Guardant®

Система защиты от компьютерного пиратства

Руководство пользователя

Часть 1

Издание 5.1.2.0

Содержание

Принятые обозначения

Чтобы облегчить восприятие текста, в Руководстве используются следующие специальные обозначения:

Клавиши и их комбинации выделены полужирным шрифтом.

Команды меню написаны таким образом, чтобы уровни иерархии разделялись символом «**|**», например, **Редактировать|Добавить**.

Элементы управления выделены полужирным шрифтом, кнопки выделены полужирным шрифтом и скобками: **[Сохранить]**.

Тексты и сообщения программ выделяются
специальным шрифтом.

Для обозначения общего вида функций Guardant API используются шаблоны вида GrdXXXX, где префикс означает «Guardant», а XXXX замещает название операции. К примеру, **GrdTransform** обозначает операцию **Transform**, которая преобразует блок данных при помощи аппаратного алгоритма.

Некоторые важные моменты выделены в специальные врезки. Пожалуйста, прочтите их внимательно – это позволит избежать многих трудностей:

Важная информация

Здесь содержится информация, советы, рекомендации, на которые следует обратить особое внимание.

Дополнительные источники информации

При возникновении вопросов, на которые вам не удалось найти ответа в этом руководстве, рекомендуем обратиться к следующим источникам информации:

Файл README

Файл **ReadMe.htm** находится на дистрибутивном носителе, а после установки ПО Guardant – и на компьютере. Содержит самые последние сведения о новых возможностях, улучшениях и дополнениях в программном и аппаратном обеспечении Guardant.

WWW: http://www.guardant.ru

Сайт разработчика содержит большой объем справочной информации о защите Guardant: часто задаваемые вопросы, методика устранения неисправностей, рекомендации и т. п.

Служба технической поддержки:

e-mail: hotline@guardant.ru
тел. +7(495)105-77-90

Глава 1
Введение

Что такое Guardant?

Guardant (по-русски читается «Гардант») переводится с латинского как «охраняющий», «защищающий».

Программно-аппаратный комплекс Guardant предназначен для защиты программного обеспечения (ПО) от компьютерного пиратства: незаконного копирования, тиражирования и использования приложения.

Основа комплекса защиты – электронный ключ. Это небольшое электронное устройство, которое подсоединяется к компьютеру. Защищаемые приложения «привязываются» к информации, записанной в электронном ключе.

Принцип работы Guardant

В самых общих чертах принцип работы защиты выглядит так:

- Защищенное приложение обращается к электронному ключу, который должен быть подсоединен к компьютеру
- Ключ возвращает приложению некоторую информацию
- Приложение при помощи этой информации идентифицирует электронный ключ. Если ключ имеет верные параметры, приложение продолжает работать. Если же параметры в ключе неверные (не тот электронный ключ), либо ключ вообще не подсоединен, защищенное приложение прекращает свою работу

Преимущества использования Guardant

Современные технологии защиты

Взяв на вооружение любой из электронных ключей семейства Guardant, вы получите защиту, использующую максимум современных технологий среди систем данной ценовой категории.

Широкий диапазон использования

Защита Guardant может быть построена на базе электронных ключей разных типов, различающихся своими характеристиками и возможностями. Выбирая защиту Guardant, вы обязательно найдете оптимальное решение ваших задач.

Легкость обучения и эксплуатации

Для того чтобы эффективно использовать защиту Guardant, не нужны специальные знания. Разумно спроектированная логика работы утилит защиты, мастер автозащиты, дружественные интерфейсы программ, множество примеров и рекомендаций – все это поможет быстро обучиться тонкостям работы с защитой Guardant и достичь хороших результатов в кратчайшие сроки.

Удобство для конечных пользователей

Конечные пользователи оценят те удобства, которыми будет сопровождаться их работа с защищенным приложением. Все, что им нужно будет сделать дополнительно после его установки – это подсоединить к своему компьютеру электронный ключ. Возможность создания резервных копий приложения и инсталляции его на другие компьютеры, отсутствие специальных настроек для работы защиты и проблем при использовании периферийных устройств, высокий уровень совместимости, возможность удаленного программирования ключей – все это делает работу защиты необременительной для конечного пользователя.

Техническая поддержка

Задав любой технический вопрос по телефону, или направив письмо по специальному адресу e-mail, вы получите оперативную и подробную техническую консультацию. Все виды технической поддержки бесплатны.

Оперативное обновление версий ПО

Все пользователи защиты имеют возможность бесплатно обновлять ПО, в том числе через Интернет, загружая его с сайта компании. Воспользуйтесь системой бесплатных обновлений ПО – и вы можете быть уверены в том, что постоянно используете наши новейшие технологии защиты.

Защитные свойства Guardant

Микропроцессорная технология

Использование микроконтроллера в качестве базового элемента конструкции резко усложняет логику работы ключа, делает ее более гибкой и интеллектуальной. В результате электронный ключ обогащается новыми возможностями, повышается его надежность и совместимость, а также стойкость защиты к взлому.

Аппаратные алгоритмы

Правильное использование аппаратных алгоритмов ключей Guardant обеспечивает надежную защиту от эмуляции – наиболее мощного на сегодняшний день метода взлома защиты. Аппаратные алгоритмы также позволяют значительно усилить стойкость защиты приложения, т. к. вводят в нее элемент кодирования данных.

Защищенный протокол обмена с электронным ключом

Применение таких методов защиты протокола, как кодирование, «зашумление», обмен хаотическими данными, случайное изменение параметров, делает протокол обмена с ключами Guardant особенное стойким к взлому.

Кодирование защищаемого приложения

Вы можете закодировать тело защищаемого приложения. Это хороший метод борьбы со средствами статического анализа (декомпиляции и дизассемблирования).

Использование псевдокода для защиты драйверов и Guardant API от анализа

Исполняемый код драйверов и Guardant API выполняется в виде псевдокода на виртуальной машине. Эта технология затрудняет как статический, так и динамический анализ, а также позволяет контролировать целостность кода.

Семейство электронных ключей Guardant

На момент написания Руководства линейка Guardant состоит из электронных ключей следующих типов:

Guardant Stealth III

Локальный электронный ключ нового поколения, разработанный с применением передовых технологий. Построен на базе микроконтроллера, имеет 2048 байтов EEPROM-памяти. Выпускается только в варианте USB.

В Guardant Stealth III реализованы симметричный алгоритм GSII64 для кодирования и декодирования данных внутри ключа и алгоритм хэширования HASH64. Технология защищенных ячеек расширяет возможности по хранению и управлению данными и аппаратными алгоритмами. Технология Trusted Remote Update (TRU) обеспечивает защищенный протокол удаленного перепрограммирования ключа на аппаратном уровне.

Guardant Net III

Сетевой электронный ключ нового поколения, разработанный с применением передовых технологий. Построен на базе микроконтроллера, имеет 2048 байтов EEPROM-памяти. Выпускается только в варианте USB.

Обеспечивает защиту и лицензирование сетевых Win32-приложений в любых локальных сетях с интерфейсами TCP/IP и NetBIOS. Для работы защищенного сетевого продукта достаточно одного Guardant Net III на локальную сеть.

Поддерживает все возможности Guardant Stealth III.

Guardant Stealth II

Локальный электронный ключ, разработанный с применением передовых технологий. Построен на базе микроконтроллера, имеет 256 байтов EEPROM-памяти. Выпускается в USB- и LPT-вариантах.

В Guardant Stealth II реализован симметричный алгоритм GSII64 для кодирования и декодирования данных внутри ключа, при этом сохранена функциональная совместимость с Guardant Stealth.

Guardant Net II

Сетевой электронный ключ на базе микроконтроллера для LPT- или USB-порта IBM-совместимого компьютера. Поддерживает все возможности Guardant Stealth II. Обеспечивает защиту и лицензирование сетевого ПО в любых локальных сетях с интерфейсами TCP/IP и NetBIOS. Не требует специальной настройки в сети. Для работы защищенного сетевого продукта достаточно одного Guardant Net II на локальную сеть.

Guardant Fidus

Локальный электронный ключ на базе микроконтроллера для USB- и LPT-портов IBM-совместимого компьютера. Является компромиссным решением, объединяющим в себе доступную цену и характеристики, присущие ключам семейства Guardant. Имеет 256 байт энергонезависимой памяти для хранения данных пользователя и 8 предопределенных аппаратных алгоритмов Y=F(X).

Комплект разработчика Guardant

Комплект разработчика Guardant позволяет разработчику ПО программировать электронные ключи и встраивать модули защиты в приложения.

В Комплект разработчика Guardant входит:

Электронный ключ Guardant

Один электронный ключ выбранного типа и интерфейса.

Программное обеспечение Guardant

Программное обеспечение для работы со всеми электронными ключами Guardant и для защиты приложений любого типа. Поставляется на диске CD-R.

Листовка с кодами доступа

В процессе инсталляции программного обеспечения Guardant необходимо ввести коды доступа к ключам. Без ввода кодов доступа Комплект разработчика установить невозможно. Рабочие коды в запечатанном конверте входят в состав Комплекта разработчика. Рабочие коды уникальны для каждого пользователя защиты.

ПО Guardant при установке на компьютер настраивается на коды доступа к ключам конкретного пользователя и может работать только с этими ключами. Таким образом, ни один пользователь защиты не может иметь доступ к ключам других пользователей.

Документация и информационные материалы

Печатное Руководство пользователя, список поддерживаемых языков программирования, инструкция по эксплуатации электронных ключей и подобные информационные и рекламные материалы.

Демонстрационный Комплект разработчика

Демонстрационный Комплект разработчика позволяет в полном объеме изучить и оценить возможности системы защиты Guardant. Единственным его отличием от рабочего Комплекта разработчика являются демонстрационные коды доступа. Они опубликованы и общеизвестны, поэтому демо-комплект нельзя использовать для защиты программного обеспечения в коммерческих целях.

Демонстрационные коды доступа:

Код \ вид	Символьный	Десятичный	Шестнадцатеричный
Public:	DEMONVK	1368487351	519175B7h
Private Read:	DEMORDO	1368487493	51917645h
Private Write:	DEMOPRF	1368487427	51917603h
Private Master:	DEMOMST	1368487308	5191758Ch

В дальнейшем можно бесплатно обменять демо-комплект на рабочий и получить собственные уникальные коды доступа. Все электронные ключи, приобретенные впоследствии, будут содержать эти рабочие коды.

Глава 2
Приступая к работе

Системные требования

Комплект разработчика Guardant предъявляет следующие минимальные требования к программному и аппаратному обеспечению:

- IBM-совместимый компьютер
- Стандартный USB- или LPT-порт
  (в зависимости от интерфейса используемых ключей)
- 32-битная операционная система Windows Vista/2003/XP/2000
- Свободное место для установки на жестком диске ~40 Мб.

Установка и обновление Комплекта разработчика

Установка программного обеспечения

Вставьте компакт-диск Guardant в CD-ROM компьютера. Установка ПО Guardant начнется автоматически. (Если автозапуск запрещен, то запустите программу AutoRun.exe из корневого каталога компакт-диска Guardant). В появившемся окне выберите элемент **Установить Комплект разработчика**.

Затем, руководствуясь указаниями программы-установщика, выполните следующие действия:

- Определите каталоги для размещения ПО Guardant
  и тип установки
- Введите коды доступа и контрольное число
  (если Комплект разработчика демонстрационный, то для автоматического ввода кодов достаточно нажать кнопку **[Демо]**)

Важная информация

При установке комплекта разработчика Public Code вводится в символьном виде, а остальные коды – в виде десятичных или шестнадцатеричных чисел (выделены **жирным шрифтом**).

- Выберите типы ключей и защищаемых приложений
- Выберите общие компоненты комплекта разработчика
- Выберите компоненты Guardant API
  и утилиты Комплекта разработчика
- После установки ПО перезагрузите операционную систему

Важная информация

Все компоненты, которые могут понадобиться для работы с электронными ключами, уже отмечены по умолчанию. Не рекомендуется изменять конфигурацию ПО без особой надобности

В процессе установки ПО на диске в рабочем каталоге Guardant будет создан файл **NvCodes.dat** с информацией о кодах доступа к ключу. Этот файл нужен для работы утилит автоматической защиты и программирования ключа, за исключением утилит дистанционного программирования, предназначенных для конечных пользователей.

Важная информация

Файл **NvCodes.dat** необходим только тем утилитам ПО Guardant, которыми пользуется разработчик. Ни защищенные приложения, ни какие-либо утилиты, предназначенные для конечных пользователей, не нуждаются в этом файле. Ни в коем случае нельзя передавать файл **NvCodes.dat** недоверенным лицам или организациям!

Также в процессе работы программы **Setup.exe** в систему будут установлены драйверы Guardant. Работа с ключом без драйвера невозможна.

Важная информация

1. Во время установки драйверов все приложения должны быть закрыты во избежание ошибки разделения файлов.

2. Пользователь, работающий с Windows Vista/2003/XP/2000, должен обладать правами администратора системы, иначе установка драйверов будет невозможна.

По окончании процесса установки ПО в разделе **Программы** меню **Пуск** будет создана программная группа **Комплект разработчика Guardant 5 – "%Public Code%"**.

Для выполнения большинства действий с электронными ключами удобно вызывать утилиты из программной группы или из оболочки **Guardant Интегратор**.

Оболочка **Guardant Интегратор** предназначена для обеспечения быстрого доступа ко всем наиболее часто используемым элементам программного обеспечения Guardant.

Обновление программного обеспечения

Для повышения уровня защищенности приложений, получения новых возможностей защиты и предотвращения возможных конфликтов с программным и аппаратным обеспечением необходимо регулярно обновлять программное обеспечение Guardant.

Важная информация

Обновлять комплект разработчика можно только на тех компьютерах, на которых он уже был инсталлирован.

Обновление с диска

Новый релиз ПО можно получить в офисе **компании Актив** или ее дилеров.

Чтобы обновить ПО Guardant, выполните следующие действия:

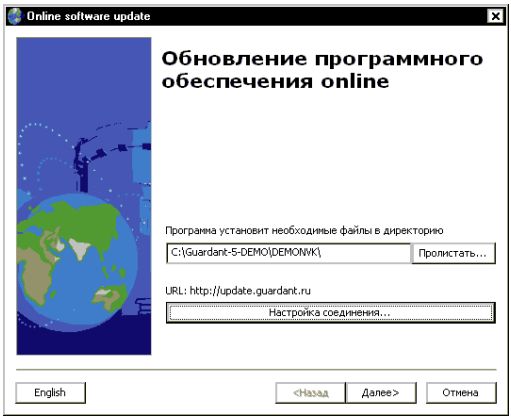
- Подключите носитель с новой версией ПО Guardant к компьютеру
- Выберите элемент программной группы Guardant **Обновление Комплекта разработчика**
- Укажите каталог, в котором находятся файлы обновления
- Выберите один из файлов и нажмите кнопку **[OK]**

Произойдет автоматическая разархивация всех файлов, находящихся в указанной директории, и обновление ПО.

Обновление в режиме On-line

Чтобы выполнить обновление ПО Guardant через Интернет, воспользуйтесь утилитой **GSoftUpd.exe**:

- Установите интернет-соединение
- Выберите элемент программной группы Guardant **Обновление Комплекта разработчика On-Line**
- Появится окно **GSoftUpd.exe**, в котором можно определить директорию размещения файлов обновления и настроить параметры интернет-соединения. Для продолжения нажмите на кнопку **[Далее]**:



- Утилита автоматически загрузит с сайта **компании Актив**) информацию о последнем релизе ПО Guardant

- Выберите из списка в левой части диалогового окна компоненты, которые необходимо загрузить (отсутствующие или устаревшие компоненты заранее отмечены флажками), и нажмите на кнопку **[Загрузить]**
- Запустите программу установки (кнопка **[ДА]** или **[Готово]**)

Важная информация

Загруженные файлы обновления сохраняются. Их можно использовать повторно, например, при аварийном завершении сеанса связи.

Установка электронного ключа

Установка USB-ключа

- Установите драйвер ключа (см. раздел **Драйверы Guardant**)
- Подсоедините ключ к USB-порту

Важная информация

Под ОС Windows XP/2003 после установки драйверов и первого присоединения USB-ключа запускается Мастер нового оборудования. Для корректной работы ключа необходимо выполнить инструкции Мастера (причем не следует обращать внимание на предупреждение о том, что драйверы не тестировались на совместимость с XP, и предложение прекратить установку).

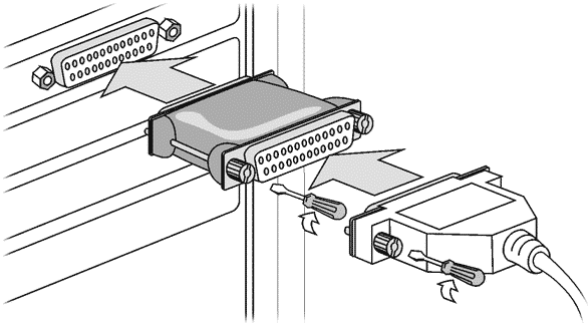
Установка LPT-ключа

- Выключите питание компьютера и периферийных устройств, подключенных к параллельному порту (принтера, сканера)

Важная информация

LPT-ключи Guardant можно подсоединять к работающему компьютеру только при отсутствии периферийных устройств на данном порту

- Отключите периферийное устройство от параллельного порта. Если на компьютере имеется несколько параллельных портов, можно подключать электронный ключ к любому из них
- Разъем электронного ключа типа «вилка» подключите к параллельному порту компьютера (разъем типа «розетка» на задней панели системного блока) и заверните крепежные винты при помощи отвертки. Подключение должно быть плотным и без перекосов. Электронные ключи можно подключать каскадно, т.е. последовательно один к другому (до 10 штук), при этом все они будут доступны для использования
- К разъему электронного ключа типа «розетка» подключите периферийное устройство и заверните крепежные винты при помощи отвертки



- Включите компьютер и загрузите операционную систему

Установка нескольких ключей на один параллельный порт

К одному параллельному порту компьютера можно подсоединять несколько электронных ключей, в том числе разных типов. Нет никаких специальных условий, касающихся последовательности каскадного подключения ключей Guardant. Если необходимо вставить в «цепочку» LPT-ключи других производителей, обратитесь к Руководству по их эксплуатации для выявления возможных нюансов подключения.

Устройство электронного ключа

В электронных ключах семейства Guardant реализованы последние достижения в области систем программно-аппаратной защиты.

Основным элементом электронных ключей Guardant является микроконтроллер. Программа, записанная в нем, осуществляет обработку информации и реализует протокол обмена с драйвером.

Все ключи Guardant имеют энергонезависимую память различного объема в зависимости от модели ключа.

Типы электронных ключей

Локальный ключ

Локальный ключ предназначен для работы на локальном компьютере, с ним можно защищать только несетевое программное обеспечение. К локальным ключам относятся Guardant Stealth III / Stealth II / Stealth и Guardant Fidus.

Сетевой ключ

Сетевой ключ имеет более широкие возможности. Он может работать как на отдельных компьютерах, так и на рабочих станциях и серверах в локальной сети. Поэтому с помощью такого ключа можно защищать как несетевое, так и сетевое ПО. Сетевым и являются ключи Guardant Net с I по III поколение.

Коды доступа

Код доступа служит «паролем», по которому защищенное приложение может найти нужный ему ключ и выполнить конкретную операцию. Такой код называется Личным.

В ключах семейства Guardant есть три разных Личных кода доступа:

Личный код для чтения **(Private Read Code)** – для чтения памяти ключа и выполнения аппаратных алгоритмов;

Личный код для записи **(Private Write Code)** – для записи в память ключа;

Личный мастер-код **(Private Master Code)** – для выполнения с ключом специальных операций.

Для того чтобы ключ выполнил ту или иную операцию, защищенное приложение должно прислать ему нужный Личный код.

Кроме того, в каждом ключе Guardant есть ***Общий код***. Его назначение – показать принадлежность ключа тому или иному владельцу. Общий код ключа можно просмотреть утилитой диагностики.

Коды доступа записываются в память ключа на этапе его предпродажной подготовки и не могут быть изменены.

Идентификационный номер (ID) ключа

В каждом ключе семейства Guardant хранится особое 4-байтовое число – идентификационный номер (ID) ключа. ID имеет уникальное значение. Он прошивается на этапе производства ключа и не может быть продублирован или изменен никем, включая разработчиков защиты Guardant. Значение ID может использоваться для «жесткой» привязки защищенного приложения к данному конкретному ключу. Посмотреть ID ключа можно утилитой диагностики, либо утилитой программирования ключей **GrdUtil.exe**.

Память и поля памяти

Все типы ключей Guardant имеют энергонезависимую память для хранения данных. Используется память типа EEPROM, которая обеспечивает сохранность данных в течение порядка 100 лет и не требует для этого никаких источников энергии. Поддерживается неограниченное число сеансов чтения и до 1.000.000 сеансов записи в память ключа.

Для удобства работы память разбита на логические поля. Обобщенно эти поля можно разделить на три типа: специального назначения, общего назначения и свободного назначения.

Более подробно карта памяти ключей Guardant рассматривается во 2-й части Руководства.

Диагностика электронного ключа

Утилита диагностики **GrdDiag.exe** предназначена для обнаружения электронных ключей Guardant, подключенных к компьютеру, проверки их работоспособности, сбора диагностической информации о системе, в которой они работают. На основе полученных данных утилита генерирует отчет для службы техподдержки ***Компании Актив***. Также с помощью **GrdDiag.exe** можно обновить драйверы ключа и программы, необходимые для работы защищенного ПО.

Утилита диагностики интегрирована с драйверами Guardant. По умолчанию в комплекте драйверов поставляется утилита диагностики **GrdDem32.exe**, работающая с демонстрационными ключами. Чтобы производить диагностику ключей с рабочими кодами доступа из утилиты установки драйверов, необходимо выполнить регистрацию **GrdDiag.exe**. Для этого достаточно запустить **GrdDiag.exe** из текущего каталога (драйверы Guardant уже должны быть установлены).

Запуск утилиты с параметром **/reg** позволяет ее зарегистрировать, не запуская интерфейсную часть. Параметр **/unreg** удаляет регистрацию утилиты. Для установки и удаления утилиты диагностики из инсталлятора защищенного приложения можно добавить соответствующие вызовы в сценарий установки.

Утилиту диагностики **GrdDiag.exe** следует включать в комплект поставки защищенного приложения. Она помогает службе технической поддержки более оперативно проводить диагностику, анализируя сгенерированные утилитой отчеты.

Состав программного обеспечения

ПО Guardant функционально делится на несколько частей:

Утилиты

Guardant Integrator, файл **Guardant.exe.** Оболочка для быстрого доступа ко всем часто используемым утилитам Комплекта разработчика. Все компоненты Интегратора снабжены всплывающими подсказками.

Утилита диагностики, файл **GrdDiag.exe.** Утилита, которая позволяет проводить комплексную диагностику электронных ключей, собирать информацию о системе, в которой они работают, подготавливать отчеты для службы техподдержки, обновлять драйверы ключа и программы, необходимые для работы защищенного ПО.

Утилита автоматической защиты, файл **NwKey32.exe.** Утилита для автозащиты готовых Win32-приложений (кроме .NET-сборок).

Мастер автоматической защиты, файл **AutoProtectWizard.exe.** Управляющая оболочка, позволяющая максимально быстро и просто задать нужные опции и произвести автоматическую защиту приложений.

Утилита программирования ключей, файл **GrdUtil.exe.** Утилита для программирования памяти ключей Guardant.

Утилиты дистанционного программирования, файлы **GrdTRU.exe** *и* **GsRemote.exe**. Утилиты для удаленного перепрограммирования ключа на стороне конечного пользователя. **GrdTRU.exe** предназначена для работы с ключами Guardant Stealth III/Net III, **GsRemote.exe** – с ключами Guardant второго и первого поколений.

