), а значение ключа и, следовательно, функций *Vi*(Key) остается неизменным, то иногда становится целесообразно заранее однократно вычислить данные значения и хранить их в оперативной памяти совместно с ключом. Поскольку эти значения зависят только от ключа, то оин в криптографии называются материалом ключа. Необходимо отметить, что данная операция никоим образом не изменяет ни длину ключа, ни криптостойкость алгоритма в целом. Здесь происходит лишь оптимизация скорости вычислений путем кеширования (англ. caching) промежуточных результатов. Описанные действия встречаются практически во многих блочных криптоалгоритмах и носят название расширение ключа (англ. key scheduling).

**Области применения**

Стандартный алгоритм шифрования должен быть применим во многих приложениях:

- Шифрование данных.
- Создание случайных чисел.
- Хэширование.

**Платформы**

Стандартный алгоритм шифрования должен быть реализован на различных платформах:

- Специализированная аппаратура.
- Большие процессоры.
- Процессоры среднего размера.
- Малые процессоры.

**Используемые критерии при разработке алгоритмов**

Считается, что алгоритм симметричного шифрования должен:

- Иметь размер блока 64 или 128 бит.
- Иметь масштабируемый ключ до 256 бит.
- Использовать простые операции, которые эффективны на микропроцессорах, Не должно использоваться сдвигов переменной длины, побитных перестановок или условных переходов.
- Должна быть возможность реализации алгоритма на 8-битном процессоре с минимальными требованиями к памяти.
- Использовать заранее вычисленные подключи.
- Состоять из переменного числа итераций.
- По возможности не иметь слабых ключей.
- Задействовать подключи, которые являются односторонним хэшем ключа.
- Не иметь линейных структур, которые уменьшают комплексность и не обеспечивают исчерпывающий поиск.
- Использовать простую для понимания разработку.