GrdCryptEx

Функция(метод) **GrdCryptEx** зашифровывает или расшифровывает блок данных при помощи аппаратно- или программно-реализованного криптог рафического алгоритма (GSII64,AES128 или AES256).

Синтаксис

C

```
int GRD_API GrdCryptEx(
  HANDLE hGrd,
  DWORD dwAlgo,
  DWORD dwDataLng,
  void *pData,
  DWORD dwMethod,
  DWORD        dwIVLng,
  void *pIV,
  void *pKeyBuf,
  void *pContext
  void *pReserved
);
```

hGrd [in]	хэндл, через который будет выполнена данная операция
dwAlgo [in]	номер дескриптора аппаратного алгоритма (GSII64,AES128) или номер программно-реализованного алгоритма (AES256), которым будет проводиться шифрование или расшифрование. Программно-реализованный алгоритм AES256 имеет номер **GrdSC_AES256**.
dwDataLng [in]	длина буфера данных в байтах
pData [in, out]	указатель на буфер данных открытого или зашифрованного текста

dwMethod [in]	следующие флаги позволяют указать режим работы алгоритма и тип операции:

Биты 0-5 - режим работы алгоритма	
GrdAM_ECB	Режим электронной кодовой книги (режим простой замены). Каждый блок открытого текста заменяется блоком шифротекста. Шифрование двух одинаковых блоков даст идентичный результат. Скорость обработки блоков в режиме ECB фиксирована. Недостаток ECB, в сравнении с другими режимами шифрования, — сохранение статистических особенностей открытого текста.
GrdAM_CBC	Режим сцепления блоков шифротекста. Каждый блок открытого текста (кроме первого) побитово складывается по модулю 2 (операция XOR) с предыдущим результатом шифрования.Таким образом, каждый блок зашифрованного текста зависит от всех блоков открытого текста, обработанных до него. Режим CBC лишён недостатка алгоритма ECB, но всё же имеет ряд недостатков с точки зрения безопасности.
GrdAM_CFB	Режим обратной связи по шифротексту (режим гаммирования с обратной связью). Для шифрования следующего блока открытого текста он складывается по модулю 2 с перешифрованным (блочным шифром) результатом шифрования предыдущего блока. Криптостойкость CFB определяется криптостойкостью используемого шифра.
GrdAM_OFB	Режим обратной связи по выходу. В этом режиме открытый текст используются только для конечного сложения. Операции блочного шифра могут быть выполнены заранее, позволяя выполнить заключительное шифрование параллельно с открытым текстом.
Бит 6 - резерв	
Бит 7 - тип операции	
GrdAM_Encrypt	Зашифровать блок данных
GrdAM_Decrypt	Расшифровать блок данных
Биты 8-9: номер блока данных	
GrdSC_First	Первый блок данных
GrdSC_Next	Следующий блок данных
GrdSC_Last	Последний блок данных
GrdSC_All	Один блок данных

dwIVLng [in]	длина вектора инициализации: для GSII64 - 8 байт, для AES128 и AES256 - 16 байт
pIV [in,out]	указатель на вектор инициализации для режимов шифрования CBC, CFB и OFB. Для режима ECB данный параметр игнорируется
pKeyBuf [in]	указатель на ключ шифрования для программно-реализованного алгоритма AES256. Длина ключа 32 байта (**GrdAES256_KEY_SIZE**). При использовании аппаратного алгоритма параметр должен быть равен **NULL**
pContext [in, out]	указатель на буфер контекста шифрования (только для программно-реализованных алгоритмов). Если предполагается шифрование нескольких блоков данных (используются флаги **GrdSC_First** / **GrdSC_Next**), то для контекста должна быть зарезервирована память размером **GrdAES_CONTEXT_SIZE**. При использовании аппаратного алгоритма параметр должен быть равен **NULL**
Reserved	не используется. Параметр должен быть равен **NULL**

GrdE_OK	Операция выполнена успешно.
GrdE_AlgoNotFound	Алгоритма с указанным номером не существует.
GrdE_InvalidArg	Задано недопустимое значение одного из аргументов функции.
GrdE_NoService	Для алгоритма/ячейки сервис не предусмотрен.
GrdE_GPis0	Счетчик алгоритма достиг нулевого значения.
GrdE_InactiveItem	Алгоритм/ячейка находятся в неактивном состоянии, команда не выполнена.
Перейти к списку всех ошибок Guardant API	