Сервер Guardant Net, файл **GrdSrv.exe.** Программный сервер, обеспечивающий работу сетевых электронных ключей Guardant в локальных сетях. Сервер служит для обеспечения связи между защищенным приложением и ключом и для контроля состояния электронных ключей с любой рабочей станции.

Утилита обновления ПО Guardant через Интернет, файл **GsoftUpd.exe.** Утилита автоматизацирует процесс обновления ПО Guardant через Интернет. Она выполняет рутинную работу, связанную с определением необходимости обновления отдельных частей ПО, а также их загрузку с сервера для последующей установки.

Интерфейс прикладного программирования (API) Guardant

Набор объектных модулей (.obj, .bin, .lib, .dll) со всеми необходимыми функциями для организации работы с электронным ключом прямо из кода защищенного приложения. Объектные модули комплектуются примерами их использования для популярных языков программирования и средств разработки.

Драйверы электронных ключей

Набор драйверов Guardant для Windows 98/ME и Windows Vista (x86)/2003(x86)/XP(x86)/2000. Программа для их установки и конфигурирования.

Служебные файлы

Служебные файлы, нужные для работы перечисленных утилит.

Драйверы Guardant

Драйверы Guardant необходимы для работы защищенных приложений и всех утилит, обращающихся к электронному ключу.

Поэтому после установки ПО необходимо убедиться, что драйверы Guardant установлены и работают корректно.

Важная информация

1. Во время установки драйверов все приложения должны быть закрыты во избежание ошибки разделения файлов.

2. Пользователь, работающий с Windows Vista/2003/XP/2000, должен обладать правами администратора системы, иначе установка драйверов будет невозможна.

3. USB-ключи Guardant нельзя подключать к компьютеру до установки драйверов.

4. В ОС Windows XP/2003 при ПЕРВОМ после установки драйверов подсоединении ключа к USB-порту В ЛЮБОМ СЛУЧАЕ запустится Мастер нового оборудования. Это штатная ситуация для данных операционных систем. После запуска Мастера следует пройти с ним все этапы инициализации драйверов, это необходимо для корректной работы ключа. Подробная, пошаговая и иллюстрированная инструкция по установке драйверов доступна на сайте www.guardant.ru

Установка драйверов

Как правило, установка драйверов происходит автоматически в процессе инсталляции ПО Guardant.

Чтобы установить (переустановить, обновить) драйверы Guardant, выполните следующие действия:

- Запустите утилиту **GrdDriversRu.msi** из каталога **"%Program Files%\Guardant\Guardant 5\%Public Code%\Drivers\x86\Windows"**
- Следуйте указаниям мастера установки

Важная информация

Под ОС Windows Vista в ходе установки драйверов будет выдано предупреждение:

Необходимо выбрать пункт «Все равно установить этот драйвер»

- После установки драйвера по необходимости перезагрузите операционную систему

Конфигурирование драйверов

При необходимости можно изменять настройку установленного драйвера, используя апплет **Драйверы Giardant** в **Панели управления Windows**. Подробно конфигурирование драйверов рассматривается в главе **Драйверы Guardant**.

Удаление драйверов

Для удаления драйвера воспользуйтесь оснасткой **Установка и удаление программ** в **Панели управления Windows**. Чтобы удалить драйверы, запустите оснастку, выберите из списка установленных программ **Драйверы Guardant** и нажмите на кнопку **[Удалить]**.

Важная информация

В Windows 98/ME драйвер остается активным до перезагрузки операционной системы

Управление драйверами из приложения

Чтобы создать конечным пользователям более комфортные условия работы с защищенными программными продуктами, можно устанавливать, настраивать и удалять драйверы Guardant из Windows-приложений.

Для обеспечения такой возможности необходимо при разработке приложений использовать функции API для работы с драйверами (см. описание этих функций в главе **Guardant API**).

Программирование электронного ключа

Утилита программирования GrdUtil.exe

Чтобы ключ мог выполнять действия, требующиеся для функционирования системы защиты, его необходимо запрограммировать.

Программирование электронных ключей Guardant осуществляется при помощи утилиты программирования **GrdUtil.exe**.

Утилита GrdUtil.exe (редактор памяти ключей) позволяет считывать из памяти и записывать в память ключа нужную информацию, создавать, редактировать и удалять логические поля и аппаратные алгоритмы, устанавливать и снимать аппаратные запреты на доступ к областям памяти, подготавливать данные для дистанционного программирования.

Редактор GrdUtil.exe также используется при подготовке данных для защиты приложений: с его помощью можно создавать таблицы (массивы) ответов аппаратных алгоритмов, кодировать данные методом взаимообратного преобразования, тестировать выполнение функций API с заданными параметрами.

Для запуска утилиты GrdUtil.exe выберите элемент **Программирование ключей** программной группы Guardant.

Работа с утилитой программирования ключей GrdUtil.exe описана в главе **Программирование электронных ключей**.

Важная информация

При предпродажной подготовке электронных ключей Guardant в память ключа записывается информация, которую можно использовать в учебных и диагностических целях. Для разработки систем защиты настоятельно рекомендуется разрабатывать и записывать в память ключей собственные данные, включая дескрипторы аппаратных алгоритмов и защищенных ячеек.

Использование аппаратных алгоритмов – это необходимое условие создания стойкой защиты.

Дистанционное программирование

Дистанционное (или удаленное) программирование применяется для оперативного обновления памяти электронных ключей Guardant. Этой функцией пользуются в случаях, когда требуется изменить условия эксплуатации защищенной программы для конечного потребителя (например, продлить аренду, разрешить работу с новой версией программы и т. п.). Для этого необходимо соответствующим образом модифицировать определенный участок памяти ключа.

Удаленное программирование существенно экономит время разработчикам и конечным пользователям программ.

В электронных ключах Guardant Stealth III реализована новая технология удаленного программирования – Trusted Remote Update (TRU), использующая аппаратное шифрование и проверку подлинности данных.

Защита программы

Guardant позволяет создавать системы защиты любого уровня сложности, в том числе и такие, взлом которых будет невозможен или экономически невыгоден. Программное обеспечение Guardant предоставляет эффективные инструменты для обеспечения защиты любого программного продукта.

Стойкость и надежность защиты напрямую зависит от степени продуманности и правильности установки системы защиты. Здесь описываются основные действия, которые необходимо выполнить для установки защиты.

Системы защиты, основанные на использовании электронных ключей, могут выполнять ряд проверок различной степени сложности. Самые простые – проверки наличия ключа с заданными свойствами. Они выполняются быстро и могут использоваться достаточно часто.

Более сложные проверки используют преобразование информации при помощи электронных ключей. Поскольку ключ является интеллектуальным устройством, он может выполнять преобразование информации по специальным алгоритмам.

Разработчики системы защиты могут сами создавать дескрипторы алгоритмов и записывать их в память ключа при помощи утилит программирования (кроме ключей Guardant Fidus). Такая возможность делает уникальной каждую систему защиты, использующую алгоритмы.

ПО Guardant поддерживает два метода защиты:

- Автоматическая защита готовых приложений
- Защита при помощи Guardant API

Автоматическая защита

Метод основан на обработке готовых приложений утилитой автоматической защиты из состава ПО Guardant. В результате приложение привязывается к электронному ключу и получает защиту от отладчиков и дизассемблеров. Автозащита имеет целый ряд опций (режимов), которые позволяют настроить приложение на параметры электронного ключа (привязать его к ID, серийному номеру и т. д.), защитить его от компьютерных вирусов, ограничить число запусков или время работы приложения, закодировать его. Также

поддерживается возможность периодической проверки ключа, режимы ограничения количества запусков или времени работы приложения и другие опции.

Главное преимущество метода – в малом времени установки защиты. На сам процесс требуется всего несколько секунд. Кроме того, для установки защиты не нужны никакие специальные знания. Этот метод применим даже в тех случаях, когда отсутствует исходный код приложения, и нет возможности разработать систему защиты на базе Guardant API.

Главный недостаток метода в том, что он не может обеспечить приложению достаточной защищенности. Сама суть метода указывает на то, что защита не может составлять с приложением единого целого. Она ставится на уже готовое приложение, как бы «приклеивается» к нему, поэтому есть вероятность того, что она может быть снята хакером. Кроме того, метод не может обеспечить нестандартной логики работы защиты, что очень важно для повышения стойкости к взлому.

Схема работы автоматической защиты:

- Утилита автоматической защиты вписывает в тело защищаемого приложения исполняемый модуль – внутреннюю вакцину.
- В момент запуска приложения внутренняя вакцина вызывает из отдельного файла внешнюю вакцину **GrdVkc32.dll**.
- Внешняя вакцина производит необходимые проверки и преобразования и запускает защищенное приложение

Персонализация автоматической защиты

При автоматической защите готовых приложений можно персонализировать систему защиты для конкретного ключа или конкретного приложения. Также возможно разрешать или запрещать запуск отдельных модулей в программных комплексах, состоящих из нескольких исполняемых файлов.

Уникальный идентификатор электронного ключа	Каждый ключ Guardant имеет собственный уникальный идентификатор. Проверяя его, можно обеспечить привязку приложения к единственному уникальному ключу.
Серийный номер электронного ключа	Серийный номер записывается в ключ при помощи утилит программирования. Может служить для идентификации конкретной копии программного продукта.
Версия программного продукта	Максимальный номер версии программного продукта, который будет работать с данным электронным ключом. Записывается в ключ при помощи утилит программирования.

Битовая маска	Если программный продукт состоит из нескольких самостоятельных программных модулей, битовая маска позволит выборочно разрешать или запрещать пользователю запуск определенных модулей из состава программного комплекса. Записывается в ключ при помощи утилит программирования.
Номер программы	Если есть несколько программных продуктов, защищаемых электронными ключами, этот номер может быть использован для идентификации конкретного программного продукта. Записывается в ключ при помощи утилит программирования.

Установка ограничений на работу защищенной программы

Для создания приложений, которые могут иметь ограниченный срок работы или количество запусков, можно воспользоваться специальными счетчиками. Эти возможности обычно используются для разработки программных продуктов, которые сдаются в аренду, демонстрационных версий программ и в других случаях.

Ограничение времени работы приложения	Определяет максимальное фактическое время работы приложения. Может корректироваться удаленно, вплоть до снятия ограничения
Ограничение количества запусков приложения	Определяет максимальное количество запусков приложения. Максимальное количество – 65534. Записывается в ключ при помощи утилит программирования. Может корректироваться удаленно, вплоть до снятия ограничения
Периодическая проверка наличия электронного ключа	Можно проверять наличие электронного ключа не только при запуске, но и в течение всего сеанса работы защищенного приложения через заданные промежутки времени

Утилита автоматической защиты

Для автоматической защиты приложений Win32 используется утилита **NwKey32.exe**.

Для установки автоматической защиты необходимо, чтобы следующие файлы находились в одном каталоге:

- Защищаемое приложение
- Утилита автоматической защиты **NwKey32.exe**
- Внешняя вакцина **GrdVkc32.dll**
- Файл с сообщениями об ошибках **NwKey32.msg**
- Файл **NvCodes.dat** с информацией о кодах доступа

Установка автоматической защиты

Запуск утилиты автозащиты из командной строки или BAT-файла

Формат вызова:

nwkey32.exe [опции] [путь]список_файлов

или: **nwkey32.exe [опции]@[путь]filename.fil**

Использование Мастера автоматической защиты

Для упрощения процесса установки автоматической защиты воспользуйтесь утилитой **AutoProtectWizard.exe** (элемент **Мастер автоматической защиты** программной группы Guardant).

После запуска **Мастера автоматической защиты**, руководствуясь указаниями утилиты, выполните следующие действия:

- Выберите исполняемый файл, который нужно защитить
- Выберите типы электронных ключей, с которыми будет работать приложение
- Установите опции автозащиты для выбранного приложения
- При необходимости можно сохранить установленные настройки в файле специального формата *.apw. Для этого нажмите на кнопку **[Сохранить конфигурацию].** Конфигурация сохранится в файле по умолчанию awizard.apw. При необходимости сохранить конфигурацию в файле с другим именем используйте меню **Файл|Сохранить конфигурацию как...**
- Для завершения процесса защиты выбранного файла нажмите на кнопку **[Защитить]**

Полностью возможности автозащиты и работа с Мастером автоматической защиты описаны в главе **Автоматическая защита**.

Важная информация

Для работы защищенного программного продукта необходимо, чтобы исполняемый файл и внешняя вакцина находились в одном каталоге

Защита при помощи Guardant API

Метод основан на использовании специальных функций Guardant API, собранных в объектных модулях. Функции API обеспечивают выполнение с ключом любых операций: поиска, чтения и записи памяти, установки аппаратных запретов, кодирования данных при помощи аппаратных алгоритмов и т. д. Для установки защиты по этому методу нужно вставить вызовы функций API в исходные тексты приложения и скомпилировать их с объектными модулями.

Главное преимущество метода в том, что он обеспечивает неизмеримо более высокий уровень защищенности. Защита (при правильной ее установке) образует с приложением неразрывное целое, следовательно, удалить ее хакеру весьма сложно. Функции Guardant API позволяют выполнить с ключом любую операцию, обработать любой доступный участок его памяти – иными словами, возможности по конструированию защиты ограничены только фантазией и трудолюбием разработчика. Можно выстроить любую, даже самую нестандартную логику работы защиты, что значительно осложнит хакеру задачу ее взлома. Наконец, только функции Guard-

ant API дают полную свободу действий по работе с аппаратными алгоритмами ключей.

Создание системы защиты, построенной на Guardant API – задача, допускающая множество различных решений. В силу этого невозможно дать универсальное и детальное, пошаговое описание процесса установки такой защиты. Ниже предлагается лишь общая схема действий, которой следует придерживаться в любом случае:

- Ознакомьтесь с описанием функций Guardant API в Руководстве пользователя (глава **Guardant API**)
- Изучите тестовые примеры программ на соответствующем языке программирования (см. каталог **"%Program Files%\Guardant\Guardant 5\%Public Code%\Samples\x86\Win32"**). В тестах содержатся примеры использования основных функций API
- Разработайте свою систему защиты, используя приобретенные знания и рекомендации, содержащиеся в приложении **Как повысить стойкость защиты при помощи API**

Какой метод выбрать?

Конечно, можно использовать эти методы по отдельности – только автоматическую защиту либо только защиту с помощью API. Однако все сказанное выше заставляет сделать вывод о необходимости совместного использования обоих методов. Только так можно «сложить» достоинства методов и «вычесть», компенсировать их недостатки. Используйте автоматическую защиту для того, чтобы защитить приложение от отладчиков и дизассемблеров, закодировать его тело, укрыть от посторонних глаз вызовы функций API. Кроме того – это прекрасный метод защиты от любопытства людей, не имеющих достаточных навыков взлома защиты.

Однако автоматическая защита должна быть лишь «внешним уровнем обороны». Ядро же ее должна составлять защита при помощи функций API. Именно ей нужно поручить всю основную работу: проверку ключа и реакцию на его отсутствие, работу с памятью и аппаратными алгоритмами и т. д. Особенно важно построить защиту таким образом, чтобы она стала неотъемлемой частью самого приложения, без которой оно просто перестало бы верно работать.

Рекомендации по повышению стойкости системы защиты

При организации системы защиты придерживайтесь следующих основных правил, которые помогут создать более эффективную и стойкую защиту.

- Комбинируйте автоматическую защиту и защиту при помощи функций API
- Используйте аппаратные алгоритмы преобразования данных
- При автоматической защите приложений по возможности используйте опции:
  - Кодирования загружаемой части приложения
  - Периодической проверки наличия ключа
- При защите с помощью API:
  - Не храните коды доступа в теле приложения в явном виде
  - Используйте сложные алгоритмы работы с функциями API
  - Распределяйте проверки по коду приложения
  - Используйте различные проверки с разной вероятностью
  - Задерживайте реакцию приложения на коды возврата функций API
  - Усложняйте логику обработки кодов возврата

Подробная информация о перечисленных правилах и дополнительные приемы повышения стойкости содержатся в приложении **Повышение стойкости защиты. Рекомендации программисту**.

Глава 3
Драйверы Guardant

Для работы защищенных приложений, а также всех утилит, работающих с электронными ключами, требуются драйверы Guardant.

Важная информация

Драйверы Guardant обязательно должны быть в комплекте поставки защищенного продукта!

Установка, конфигурирование, удаление драйверов

Установка

При инсталляции Комплекта разработчика Guardant драйверы автоматически устанавливаются в операционную систему.

Чтобы переустановить драйверы (установить драйверы на другом компьютере), запустите файл **GrdDriversRU.msi** из каталога **"%Program Files%\Guardant\Guardant 5\%Public Code%\Drivers\x86\Windows"** и следуйте указаниям программы установки*.

После запуска инсталлятор запросит каталог назначения, после чего будет произведено копирование и установка драйверов для всех типов электронных ключей Guardant независимо от интерфейса подключения.

Важная информация

Во время установки драйверов ВСЕ приложения и особенно DOS-сессии должны быть закрыты! В противном случае возможны ошибки разделения файлов.

Для Windows Vista/2003/XP/2000 также необходимо, чтобы пользователь, который работает с программой установки, обладал правами администратора системы.

* Если в системе отсутствует Windows Installer (например, в Windows 98), запустите файл **Setup.exe**, находящийся в той же директории. Setup.exe установит Windows Installer и запустит **GrdDriversRU.msi**

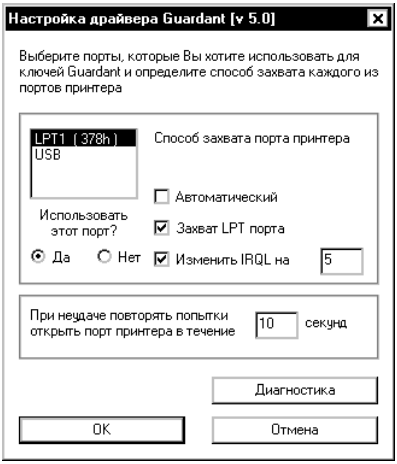
Под ОС Windows Vista в ходе установки драйверов будет выдано предупреждение:

Необходимо выбрать пункт «Все равно установить этот драйвер»

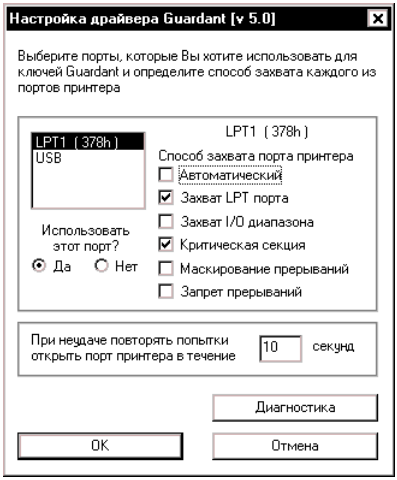
Конфигурирование

Для настройки конфигурации драйверов Guardant воспользуйтесь апплетом **Драйверы Guardant** из **Панели управления** Windows.

Ниже показан вид диалога конфигурирования драйвера. Диалоги в Windows Vista/2003/XP/2000 и 98/ME имеют отличия в интерфейсе (способ захвата LPT портов).



Вид диалога в Windows Vista/2000/ XP/2003



Вид диалога в Windows 98/ME

Процесс конфигурирования драйвера состоит из:

- Задания портов для поиска ключа
- Задания способов захвата LPT-порта
- Установки таймаута (для LPT-порта)

Задание портов для поиска ключа

В левой верхней части окна отображается список найденных LPT- и USB-портов. Можно указать, на каких портах драйвер должен производить поиск ключей. Это бывает полезно для уменьшения времени ответа ключа. Например, если ключ подсоединен к порту LPT2, а к LPT1 подсоединен принтер, то можно запретить поиск ключа для порта LPT1. Если используются только USB-ключи Guardant, можно запретить поиск ключей на всех LPT-портах. По умолчанию поиск разрешен на всех USB- и LPT-портах.

Задание способов захвата LPT-порта

Захват параллельного порта необходим во избежание конфликтов разделения ресурсов параллельного порта. Т. е., если защищенное приложение пытается обратиться к ключу в тот момент, когда другое приложение использует тот же порт, к которому присоединен ключ (например, происходит печать на принтер). Захват порта драйвером производится только на время общения с ключом, после чего порт освобождается.

По умолчанию утилита конфигурации устанавливает автоматический способ захвата. Остальные способы имеет смысл использовать только опытным пользователям, хорошо знакомым с тонкостями работы операционной системы и периферийных устройств.

Каждому из выбранных параллельных портов можно задать свое сочетание способов захвата.

Установка таймаута

В случае если параллельный порт, к которому драйвер Guardant пытается получить доступ, оказывается занятым, драйвер будет ожидать освобождения порта. Время ожидания драйвера называется таймаутом. В течение этого времени он будет постоянно проверять состояние порта и, если порт так и остался занятым, защищенному приложению будет возвращена ошибка #24 (Port busy).

Значение таймаута в секундах задается в нижней части окна **Настройка драйвера Guardant**. Допустимые значения – от 0,001 до 600 секунд. Значение таймаута по умолчанию – 10 секунд.

Важная информация

Под временем таймаута подразумевается общее время ожидания для всех параллельных портов, а не для каждого из них в отдельности (например, таймаут по умолчанию – 10 секунд всего, а не по 10 секунд для каждого из портов).

Диагностика

По нажатию кнопки **[Диагностика]** вызывается утилита диагностики электронных ключей.

Драйверы Guardant комплектуются демонстрационной версией утилиты диагностики - **GrdDem32.exe**. Она обладает всеми возможностями рабочей утилиты, кроме, собственно, тестирования ключей с кодами доступа, отличными от демонстрационных. Для того чтобы запуск рабочей утилиты диагностики - **GrdDiag.exe** - был возможен из окна конфигурирования драйвера, необходимо выполнить ее регистрацию. Для этого достаточно запустить **GrdDiag.exe** из текущего каталога (драйверы Guardant уже должны быть установлены).

Запуск утилиты с параметром **/reg** позволяет ее зарегистрировать, не запуская интерфейсную часть. Параметр **/unreg** удаляет регистрацию утилиты. Для установки и удаления утилиты диагностики из инсталлятора защищенного приложения можно добавить соответствующие вызовы в сценарий установки.

Утилиту диагностики **GrdDiag.exe** следует включать в комплект поставки защищенного приложения. Она помогает службе технической поддержки более оперативно проводить диагностику, анализируя сгенерированные утилитой отчеты.

Удаление

Для удаления драйверов Guardant необходимо воспользоваться меню **Установка и удаление программ** Панели Управления Windows.

Выберите из списка установленных программ **Драйверы Guardant** и нажмите на кнопку **[Удалить]**.

Передача драйверов конечным пользователям

При установке ПО Guardant файлы драйверов, предназначенные для передачи конечным пользователям, копируются в подкаталог **"%Program Files%\Guardant\Guardant 5\%Public Code%\Drivers\x86\Windows"**, а файлы, обеспечивающие управление драйверами из приложений, в подкаталог **"%Program Files%\Guardant\Guardant 5\%Public Code%\Drivers\x86\ Windows"\Driver Installation API**.

Разработчикам предоставляется несколько способов передачи драйверов конечным пользователям.

Дистрибутив установки драйверов

Наиболее простым является способ распространения дистрибутива драйверов Guardant.

Список файлов дистрибутива драйверов Guardant приведён ниже:

GrdDriversRU.msi	MSI-пакет (драйверы Guardant)
Setup.exe	Программа установки
Setup.ini	Файл конфигурации

Дистрибутив состоит из MSI пакета и стандартных файлов системы Microsoft Windows Installer. По умолчанию программа отображает ход процесса установки, но также существует возможность тихой (quiet) установки. Для этого необходимо задать параметр **/quiet** в командной строке. Это может оказаться полезным при запуске **Setup.exe** из других приложений. Более подробная информация опций командной строки приведена в документации Microsoft.

