**Установка**

Windows

* Из командной строки:  
  winget install openssl
* Через установщик с сайта <https://slproweb.com/products/Win32OpenSSL.html>

Linux

* sudo apt install openssl libssl3 libssl-dev libsssl-doc

**Получение списка команд**

openssl help

**Получение подробной информации об определенной команде**

openssl enc -help

**Получение списка поддерживаемых версией стандартов**

openssl list -cipher-algorithms

**Команды шифрования/расшифрования (полный список опций - https://wiki.openssl.org/index.php/Enc)**

openssl enc -ШИФР -e -in ДЛЯ-ШИФРОВАНИЯ -out ЗАШИФРОВАНЫЕ-ДАННЫЕ

openssl enc -ШИФР -d -in ЗАШИФРОВАНЫЕ-ДАННЫЕ -out РАСШИФРОВАННЫЕ-ДАННЫЕ

**Шифрование и расшифрование с применением AES в командной строке**

Сгенерируем 256-битовый ключ шифрования:

openssl rand -hex 32

# 74c8a19fee9e0710683afe526462ce8a960ca1356e113bf3a08736a68a48eca0

А так же 128-битовый IV той же длины, что размер блока:

openssl rand -hex 16

# 4007fbd36f08cb04869683b1f8a15c99

Зашифруем файл:

openssl enc ^

-aes-256-cbc ^

–K 74c8a19fee9e0710683afe526462ce8a960ca1356e113bf3a08736a68a48eca0 ^

-iv 4007fbd36f08cb04869683b1f8a15c99 ^

-e ^

-in somefile.txt ^

-out somefile.txt.encrypted

Попробуем расшифровать файл:

openssl enc ^

-aes-256-cbc ^

-K 74c8a19fee9e0710683afe526462ce8a960ca1356e113bf3a08736a68a48eca0 ^

-iv 4007fbd36f08cb04869683b1f8a15c99 ^

-d ^

-in somefile.txt.encrypted ^

-out somefile.txt.decrypted

Проверим, что длина расшифрованного файла совпадает с  длиной исходного. Это можно сделать, взглянув на содержимое файла в редакторе, но надежнее вычислить криптографически стойкое хеш-значение сообщения:

openssl dgst somefile.txt\*

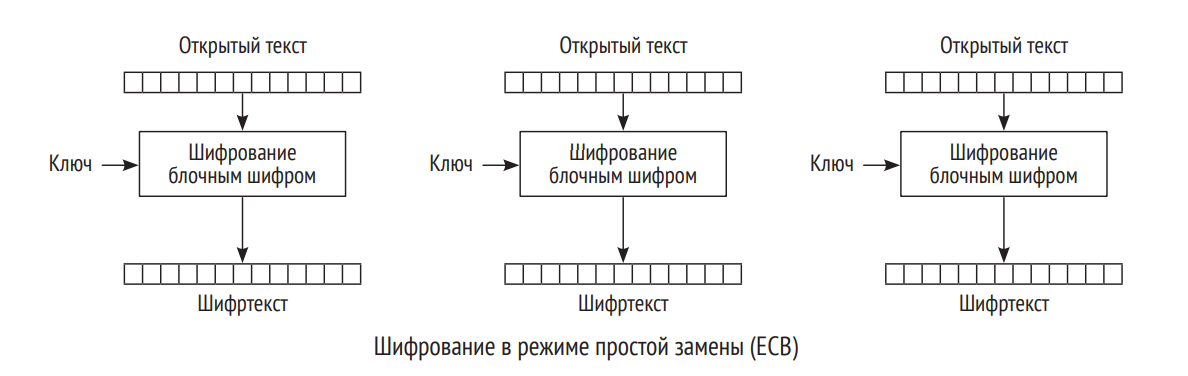
SHA256(somefile.txt)= 67d4ff71d43921d5739f387da09746f405e425b-07d727e4c69d029461d1f051f

SHA256(somefile.txt.decrypted)= 67d4ff71d43921d5739f-387da09746f405e425b07d727e4c69d029461 d1f051f

SHA256(somefile.txt.encrypted)= 5a6a88e37131407c9dc9b8601b-3195c8a33c75a20421a1733233a963b a42aef3