Функция **GrdCryptEx** позволяет зашифровать или расшифровать данные с помощью аппаратно- или программно-реализованного симметричного алгоритма шифрования. Если параметр *dwAlgo* соответствует аппаратному алгоритму, вызов переадресуется функции GrdTransformEx. Иначе, если алгоритм реализован программно (установлен старший бит в параметре *dwAlgo*), вызывается соответствующая функция программного шифрования.

Длина буфера данных задаётся параметром *dwLng*. Для режимов ECB и CBC она должна быть кратна размеру блока алгоритма шифрования, в противном случае возвращается ошибка GrdE_InvalidArg. Для режимов CFB и OFB длина может быть произвольной, т.к. в этих режимах открытый или зашифрованный текст складываются по модулю 2 с ключевыми блоками полученными в результате работы операции блочного шифра. Открытый или зашифрованный текст должны находиться по адресу, указанному в параметре *pData*. Если функция выполнена успешно, то по этому адресу будут размещены зашифрованные или расшифрованные данные той же длины. В этом случае функция возвращает GrdE_OK.

Если шифрование выполнялось блоками произвольной длины, то для корректного расшифрования длина и порядок обработки блоков должны сохраняться. При выполнении операций шифрования и расшифрования следует использовать один и тот же вектор инициализации, если предполагается получить исходный открытый текст.

Для аппаратных алгоритмов:

Шифрование выполняется алгоритмом с номером, заданным в параметре *dwAlgoNum*. Этот алгоритм предварительно должен быть создан, в противном случае возвращается ошибка GrdE_AlgoNotFound. Если в дескрипторе алгоритма установлен флаг "уменьшение счетчика", то вычитание счетчика алгоритма происходит при каждом вызове **GrdCryptEx**. При достижении счетчиком нулевого значения, возвращается ошибка GrdE_GPis0. Если при создании алгоритма или в процессе работы он был переведен в неактивное состояние, возвращается ошибка GrdE_InactiveItem.

В целях безопасности, в электронном ключе существует возможность создавать алгоритмы позволяющие только зашифровывать или только расшифровывать данные. Для таких алгоритмов тип операции, передаваемый в параметре *dwMethod,* должен соответствовать типу алгоритма внутри электронного ключа, в противном случае возвращается ошибка GrdE_NoService.

В таких режимах шифрования, как CBC, CFB и OFB функции необходимо передать указатель на вектор инициализации *pIV* и длину вектора инициализации в параметре *dwIVLng*. После выполнения операции буфер, на который указывает *pIV*, будет содержать значение необходимое функции для повторного вызова. Таким образом, начальное значение вектора инициализации будет изменено. Длина вектора инициализации должна быть равна размеру блока алгоритма шифрования. Если длина окажется больше указанной величины, то лишние байты будут проигнорированы. Если длина окажется меньше указанной величины, то исходный буфер будет дополнен нулями. При вызове функции с нулевым указателем на вектор инициализации, ситуация равноценна использованию вектора инициализации состоящего из нулевых значений или вектора инициализации нулевой длины.

Скорость работы функции зависит от быстродействия электронного ключа и длины переданных данных. Следует заметить, что существует максимальная длина пакета данных передаваемых в ключ. Для достижения максимального быстродействия, длина передаваемых функции данных должна быть кратна этой величине. Если длина переданных данных превышает максимальную длину пакета, то поведение функции аналогично её многократному вызову.