Пакет драйверов MSI

Если для установки программного продукта предполагается использование системы Microsoft Windows Installer, необходимо включить MSI пакет с драйверами Guardant в свой комплект установки без изменений, либо соединить (merge) с одним из уже существующих MSI пакетов.

Установка драйверов из приложений

Драйверы Guardant можно устанавливать, конфигурировать и удалять из Windows-приложений (например, из программы установки программного продукта). Для этого в комплект поставки включена библиотека **GrdDrv.dll**, содержащая все необходимые функции API.

Описания всех используемых констант и структур данных находятся в файле **GrdDrv.h**. Можно включить эти файлы в приложение, использующее библиотеку **GrdDrv.dll**.

Подробно процесс работы с библиотекой описан в примере, написанном на языке C (файл **InstDrvTest.c**).

В процессе установки приложения файлы **GrdDrv.dll** и **GrdDriversRU.msi** (или английская версия **GrdDriversEN.msi**) должны находиться в одном каталоге.

Глава 4
Программирование электронных ключей

Чтобы электронный ключ мог работать с защищенным приложением согласно принятой схеме защиты, он должен быть предварительно запрограммирован. Для программирования ключей Guardant предназначена утилита GrdUtil.exe.

Утилита GrdUtil.exe предоставляет широкие возможности для редактирования памяти ключа и подготовки данных для защиты:

- Работа с маской (внешним образом памяти) ключа:
  - Считывание данных из ключа/запись данных в ключ
  - Создание/редактирование/удаление полей памяти ключа
  - Работа с аппаратными алгоритмами
  - Работа с защищенными ячейками
  - Работа с таблицей сетевых лицензий
  - Работа с дампами, числами, строками и счетчиками
  - Установка аппаратных запретов на чтение/запись участков памяти
  - Сохранение маски во встроенной базе данных или в отдельном файле
  - Получение информации о подсоединенных ключах
- Программирование ключа (в том числе, из командной строки):
  - Запись данных в ключ
  - Пакетный режим записи
  - Локальное и удаленное обновление памяти ключа
- Работа со встроенной базой данных:
  - Ведение базы масок
  - Ведение базы конечных пользователей
  - Хранение прошивок (всех фактов записи маски в ключ)
  - Поиск прошивок по заданным критериям и вывод результатов в виде списка
  - Удаленное и локальное обновление памяти ключа по любому факту прошивки

- Подготовка данных для защиты приложений:
  - Генерация массивов вопросов и ответов аппаратных алгоритмов
  - Кодирование/декодирование данных алгоритмами
  - Проверка выполнения функций Guardant API с заданными параметрами

Утилита GrdUtil.exe поддерживает работу со всеми типами ключей Guardant: Guardant Stealth III / Net III, Guardant Stealth II / Net II, Guardant Stealth / Net* и Guardant Fidus.

Системные требования

Утилита программирования GrdUtil.exe версии 5.x предъявляет следующие минимальные системные требования к аппаратному и программному обеспечению компьютера:

- 32-битная операционная система Windows Vista/2003/XP/2000
- Драйверы Microsoft Jet версии 4.0 или выше, необходимые для доступа к базе данных (как правило, эти драйверы уже установлены в операционной системе)
- 10 M свободного дискового пространства, без учета места для базы данных GrdUtil.exe
- Разрешение экрана – не ниже 800x600, глубина цвета – 16 бит
- Манипулятор типа «мышь»

Основные термины

При работе с GrdUtil.exe используются следующие термины:

Поля памяти ключа: для удобства работы память ключа в утилите GrdUtil.exe логически разделена на отдельные поля. Каждое поле представляет собой участок памяти, содержащий данные, которые относятся к определенному типу. Чтобы записать данные в ключ с помощью утилиты, предварительно их необходимо занести в заранее созданное поле. Структура полей ключа образует его маску.

Маска ключа – внешний образ содержимого памяти электронного ключа Guardant; совокупность полей памяти, их структуры и значений, представленная в удобной для восприятия форме.

Вся работа с данными ключа в GrdUtil.exe происходит на уровне масок: информация, находящаяся в памяти ключа, считывается в маску, маска редактируется и сохраняется, измененные данные записываются из маски в память ключа.

* Guardant Stealth / Guardant Net не выпускаются с февраля 2005 года

Существует несколько подвидов маски, различающихся по месту и типу хранения. Маска ключа может храниться во встроенной базе данных GrdUtil.exe как шаблон или прошивка, а также в отдельном файле.

Прошить ключ, ***запрограммировать ключ*** – записать в память ключа данные шаблона маски, прошивки или файла формата .nsd.

Шаблон маски – маска с заданной структурой и начальным набором данных, а также уникальным именем и / или версией, сохраненная в базе GrdUtil.exe. Это может быть маска для программирования определенной модели ключей Guardant или, к примеру, ключей, предназначенных для определенной версии защищенной программы и т. д. Шаблон маски используется как основа, трафарет для программирования. Каждая запись шаблона в ключ фиксируется в базе данных в виде прошивки.

Прошивка – факт записи маски в память электронного ключа, а также совокупность самих данных записанной маски, автоматически сохраняемых в базе данных GrdUtil.exe в ходе операции записи. Прошивки сохраняются только при работе в режиме базы данных. Каждая прошивка может содержать уникальный набор данных. Прошивки используются для программирования и обновления памяти ключей, в том числе дистанционного.

Файл маски – файл формата *.nsd, в котором может быть сохранена маска ключа. Работа с маской, сохраненной в файле, имеет серьезные ограничения по сравнению с масками, хранящимися в базе данных GrdUtil.exe. Так, к примеру, не сохраняются факты прошивок файлом маски и, поэтому трудно вести учет данных, индивидуальных для каждого ключа (случайные пароли для сервисов защищенных ячеек), нельзя проводить обновление отдельных участков памяти ключа, не затрагивая остальную память.

Режим базы данных GrdUtil.exe – порядок работы утилиты, при котором шаблоны масок, прошивки ключей и список конечных пользователей хранятся во встроенной базе формата Microsoft Access 2000 (*.mdb) и загружаются из нее. Информация из базы данных может быть в любой момент использована для программирования и обновления памяти ключей, в том числе дистанционного.

Разработчик – создатель коммерческого приложения; программист, использующий электронные ключи Guardant для защиты и лицензирования своего продукта.

Конечный пользователь – клиент разработчика, покупатель программного продукта, защищенного ключами Guardant.

Главное окно утилиты GrdUtil.exe

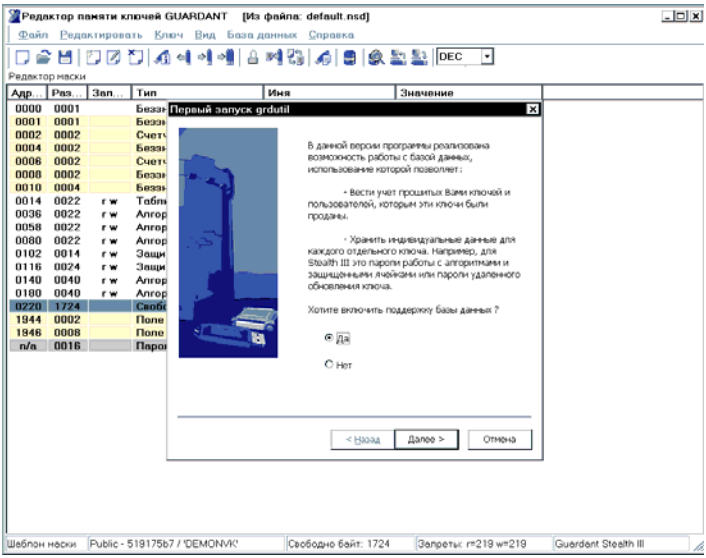
Важная информация

Для успешной работы с утилитой к компьютеру должен быть подсоединен любой электронный ключ Guardant с заданными кодами доступа.

Чтобы начать работу с утилитой программирования ключа, выполните одно из следующих действий:

- Откройте файл GrdUtil.exe, находящийся в каталоге **“%Program Files%\Guardant\Guardant 5\%Public Code%\Bin”**
- Выберите элемент **Программирование ключей Guardant** в оболочке **Guardant Интегратор**
- Выберите элемент **Программирование ключей Guardant** из программной группы **Комплект разработчика Guardant 5 – “%Public Code%”** в меню **Пуск**

На экране появится главное окно утилиты. При первом запуске GrdUtil.exe вместе с главным окном утилиты на экране появляется диалог мастера подключения встроенной базы данных:



Первая страница диалога содержит переключатель, позволяющий включить и настроить базу данных немедленно, или отложить это действие.

Важная информация

Выполнить настройку базы данных можно в любой момент работы с GrdUtil.exe при помощи команды меню **База данных | Настройка базы данных**.

Режим базы данных является основным и рекомендуемым режимом работы с утилитой GrdUtil.exe. Описание базы данных и процесса ее настройки см. в разделе **База данных GrdUtil.exe**.

В главном окне GrdUtil.exe отображаются следующие элементы управления:

- Меню
- Панель инструментов
- Редактор маски
- Инструменты базы данных (в режиме базы данных)
- Прошивки (в режиме базы данных)
- Панель состояния

Меню и панель инструментов. Способы выполнения операций

Меню и панель инструментов GrdUtil.exe имеют следующую структуру:

Раздел меню	Команда меню	Горячие клавиши	Панель инструментов	Описание команды
Файл	Новый	Ctrl+N		Создать маску
	Открыть	Ctrl+O		Загрузить маску из файла
	Сохранить	Ctrl+S		Сохранить маску в файле
	Сохранить как...	Ctrl+A	-	Сохранить маску в новом файле
	Конвертировать маску	Alt+C	-	Перенести данные в маску для другого типа ключей
	Выход	Alt+F4	-	Закрыть приложение
Редактировать	Добавить поле	Ins		Создать поле нужного типа
	Свойства поля	Ctrl+P		Редактировать поле
	Удалить поле	Del		Удалить выбранное поле
Ключ	Запись в ключ	Ctrl+W		Перенести данные из текущей маски в память ключа
	Чтение из ключа	Ctrl+R		Перенести данные из памяти ключа в текущую маску
	Пакетный режим	-		Поочередно перенести данные из текущей маски в память нескольких ключей
	Информация о ключе	Ctrl+I		Получить информацию о ключе(-ах)
	Установить запреты	Ctrl+L		Установить аппаратные запреты на чтение и/или запись с выбранного участка памяти ключа
	Создать отчет алгоритма	Ctrl+Q		Получить ответы аппаратного алгоритма

Раздел меню	Команда меню	Горячие клавиши	Панель инструментов	Описание команды
	Преобразование	Ctrl+T		Кодировать/декодировать данные выбранным аппаратным алгоритмом
	Выполнить функцию Guardant API	Ctrl+F		Проверить корректность выполнения функций Guardant API с заданными параметрами
	Обновление ключа	Ctrl+M		Выполнить удаленное /локальное программирование памяти ключа
	Завершить обновление ключа	Ctrl+E		Завершить удаленное программирование памяти ключа по факту прошивки
Вид	Панель инструментов	Alt+B	-	Показать/скрыть панель инструментов
	Строка статуса	Alt+V	-	Показать/скрыть строку статуса
	Система счисления: BIN, OCT, DEC, HEX	Ctrl+2, Ctrl+8, Ctrl+0, Ctrl+6	-	Выбрать систему счисления: двоичную, восьмеричную, десятичную и шестнадцатеричную соответственно
	Выбрать язык интерфейса	Ctrl+G	-	Выбрать русский/английский язык
База данных	Включить / выключить базу данных	Ctrl+D		Перейти в режим/выйти из режима базы данных GrdUtil.exe
	Получить список прошивок	Alt+L	-	Вывести на экран перечень прошивок, удовлетворяющих заданным критериям
	Загрузить прошивку по числу-вопросу	Alt+F		Загрузить в Редактор маски прошивку, сведения о которой содержатся в запросе на обновление ключа
	Записать маску в базу данных	Alt+W		Сохранить редактируемую маску в базе данных GrdUtil.exe
	Записать маску в базу данных как..	Alt+S		Сохранить редактируемую маску в базе данных под новым именем и/или другой версией
	Загрузить маску из базы данных	Alt+R		Загрузить выбранную маску в Редактор
	Удалить маску из базы данных	Alt+M		Удалить выбранную маску из базы данных
	Выбрать клиента	Alt+U		Выбрать конечного пользователя из базы данных и сделать его текущим
	Добавить клиента	Alt+N		Завести нового конечного пользователя в базе данных
	Информация о пользователе	Alt+I		Вывести на экран учетную запись указанного конечного пользователя

Раздел меню	Команда меню	Горячие клавиши	Панель инструментов	Описание команды
	Удалить клиента	Alt+X		Удалить учетную запись конечного пользователя из базы данных
	Настройка базы данных	Alt+D	-	Запустить мастер настройки базы данных
Справка	Главное окно	F1	-	Открыть главную страницу справочного файла
	Содержание	Ctrl+F1	-	Открыть содержание справочного файла
	Поиск	Alt+F1	-	Открыть справку на вкладке **Поиск**
	О программе	Alt+A	-	Получить информацию о версии GrdUtil.exe

Таким образом, основными способами выполнения операций в GrdUtil.exe являются:

- Выбор команды меню
- Нажатие «горячих» клавиш
- Щелчок мышью по пиктограмме на панели инструментов

Для некоторых операций существуют дополнительные способы выполнения:

- Выбор пункта контекстного меню, которое появляется по щелчку правой кнопки мыши (при этом должно быть выделено нужное поле или строка)

Выбор нужного поля памяти и просмотр его свойств происходит с помощью:

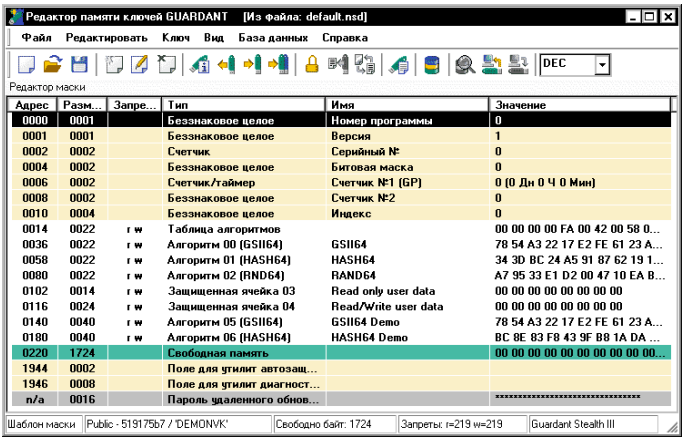
- Нажатия на клавишу пробела после выделения поля клавишами со стрелками
- Двойного щелчка левой кнопки мыши на выделенном поле

Редактор маски

В Редакторе маски происходит вся работа по программированию ключа:

- Загрузка маски в Редактор из базы данных или файла
- Считывание данных из подсоединенного ключа в загруженную маску
- Редактирование структуры и содержимого маски

- Сохранение текущей маски в базе данных или файле
- Запись данных маски в подсоединенный ключ
- Процедура обновления ключа на основе данных загруженной маски



При работе в режиме базы данных к Редактору маски добавляются окна: **Инструменты базы данных** и **Прошивки**, которые рассматриваются в разделе **База данных GrdUtil.exe**.

Редактор маски организован в виде таблицы, строки которой образуют поля памяти ключа (причем каждое поле занимает одну строку), а столбцы – параметры полей:

Параметр поля	**Описание**
Адрес	Адрес поля в режиме UAM
Размер	Размер поля в байтах
Запреты	Аппаратные запреты*, установленные на данную область памяти
Тип	Тип информации, которую содержит поле. Возможные варианты**: аппаратный алгоритм, защищенная ячейка, таблица лицензий, целое число, строка, счетчик, дамп
Имя	Любое подходящее по смыслу название поля
Значение	Отображение содержимого поля

* Детальное описание аппаратных запретов и их обозначений см. в пункте **Установка аппаратных запретов**

** Более подробно каждый тип поля будет рассмотрен далее

Цветовая индикация полей

Для удобства восприятия поля памяти имеют цветовые обозначения:

Цвет	Что означает
Желтый	Поля общего и специального назначения
Зеленый	Область свободной памяти ключа
Белый	Поля, созданные разработчиком
Синий	Выделенное поле
Серый	Поле, содержащее пароль удаленного обновления (Guardant Stealth III / Net III)

Отдельные виды полей могут выделяться шрифтом: так деактивированные алгоритмы и защищенные ячейки обозначаются бледно-серым шрифтом.

Панель состояния

В панели состояния отображается следующая информация о маске и текущем ключе:

- Индикатор маски:
  - Состояние ***Маска***:
    если маска загружена из списка прошивок
  - Состояние ***Шаблон маски***:
    если маска загружена из базы данных или файла
- Общий код доступа в десятичном и символьном виде
- Объем свободной памяти
- Нижняя граница аппаратных запретов на чтение и запись
- Тип ключа

Маска ключа

Совокупность всех полей памяти и их содержимого образуют так называемую ***маску*** – структуру, внешний образ памяти ключа.

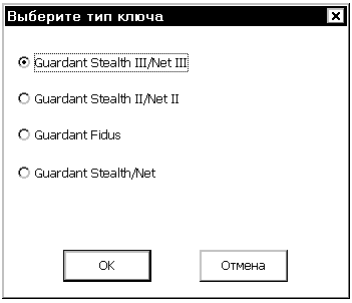
Маска делает работу с памятью ключа более наглядной и удобной. Маска может храниться в базе данных GrdUtil.exe или в специальном файле (с расширением *.nsd).

Маски ключей различных типов (Guardant Stealth III/ Net III, Guardant Stealth II/ Net II, Guardant Stealth/ Net, Guardant Fidus) отличаются друг от друга. Поэтому важно следить, чтобы в ключ записывалась подходящая по типу маска.

Создание маски

При первом запуске утилиты автоматически создается маска по умолчанию. Тип маски по умолчанию зависит от типа электронного ключа, который на тот момент подсоединен к порту.

Чтобы создать новую маску, выполните команду меню **Файл | Новый**. В появившемся диалоге с помощью селектора выберите тип маски для работы с определенным типом ключей:



Важная информация

В GrdUtil.exe не различаются маски для локальных и сетевых ключей. Поэтому при поддержке обеих разновидностей ключей рекомендуется создавать отдельные маски для программирования локальных и сетевых ключей.

После нажатия кнопки **[OK]** новая маска загружается в **Редактор маски** и ей присваивается имя ***default.nsd***, которое потом можно изменить. Имя маски отображается в заголовке главного окна утилиты.

Сохранение маски

1. Сохранение маски в базе данных GrdUtil.exe в виде шаблона

Чтобы сохранить маску в базе данных GrdUtil.exe, выполните команду меню **База данных | Записать маску в базу данных**.

Чтобы сохранить маску в базе данных GrdUtil.exe под другим именем, выполните команду меню **База данных | Записать маску в базу данных как...**

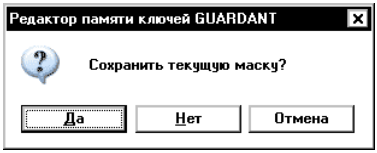
Режим базы данных является предпочтительным режимом работы с GrdUtil.exe. Подробное описание см. в разделе **База данных GrdUtil.exe**.

2. Сохранение маски в базе данных GrdUtil.exe в виде прошивки

При работе в режиме базы данных маска, записываемая в ключ, автоматически сохраняется в базе в виде ***прошивки***. В отличие от шаблона маски в прошивке хранятся данные, индивидуальные для каждого ключа (например, случайные пароли к защищенным ячейкам или пароли доверенного удаленного обновления). Прошивка используется в дальнейшем для программирования и обновления памяти ключей. Подробное описание см. в разделе **База данных GrdUtil.exe**.

3. Сохранение маски в файле

При завершении работы с текущей маской GrdUtil.exe запрашивает подтверждение на ее сохранение, если изменения в маске не были сохранены:



Внешним признаком того, что маска была изменена, и перед закрытием ее необходимо сохранить, служит звездочка после названия маски в главном окне заголовке главного окна GrdUtil.exe:

Чтобы сохранить отредактированную маску в текущем файле, выполните команду меню **Файл | Сохранить**.

Чтобы сохранить маску в новом файле, выполните команду меню **Файл | Сохранить как...** В появившемся стандартном диалоге сохранения файла с другим названием задайте новое имя файла и нажмите на кнопку **[Сохранить]**:

После операции сохранения в заголовке главного окна GrdUtil.exe будет отображаться имя сохраненной маски, а также указание на то, что маска сохранена в файле:

Загрузка маски

Во время запуска GrdUtil.exe происходит автоматическая загрузка маски, которая редактировалась во время последнего сеанса работы с утилитой (или маски default.nsd).

После загрузки маски Редактор показывает список полей и значения, записанные в них на момент создания или последней модификации маски. Чтобы занести в маску реальные значения из подсоединенного ключа, выполните операцию чтения памяти ключа.

1. Загрузка шаблона маски из базы данных GrdUtil.exe

Чтобы загрузить маску из базы данных в Редактор маски, выделите нужную строку в списке масок (окно **Инструменты базы данных**, вкладка **Маски**) и выполните двойной щелчок левой кнопкой мыши или выполните команду **База данных | Загрузить маску из базы данных**.

Режим базы данных является предпочтительным режимом работы с GrdUtil.exe. Подробное описание см. в разделе **База данных GrdUtil.exe.**

2. Загрузка маски из списка прошивок

Любую прошивку, т. е. маску, прошитую в ключ и содержащую уникальные данные (случайные пароли и проч.), можно найти в базе данных по заданным параметрам и загрузить из окна **Прошивки** в Редактор маски.

Чтобы загрузить прошивку в Редактор маски, используйте один из следующих вариантов выполнения операции:

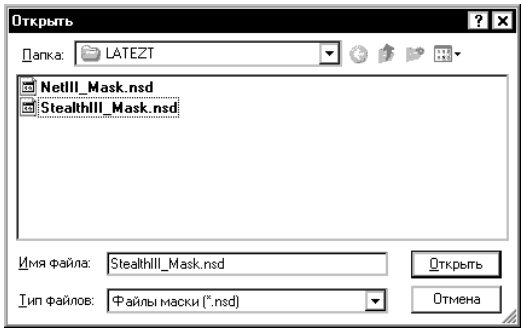
- Двойной щелчок левой кнопкой мыши по нужной записи
- Щелчок правой кнопкой мыши по нужной записи и выбор пункта **Загрузить** в открывшемся контекстном меню

Подробное описание см. в разделе **База данных GrdUtil.exe**.

3. Загрузка маски из файла

Чтобы загрузить маску из файла в Редактор маски, выполните команду меню **Файл | Открыть**. В появившемся стандартном диалоге открытия файла выберите нужный файл из списка и нажмите на кнопку **Открыть**.

Диалог открытия файла маски:



Выбранный файл будет загружен в Редактор маски.

Поля памяти

Для наглядности и удобства работы память ключа в утилите GrdUtil.exe логически разделена на отдельные поля. Каждое поле представляет собой участок памяти, содержащий данные, которые относятся к определенному типу. Чтобы записать данные в ключ с помощью утилиты, предварительно их необходимо занести в заранее созданное поле. Перечень и структура полей ключа образуют его маску.