**Обзор режима простой замены**

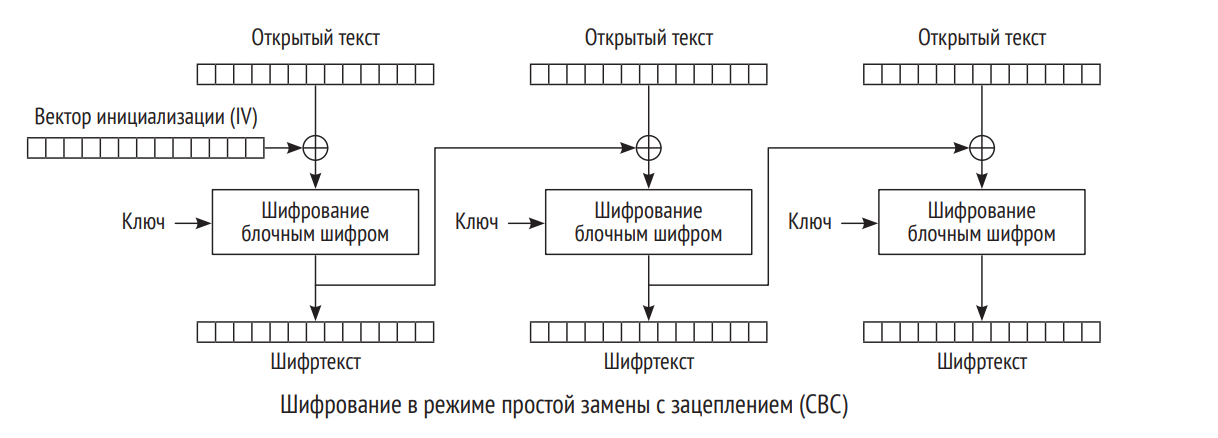
Простейшим режимом работы является режим простой замены (Electronic Code Book – ECB). В этом режиме каждый блок открытого текста преобразуется в блок шифртекста с применением только ключа шифрования, без использования IV или предыдущих блоков открытого или шифртекста. Таким образом, порожденные блоки шифртекста конкатенируются. Режим ECB показан на следующем рисунке.



В режиме ECB один и тот же открытый текст всегда порождает один и тот же шифртекст. С точки зрения безопасности, это проблема, потому что любые закономерности в открытом тексте сохраняются в шифртексте и видны наблюдателю.

**Обзор режима простой замены с зацеплением**

В режиме CBC шифртекст текущего блока зависит от шифртекста предыдущего блока. В  отличие от режима ECB, где каждый блок открытого текста шифруется сам по себе, в режиме CBC к текущему блоку открытого текста и к предыдущему блоку шифртекста применяется операция XOR, а ее результат зашифровывается, порождая текущий блок шифртекста. Что делать с  самым первым блоком открытого текста, для которого нет предыдущего шифртекста? Он объединяется операцией XOR с вектором инициализации IV, т. е. с блоком данных, который используется для инициализации шифра. Во многих режимах, включая и CBC, IV обычно генерируется случайным образом и имеет такой же размер, как блок шифра. Как правило, IV не является секретом и хранится вместе с шифртекстом, потому что необходим для расшифрования. Но иногда, если это возможно и имеет смысл, например в протоколе TLS версии 1.1 и выше, IV передается в зашифрованном виде. Режим работы CBC показан на следующем рисунке

****

Видно, что характерные части исходного изображения не сохраняются при шифровании. Глядя на зашифрованное изображение, невозможно сказать, что было изображено первоначально. Таким образом, режим CBC обеспечивает хорошую защиту шифрования. Он скрывает от наблюдателя как сами исходные данные, так и  связи между их частями. Другие режимы работы, отличные от ECB, также дают похожие случайные изображения. Какой IV использовался в  режиме CBC для данного сообщения, должно быть непредсказуемо, чтобы не пасть жертвой атаки на основе подобранного открытого текста. Проще всего удовлетворить этому требованию, генерируя IV случайным образом. Почти во всех режимах работы, включая CBC, IV должен использоваться лишь для шифрования одного сообщения заданным ключом, повторное использование запрещено. Поэтому можно сказать, что IV обычно является одноразовым значением. Для некоторых режимов работы говорят, что IV формируется из одноразового значения. Режим CBC был предложен в 1976 году и на протяжении многих лет был самым популярным режимом работы для шифрования файлов и  сетевого трафика. Сегодня набирает популярность режим GCM (описан ниже), особенно в сетевых приложениях. Но CBC по-прежнему хорош для шифрования файлов. GCM основан на режиме гаммирования (Counter – CTR), который рассматривается ниже.