Для программно-релизованных алгоритмов:

Ключ шифрования передаётся функции через параметр *pKeyBuf*. Длина ключа фиксирована и равна 32 байтам. Для хранения ключа шифрования и буфера обратной связи (в зависимости от режима работы алгоритма) используется контекст (параметр *pContext*). Если вызов осуществляется с флагом **GrdSC_All**, то параметр *pContext* может быть равен **NULL**. В этом случае функция позволяет зашифровать или расшифровать только один блок данных. Если предполагается использовать многократный вызов функции для шифрования нескольких блоков данных, то память для контекста должна быть зарезервирована заранее, а функцию следует вызывать с одним из установленных флагов **GrdSC_First** или **GrdSC_Next** в параметре *dwMethod,* в противном случае возвращается ошибка GrdE_InvalidArg. Первый вызов всегда следует выполнять с флагом **GrdSC_First**, в этом случае функция считывает ключ шифрования, вектор инициализации и выполняет шифрование. Последующие вызовы (**GrdSC_Next**) позволяют продолжить шифрование используя заданный ключ и режим работы алгоритма.

В таких режимах шифрования, как CBC, CFB и OFB функции необходимо передать указатель на вектор инициализации *pIV* и длину вектора инициализации в параметре *dwIVLng*. Длина вектора инициализации должна быть равна размеру блока алгоритма шифрования, в противном случае возвращается ошибка GrdE_InvalidArg.

Для алгоритмов GSII64

Размер блока алгоритма шифрования равен 8 байт. Максимальная длина пакета 248 байт (**GrdAMRS_GSII64**).

Для алгоритмов AES128

Размер блока алгоритма шифрования равен 16 байт. Максимальная длина пакета 16384 байт.

Для алгоритмов AES256

Размер блока алгоритма шифрования равен 16 байт.

Для алгоритмов ECC160

Для работы с аппаратно-реализованными алгоритмами ECC160 используйте функцию GrdSign.

Для алгоритмов HASH64

Для работы с аппаратно-реализованными алгоритмами HASH64 используйте функцию GrdHashEx.

C#

```
public static GrdE GrdCryptEx(Handle grdHandle, GrdAlgNum algNum, byte[] data, GrdAM method, byte[] iv)
public static GrdE GrdCryptEx(Handle grdHandle, GrdAlgNum algNum, byte[] data, GrdAM method, byte[] iv, byte[]
key)
public static GrdE GrdCryptEx(Handle grdHandle, GrdAlgNum algNum, byte[] data, GrdAM method, byte[] iv, byte[]
key, byte[] context)
```

grdHandle [in]

Тип: Handle

хэндл, через который будет выполнена данная операция.

algNum [in]

Тип: GrdAlgNum, GrdAN, GrdSA

Номер дескриптора аппаратного алгоритма (GSII64,AES128) или номер программно-реализованного алгоритма (AES256), которым будет проводиться шифрование или расшифрование. Для наглядности следует использовать GrdSA.AES256 для программного-реализованного алгоритма AES256. В примерах используются объекты класса GrdAN для передачи номеров демонстрационных алгоритмов.

data [in, out]

Тип: byte [ ]

Массив данных открытого или зашифрованного текста.

method [in]

Тип: GrdAM

Для одновременной передачи методу режима работы алгоритма, типа операции шифрования и номера блока данных (см. описание GrdSC), класс GrdAM содержит перегрузки оператора сложения. Например, чтобы программно зашифровать один блок в режиме электронной кодовой книги, методу достаточно передать GrdAM.ECB + GrdAM.Encrypt + GrdSC.All .

iv [in]

Тип: byte [ ]

Вектор инициализации для режимов шифрования CBC, CFB и OFB:

для GSII64 - 8 байт, для AES128 и AES256 - 16 байт.

key [in]

Тип: byte [ ]

Буфер для передачи ключа шифрования для программно-реализованного алгоритма AES256. Длина ключа 32 байта (GrdAES256.KEY_SIZE).

context [in]

Тип: byte [ ]

Буфер для контекста при использовании программно-реализованных алгоритмов (только в режиме шифрования нескольких блоков данных, когда используется последовательнось GrdSC.First / GrdSC.Next). Для контекста должна быть зарезервирована память размером GrdAES.CONTEXT_SIZE.