По своему назначению память ключа логически делится на несколько областей (перечисляются последовательно, начиная с младших адресов):

Область памяти	**Краткое описание**
Поля только для чтения	Доступны для чтения и недоступны для записи функциями Guardant API. Содержат служебную информацию, которая может использоваться в качестве параметров поиска ключа из приложения. Из GrdUtil.exe значения полей можно прочитать с помощью команды меню **Ключ | Информацию о ключе**
Поля специальных операций	Доступны для чтения и выполнения специальных операций Guardant API (GrdInit, GrdProtect). Используются для определения числа аппаратных алгоритмов в ключе и адресов аппаратных запретов. Из GrdUtil.exe значения полей можно прочитать с помощью команды меню **Ключ | Информацию о ключе**
Поля общего назначения	Доступны для чтения и записи. Содержат номер и версию приложения, серийный номер ключа, счетчик запусков и сетевой ресурс и т. д. Используются утилитой автозащиты и функциями Guardant API
Поля свободного назначения	Свободная область памяти ключа. Позволяет хранить любые данные, необходимые для защиты приложения. В этой области памяти можно создавать поля различных типов, редактировать их содержимое и удалять эти поля.
Поля специального назначения	Служебные поля, которые используются утилитами автозащиты, дистанционного программирования и диагностики ключа.

Далее рассматриваются только те категории полей, которые доступны для редактирования из GrdUtil.exe: поля общего и свободного назначения. Остальные области памяти подробно рассматриваются в разделе **Устройство памяти ключей Guardant**.

Поля общего назначения

Группа полей, назначение которых предопределено. Эти поля «закреплены» за автоматической защитой*. Поэтому GrdUtil.exe позволяет только редактировать содержимое полей общего назначения, но не удалять сами поля из маски.

При использовании автозащиты содержимое полей проверяется на соответствие заданным требованиям. «Привязка» к полю осуществляется с помощью определенной опции.

Поля общего назначения можно использовать и функциями Guardant API (к примеру, устанавливать эти поля в качестве критериев поиска ключа приложением).

Таблица характеристик полей общего назначения:

Адрес (UAM)	Название и тип поля	Диапазон значений, DEC	Назначение	Работа с полем из приложения	
				Авто-защита	API
0000	Номер программы, беззнаковое целое	0 - 255	«Привязка» копии приложения к ключу для поддержки нескольких программных продуктов	/UN [=[0x]...]	GrdSetFindMode, GrdGetInfo, GrdRead, GrdWrite
0001	Версия программы, беззнаковое целое	0 - 255	«Привязка» копии приложения к ключу для поддержки новых версий программы	/UV [=[0x]...]	
0002	Серийный номер, счетчик	0 - 65535	«Привязка» конкретной копии приложения к электронному ключу	/US [=[0x]...]	
0004	Битовая маска, беззнаковое целое	0 - 65535	Разрешение/запрет на работу с отдельными, независимыми модулями программного комплекса	/UM [=[0x]..]	
0006	Счетчик запусков (счетчик №1, GP)	0 – 65535	Ограничение времени работы / числа запусков защищенного приложения	/DC, /DT	GrdSetFindMode, GrdGetInfo, GrdRead, GrdWrite GrdDecGP

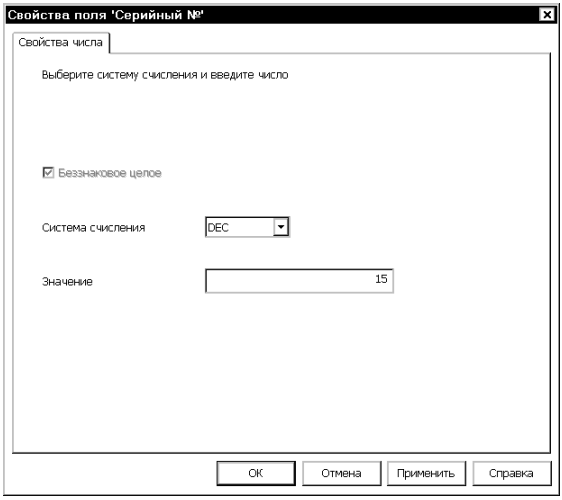
* Кроме поля **Индекс**, которое используется утилитой дистанционного программирования

Адрес (UAM)	Название и тип поля	Диапазон значений, DEC	Назначение	Работа с полем из приложения	
				Авто-защита	API
0008	Сетевой ресурс (счетчик №2), беззнаковое целое	0 - 65535	Лицензирование – ограничение числа одновременно используемых в ЛВС копий приложения. Только для сетевых ключей	/GN3, /GN2	GrdSetFindMode, GrdGetInfo, GrdRead, GrdWrite
0010	Индекс, беззнаковое целое	-	Предназначено для дистанционного программирования ключа. **Нельзя использовать для хранения данных!**	-	-

Редактирование полей общего назначения

Чтобы отредактировать поле общего назначения, выделите его в списке полей и выполните команду **Редактировать | Свойства поля**.

В появившемся диалоге задайте новое значение поля:



Вид диалога **Свойства поля** одинаков для всех полей общего назначения, кроме поля **Счетчик № 1 (GP)**, в котором содержится информация о времени работы (или числе запусков) приложения.

Ограничение времени работы приложения

Электронные ключи Guardant позволяют ограничивать время работы защищенного приложения. Этот режим реализован программным способом: нужное значение записывается в предопределенное поле памяти ключа – **Счетчик №1 (GP)**. Во время работы приложения счетчик каждые 5 минут декрементируется на единицу с помощью специальной функции Guardant API или утилитой автозащиты.

Время работы программы для удобства и наглядности задается в формате ***Дни:Часы:Минуты***. При необходимости можно также задавать продолжительность рабочего дня конечного пользователя приложения.

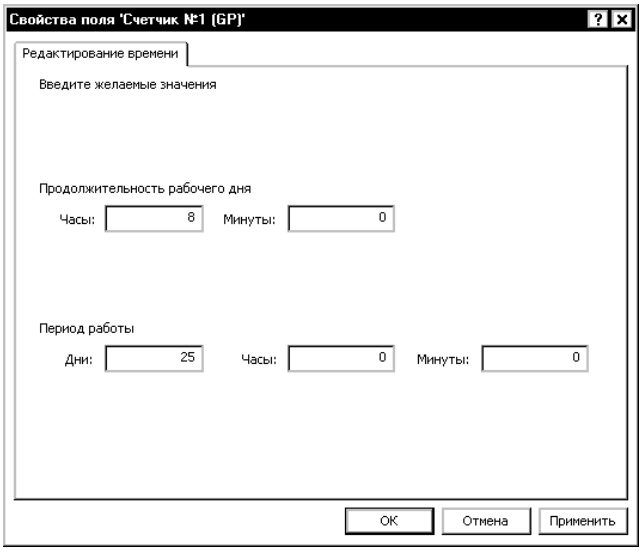
GrdUtil.exe автоматически переводит время работы в числовой формат из расчета 1 единица=5 минутам.

Пример:

Так, при «чистом» времени функционирования программы, равном 27 дням, 7 часам и 30 минутам, и 8-часовом рабочем дне, показание счетчика будет равно 2682 (т. е., 60x(27x8 + 7 + 0,5) / 5=2682).

Чтобы запрограммировать время работы защищенного приложения, выделите в списке полей **Счетчик GP** и выполните команду **Редактировать | Свойства поля**.

В появившемся диалоге **Обновление времени работы** определите значения для «чистого» времени работы приложения, а также продолжительности рабочего дня, если это необходимо:



После нажатия на кнопку **[Применить]** новое значение счетчика отобразится в маске.

Изменить заданное время работы приложения можно путем обновления памяти ключа.

Элементы управления диалога **Обновление времени работы**:

Элемент интерфейса	Назначение
Поля ввода **Продолжительность рабочего дня: Часы и Минуты**	Задать продолжительность рабочего дня пользователя программы с точностью до минуты
Поля ввода **Время работы: Дни, Часы, Минуты**	Задать общее «чистое» время работы программы с точностью до минуты
Флаг **Установить неограниченное время работы**	Не ограничивать время работы приложения. По умолчанию флаг не установлен

Поля свободного назначения

В этой области памяти могут храниться любые данные, необходимые для защиты приложений: в том числе, дескрипторы аппаратных алгоритмов, таблицы лицензий, ключевые слова, наборы данных, константы, участки кода и проч.

Здесь можно создавать поля различных типов, редактировать их содержимое и удалять эти поля.

Важная информация

Все модели современных ключей Guardant (кроме Guardant Fidus), поступающие в продажу, по умолчанию содержат в области полей свободного назначения несколько дескрипторов стандартных аппаратных алгоритмов (и защищенных ячеек в Guardant Stealth III / Net III).

Доступ из приложения к данным, хранящимся в области полей свободного назначения, осуществляется при помощи функций Guardant API.

Типы и основные характеристики полей свободного назначения:

Тип поля	**Размер, байтов**	**Содержимое**	**Аппаратные запреты**	**Работа с полем из приложения (Guardant API)**
Алгоритм	Размер поля определяется размером дескриптора алгоритма	Дескриптор аппаратного алгоритма того или иного типа.	Обязательные запреты на запись и чтение	1. GrdTransform, GrdCrypt, GrdHash, GrdCodeInit 2. GrdPI_Activate, GrdPI_Deactivate, GrdPI_Read, GrdPI_Update 3. GrdTRU_DecryptQuestion, GrdTRU_EncryptAnswer, GrdTRU_ApplyAnswer
Защищенная ячейка	1 – 255 + служебные поля	Дескриптор защищенной ячейки	Обязательные запреты на запись и чтение	GrdPI_Activate, GrdPI_Deactivate, GrdPI_Read, GrdPI_Update
Таблица лицензий (Guardant Net III)	1, 2	1. Реальный сетевой ресурс 2. Количество модулей программного комплекса и их ресурсы лицензий	Обязательные запреты на запись и чтение	GrdPI_Activate, GrdPI_Deactivate, GrdPI_Read, GrdPI_Update
Таблица лицензий (Guardant Net II / Net)		Количество модулей программного комплекса и их ресурс лицензий	Обязательный запрет на запись	GrdRead
Целое число	1, 2, 4, 8	Целое число со знаком или без знака	Запрет на запись (при необходимости)	GrdRead, GrdWrite
Строка	Произвольный размер	Последовательность символов в кодировке ANSI или Unicode	Запрет на запись (при необходимости)	GrdRead, GrdWrite
Счетчик	1, 2, 4, 8	Беззнаковое целое число. Автоматически увеличивается на 1 после каждой операции записи маски в память ключа	Запрет на запись (при необходимости)	GrdRead, GrdWrite
Дамп	Произвольный размер	Последовательность шестнадцатеричных символов	Запрет на запись (при необходимости)	GrdRead, GrdWrite

Расположение полей свободного назначения

Существует определенный порядок следования полей свободного назначения. Это связано с тем, что есть поля, которые должны быть обязательно защищены аппаратными запретами. Причем особенность аппаратных запретов состоит в том, что они могут устанавливаться только с начала области полей свободного назначения (с адреса 14 UAM) и только непрерывным блоком.

Таким образом,

- В начале области полей свободного назначения группируются поля, по умолчанию защищенные аппаратными запретами на чтение и запись: аппаратные алгоритмы, защищенные ячейки и таблица лицензий формата Net III[*] . Причем GrdUtil.exe не позволяет вставить между ними поля, тип которых отличен от вышеперечисленных.
- После полей, защищенных запретами на чтение и запись, располагаются поля, защищенные запретами на запись. К примеру, таблица лицензий формата Guardant Net II/ Net
- Далее, в произвольном порядке могут располагаться поля других типов: целое число, строка, дамп, счетчик. Причем GrdUtil.exe не позволяет вставить между ними поля, которые по умолчанию должны быть защищены аппаратными запретами.

Создание полей

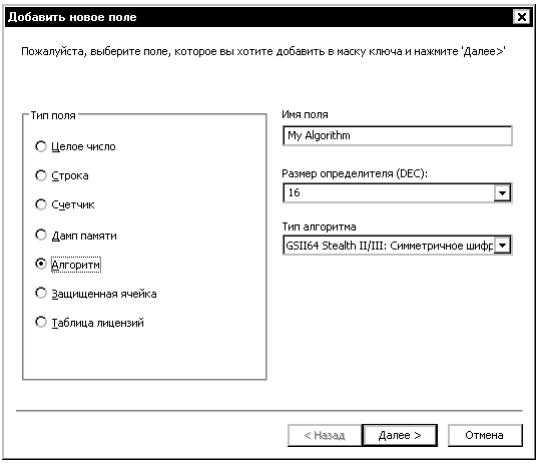
Новое поле добавляется в маску перед полем, выделенным в списке.

Чтобы создать новое поле, выделите в списке поле **Свободная память** (или любое созданное поле из области полей свободного назначения) и выполните команду **Редактировать | Добавить поле**.

В появившемся диалоге **Добавить новое поле** выберите тип поля, укажите его имя и размер.

[*] Два последних типа полей можно создавать только в ключах Guardant Stealth III / Net III

Диалог **Добавить новое поле**:



Элементы управления диалога **Добавить новое поле**:

Элемент интерфейса	**Назначение**
Поле **Имя**	Задать любое подходящее имя для создаваемого поля
Список / поле ввода **Размер в байтах**	Выбрать возможный / задать произвольный размер поля. Размер зависит от выбранного типа поля
Селектор **Тип поля**	Выбрать тип поля из возможных вариантов

В зависимости от типа маски и создаваемого поля интерфейс диалога может незначительно отличаться от приведенного на иллюстрации. К примеру, в маске Guardant Fidus нельзя создать следующие типы полей: аппаратный алгоритм, защищенная ячейка и таблица лицензий.

После заполнения текущего диалога необходимо нажать на кнопку **[Далее]** для перехода к диалогу определения свойств поля.

Более подробно создание полей различных типов и работа с ними рассматриваются в соответствующих разделах.

Редактирование полей

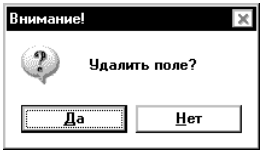
Чтобы изменить содержимое или отредактировать свойства поля, выделите поле в текущей маске и выполните команду **Редактировать | Свойства поля**.

В появившемся диалоге **Свойства поля** произведите необходимые изменения.

Подробно редактирование полей различных типов рассматривается в соответствующих разделах.

Удаление полей

Чтобы удалить поле и его содержимое, выделите поле в маске и выполните команду **Редактировать | Удалить поле**. Удаление поля необходимо подтвердить в соответствующем диалоге:



Важная информация

Любые изменения в маске – создание, удаление или редактирование содержимого полей – будут отражены в памяти ключа только после выполнения команды **Ключ | Запись**.

База данных GrdUtil.exe

Утилита GrdUtil.exe позволяет сохранять ***шаблоны масок***, ***прошивки*** ключей и списки конечных пользователей во встроенной базе формата Microsoft Access 2000 (*.mdb). Информация из базы данных может быть в любой момент использована для программирования и обновления памяти ключей, в том числе дистанционного.

Важная информация

Режим базы данных является основным и рекомендуемым способом работы с GrdUtil.exe.

Каждый факт записи шаблона маски в ключ фиксируется и сохраняется в базе в виде так называемой ***прошивки***. В прошивках могут содержаться данные, уникальные для каждого ключа. Поэтому в дальнейшем прошивки активно используются для обновления памяти ключей, а также для получения статистических отчетов по любым параметрам прошивок.

Если же база данных отключена, то маска, которую нужно записать в ключ, загружается из файла *.nsd, и результаты программирования не сохраняются. Это сужает возможности обновления и программирования памяти ключей и увеличивает временные и трудозатраты на обслуживание ключей.

Настройка базы данных

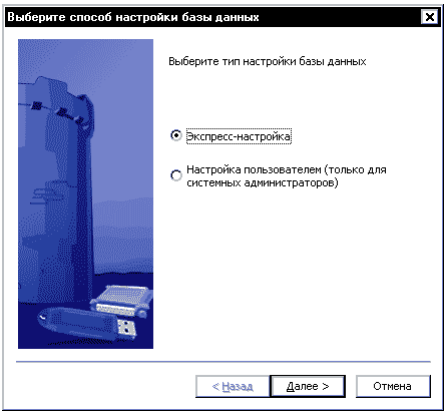
Чтобы настроить базу данных GrdUtil.exe, выполните команду меню **База данных | Настройка базы данных**.

Важная информация

Настройка базы данных возможна только при отключенном режиме базы данных.

На экране появится диалог, выполненный в виде Мастера, состоящего из нескольких страниц. Переход между страницами осуществляется с помощью кнопок **[Далее]** и **[Назад]**, расположенных в нижней части диалога.

1. На первой странице Мастера выберите тип настройки базы данных **Экспресс-настройка**:

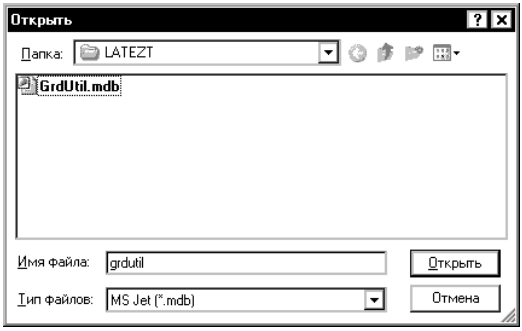


Важная информация

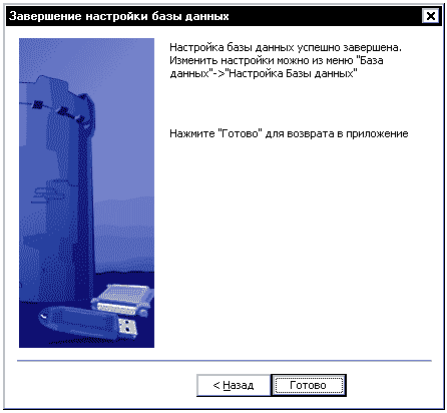
1. Вариант **Пользовательская настройка** в данном руководстве подробно не рассматривается. Он необходим только в ситуации, когда предполагается использование базы данных с какими-либо специфичными настройками, например, при использовании Microsoft SQL Server. В таком случае настройка базы данных должна выполняться системным администратором.

2. В составе ПО Guardant входит SQL-скрипт для создания БД (файл db_setup.sql)

2. После нажатия кнопки **[Далее]** появляется стандартный системный диалог открытия файла, в котором надо указать путь к файлу базы данных по умолчанию **“%Program Files%\Guardant\Guardant 5\%Public Code%\Bin\GrdUtil.mdb”**:



3. После указания файла базы данных появляется последняя страница диалога. Для завершения процедуры настройки базы данных нажмите на кнопку **[Готово]**:



После завершения настройки база данных готова к работе.

Работа в сети

Важная информация

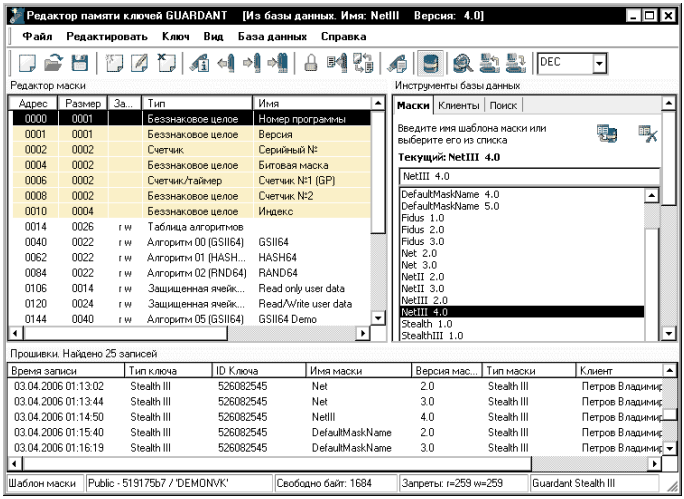
Работа в сетевом режиме является побочной возможностью базы GrdUtil.exe и не тестировалась досконально

База данных GrdUtil.exe поддерживает возможность работы в сети. В этом случае файл *.mdb помещается на сетевой диск. Настройка базы данных для работы в сети происходит аналогичным образом (см. выше).

Включение базы данных

Чтобы включить режим базы данных, выполните команду **База данных | Включить базу данных**.

После выполнения команды к окну Редактора маски добавятся окна **Инструменты базы данных** (справа) и **Прошивки** (внизу):



Инструменты базы данных

В окне **Инструменты базы данных** происходит вся работа по управлению базой шаблонов масок и конечных пользователей.

Окно **Инструменты базы данных** состоит из вкладок:

- Маски
- Клиенты
- Поиск

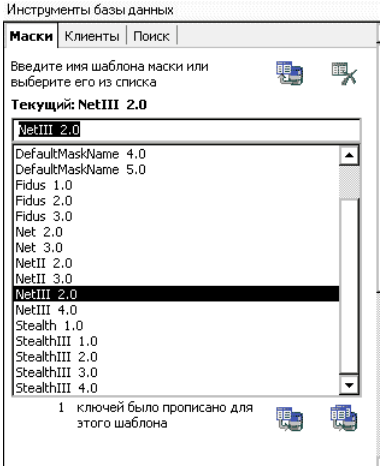
Переключение между вкладками происходит с помощью комбинации клавиш **Ctrl+Tab** или щелчком мышью на заголовке нужной вкладки.

В нижней части секции **Инструменты базы данных** расположены:

- Кнопка **Вывести список прошивок согласно критериям**, при нажатии на которую в окне **Прошивки** выводится список прошивок в соответствии с заданными критериями поиска
- Поле, общее для всех вкладок, в котором отображаются текущие критерии поиска прошивок

Маски

Вкладка **Маски** служит для работы с базой шаблонов масок, которые используются для программирования ключей. Элементы управления вкладки позволяют сохранять шаблоны масок в базе данных, удалять шаблоны и загружать их в Редактор маски:



Элементы управления вкладки **Маски**:

Элемент интерфейса	**Назначение**
Кнопка	Загрузить выбранный шаблон маски в Редактор
Кнопка	Удалить выбранный шаблон маски из базы данных
Поле ввода имени и версии шаблона маски	Ввести имя шаблона маски, чтобы выбрать его из списка
Окно, отображающее список шаблонов	Основное окно вкладки, в котором отображается список шаблонов масок. Под окном выводится информация о количестве ключей, запрограммированных текущим шаблоном
Кнопка	Сохранить шаблон маски в базе данных
Кнопка	Сохранить шаблон маски в базе данных с новым именем и / или версией

Сохранение шаблона маски в базе данных

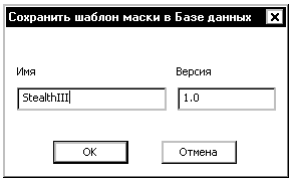
При завершении работы с текущей маской GrdUtil.exe запрашивает подтверждение на ее сохранение, если изменения в маске не были сохранены:

Внешним признаком того, что маска была изменена, и перед закрытием ее необходимо сохранить, служит звездочка после названия маски в заголовке главного окна GrdUtil.exe:

Чтобы сохранить редактируемую маску в базе данных, выполните команду меню **База данных | Записать маску в базу данных**.

1. Если маска ранее уже сохранялась в базе данных, и сейчас редактируется только содержимое полей, без изменения их структуры, то сохранение маски происходит сразу после выполнения команды **База данных | Записать маску в базу данных**, без выдачи диалога.

2. Если маска ранее не сохранялась в базе данных, то на экране появится диалоговое окно **Сохранить шаблон в базе данных**, содержащее поля ввода **Имя** и **Версия**. Первое служит для задания имени (по умолчанию *DefaultMaskName*), второе – для задания версии маски (по умолчанию *1.0*).



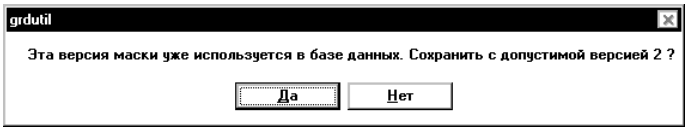
По нажатию кнопки **[OK]** происходит сохранение маски в базе данных. Сохраненная маска появляется в списке шаблонов во вкладке **Маски**.

Имя и версия сохраненной маски, а также указание на то, что маска сохранена в базе данных, отображаются в заголовке главного окна GrdUtil.exe:

Имя и версия сохраненной маски также будут отображаться в секции критериев поиска (окно **Инструменты базы данных**).

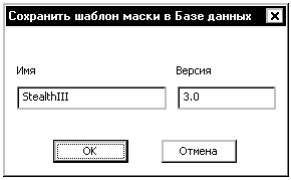
Сохранение шаблона маски в базе данных под другим именем

В базе данных нельзя сохранить маску, имя и версия которой совпадают с именем и версией шаблона, уже имеющегося в базе. В этом случае GrdUtil.exe выдает предупреждение и предлагает сохранить шаблон с другой версией.



Чтобы сохранить редактируемую маску с другим именем и/или новой версией, выполните команду **База данных | Записать маску в базу данных, как...**.

На экране появляется диалог **Сохранить шаблон в базе данных**, содержащий 2 поля ввода: **Имя** и **Версия**. Первое служит для задания имени (по умолчанию содержит текущее имя маски), второе – для задания версии маски (по умолчанию содержит увеличенное на 1.0 текущее значение версии маски).



По нажатию кнопки **[OK]** происходит сохранение маски в базе данных. Сохраненная маска появляется в списке шаблонов в окне вкладки **Маски**.

Имя и версия сохраненной маски, а также указание на то, что маска сохранена в базе данных, отображаются в заголовке главного окна GrdUtil.exe:

Загрузка шаблона маски из базы данных

Чтобы загрузить маску из базы данных в Редактор, выделите нужную строку в списке шаблонов масок и выполните команду **База данных | Загрузить маску из базы данных** (или дважды щелкните левой кнопкой мыши на выделенной строке).

Выбранная маска будет загружена в Редактор. В заголовке главного окна GrdUtil.exe отобразятся имя и версия маски, а также указание на то, что маска загружена в Редактор из базы данных:

Имя и версия загруженной в Редактор маски также будут отображаться в секции критериев поиска (окно **Инструменты базы данных**).

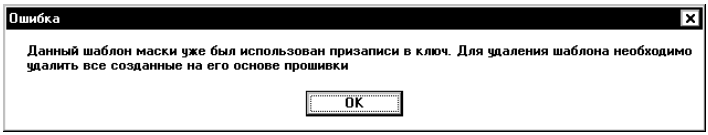
После загрузки маски Редактор показывает список полей и значения, записанные в них на момент создания или последней модификации маски. Чтобы занести в маску реальные значения из подсоединенного ключа, выполните операцию чтения памяти ключа.

Удаление шаблона маски из базы данных

Чтобы удалить маску из базы данных, выделите нужную строку в списке масок и выполните команду **База данных | Удалить маску из базы данных**.

1. Если удаляемый шаблон маски уже был использован для программирования ключей, то с ним связаны прошивки – записи в базе данных о каждом факте программирования электронного ключа определенной маской.

В этом случае шаблон маски невозможно удалить до удаления всех прошивок, которые на нем основаны. При попытке удаления GrdUtil.exe выдаст сообщение:



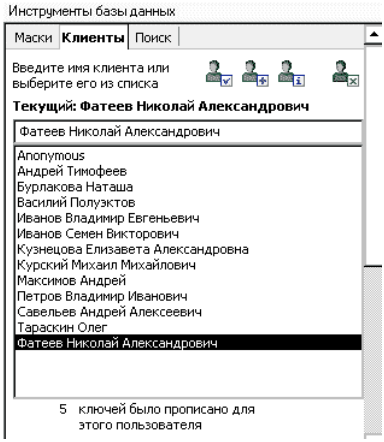
После удаления всех прошивок (см. пункт **Удаление прошивки**), связанных с данным шаблоном, шаблон можно будет удалить. При этом утилита запросит подтверждение на удаление из базы данных:

После нажатия кнопки **[Да]** в диалоге подтверждения шаблон маски будет удален из базы данных.

2. Если удаляемый шаблон маски не использовался для программирования ключей, то маску можно удалить сразу же, после вывода на экран формального диалога подтверждения на удаление маски (см. выше).

Клиенты

Вкладка **Клиенты** служит для работы базой конечных пользователей защищенного приложения. Элементы управления вкладки позволяют добавлять и удалять конечных пользователей из списка, а также определять пользователя, на которого будет зарегистрирована прошивка:



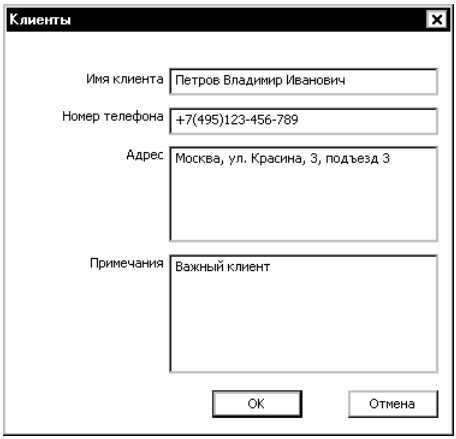
Элементы управления вкладки **Клиенты**:

Элемент интерфейса	Назначение
Кнопка	Сделать текущим выбранного конечного пользователя. Все прошивки будут регистрироваться на текущего клиента до тех пор, пока не будет выбран другой конечный пользователь. Информация о текущем клиенте отображается напротив кнопки
Кнопка	Добавить в базу нового конечного пользователя
Кнопка	Получить информацию о конечном пользователе
Кнопка	Удалить из базы выбранного конечного пользователя
Поле ввода имени клиента	Ввести имя конечного пользователя, чтобы выбрать его из списка
Окно, отображающее список клиентов	Основное окно вкладки, в котором отображается список конечных пользователей. Под окном выводится информация о количестве ключей, запрограммированных для текущего пользователя

В начале работы с базой данных вкладка **Клиенты** содержит только пользователя по умолчанию (запись ***Anonymous***).

Добавление клиента в базу данных

Чтобы добавить нового конечного пользователя в базу данных, выполните команду **База данных | Добавить клиента**.



В появившемся диалоге Клиенты заполните следующие поля ввода: **Имя клиента**, **Номер телефона**, **Адрес**, **Примечания**.

Назначение этих полей следует из их названий.

Обязательным для заполнения является только поле **Имя клиента**.

После заполнения полей нажмите на кнопку **[OK]** для завершения диалога: имя нового клиента появится в списке вкладки **Клиенты**.

Важная информация

В базе данных не может быть нескольких конечных пользователей с одинаковыми именами. При попытке добавления клиента, имя которого совпадает с именем клиента, уже сохраненного в базе, выдается предупреждение.

Редактирование информации о клиенте

Чтобы получить или изменить информацию о конечном пользователе, выберите его имя в списке и выполните команду **База данных | Информация о пользователе**.

В появившемся диалоге **Клиенты** (см. предыдущий пункт), можно изменить содержимое всех полей ввода, кроме поля **Имя клиента**.

Регистрация прошивки на выбранного клиента

Чтобы программируемые ключи регистрировались на определенного конечного пользователя, необходимо предварительно выбрать его из списка клиентов – сделать его ***текущим***.

Для этого выполните команду **База данных | Выбрать пользователя**. Команду удобно выполнять двойным щелчком мыши или нажатием клавиши **[Enter]** на нужной строке списка.

После выбора клиента его имя устанавливается в поле ввода (вкладка Клиенты), а также отображается над этим полем с пометкой ***текущий***.

Все программируемые в дальнейшем ключи будут регистрироваться в базе данных на текущего пользователя до момента выбора другого клиента.

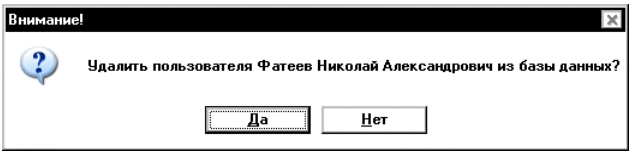
Удаление клиента из базы данных

Чтобы удалить конечного пользователя из базы данных, выделите мышью нужную строку в списке и выполните команду **База данных | Удалить клиента**.

1. Если на удаляемого клиента уже регистрировались программируемые ключи, то с ним связаны ***прошивки*** – записи в базе данных о каждом факте программирования электронного ключа для этого клиента.

В таком случае удалить клиента будет невозможно до удаления всех прошивок, которые на него зарегистрированы. При попытке удаления клиента GrdUtil.exe выдаст сообщение:

После удаления всех прошивок, связанных с данным клиентом, его можно будет удалить. При этом утилита запросит подтверждение на удаление клиента из базы данных:

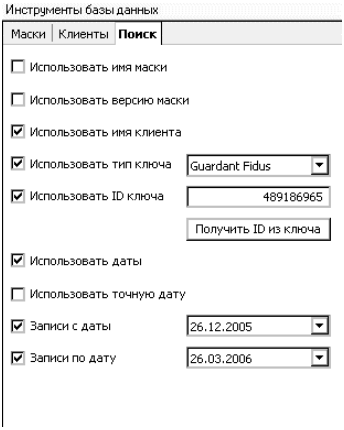


После нажатия кнопки **[Да]** в диалоге подтверждения клиент будет удален из базы данных.

2. Если на удаляемого клиента не регистрировались программируемые ключи, то его можно удалить сразу же после вывода на экран формального диалога подтверждения на удаление клиента.

Поиск

Вкладка **Поиск** служит для задания параметров поиска прошивок, хранящихся в базе данных. Вкладка содержит набор флагов, установка / снятие которых определяет критерии поиска:



Можно задавать, как отдельные параметры поиска, так и их произвольные комбинации. При этом необходимо учитывать, что поиск осуществляется по строгому соответствию заданным параметрам, т. е. ***чтобы получить положительный результат поиска, необходимо выполнение ВСЕХ его условий***.

Таким образом, к примеру, всегда можно узнать в какой именно ключ (ID, тип) записана та или иная маска (имя, версия), и у какого конечного пользователя этот ключ должен находиться.

Важная информация

1. При первом запуске базы данных параметры поиска не заданы, в дальнейшем состояние критериев поиска запоминается, и они содержат те значения, которые имели на момент последней установки.

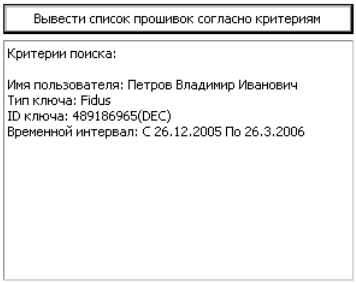
2. Если параметры поиска не заданы, то после выполнения команды **База данных | Получить список прошивок** в окне **Прошивки** будут выведены все прошивки, хранящиеся в базе данных.

Элементы управления вкладки **Поиск**:

Элемент интерфейса	Назначение
Флаг **Использовать имя маски**	Вывести в результатах поиска все прошивки для текущей маски. При поиске прошивок учитывается имя текущей маски
Флаг **Использовать версию маски**	Вывести в результатах поиска все прошивки для текущей маски. При поиске прошивок учитывается версия текущей маски
Флаг **Использовать имя клиента**	Вывести в результатах поиска все прошивки для текущего конечного пользователя. Выбор текущего пользователя происходит на вкладке Клиенты
Флаг **Использовать тип ключа**	Вывести в результатах поиска все прошивки для ключей заданного типа. Тип ключа выбирается из списка, который становится доступным после установки флага
Список для выбора типа ключа	Выбрать тип ключа для использования в качестве критерия поиска. Возможные варианты*: Guardant Stealth III, Guardant Stealth II, Guardant Stealth, Guardant Fidus
Флаг **Использовать ID ключа**	Вывести в результатах поиска все прошивки для ключа с заданным ID. Идентификатор считывается из ключа, подсоединенного к порту, либо вводится вручную в поле, которое становится доступным после установки флага
Поле ввода **ID ключа**	Ввести идентификатор ключа, который будет использоваться в качестве критерия поиска
Кнопка **Получить ID из ключа**	Считать идентификатор из ключа, подсоединенного к порту. В случае, когда ключей несколько, считывается ID первого найденного ключа
Флаг **Использовать даты**	Вывести в результатах поиска все прошивки за указанное время. Время задается при помощи флагов и списков выбора дат, которые становятся доступны после установки флага Использовать даты
Флаг **Использовать Точную дату**	Вывести в результатах поиска все прошивки за указанную дату (число, месяц, год). Дата задается при помощи календаря, открывающегося по нажатию кнопки выпадающего списка, либо вручную, путем редактирования даты в поле ввода
Флаги **Запись с даты/ Запись по дату**	Вывести в результатах поиска все прошивки за указанный временной диапазон. Начальное и/ или конечное значение диапазона задается при помощи календаря, открывающегося по нажатию кнопки выпадающего списка, либо вручную, путем редактирования даты в поле ввода

* В GrdUtil.exe не различаются локальные и сетевые ключи. Поэтому при поддержке обеих разновидностей ключей (например, Guardant Stealth III и Guardant Net III) рекомендуется создавать отдельные маски для программирования локальных и сетевых ключей.

Текущие заданные параметры поиска и их значения отображаются в секции **Критерии поиска**, расположенной в нижней части окна **Инструменты базы данных**.



Вывод результатов поиска

Чтобы найти прошивки согласно заданным критериям поиска, выполните команду меню **База данных | Получить список прошивок** или нажмите на кнопку **[Вывести результаты поиска согласно критериям]** в нижней части окна **Инструменты базы данных**.

Результаты поиска в виде списка прошивок выводятся в окне **Прошивки**, расположенном в нижней части окна GrdUtil.exe (см. скриншот в разделе **Прошивки**).

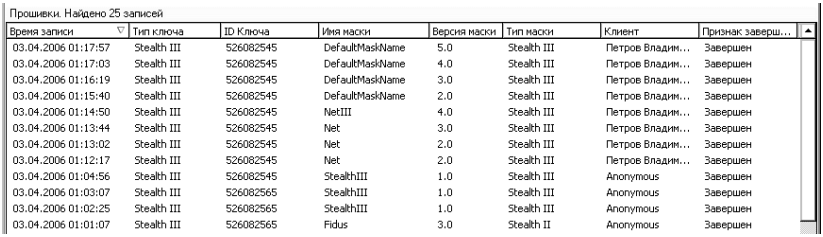
Сброс результатов поиска

Чтобы очистить окно **Прошивки** от текущих результатов поиска, щелкните на любой строке списка правой кнопкой мыши и выберите пункт **Очистить результаты поиска** в появившемся контекстном меню.

Прошивки

Прошивка – это совокупность данных маски, записанных в ключ, а также сам факт записи маски в память электронного ключа. При работе в режиме базы данных все прошивки сохраняются и могут использоваться в дальнейшем для программирования и обновления памяти ключей.

Для работы с прошивками служит окно **Прошивки** в нижней части GrdUtil.exe, в котором выводятся результаты поиска прошивок:



Прошивки. Найдено 25 записей

Время записи	Тип ключа	ID Ключа	Имя маски	Версия маски	Тип маски	Клиент	Признак заверш...
03.04.2006 01:17:57	Stealth III	526082545	DefaultMaskName	5.0	Stealth III	Петров Владим...	Завершен
03.04.2006 01:17:03	Stealth III	526082545	DefaultMaskName	4.0	Stealth III	Петров Владим...	Завершен
03.04.2006 01:16:19	Stealth III	526082545	DefaultMaskName	3.0	Stealth III	Петров Владим...	Завершен
03.04.2006 01:15:40	Stealth III	526082545	DefaultMaskName	2.0	Stealth III	Петров Владим...	Завершен
03.04.2006 01:14:50	Stealth III	526082545	NetIII	4.0	Stealth III	Петров Владим...	Завершен
03.04.2006 01:13:44	Stealth III	526082545	Net	3.0	Stealth III	Петров Владим...	Завершен
03.04.2006 01:13:02	Stealth III	526082545	Net	2.0	Stealth III	Петров Владим...	Завершен
03.04.2006 01:12:17	Stealth III	526082545	Net	2.0	Stealth III	Петров Владим...	Завершен
03.04.2006 01:04:56	Stealth III	526082545	StealthIII	1.0	Stealth III	Anonymous	Завершен
03.04.2006 01:03:07	Stealth III	526082565	StealthIII	1.0	Stealth III	Anonymous	Завершен
03.04.2006 01:02:25	Stealth III	526082565	StealthIII	1.0	Stealth III	Anonymous	Завершен
03.04.2006 01:01:07	Stealth III	526082565	Fidus	3.0	Stealth II	Anonymous	Завершен

Окно организовано в виде таблицы, строки которой образует перечень прошивок, удовлетворяющих заданным условиям поиска, а столбцы – параметры прошивок:

Столбец	**Содержимое**
Время записи	Точная дата записи данных маски в ключ. Формат: дд. мм. гг. чч :мм : сс
Тип ключа	Тип программируемого ключа Guardant без указания интерфейса
ID ключа	Уникальный идентификатор программируемого ключа
Имя маски	Название шаблона маски, записанной в ключ
Версия маски	Версия шаблона маски, которая была записана в ключ
Тип маски	Тип маски, данные из которой были записаны в ключ
Клиент	Имя конечного пользователя, для которого программировался ключ
Признак завершенности	Статус операции дистанционного программирования: завершен, не завершен

Перечень прошивок появляется в окне после выполнения команды меню **База данных | Получить список прошивок** (см. раздел **Инструменты базы данных**, вкладка **Поиск**).

Со списком прошивок можно выполнять следующие действия:

- Загружать прошивки из списка в Редактор маски
- Сортировать список прошивок
- Перерегистрировать прошивки на другого пользователя
- Удалять прошивки

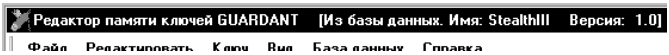
Загрузка прошивки в Редактор маски

Любую прошивку, т. е. маску, содержащую уникальные данные (случайные пароли и проч.) и прошитую в ключ, можно найти в базе GrdUtil.exe по заданным параметрам и загрузить из окна прошивок в Редактор маски.

Чтобы загрузить прошивку в Редактор маски, выполните одно из следующих действий:

- Двойной щелчок левой кнопкой мыши по нужной записи
- Щелчок правой кнопкой мыши по нужной записи и выбор пункта **Загрузить** в открывшемся контекстном меню

После этого прошивка будет загружена в Редактор. Причем индикатор маски в нижнем левом углу статусной строки переключится в положение «маска», а в заголовке главного окна GrdUtil.exe появится название загруженной маски и признак того, что она взята из базы данных:



Сортировка списка прошивок

Чтобы отсортировать записи о прошивках в требуемом порядке, щелкните левой кнопкой мыши на заголовке нужного столбца.

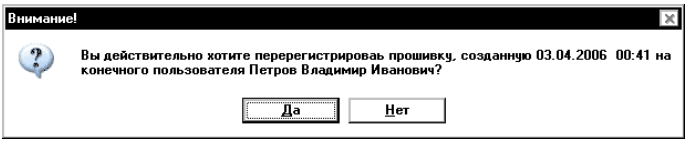
Сортировка допустима по любому из столбцов таблицы и может выполняться по убыванию или возрастанию значений параметров.

Направление сортировки указывается стрелкой-индикатором в заголовке выбранного столбца.

Перерегистрация прошивки на другого конечного пользователя

Чтобы перерегистрировать прошивку на другого пользователя, выберите его имя на вкладке **Клиенты** (окно **Инструменты базы данных**) командой меню **База данных | Выбрать клиента**. Далее щелкните на нужной прошивке правой кнопкой мыши и выберите пункт **Зарегистрировать на другого клиента** в открывшемся контекстном меню.

На экране появится запрос на подтверждение выбранного действия:

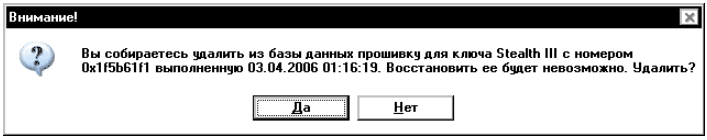


По нажатию кнопки **[Да]** прошивка будет зарегистрирована на выбранного конечного пользователя. Факт перерегистрации отразится в списке прошивок.

Удаление прошивки из базы данных

Чтобы удалить прошивку из базы данных, щелкните на нужной строке списка прошивок правой кнопкой мыши и выберите пункт **Удалить** в открывшемся контекстном меню.

На экране появится запрос на подтверждение выбранного действия:



По нажатию кнопки **[Да]** прошивка будет удалена из базы данных, запись о прошивке будет удалена из списка.

Установка аппаратных запретов

Аппаратные запреты – это программируемая блокировка чтения и записи выбранного участка памяти. Использование аппаратных запретов – эффективная мера защиты содержимого ключа.

Установка аппаратных запретов производится на нижнем уровне, что гарантирует невозможность их обхода обычными программными средствами. Ключ просто не отвечает на попытки чтения или записи защищенной области.

Важная информация

В ключах Guardant Stealth III/ Net III с помощью специальных операций Guardant API можно получить доступ к защищенным ячейкам - особой разновидности полей памяти, защищенных запретами.

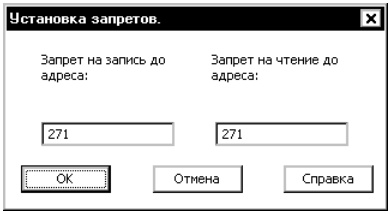
Запреты можно устанавливать на непрерывную область памяти свободного назначения, начиная с младших адресов (т. е. с адреса 14 UAM до области полей специальных операций).

GrdUtil.exe автоматически устанавливает аппаратные запреты на чтение и запись на следующие типы полей: аппаратные алгоритмы, защищенные ячейки памяти, таблица лицензий.

Важная информация

Категорически не рекомендуется оставлять незащищенной аппаратными запретами на чтение и запись область памяти, занятую аппаратными алгоритмами, таблицей лицензий и защищенными ячейками памяти!

Чтобы установить аппаратные запреты, выполните команду меню **Ключ | Установить запреты**. В появившемся диалоге **Установка запретов** отображаются текущие аппаратные запреты:



Числа в полях ввода означают адрес последнего байта (в режиме UAM), защищенного от записи и чтения соответственно.

Чтобы изменить размеры защищенной запретами области памяти, отредактируйте значение в нужном поле ввода.

Можно устанавливать как отдельные виды запретов (только на чтение, только на запись), так и комбинировать их (и на чтение, и на запись).

Виды и обозначения аппаратных запретов в Редакторе маски:

Аппаратный запрет	**Описание**
r	Поле целиком защищено от чтения
r-	Часть поля защищена от чтения
w	Поле целиком защищено от записи
w-	Часть поля защищена от записи
rw	Поле целиком защищено от чтения и записи
r-w-	Часть поля защищена от чтения и записи

Согласно идеологии ключей Guardant установка новых аппаратных запретов ведет к инициализации памяти ключа, т. е. все пользовательские данные удаляются из памяти.

Таким образом, после установки новых значений запретов в Редакторе маски и записи маски в ключ, память ключа полностью обновляется. При этом выполняется целая цепочка операций Guardant API:

- Память ключа очищается (операция GrdInit)
- Информация из маски пишется в ключ (операция GrdWrite)
- На указанную область памяти устанавливаются аппаратные запреты (операция GrdProtect)

Запись в ключ

Чтобы перенести в память ключа данные из текущей маски, выполните команду меню **Ключ | Запись в ключ**.

В процессе записи данных GrdUtil.exe автоматически переносит в ключ всю информацию из маски, загруженной в Редактор, и корректирует границы аппаратных запретов в случае их изменения.