GrdE.OK	Операция выполнена успешно.
GrdE.AlgoNotFound	Алгоритма с указанным номером не существует.
GrdE.InvalidArg	Задано недопустимое значение одного из аргументов функции.
GrdE.NoService	Для алгоритма/ячейки сервис не предусмотрен.
GrdE.GPis0	Счетчик алгоритма достиг нулевого значения.
GrdE.InactiveItem	Алгоритм/ячейка находятся в неактивном состоянии, команда не выполнена.
Перейти к списку всех ошибок Guardant API	

Метод **GrdCryptEx** позволяет зашифровать или расшифровать данные с помощью аппаратно- или программно-реализованного симметричного алгоритма шифрования. Если параметр *algNum* соответствует аппаратному алгоритму, вызов переадресуется функции GrdTransformEx. Иначе, если алгоритм реализован программно (установлен старший бит в параметре *algNum*), вызывается соответствующая функция программного шифрования.

Длина буфера данных для режимов ECB и CBC должна быть кратна размеру блока алгоритма шифрования, в противном случае возвращается ошибка GrdE.InvalidArg. Для режимов CFB и OFB длина может быть произвольной, т.к. в этих режимах открытый или зашифрованный текст складываются по модулю 2 с ключевыми блоками полученными в результате работы операции блочного шифра. Открытый или зашифрованный текст должны находиться в массиве передаваемом по ссылке в параметре *data*. Если метод выполнен успешно, то в этом же массиве будут размещены зашифрованные или расшифрованные данные той же длины. В этом случае метод возвращает GrdE.OK.

Если шифрование выполнялось блоками произвольной длины, то для корректного расшифрования длина и порядок обработки блоков должны сохраняться. При выполнении операций шифрования и расшифрования следует использовать один и тот же вектор инициализации, если предполагается получить исходный открытый текст.

Для аппаратных алгоритмов:

Шифрование выполняется алгоритмом с номером, заданным в параметре *algNum*. Этот алгоритм предварительно должен быть создан, в противном случае возвращается ошибка GrdE.AlgoNotFound. Если в дескрипторе алгоритма установлен флаг "уменьшение счетчика", то вычитание счетчика алгоритма происходит при каждом вызове **GrdCryptEx**. При достижении счетчиком нулевого значения, возвращается ошибка GrdE.GPis0. Если при создании алгоритма или в процессе работы он был переведен в неактивное состояние, возвращается ошибка GrdE.InactiveItem.

В целях безопасности, в электронном ключе существует возможность создавать алгоритмы позволяющие только зашифровывать или только расшифровывать данные. Для таких алгоритмов тип операции, передаваемый в параметре *method,* должен соответствовать типу алгоритма внутри электронного ключа, в противном случае возвращается ошибка GrdE.NoService.

В таких режимах шифрования, как CBC, CFB и OFB методу необходимо передать ссылку на массив, содержащий вектор инициализации (параметр *iv*). После выполнения операции данный массив будет содержать значения необходимые методу для повторного вызова. Длина вектора инициализации должна быть равна размеру блока алгоритма шифрования. Если длина окажется больше указанной величины, то лишние байты будут проигнорированы. Если длина окажется меньше указанной величины, то исходный массив будет дополнен нулями. При вызове метода с нулевым указателем на вектор инициализации, ситуация равноценна использованию вектора инициализации состоящего из нулевых значений или вектора инициализации нулевой длины.

Скорость работы метода зависит от быстродействия электронного ключа и длины переданных данных. Следует заметить, что существует максимальная длина пакета данных передаваемых в ключ. Для достижения максимального быстродействия, длина передаваемых методом данных должна быть кратна этой величине. Если длина переданных данных превышает максимальную длину пакета, то поведение метода аналогично его многократному вызову.

Для программно-релизованных алгоритмов:

Ключ шифрования передаётся методу через параметр *key*. Длина ключа фиксирована и равна 32 байтам. Для хранения ключа шифрования и буфера обратной связи (в зависимости от режима работы алгоритма) используется контекст (параметр *context*). Если вызов осуществляется с флагом GrdSC.All, то параметр *context* может быть не задан. В этом случае метод позволяет зашифровать или расшифровать только один блок данных. Если предполагается использовать многократный вызов метода для шифрования нескольких блоков данных, то память для контекста должна быть зарезервирована заранее, а метод следует вызывать с одним из установленных флагов GrdSC.First или GrdSC.Next в параметре *м ethod,* в противном случае возвращается ошибка GrdE.InvalidArg. Первый вызов всегда следует выполнять с флагом GrdSC.First, в этом случае метод считывает ключ шифрования, вектор инициализации и выполняет шифрование. Последующие вызовы (GrdSC.Next) позволяют продолжить шифрование используя заданный ключ и режим работы алгоритма.