Согласно идеологии ключей Guardant запись данных в области памяти ключа, защищенные аппаратными запретами на запись, невозможна. К примеру, при попытке выполнения операции записи из приложения по адресу, на который установлен запрет, функция GrdWrite сообщает об успешном завершении операции, но на самом деле ничего не записывает.

Поэтому каждый раз при необходимости записи новых данных в защищенную область память ключа перезаписывается полностью. При этом GrdUtil.exe выполняет целую цепочку операций Guardant API:

- Память ключа очищается (операция GrdInit)
- Информация из маски пишется в ключ (операция GrdWrite)
- На указанную область памяти устанавливаются аппаратные запреты (операция GrdProtect)

Соответствие типов маски и ключа

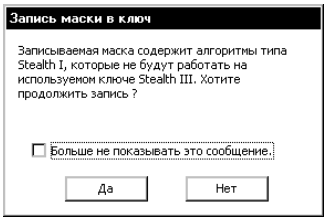
Маски для ключей разных типов отличаются друг от друга. Поэтому важно следить, чтобы в ключ записывалась подходящая по типу маска (т. е. маска Guardant Stealth III/ Net III должна прошиваться в ключ Guardant Stealth III/ Net III). Это правило следует соблюдать в абсолютном большинстве ситуаций.

Однако в некоторых случаях может возникнуть необходимость записывать в ключ маску несоответствующего типа. Допустимые варианты записи представлены в таблице и обозначены знаком «+»:

		Тип записываемой маски			
		Guardant Stealth III / Net III	Guardant Stealth II/Net II	Guardant Stealth/Net	Guardant Fidus
Тип ключа	Guardant Stealth III/Net III	+	+	+	
	Guardant Stealth II/ Net II		+	+	
	Guardant Stealth/ Net			+	
	Guardant Fidus				+

Следует иметь в виду, что после записи в ключ неподходящей маски, могут появиться серьезные ограничения и трудности при работе с таким ключом. Например:

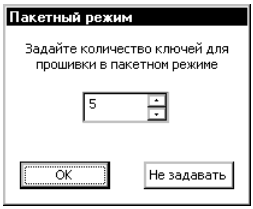
- Возможны неоднозначные ситуации при попытке чтения данных из ключа в текущую маску
- Ключи Guardant Stealth III/Net III не поддерживают алгоритмы Stealth I. Предупреждение об этом выдается при попытке записи маски Guardant Stealth II/ Net II или Guardant Stealth / Net в ключ 3-го поколения:



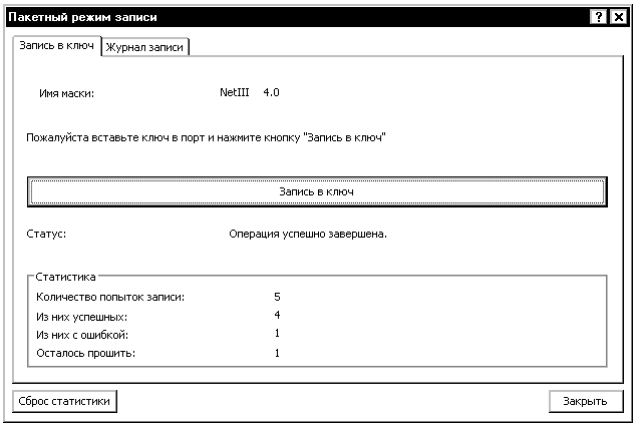
Пакетный режим записи

Чтобы запрограммировать партию ключей с одинаковыми параметрами, выполните команду меню **Ключ | Пакетный режим**.

В появившемся диалоге **Пакетный режим** можно назначить количество ключей, прошиваемых за сеанс (поле ввода со значением по умолчанию 5 и кнопка **[ОК]**), либо не лимитировать это количество (кнопка **[Не задавать]**):



Основной диалог **Пакетный режим записи**, который появляется после нажатия любой из кнопок, состоит из двух вкладок – **Запись в ключ** и **Журнал записи**:



Запись в ключ

Запись данных маски в память ключа происходит после нажатия на кнопку **[Запись в ключ]**. Когда операция записи завершится, ключ необходимо отсоединить от порта, вставить следующий ключ и произвести новую запись.

В процессе работы на вкладке отображается следующая информация о ходе выполнения программирования ключей:

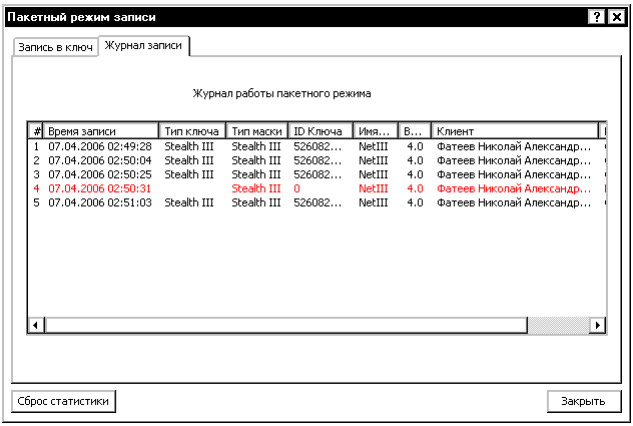
Имя маски		Имя маски, содержимое которой будет записано в ключ
Статус		Статус выполненной операции записи: успешно или неудачно
Статистика	**Количество попыток записи**	Общее число попыток записи
	Из них успешных	Число успешных попыток
	Из них с ошибкой	Число неудачных попыток
	Осталось прошить	Количество ключей, которое необходимо прошить*

Для сброса статистики используется одноименная кнопка в левом нижнем углу диалога.

* Этот пункт появляется в случае, если ранее было назначено количество ключей, прошиваемых за сеанс

Журнал записи

На вкладке **Журнал записи** отображается статистическая информация по итогам сеанса записи партии ключей:



Вкладка организована в виде таблицы, строки которой образует перечень записей о прошитых ключах, а столбцы – параметры записей:

Столбец	Описание
Время записи	Дата записи маски в ключ в формате число. месяц. год час. минута. секунда
Тип ключа	Тип ключа Guardant, в который было записано содержимое маски
Тип маски	Тип маски, записанной в ключ
ID ключа	Идентификатор ключа
Имя маски	Имя маски, записанной в ключ
Версия маски	Версия маски, записанной в ключ*
Клиент	Данные о конечном пользователе, которому предназначен ключ*
Результат операции	Статус выполненной операции записи: успешно или неудачно. Неудачные попытки записи выделяются в журнале красным цветом

Чтение из ключа

После загрузки маски в Редакторе отображаются значения и структура полей маски на момент ее последней модификации (или данные по умолчанию, если маска только создана).

Чтобы занести в маску актуальную информацию из подсоединенного ключа, выполните команду меню **Ключ | Чтение из ключа**.

* Эти пункты заполняются при работе в режиме базы данных

Важная информация

1. Чтобы чтение ключа происходило корректно, рекомендуется загружать в Редактор последнюю прошивку для данного ключа.

2. Согласно идеологии ключей Guardant содержимое участков памяти ключа, защищенных аппаратными запретами на чтение, не считывается в маску. Ключ не отвечает на попытки чтения таких полей: в этом случае в маске просто остаются текущие значения защищенных полей, а при чтении таких участков из приложения операция GrdRead возвращает нули.

Соответствие структуры маски и данных в ключе

Для корректного переноса значений из памяти ключа в маску необходимо, чтобы ее структура (количество, длина и тип полей) совпадала с расположением данных в ключе. В противном случае в большинстве ситуаций чтение из ключа будет неполным и/или неверным.

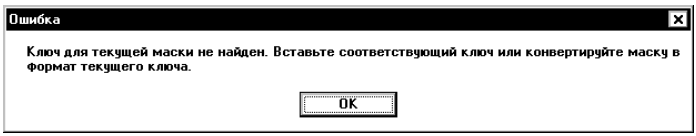
Основные варианты несоответствия структуры маски и данных в ключе:

Несоответствие структуры маски и данных в ключе	Результат чтения из ключа
Поле в маске отсутствует по тому адресу, где в ключе расположены данные	Ни само поле, ни его содержимое в маске отображено не будет
Поле в маске присутствует, но его тип не совпадает с типом данных, содержащихся в ключе	Данные будут отображены в маске некорректно в большинстве случаев
Поле в маске присутствует, но его длина не совпадает длиной данных в ключе	Данные могут быть отображены в маске некорректно (если длина поля меньше)
Не совпадает адресация защищенных от чтения полей	Утилита выдает предупреждение и считывает только те данные, расположение которых совпадает со структурой маски

Соответствие типов маски и ключа

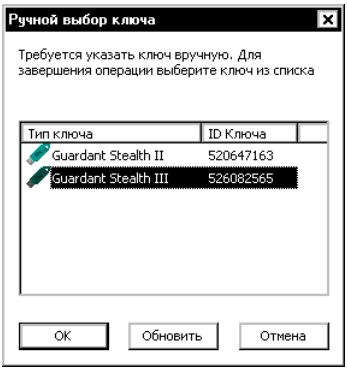
Маски для ключей разных типов отличаются друг от друга. Поэтому важно следить, чтобы текущая маска, в которую считываются данные, подходила к ключу. В противном случае, могут возникать неоднозначные ситуации при попытке чтения данных из ключа в текущую маску, либо чтение вовсе будет невозможно.

Например, чтение из ключа Guardant Stealth III, в который записана маска Guardant Stealth, можно выполнить, только если в Редактор загружена прошивка этого ключа, - при попытке чтения в файл или шаблон маски утилита выдаст ошибку:



Работа с несколькими ключами

GrdUtil.exe может обрабатывать ситуацию, когда к портам компьютера подсоединено несколько ключей. В этом случае при выполнении команд меню **Ключ | Чтение из ключа**, **Ключ | Запись в ключ** и **Ключ | Пакетный режим** на экране появляется диалог **Выбор ключа**, в котором отображены все ключи, подсоединенные к портам, указан их тип и интерфейс, а также ID:



Для завершения выбранной операции укажите нужный ключ в списке и нажмите на кнопку **[OK]**.

Аппаратные алгоритмы

Аппаратные алгоритмы – это математические функции преобразования данных, выполняющиеся в самом ключе, без использования ресурсов компьютера.

Аппаратные алгоритмы служат для кодирования информации, необходимой для работы защищенного приложения. При правильной организации системы защиты использование аппаратных алгоритмов делает бессмысленным удаление из тела приложения вызовы функций API: в этом случае не произойдет декодирование нужных приложению данных. Кроме того, сам факт наличия аппаратных алгоритмов сильно усложняет логику работы электронных ключей Guardant.

Использование аппаратных алгоритмов – это основной путь создания качественной и эффективной защиты приложения.

Аппаратные алгоритмы реализуются микропрограммой ключа, записанной в микроконтроллер, в сочетании с дескрипторами, которые хранятся в памяти электронного ключа. Микропрограмма ключа Guardant является неизменяемой частью алгоритма, ее невозможно ни считать, ни модифицировать. Для формирования конкретного вида алгоритма и его параметров, а также для управления его «поведением», служит дескриптор – набор данных, хранящихся в памяти ключа, доступной разработчику защиты.

Подробное описание дескрипторов аппаратных алгоритмов см. в главе **Аппаратные алгоритмы**.

GrdUtil.exe позволяет создавать, редактировать и удалять дескрипторы аппаратных алгоритмов.

Диалог создания алгоритма* выполнен в виде мастера, состоящего из нескольких страниц:

- Добавить алгоритм (новое поле)
- Свойства алгоритма
- Определитель алгоритма

Переход к следующей странице происходит при помощи нажатия кнопки **[Далее]** после выполнения текущего диалога.

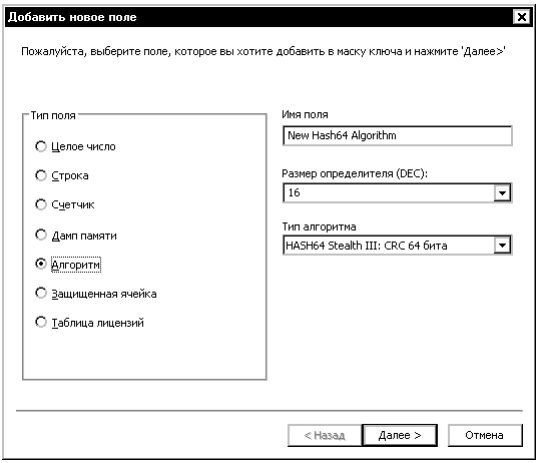
Создание алгоритма

Чтобы создать аппаратный алгоритм, выполните команду **Редактировать | Добавить новое поле**.

В появившемся диалоге **Добавить новое поле** выберите тип поля **Алгоритм**, задайте имя и тип нового алгоритма, а также размер его определителя.

* При работе с аппаратными алгоритмами подразумевается, что все действия выполняются с их дескрипторами.

Диалог добавления нового алгоритма:



Тип алгоритма

Выберите тип алгоритма при помощи разворачивающегося списка.

Тип создаваемого алгоритма зависит от маски ключа:

Тип маски	Тип алгоритма
Guardant Stealth III / Net III	1. Симметричный алгоритм GSII64 2. Однонаправленные алгоритмы Stealth III: HASH64, RND64
Guardant Stealth II / Net II	1. Симметричный алгоритм GSII64 2. Однонаправленный алгоритм Stealth и его разновидности: Fast, Random, AutoProtect
Guardant Stealth / Net	Однонаправленный алгоритм Stealth и его разновидности: Fast, Random, AutoProtect
Guardant Stealth Fidus	-

Размер определителя

Определитель – основная составляющая дескриптора аппаратного алгоритма, которая задает конкретный вид функции преобразования. Определители алгоритмов в ключах Guardant последнего поколения имеют фиксированную четную длину 16 или 32 байта. С помощью GrdUtil.exe можно задавать размер определителя и редактировать его вид.

Чтобы выбрать (или задать – для однонаправленного алгоритма Stealth) размер определителя, воспользуйтесь одноименным комбинированным полем-списком, слева от которого указывается система счисления.

Размер определителя зависит от типа алгоритма:

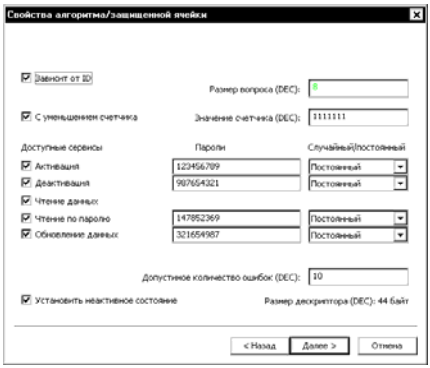
Тип алгоритма	Размер определителя, байтов
Симметричный GSII64	16 или 32
Однонаправленные Stealth III (HASH64, RND64)	16 или 32
Однонаправленный Stealth* и его разновидности	4 – 200 (оптимально - 32)

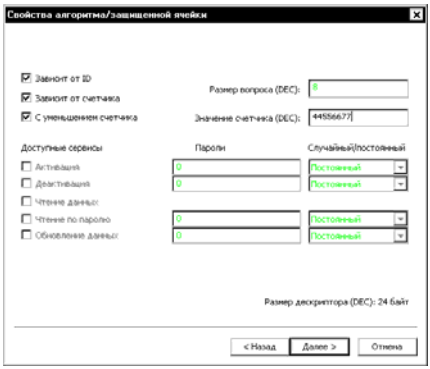
***

После задания типа и размера определителя нового алгоритма необходимо отредактировать его свойства. Для перехода к следующей странице нажмите на кнопку **[Далее]** в нижней части диалога.

Свойства алгоритма

Диалог **Свойства алгоритма** служит для определения свойств создаваемого алгоритма: задания комбинации флагов, размера вопроса, а также набора сервисов для алгоритмов, создаваемых в ключах Guardant Stealth III/ Net III:





Флаги свойств алгоритмов

Группа флагов в левой верхней части диалога (**Зависит от ID**, **Зависит от счетчика** и **С уменьшением счетчика**) служит для задания свойств аппаратного алгоритма. Комбинирование этих свойств позволяет создавать алгоритмы с различными режимами работы, адаптированные для решения разных задач (получение случайных чисел, ограничение числа запусков, и проч.).

* Однонаправленный алгоритм Stealth является морально устаревшим, его рекомендуется использовать только для поддержания существующей системы защиты на ключах Guardant Stealth / Net.

Возможные комбинации флагов свойств алгоритмов:

Комбинация флагов	**Режим работы алгоритма**	**Описание режима**
Флаги не установлены	Режим по умолчанию	Алгоритм от флагов не зависит
Установлен флаг **Зависит от ID**	Уникальность алгоритма по ID	Кодирование зависит от ID ключа. При одинаковых определителях алгоритмы в разных ключах кодируют данные по-разному
Установлен флаг **Зависит от счетчика**	Зависимость алгоритма от счетчика	В поле счетчика алгоритма записывается 4-байтовое значение, от которого зависит вид преобразования. При одинаковых определителях алгоритмы с разными значениями счетчиков будут кодировать данные по-разному
Установлен флаг **С уменьшением счетчика**	Ограничение числа запусков алгоритма	В поле счетчика алгоритма записывается 4-байтовое начальное значение. Счетчик декрементируется при каждом вызове GrdTransform, а по достижении счетчиком значения 0 алгоритм перестает выполняться
Установлены флаги **Зависит от ID** и **Зависит от счетчика**	Зависимость алгоритма от счетчика + уникальность по ID	Кодирование зависит от ID ключа. В поле счетчика алгоритма записывается 4-байтовое значение, от которого зависит вид преобразования. При одинаковых определителях алгоритмы с разными значениями счетчиков будут кодировать данные по-разному
Установлены флаги **Зависит от ID** и **С уменьшением счетчика**	Ограничение числа запусков алгоритма + уникальность по ID	Кодирование зависит от ID ключа. При одинаковых определителях алгоритмы в разных ключах кодируют данные по-разному. В поле счетчика алгоритма записывается 4-байтовое начальное значение. Счетчик декрементируется при каждом вызове GrdTransform, а по достижении счетчиком значения 0 алгоритм перестает выполняться.
Установлены флаги **Зависит от счетчика** и **С уменьшением счетчика**	Генератор псевдослучайных чисел	В поле счетчика алгоритма записывается большое 4-байтовое начальное значение. Счетчик декрементируется при каждом вызове GrdTransform, а по достижении счетчиком значения 0 алгоритм перестает выполняться. При этом преобразование выполняется с каждым уменьшением счетчика по-разному
Установлены флаги **Зависит от ID**, З**ависит от счетчика** и **С уменьшением счетчика**	Генератор псевдослучайных чисел	Кодирование зависит от ID ключа. При одинаковых определителях алгоритмы в разных ключах кодируют данные по-разному. В поле счетчика алгоритма записывается большое 4-байтовое начальное значение. Счетчик декрементируется при каждом вызове GrdTransform, а по достижении счетчиком значения 0 алгоритм перестает выполняться. При этом преобразование выполняется с каждым уменьшением счетчика по-разному

Важная информация

Перечисленные флаги и их комбинации доступны не для всех типов ключей.

Значение счетчика

Счетчик алгоритма – специальное 4-байтовое поле, входящее в состав дескриптора алгоритма. Счетчик обычно используется для ограничения числа запусков алгоритма, а также в режимах зависимости алгоритма от счетчика.

Поле ввода **Значение счетчика** становится доступным при использовании флагов **Зависит от счетчика** или **С уменьшением счетчика**. Оно располагается в верхней правой части диалога. Рядом с полем указывается система счисления.

Размер вопроса алгоритму

Вопрос алгоритму – это блок данных определенной длины, которую аппаратный алгоритм может преобразовать за один прием (ср. с понятием **Размер вопроса GrdTransform**).

Для ввода размера вопроса служит одноименное поле в верхней правой части диалога. Рядом с полем указывается система счисления.

Размер вопроса зависит от типа алгоритма:

Тип алгоритма	Размер вопроса, байтов
Симметричный GSII64	8
ОднонаправленныеHASH64, RND64	8
Однонаправленный Stealth*	4 – 255 (желательно использовать четные числа)

Сервисы аппаратных алгоритмов Guardant Stealth III/ Net III

В группе **Доступные сервисы** расположены флаги, управляющие сервисами алгоритмов. Использование сервисов значительно расширяет функциональность аппаратных алгоритмов Guardant Stealth III/ Net III.

Сервисы позволяют:

- Задавать состояние алгоритма (активное/неактивное) и, в дальнейшем, управлять им из приложения или путем удаленного обновления
- Получать доступ к содержимому дескриптора алгоритма и обновлять его, не затрагивая (не перезаписывая) остальную память – ср. с идеологией аппаратных запретов.

* Однонаправленный алгоритм Stealth на сегодняшний день является морально устаревшим, его рекомендуется только для поддержания существующей системы защиты.

Такие возможности стали доступны благодаря ***технологии защищенных ячеек***, частным случаем которых являются аппаратные алгоритмы Guardant Stealth III/ Net III.

Активация

Если в свойствах алгоритма включен сервис **Активация**, то этот алгоритм в определенный момент можно сделать активным, обратившись к нему из приложения с помощью специальной команды Guardant API или выполнив процедуру обновления.

После активации с алгоритмом можно выполнять все ранее запрограммированные для него действия: выполнять преобразования, читать содержимое дескриптора алгоритма, обновлять определенные участки дескриптора, деактивировать алгоритм.

Примеры использования

1. Удаленная активация алгоритма, который отвечает за работоспособность дополнительных модулей приложения, после получения оплаты от конечного пользователя.

2. Активация дополнительного алгоритма по определенному событию для усложнения логики работы защищенной программы.

3. Активация нового и деактивация старого алгоритма при выходе новой версии приложения. При этом все алгоритмы должны быть заранее созданы и их свойства и сервисы должны быть определены.

Активация алгоритма из приложения выполняется с помощью команды ***GrdPI_Activate***.

Установка флага **Активация** включает одноименный сервис, при этом становятся доступными опции (подробнее о них см. ниже):

- Поле ввода **Пароль активации**
- Поле ввода **Постоянный / случайный пароль**
- Флаг **Установить неактивное состояние**

А также появляется опция, общая для всех сервисов:

- Поле **Допустимое количество ошибок при наборе паролей**

После включения сервиса **Активация** задайте пароль активации, определите его вид и допустимое число ошибочного ввода.

Если схема защиты приложения предполагает, что алгоритм должен быть деактивирован изначально, установите флаг **Установить неактивное состояние**. Неактивный алгоритм выделяется в Редакторе маски бледно-серым шрифтом.

Деактивация

Если в свойствах алгоритма включен сервис **Деактивация**, то алгоритм в определенный момент можно сделать неактивным, обратившись к нему из приложения с помощью специальной команды Guardant API или выполнив процедуру обновления.

После деактивации алгоритм можно только активировать (если сервис активации был предварительно включен в свойствах алгоритма), но нельзя выполнять остальные действия, запрограммированные для этого алгоритма.

Деактивация алгоритма из приложения выполняется с помощью команды ***GrdPI_Deactivate***.

Установка флага **Деактивация** включает одноименный сервис, при этом становятся доступными опции (подробнее о них см. ниже):

- Поле ввода **Пароль деактивации**
- Поле ввода **Постоянный / случайный пароль**

А также появляется опция*, общая для всех сервисов:

- Поле **Допустимое количество ошибок при наборе паролей**

После включения сервиса **Деактивация** задайте пароль деактивации, определите его вид и допустимое число ошибочного ввода.

Чтение данных. Чтение по паролю

Если в свойствах алгоритма включен сервис **Чтение данных**, то можно получить информацию о содержимом определителя этого алгоритма, обратившись к нему из приложения с помощью специальной команды Guardant API.

Если при этом будет включен сервис **Чтение по паролю**, то для выполнения команды чтения необходимо будет указать верный пароль.

Доступ к содержимому аппаратного алгоритма из приложения выполняется с помощью команды ***GrdPI_Read***.

Установка флага **Чтение данных** включает одноименный сервис, при этом становится доступной опция **Чтение по паролю**.

Установка флага **Чтение по паролю** включает одноименный сервис, при этом становится доступными опции (подробнее о них см. ниже):

- Поле ввода **Пароль для чтения данных**
- Поле ввода **Постоянный / случайный пароль**

* Если она уже не появилась после включения другого сервиса

А также появляется опция*, общая для всех сервисов:

- Поле **Допустимое количество ошибок при наборе паролей**

После включения сервиса **Чтение данных** включите сервис **Чтение по паролю**, задайте пароль для чтения данных, определите его вид и допустимое число ошибочного ввода.

Обновление данных

Если в свойствах алгоритма включен сервис **Обновление данных**, то содержимое определителя такого алгоритма можно будет изменить, обратившись к нему из приложения с помощью специальной команды Guardant API.

Запись новых данных в дескриптор алгоритма из приложения выполняется с помощью команды ***GrdPI_Update***.

Установка флага **Обновление данных** включает одноименный сервис, при этом становятся доступными опции (подробнее о них см. ниже):

- Поле ввода **Пароль для обновления данных**
- Поле ввода **Постоянный/случайный пароль**

А также появляется опция*, общая для всех сервисов:

- Поле **Допустимое количество ошибок при наборе паролей**

После включения сервиса **Обновление данных** задайте пароль для обновления, определите его вид и допустимое число ошибочного ввода.

Пароли

Для выполнения команд Guardant API, изменяющих статус алгоритма, либо его содержимое, требуется указать пароль, предварительно заданный при включении сервиса(-ов). Для ввода паролей служат поля, расположенные напротив флагов, которые управляют тем или иным сервисом.

Поля ввода в секции **Пароли** становятся доступными при использовании соответствующего сервиса.

Максимальная длина каждого пароля составляет 4 байта.

Постоянный/случайный пароль

Пароль для сервиса может быть постоянным, т. е. одинаковым для всех ключей, использующих данный алгоритм, либо случайным, т. е. для каждого следующего ключа, который прошивается данным шаблоном маски, автоматически задается случайный пароль.

* Если она уже не появилась после включения другого сервиса

Важная информация

Использование случайных паролей предполагает работу в режиме базы данных GrdUtil.exe. Только в этом случае можно отследить, какой пароль был задан при программировании конкретного ключа, т. к. каждый факт прошивки ключа регистрируется в базе данных.

Для выбора вида пароля служит разворачивающийся список, расположенный напротив выбранного сервиса.

Списки в секции **Постоянный/случайный пароль** становятся доступными при использовании соответствующего сервиса.

Допустимое количество ошибок. Алгоритм с неизменяемым статусом

Значение поля **Допустимое количество ошибок** задает предельное число попыток ошибочного ввода пароля при попытке доступа к алгоритму (значение по умолчанию – 10).

В том случае, если при активации, деактивации, обновлении или других действиях, запрограммированных для данного алгоритма, число попыток набора неправильного пароля было превышено, то алгоритм блокируется и приобретает статус неизменяемой ячейки.

При этом если алгоритм до блокирования имел статус активного, то такой блокированный алгоритм можно выполнять при помощи команды GrdTransform, однако остальные действия, запрограммированные для этого алгоритма (изменение статуса или содержания), будут недоступны.

Если же алгоритм до блокирования не обладал статусом активного, то любая работа с этим алгоритмом будет невозможна.

Установить неактивное состояние

Флаг **Установить неактивное состояние** становится доступным после установки флага **Активация**.

Если схема защиты приложения предполагает, что алгоритм должен быть деактивирован изначально, установите флаг **Установить неактивное состояние**. Неактивный алгоритм выделяется в Редакторе маски бледно-серым шрифтом.

Размер дескриптора алгоритма

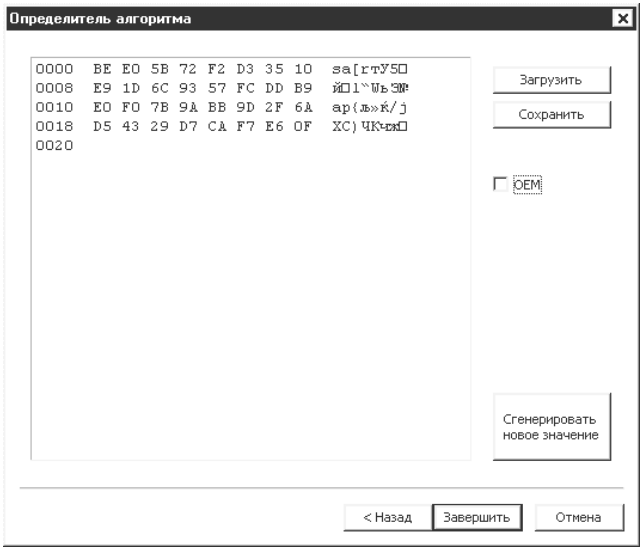
В нижней части диалога **Свойства алгоритма** отображается статистическая информация о размере дескриптора алгоритма (с указанием выбранной системы счисления). Размер дескриптора алгоритма складывается из размеров его составных частей: определителя, наборов флагов и сервисов.

***

После определения свойств алгоритма остается отредактировать его определитель (если в этом есть необходимость). Для перехода к следующей странице нажмите на кнопку **[Далее]** в нижней части диалога.

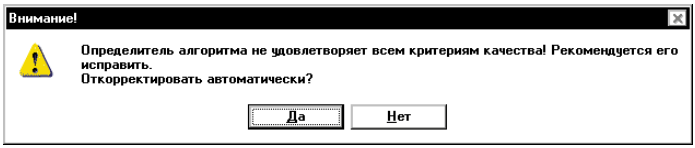
Редактирование определителя алгоритма

Диалог **Определитель алгоритма** представляет собой шестнадцатеричный редактор для ввода и изменения определителя:



По умолчанию при создании алгоритма в определитель записываются случайные числа. Их можно изменить, сформировав определитель самостоятельно, путем ввода новых значений непосредственно в окне редактора, или автоматически создать новый определитель (кнопка **[Сгенерировать новое значение]**).

GrdUtil.exe отслеживает «качественность» определителя и, в случае несоответствия критериям, выдает предупреждение:



Элементы управления диалога **Определитель алгоритма**:

Элемент интерфейса	Назначение
Окно шестнадцатеричного редактора	Ввести значение определителя аппаратного алгоритма
Кнопка **[Загрузить]**	Загрузить дамп из файла с расширением *.dmp
Кнопка **[Сохранить]**	Сохранить дамп в файле с расширением *.dmp
Флаг **OEM**	Выбрать Windows- / DOS-кодировку. По умолчанию используется Windows-кодировка (ANSI) – опция OEM отключена
Кнопка **[Сгенерировать новое значение]**	Автоматически создать новый определитель алгоритма

***

После редактирования определителя и нажатия на кнопку **[Завершить]** диалог создания алгоритма закрывается, и новый алгоритм появляется в списке полей маски. При этом GrdUtil.exe автоматически присваивает алгоритму порядковый номер* и корректирует границу аппаратных запретов с учетом добавленного алгоритма.

Теперь остается записать маску в ключ, и созданный аппаратный алгоритм можно будет использовать.

Важная информация

Вызов алгоритма из приложения происходит по его порядковому номеру, поэтому не рекомендуется добавлять новые алгоритмы или защищенные ячейки между уже существующими и использующимися. Это повлечет изменение номеров защищенных полей, расположенных после созданного, и, как следствие, будет необходимо поправить параметры вызова функций Guardant API.

Получение ответов аппаратных алгоритмов

Чтобы использовать аппаратные алгоритмы ключа, необходимо знать, какую последовательность вернет алгоритм в ответ на заданный вопрос. Затем эту последовательность (ответ алгоритма) можно использовать для усложнения логики работы защиты.

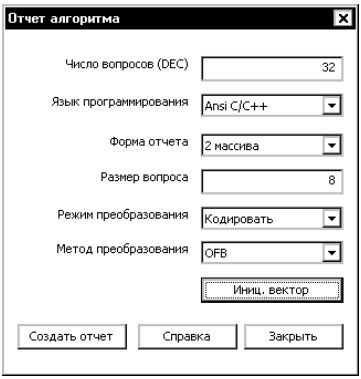
GrdUtil.exe предоставляет удобный сервис для получения ответов аппаратных алгоритмов. Утилита обращается к выбранному алгоритму, получает его ответы и сохраняет результаты в специальном файле отчета.

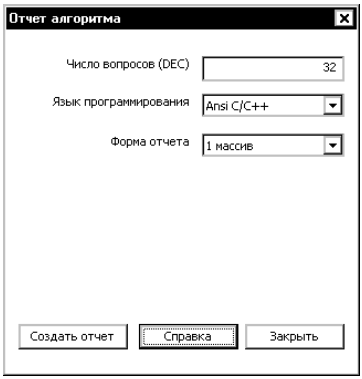
Важная информация

Если алгоритм, для которого выполняется отчет, еще не был записан в память ключа, или свойства и/или определитель алгоритма были изменены в ходе редактирования маски, то перед генерацией отчета выполните команду меню **Ключ | Запись в ключ**.

* Для защищенных ячеек и аппаратных алгоритмов действует единая нумерация

Чтобы получить массив ответов алгоритма, выделите его в списке и выполните команду меню **Ключ | Создать отчет алгоритма**:





В появившемся диалоге укажите число вопросов к алгоритму, нужный язык программирования и форму отчетов. Дополнительно для алгоритмов типа GSII64 укажите размер вопроса, а также метод и режим преобразования.

Вопросы к алгоритму представляют собой последовательности случайных чисел.

Число вопросов

В поле **Число вопросов** укажите число обращений в выбранной системе счисления к алгоритму функции ***GrdTransform***.

На каждое обращение (вопрос) алгоритм генерирует ответную последовательность, длина которой равна длине вопроса.

Язык программирования

С помощью раскрывающегося списка **Язык программирования** выберите язык, по правилам синтаксиса которого будет создан файл отчета.

Возможные варианты выбора: C/C++, Pascal/Delphi, Ассемблер.

Форма отчета

Вопросы и полученные ответы алгоритма сохраняются в файле отчета в виде одного или двух массивов. Выберите форму отчета с помощью одноименного раскрывающегося списка.

Форма отчета	Описание
1 массив	Вопрос и ответ алгоритма представляют чередующиеся элементы массива. Количество элементов массива равно удвоенному числу вопросов
2 массива	Вопросы алгоритма составляют первый массив элементов, соответствующие им ответы – второй. Количество элементов каждого массива равно числу вопросов

Дополнительные параметры для алгоритмов GSII64

Размер вопроса GrdTransform

Под размером вопроса (поле **Размер вопроса**) в данном случае подразумевается максимальная длина данных на входе операции GrdTransform, которую эта операция может обработать за один прием (ср. с понятием **Размер вопроса алгоритму**).

Для однонаправленных аппаратных алгоритмов длина вопроса GrdTransform – это величина постоянная, в отличие от алгоритмов типа GSII64, которые могут принимать от GrdTransform блоки данных разной длины:

Режимы работы алгоритма GSII64	Размер вопроса GrdTransform, байтов
ECB и CBC	Число, кратное 8. Максимальное значение – 248
CFB и OFB	Произвольное число, не превышающее 255

Задайте размер вопроса в одноименном поле ввода (значение по умолчанию 8 байтов).

Метод преобразования

Алгоритмы типа GSII64 имеют 4 режима работы, которые отличаются по своим характеристикам и назначению. Подробнее об алгоритме GSII64 см. в главе **Аппаратные алгоритмы**.

Выберите метод преобразования с помощью одноименного разворачивающегося списка.

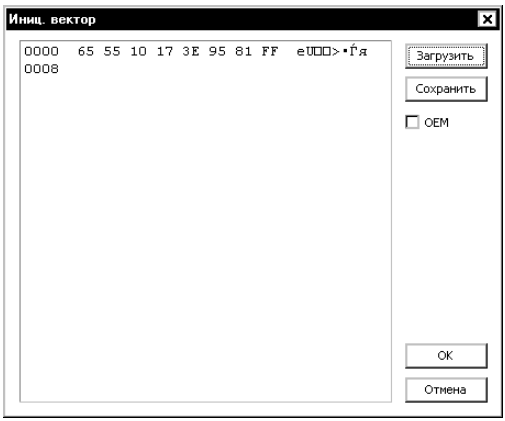
Режим преобразования

Алгоритмы типа GSII64 являются симметричными: преобразование вопроса алгоритма GSII64 в его ответ – взаимообратно.

Выберите направление преобразования (кодирование или декодирование) с помощью одноименного разворачивающегося списка.

Вектор инициализации

По нажатию кнопки **[Инит. вектор]** появляется шестнадцатеричный редактор, позволяющий задать значение вектора инициализации:



Элементы управления диалога **Вектор инициализации**:

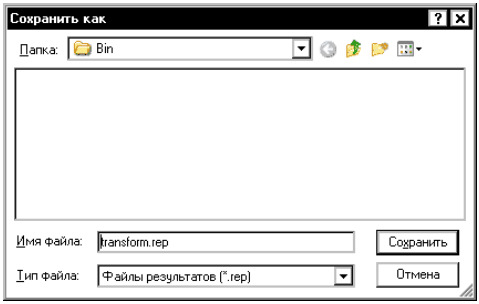
Элемент интерфейса	**Назначение**
Окно шестнадцатеричного редактора	Ввести значение вектора инициализации
Кнопка **[Загрузить]**	Загрузить дамп из файла с расширением *.dmp
Кнопка **[Сохранить]**	Сохранить дамп в файле с расширением *.dmp
Флаг **OEM**	Выбрать Windows- / DOS-кодировку. По умолчанию используется Windows-кодировка (ANSI) – опция OEM отключена

Зависимость режимов работы алгоритма GSII64 от вектора инициализации:

Режимы работы алгоритма GSII64	**Зависимость от вектора инициализации**
ECB	Не зависит
CBC и OFB	Зависят. Для преобразования информации следует использовать один и тот же вектор инициализации. В противном случае данные будут декодированы неверно
CFB	Зависит. Для преобразования информации следует использовать один и тот же вектор инициализации. В противном случае первые 8 байтов данных будут декодированы неверно

Создание отчета

После нажатия кнопки **[Создать отчет]** появляется стандартный системный диалог сохранения файла (имя файла по умолчанию *Transform.rep*):



После этого начинается создание отчета. Используя операцию GrdTransform, GrdUtil.exe обращается к выбранному алгоритму в ключе, получает ответы этого алгоритма и сохраняет их в файле отчета.

Процесс генерации отчета иллюстрирует индикатор выполнения.

Использование данных отчета

Массивы, записанные в файле отчета, используются в защищаемом приложении.

Массив вопросов хранится в теле приложения и применяется для последующих обращений к ключу (настоятельно рекомендуется хранить его в закодированном виде).

Массив ответов не следует хранить в приложении, в противном случае уровень защищенности не может быть приемлемым. При помощи этого массива лучше закодировать какие-либо важные данные, используемые приложением (можно, к примеру, использовать быстрое взаимообратное преобразование, паролем которого будет массив ответов).

Аппаратное взаимообратное преобразование данных

В электронных ключах Guardant Stealth III / Net III, Stealth II / Net II реализован симметричный аппаратный алгоритм GSII64. Он выполняет аппаратное преобразование данных небольшого объема (несколько сотен килобайтов)*. Подробную информацию об алгоритме GSII64 см. в разделе **Аппаратные алгоритмы**.

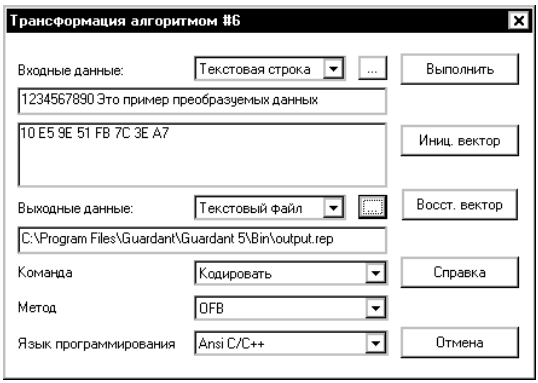
Предварительно закодированные данные могут храниться в защищенном приложении или отдельных файлах и декодироваться непосредственно перед использованием.

Для вызова алгоритма GSII64 из приложения используется операция ***GrdTransform***. Подробную информацию об операциях Guardant API см. в разделе **Guardant API**.

GrdUtil.exe предоставляет удобный сервис для предварительной подготовки данных преобразования. С помощью утилиты можно закодировать и декодировать информацию. Подготовленные данные в дальнейшем используются при защите приложения.

Подготовка данных для преобразования

Выделите в списке полей Редактора маски алгоритм типа GSII64 и выполните команду **Ключ | Преобразование**. На экране появится диалоговое окно **Трансформация алгоритмом №N****:



* Такой размер преобразуемых данных обусловлен относительно невысокой скоростью работы аппаратного алгоритма (~200 байтов в секунду). Больший размер данных может негативно сказаться на быстродействии защищенного приложения.

** Где N – порядковый номер алгоритма GSII64.

В диалоге определите следующие параметры:

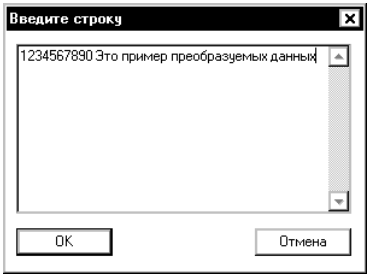
- Входные данные и их вид
- Вектор инициализации
- Выходные данные и их вид
- Направление и метод кодирования
- Язык программирования (если выходные данные представлены в виде исходного текста)

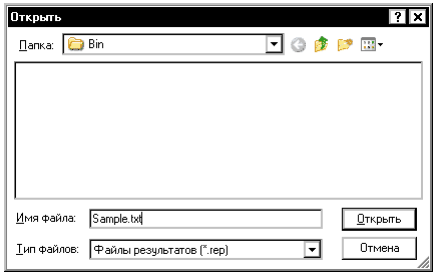
Входные данные

Данные, которые необходимо преобразовать, могут быть представлены в виде: строки символов или файла любого формата.

Для выбора вида данных на входе служит раскрывающийся список в верхней части диалога.

По нажатию кнопки **[...]**, расположенной напротив списка, открывается, либо диалог **Введите строку** для ввода строки символов, либо стандартный системный диалог для указания имени файла данных и пути к нему:





Заданная строка или имя файла с данными и путь к нему отображаются в поле ввода **Входные данные**.

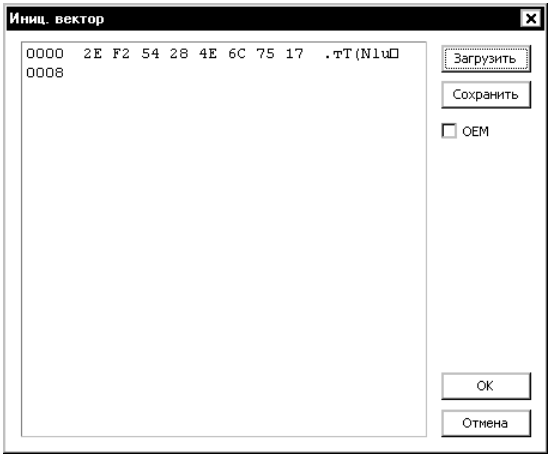
Вектор инициализации

Вектор инициализации – 8-байтовое случайное число, которое используется для работы алгоритма GSII64 в режимах работы CFB, CBC и OFB.

Вектор инициализации генерируется автоматически при открытии диалога **Трансформация алгоритмом №N** и отображается в соответствующем поле ввода.

При необходимости вектор инициализации по умолчанию можно изменить. По нажатию на кнопку **[Иниц. вектор]** на экране появляется шестнадцатеричный редактор, в окне которого можно скорректировать значение или ввести новый вектор.

Диалог **Вектор инициализации**:



Элементы управления диалога **Вектор инициализации**:

Элемент интерфейса	**Назначение**
Окно шестнадцатеричного редактора	Ввести значение вектора инициализации
Кнопка **[Загрузить]**	Загрузить дамп из файла с расширением *.dmp
Кнопка **[Сохранить]**	Сохранить дамп в файле с расширением *.dmp
Флаг **OEM**	Выбрать Windows- / DOS-кодировку. По умолчанию используется Windows-кодировка (ANSI) – опция OEM отключена

При выполнении преобразования значение вектора изменяется. Кнопка **[Восстановить]** служит для восстановления исходного вектора инициализации.

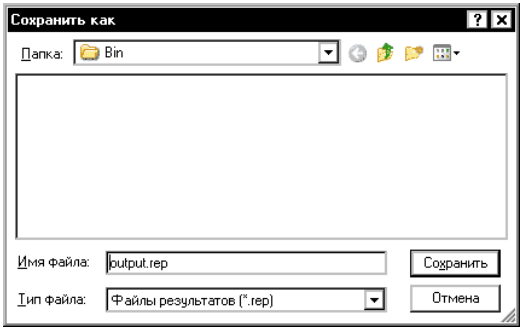
Выходные данные

Преобразованные данные могут иметь следующий вид:

Данные на выходе	**Описание**
Исходный текст	Текстовый файл, содержащий закодированные данные в виде массива чисел и созданный по правилам синтаксиса одного из основных языков программирования: Assembler, C/C++, Pascal/Delphi
Двоичный файл	Закодированная последовательность байтов

Для выбора представления данных на выходе служит раскрывающийся список в средней части диалога.

По нажатию кнопки **[...]**, расположенной напротив списка, открывается стандартный системный диалог для указания имени файла с преобразованными данными (по умолчанию - *Output.rep*) и пути к нему:



Имя файла с данными и путь к нему отображаются в поле ввода **Выходные данные**.

Кнопка [Выполнить]

По нажатию на кнопку **[Выполнить]** начинается процесс кодирования (декодирования) данных. Кнопка становится доступной после заполнения секций **Входные данные** и **Выходные данные**.

Язык программирования

Развертывающийся список **Язык программирования** расположен в нижней части диалога и становится доступным в том случае, если выбрано представление закодированных данных в виде исходного текста.

В списке представлены следующие языки программирования: Assembler, C/C++, Pascal/Delphi.

Кодирование и декодирование

Разворачивающийся список **Команда** служит для выбора операции, которая будет совершена над входными данными: кодирование или декодирование.

Метод преобразования

Алгоритмы типа GSII64 имеют 4 режима работы, которые отличаются по своим характеристикам и назначению. Подробнее об алгоритме GSII64 см. в главе **Аппаратные алгоритмы**.

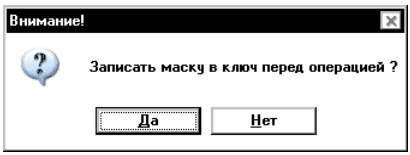
Для выбора метода преобразования предназначен одноименный разворачивающийся список.

Выполнение преобразования

Преобразование начинается после нажатия на кнопку **[Выполнить]**, расположенную в верхней части диалога.

Запись данных маски в память ключа

Перед выполнением преобразования утилита выдает запрос о необходимости записи данных маски в ключ:

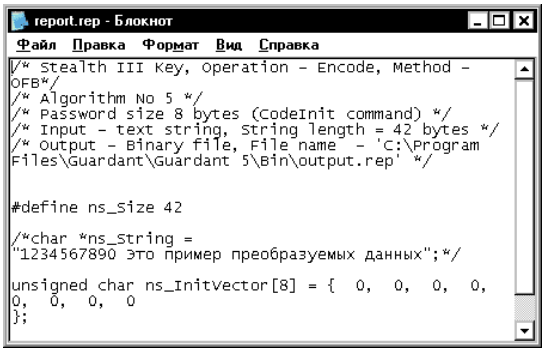


Предварительная запись маски в ключ необходима в случае, если используется новый алгоритм GSII64, либо алгоритм GSII64, у которого был изменен определитель.