В таких режимах шифрования, как CBC, CFB и OFB методу необходимо передать ссылку на массив, содержащий вектор инициализации (параметр *iv*). Длина вектора инициализации должна быть равна размеру блока алгоритма шифрования, в противном случае возвращается ошибка GrdE.InvalidArg.

Для алгоритмов GSII64

Размер блока алгоритма шифрования равен 8 байт. Максимальная длина пакета 248 байт.

Для алгоритмов AES128

Размер блока алгоритма шифрования равен 16 байт. Максимальная длина пакета 16384 байт.

Для алгоритмов AES256

Размер блока алгоритма шифрования равен 16 байт.

Для алгоритмов ECC160

Для работы с аппаратно-реализованными алгоритмами ECC160 используйте метод GrdSign.

Для алгоритмов HASH64

Для работы с аппаратно-реализованными алгоритмами HASH64 используйте метод GrdHashEx.

Java

```
public static GrdE GrdCryptEx(Handle grdHandle, int algoNum, byte[] data, int method, byte[] iv)
public static GrdE GrdCryptEx(Handle grdHandle, int algoNum, byte[] data, int method, byte[] iv, byte[] key)
public static GrdE GrdCryptEx(Handle grdHandle, int algoNum, byte[] data, int method, byte[] iv, byte[] key,
byte[] context)
```

grdHandle [in]

Тип: Handle

хэндл, через который будет выполнена данная операция.

algNum [in]

Тип: int

Номер дескриптора аппаратного алгоритма (GSII64,AES128) или номер программно-реализованного алгоритма (AES256), которым будет проводиться шифрование или расшифрование. GrdAN и GrdSA содержат предопределенные статические объекты, которые могут быть использованы для передачи методу.

data [in, out]

Тип: byte [ ]

Массив данных открытого или зашифрованного текста.

method [in]

Тип: int

Режим работы алгоритма и тип операции.

iv [in]

Тип: byte [ ]

Вектор инициализации для режимов шифрования CBC, CFB и OFB:

для GSII64 - 8 байт, для AES128 и AES256 - 16 байт.

key [in]

Тип: byte [ ]

Буфер для передачи ключа шифрования для программно-реализованного алгоритма AES256 (32 байта).

context [in]

Тип: byte [ ]

Буфер для контекста при шифровании больших массивов данных, которые разбиваются на несколько блоков.

GrdE.OK	Операция выполнена успешно.
GrdE.AlgoNotFound	Алгоритма с указанным номером не существует.
GrdE.InvalidArg	Задано недопустимое значение одного из аргументов функции.
GrdE.NoService	Для алгоритма/ячейки сервис не предусмотрен.
GrdE.GPis0	Счетчик алгоритма достиг нулевого значения.
GrdE.InactiveItem	Алгоритм/ячейка находятся в неактивном состоянии, команда не выполнена.
Перейти к списку всех ошибок Guardant API	

Метод **GrdCryptEx** позволяет зашифровать или расшифровать данные с помощью аппаратно- или программно-реализованного симметричного алгоритма шифрования. Если параметр *algNum* соответствует аппаратному алгоритму, вызов переадресуется функции GrdTransformEx. Иначе, если алгоритм реализован программно (установлен старший бит в параметре *algNum*), вызывается соответствующая функция программного шифрования.

Длина буфера данных для режимов ECB и CBC должна быть кратна размеру блока алгоритма шифрования, в противном случае возвращается ошибка GrdE.InvalidArg. Для режимов CFB и OFB длина может быть произвольной, т.к. в этих режимах открытый или зашифрованный текст складываются по модулю 2 с ключевыми блоками полученными в результате работы операции блочного шифра. Открытый или зашифрованный текст должны находиться в массиве передаваемом по ссылке в параметре *data*. Если метод выполнен успешно, то в этом же массиве будут размещены зашифрованные или расшифрованные данные той же длины. В этом случае метод возвращает GrdE.OK.

Если шифрование выполнялось блоками произвольной длины, то для корректного расшифрования длина и порядок обработки блоков должны сохраняться. При выполнении операций шифрования и расшифрования следует использовать один и тот же вектор инициализации, если предполагается получить исходный открытый текст.

Для аппаратных алгоритмов:

Шифрование выполняется алгоритмом с номером, заданным в параметре *algNum*. Этот алгоритм предварительно должен быть создан, в противном случае возвращается ошибка GrdE.AlgoNotFound. Если в дескрипторе алгоритма установлен флаг "уменьшение счетчика", то вычитание счетчика алгоритма происходит при каждом вызове **GrdCryptEx**. При достижении счетчиком нулевого значения, возвращается ошибка GrdE.GPis0. Если при создании алгоритма или в процессе работы он был переведен в неактивное состояние, возвращается ошибка GrdE.InactiveItem.

В целях безопасности, в электронном ключе существует возможность создавать алгоритмы позволяющие только зашифровывать или только расшифровывать данные. Для таких алгоритмов тип операции, передаваемый в параметре *method,* должен соответствовать типу алгоритма внутри электронного ключа, в противном случае возвращается ошибка GrdE.NoService.

В таких режимах шифрования, как CBC, CFB и OFB методу необходимо передать ссылку на массив, содержащий вектор инициализации (параметр *iv*). После выполнения операции данный массив будет содержать значения необходимые методу для повторного вызова. Длина вектора инициализации должна быть равна размеру блока алгоритма шифрования. Если длина окажется больше указанной величины, то лишние байты будут проигнорированы. Если длина окажется меньше указанной величины, то исходный массив будет дополнен нулями. При вызове метода с нулевым указателем на вектор инициализации, ситуация равноценна использованию вектора инициализации состоящего из нулевых значений или вектора инициализации нулевой длины.

Скорость работы метода зависит от быстродействия электронного ключа и длины переданных данных. Следует заметить, что существует максимальная длина пакета данных передаваемых в ключ. Для достижения максимального быстродействия, длина передаваемых методом данных должна быть кратна этой величине. Если длина переданных данных превышает максимальную длину пакета, то поведение метода аналогично его многократному вызову.

Для программно-релизованных алгоритмов:

Ключ шифрования передаётся методу через параметр *key*. Длина ключа фиксирована и равна 32 байтам. Для хранения ключа шифрования и буфера обратной связи (в зависимости от режима работы алгоритма) используется контекст (параметр *context*). Если вызов осуществляется с флагом **GrdSC_All**, то параметр *context* может быть равен **NULL**. В этом случае метод позволяет зашифровать или расшифровать только один блок данных. Если предполагается использовать многократный вызов метода для шифрования нескольких блоков данных, то память для контекста должна быть зарезервирована заранее, а метод следует вызывать с одним из установленных флагов **GrdSC_First** или **GrdSC_Next** в параметре *method,* в противном случае возвращается ошибка GrdE.InvalidArg. Первый вызов всегда следует выполнять с флагом **GrdSC_First**, в этом случае метод считывает ключ шифрования, вектор инициализации и выполняет шифрование. Последующие вызовы (**GrdSC_Next**) позволяют продолжить шифрование используя заданный ключ и режим работы алгоритма.

В таких режимах шифрования, как CBC, CFB и OFB методу необходимо передать ссылку на массив, содержащий вектор инициализации (параметр *iv*). Длина вектора инициализации должна быть равна размеру блока алгоритма шифрования, в противном случае возвращается ошибка GrdE.InvalidArg.

Для алгоритмов GSII64

Размер блока алгоритма шифрования равен 8 байт. Максимальная длина пакета 248 байт.

Для алгоритмов AES128

Размер блока алгоритма шифрования равен 16 байт. Максимальная длина пакета 16384 байт.

Для алгоритмов AES256

Размер блока алгоритма шифрования равен 16 байт.

Для алгоритмов ECC160

Для работы с аппаратно-реализованными алгоритмами ECC160 используйте метод GrdSign.

Для алгоритмов HASH64

Для работы с аппаратно-реализованными алгоритмами HASH64 используйте метод GrdHashEx.