Сохранение отчета

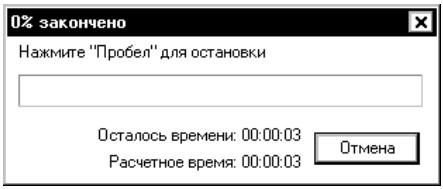
Далее на экране появится диалог сохранения отчета о преобразовании, в котором нужно указать имя (по умолчанию report.rep) и путь к файлу отчета.

Отчет о преобразовании представляет собой текстовый файл, созданный по правилам синтаксиса указанного ранее языка программирования. В отчете содержится статистическая информация о параметрах преобразования и заданный пароль в виде массива:



Процесс преобразования

После сохранения отчета на экране появится окно индикатора выполнения:



Преобразованные данные помещаются в заданный выходной файл в виде массива или последовательности байтов.

Декодирование

Процесс декодирования выполняется аналогично процессу кодирования (см. выше). В качестве данных на входе используется файл с кодированными данными. Направление преобразования меняется на декодирование (список **Команда**).

Важная информация

Для корректного декодирования данных необходимо пользоваться тем же алгоритмом, вектором инициализации и методом преобразования, которые использовались для кодирования.

Защищенные ячейки

Защищенные ячейки – это новая технология Guardant, которая впервые реализована в ключах Guardant Stealth III / Net III.

Защищенные ячейки служат для хранения данных, как и обычные поля памяти (дамп, строка, число), но при этом они защищены аппаратными запретами на чтение и запись.

Состояние (статус) защищенной ячейки можно программировать и, в дальнейшем, управлять им из приложения. Кроме того, технология защищенных ячеек предусматривает получение доступа к содержимому ячейки и обновление ячейки без перезаписи памяти ключа целиком.

Для работы с защищенными ячейками из приложения предназначены несколько команд Guardant API: GrdPI_Activate, GrdPI_Deactivate, GrdPI_Update и GrdPI_Read.

Диалог создания защищенной ячейки выполнен в виде мастера, состоящего из нескольких страниц:

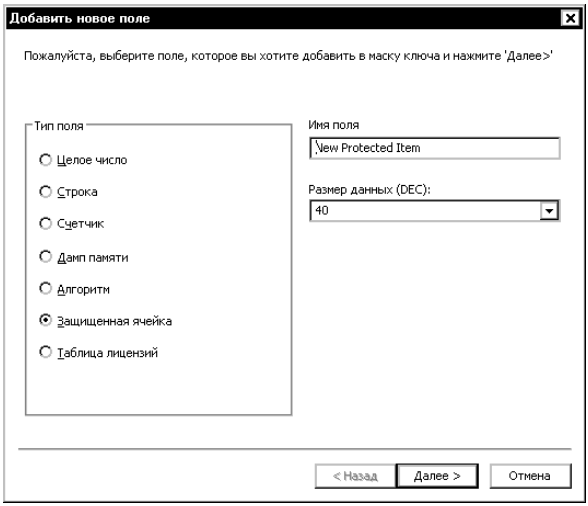
- Добавить защищенную ячейку (новое поле)
- Свойства защищенной ячейки
- Содержимое защищенной ячейки

Переход к следующей странице происходит при помощи нажатия кнопки **[Далее]** после выполнения текущего диалога.

Создание защищенной ячейки

Чтобы создать защищенную ячейку, выполните команду меню **Редактировать | Добавить новое поле**.

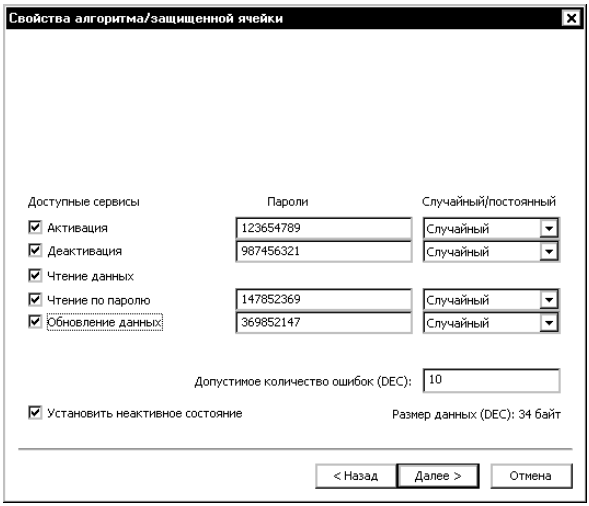
В появившемся диалоге **Добавить новое поле** выберите тип поля **Защищенная ячейка**, задайте ее имя и размер:



Размер данных защищенной ячейки может составлять от 1 до 255 байтов.

Свойства защищенной ячейки

В окне **Свойства защищенной ячейки** расположены флаги, управляющие сервисами защищенных ячеек Guardant Stealth III/ Net III:



Сервисы позволяют:

- Задавать состояние защищенной ячейки (активное/ неактивное) и, в дальнейшем, управлять им из приложения или путем удаленного обновления
- Получать доступ к содержимому ячейки и обновлять его, не затрагивая (не перезаписывая) остальную память – ср. с идеологией аппаратных запретов.

Активация

Если в свойствах защищенной ячейки включен сервис **Активация**, то эту ячейку в определенный момент можно сделать активной, обратившись к ней из приложения с помощью специальной команды Guardant API или выполнив процедуру обновления.

После активации с защищенной ячейкой можно выполнять все ранее запрограммированные для нее действия: читать содержимое ячейки, обновлять содержимое ячейки, деактивировать ячейку.

Активация ячейки из приложения выполняется с помощью команды ***GrdPI_Activate***.

Установка флага **Активация** включает одноименный сервис, при этом становятся доступными опции (подробнее о них см. ниже):

- Поле ввода **Пароль активации**
- Поле ввода **Постоянный/случайный пароль**

- Флаг **Установить неактивное состояние**

А также появляется опция*, общая для всех сервисов:

- Поле **Допустимое количество ошибок при наборе паролей**

После включения сервиса **Активация** задайте пароль активации, определите его вид и допустимое число ошибочного ввода.

Если схема защиты приложения предполагает, что защищенная ячейка должна быть деактивирована изначально, установите флаг **Установить неактивное состояние**. Неактивная ячейка выделяется в Редакторе маски бледно-серым шрифтом.

Деактивация

Если в свойствах защищенной ячейки включен сервис **Деактивация**, то эту ячейку в определенный момент можно сделать неактивной, обратившись к ней из приложения с помощью специальной команды Guardant API или выполнив процедуру обновления.

После деактивации защищенную ячейку можно только активировать (если сервис активации был предварительно включен в свойствах ячейки), но нельзя выполнять остальные действия, запрограммированные для этой ячейки.

Деактивация защищенной ячейки из приложения выполняется с помощью команды ***GrdPI_Deactivate***.

Установка флага **Деактивация** включает одноименный сервис, при этом становятся доступными опции (подробнее о них см. ниже):

- Поле ввода **Пароль деактивации**
- Поле ввода **Постоянный/случайный пароль**

А также появляется опция*, общая для всех сервисов:

- Поле **Допустимое количество ошибок при наборе паролей**

После включения сервиса **Деактивация** задайте пароль деактивации, определите его вид и допустимое число ошибочного ввода.

Чтение данных. Чтение по паролю

Если в свойствах защищенной ячейки включен сервис **Чтение данных**, то можно считать данные этой ячейки, обратившись к ней из приложения с помощью специальной команды Guardant API.

Если при этом будет включен сервис **Чтение по паролю**, то для выполнения команды чтения необходимо будет указать верный пароль.

* Если она не появилась ранее, после включения какого-либо другого сервиса

Доступ к содержимому ячейки из приложения выполняется с помощью команды ***GrdPI_Read***.

Установка флага **Чтение данных** включает одноименный сервис, при этом становится доступной опция **Чтение по паролю**.

Установка флага **Чтение по паролю** включает одноименный сервис, при этом становится доступными опции (подробнее о них см. ниже):

- Поле ввода **Пароль для чтения данных**
- Поле ввода **Постоянный / случайный пароль**

А также появляется опция[**], общая для всех сервисов:

- Поле **Допустимое количество ошибок при наборе паролей**

После включения сервиса **Чтение данных** включите сервис **Чтение по паролю**, задайте пароль для чтения данных, определите его вид и допустимое число ошибочного ввода.

Обновление данных

Если в свойствах ячейки включен сервис **Обновление данных**, то данные такой ячейки можно будет изменить, обратившись к ней из приложения с помощью специальной команды Guardant API.

Запись новых данных в дескриптор ячейки из приложения выполняется с помощью команды ***GrdPI_Update***.

Установка флага **Обновление данных** включает одноименный сервис, при этом становятся доступными опции (подробнее о них см. ниже):

- Поле ввода **Пароль для обновления данных**
- Поле ввода **Постоянный / случайный пароль**

А также появляется опция[**], общая для всех сервисов:

- Поле **Допустимое количество ошибок при наборе паролей**

После включения сервиса **Обновление данных** задайте пароль для обновления, определите его вид и допустимое число ошибочного ввода.

Пароли

Для выполнения команд Guardant API, изменяющих статус ячейки, либо ее содержимое, требуется указать пароль, предварительно заданный при включении сервиса(-ов). Для ввода паролей служат поля, расположенные напротив флагов, которые управляют тем или иным сервисом.

[**] Если она не появилась ранее, после включения какого-либо другого сервиса

Поля ввода в секции **Пароли** становятся доступными при использовании соответствующего сервиса.

Максимальная длина каждого пароля составляет 4 байта.

Постоянный/случайный пароль

Пароль для сервиса может быть постоянным, т. е. одинаковым для всех ключей, использующих данную ячейку, либо случайным, т. е. для каждого следующего ключа, который прошивается данным шаблоном маски, автоматически задается случайный пароль.

Важная информация

Использование случайных паролей предполагает работу в режиме базы данных GrdUtil.exe. Только в этом случае можно отследить, какой пароль был задан при программировании конкретного ключа, т. к. каждый факт прошивки ключа регистрируется в базе данных.

Для выбора вида пароля служит разворачивающийся список, расположенный напротив выбранного сервиса.

Списки в секции **Постоянный/случайный пароль** становятся доступными при использовании соответствующего сервиса.

Допустимое количество ошибок. Ячейка с неизменяемым статусом

Значение поля **Допустимое количество ошибок** задает предельное число попыток ошибочного ввода пароля при попытке доступа к ячейке (значение по умолчанию – 10).

В том случае, если при активации, деактивации, обновлении или других действиях, запрограммированных для данной ячейки, число попыток набора неправильного пароля было превышено, то ячейка блокируется и приобретает статус неизменяемой.

Любая работа с неизменяемой ячейкой невозможна.

Установить неактивное состояние

Флаг **Установить неактивное состояние** становится доступным после установки флага **Активация**.

Если схема защиты приложения предполагает, что ячейка должна быть деактивирована изначально, установите флаг **Установить неактивное состояние**. Неактивная ячейка выделяется в Редакторе маски бледно-серым шрифтом.

Размер данных защищенной ячейки

В нижней части диалога **Свойства защищенной ячейки** отображается статистическая информация о размере ее содержимого (с указанием выбранной системы счисления).

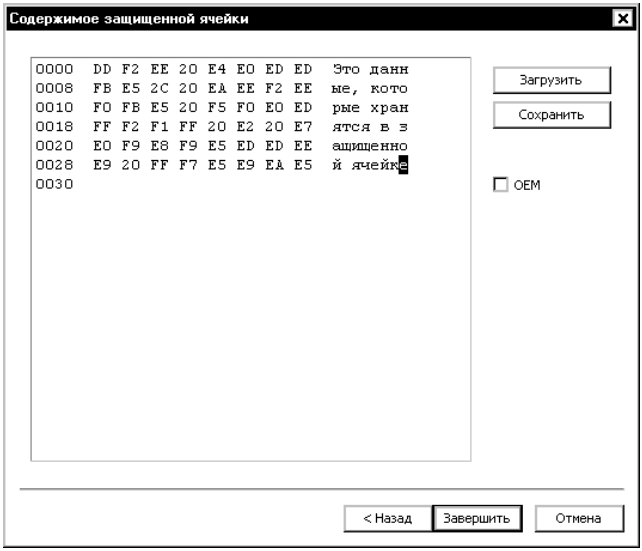
Размер дескриптора ячейки складывается из размеров его составных частей: определителя и набора сервисов.

***

После определения свойств ячейки остается поместить в нее данные. Для перехода к следующей странице служит кнопка **[Далее]** в нижней части диалога.

Редактирование содержимого ячейки

Диалог **Содержимое защищенной ячейки** представляет собой шестнадцатеричный редактор для ввода и изменения данных ячейки:



Данные вводятся непосредственно в окно редактора в виде шестнадцатеричных чисел или набора символов.

Элементы управления диалога **Содержимое защищенной ячейки**:

Элемент интерфейса	Назначение
Окно шестнадцатеричного редактора	Ввести значение защищенной ячейки
Кнопка **[Загрузить]**	Загрузить дамп из файла с расширением *.dmp
Кнопка **[Сохранить]**	Сохранить дамп в файле с расширением *.dmp
Флаг **OEM**	Выбрать Windows- / DOS-кодировку. По умолчанию используется Windows-кодировка (ANSI) – опция OEM отключена

***

После редактирования содержимого ячейки и нажатия на кнопку **[Завершить]** диалог создания защищенной ячейки закрывается, и ячейка появляется в списке полей маски. При этом GrdUtil.exe автоматически присваивает новой ячейке порядковый номер* и корректирует границу аппаратных запретов.

Теперь остается записать маску в ключ, и созданную ячейку можно будет использовать.

Важная информация

Следует учитывать, что вызов защищенной ячейки из приложения происходит по ее порядковому номеру, поэтому не рекомендуется добавлять новые алгоритмы или ячейки между уже существующими. Это повлечет изменение номеров защищенных полей, расположенных после созданного, и, как следствие, будет необходимо поправить параметры вызова функций Guardant API.

Таблица лицензий

Таблица лицензий – технология, применяющаяся в сетевых ключах Guardant Net III/ Net II/ Net. Ее назначение - контроль и управление лицензиями в сетевых программных комплексах, состоящих из нескольких модулей. *В Guardant Net III наличие таблицы лицензий обязательно, т. к. в ней содержится реальный ресурс лицензий ключа!*

При использовании таблицы лицензий сетевое ПО Guardant Net осуществляет двухуровневый контроль лицензий:

- Общее количество копий приложения или рабочих станций, на которых одновременно запущены приложения, ограничивается реальным сетевым ресурсом ключа, записанным в счетчике №2
- Количество копий приложения, либо рабочих станций, использующих определенный модуль программы, ограничивается ресурсом этого модуля, записанным в таблице лицензий

Таблица лицензий в ключах Guardant Net III отличается от таблицы лицензий в Guardant Net II / Net:

Ключ	Аппаратные запреты	Расположение в памяти ключа	Работа из приложения (Guardant API)
Guardant Net III	На чтение и запись (защищенная ячейка)	Между любыми аппаратными алгоритмами / защищенными ячейками, до начала полей других типов	GrdPI_Activate, GrdPI_Deactivate, GrdPI_Read, GrdPI_Update, GrdLogin, GrdLogout
Guardant Net / Guardant Net II	На запись	Строго после аппаратных алгоритмов и до начала полей других типов	GrdRead, GrdLogin, GrdLogout

* Для защищенных ячеек и алгоритмов действует единая нумерация

Таким образом, таблица лицензий в Guardant Net III представляет собой частный случай защищенной ячейки.

Создание таблицы лицензий

Важная информация

Для Guardant Net III наличие таблицы лицензий обязательно! В ней содержится реальный сетевой ресурс ключа!

Диалог создания таблицы лицензий выполнен в виде мастера, состоящего из нескольких страниц:

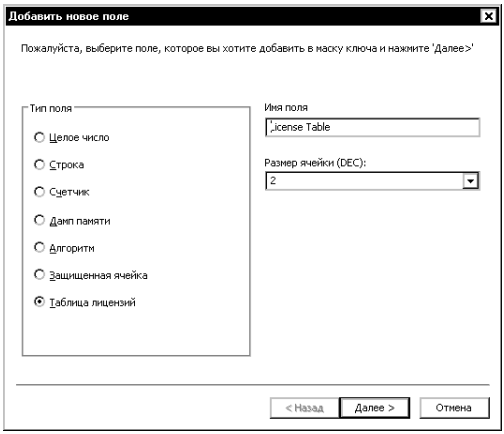
- Добавить таблицу лицензий
- Свойства таблицы лицензий
- Свойства защищенной ячейки (для маски Guardant Net III)

Переход к следующей странице происходит при помощи нажатия кнопки **[Далее]** после выполнения текущего диалога.

Добавление таблицы лицензий

Чтобы создать таблицу лицензий, выполните команду **Редактировать | Добавить новое поле**.

В появившемся диалоге выберите тип поля **Таблица лицензий**, задайте имя таблицы и выберите размер ее ячейки:



Размер ячейки

Возможный размер ячейки таблицы лицензий – 1 или 2 байта. Размер ячейки влияет на ресурс лицензий модуля защищенного приложения:

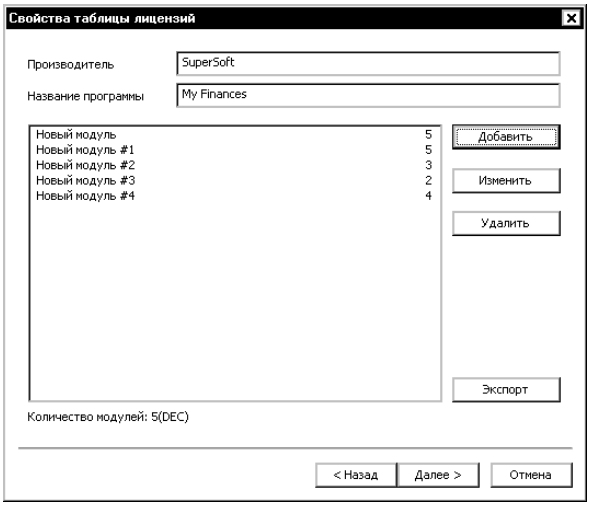
Размер ячейки, байтов	Максимальный ресурс лицензий
1	254
2	65534

Свойства таблицы лицензий

Диалог **Свойства таблицы лицензий** служит для определения параметров таблицы:

- **Реального сетевого ресурса ключа Guardant Net III**
- Названия производителя и приложения
- Количества модулей приложения
- Названия и ресурса каждого модуля

Также в диалоге предусмотрена возможность экспорта заданных параметров в конфигурационный файл сервера (GrdSrv.ini).



Для ввода названий производителя и приложения служат одноименные поля, расположенные в верхней части диалога.

Список модулей таблицы лицензий отображается в окне, занимающем центральную часть диалога. Справа от списка расположены кнопки для работы с модулями:

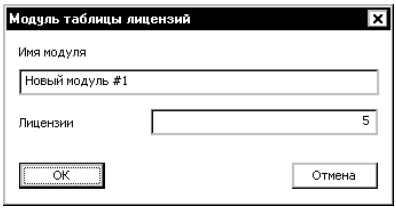
Кнопка	Назначение
[Добавить]	Вывести на экран диалог **Модуль таблицы лицензий** для создания нового модуля
[Изменить]	Вывести на экран диалог **Модуль таблицы лицензий** для редактирования параметров выбранного модуля
[Удалить]	Удалить выбранный модуль
[Экспорт]	Задать дополнительные параметры поиска ключа и перенести их вместе с данными таблицы лицензий в файл настроек сервера GrdSrv32.ini

Ниже списка модулей выводится статистическая информация о количестве модулей в выбранной системе счисления.

Модуль таблицы лицензий

Модули таблицы лицензий содержат информацию о сетевом ресурсе каждого из модулей защищенного приложения. Максимальное количество модулей таблицы лицензий – 127.

Чтобы добавить новый модуль в таблицу лицензий, нажмите кнопку **[Добавить]**, расположенную в правой части диалога **Свойства таблицы лицензий**. На экране появится диалог **Модуль таблицы лицензий**:



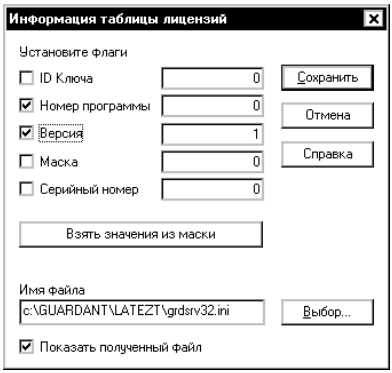
Поля **Имя модуля** и **Лицензии** служат для ввода названия модуля и количества лицензий соответственно.

По нажатию кнопки **[OK]** диалог закрывается, и созданный модуль появляется в списке.

Экспорт данных в файл настроек сервера ключа

Данные таблицы лицензий и дополнительные параметры поиска ключа необходимо перенести в конфигурационный файл сервера Guardant Net – ***GrdSrv.ini***.

По нажатию кнопки **[Экспорт]** появляется диалог **Информация таблицы лицензий**:



Дополнительные параметры поиска ключа задаются установкой соответствующих флагов. Установленный флаг означает, что при поиске сетевых ключей сервер должен учитываться указанный параметр.

Флаги расположены в диалоге по убыванию приоритета. Т. о., ID ключа имеет абсолютный приоритет, а битовая маска – низший. Уровень более приоритетного параметра выше суммы всех менее приоритетных.

Значения параметров поиска берутся из загруженной в редактор маски, кроме ID ключа, значение которого вводится вручную. Для автоматического ввода этих значений в соответствующие поля служат кнопки **[Маска]** и **[Ключ]** в секции **Брать значения из...** При необходимости поля ввода можно заполнить вручную.

Поле ввода **Имя файла** и кнопка **[Выбор]** служат для указания имени и пути к файлу настроек.

При установленном флаге **Показать полученный файл** после экспорта данных на экран выводится содержимое GrdSrv.ini с добавленным описанием ключа (секция [Key_xx], где xx – номер описания).

По нажатию кнопки **[Сохранить]** диалог закрывается, и описание ключа экспортируется в файл настроек GrdSrv.ini.

При запуске сервер Guardant Net считывает данные из подсоединенных ключей и сопоставляет их с описаниями ключей, содержащимися в конфигурационном файле. Информация из подходящего описания (владелец кода ключа, название приложения и его модулей) отображается в главном окне сервера ключа.

***

На этом создание таблицы лицензий в масках Guardant Net II / Net закончено. По нажатию на кнопку **[Завершить]** диалог создания нового поля закрывается, и таблица лицензий появляется в списке полей маски.

Диалог создания таблицы лицензий в маске Guardant Net III имеет дополнительное окно, в котором можно задействовать сервисы защищенных ячеек.

Свойства защищенной ячейки

Для таблицы лицензий формата Guardant Net III доступны все сервисы, которые управляют состоянием защищенной ячейки и обеспечивают доступ к ее содержимому из приложения:

- Активация/деактивация
- Чтение/обновление

Работа с сервисами таблицы лицензий полностью идентична работе с сервисами защищенных ячеек и аппаратных алгоритмов. Подробную информацию см. в разделе **Свойства защищенной ячейки**.

***

После активации необходимых сервисов создание таблицы лицензий в маске Guardant Net III закончено. По нажатию на кнопку **[Завершить]** диалог создания нового поля закрывается, и таблица лицензий появляется в списке полей маски.

Дампы, целые числа, строки и счетчики

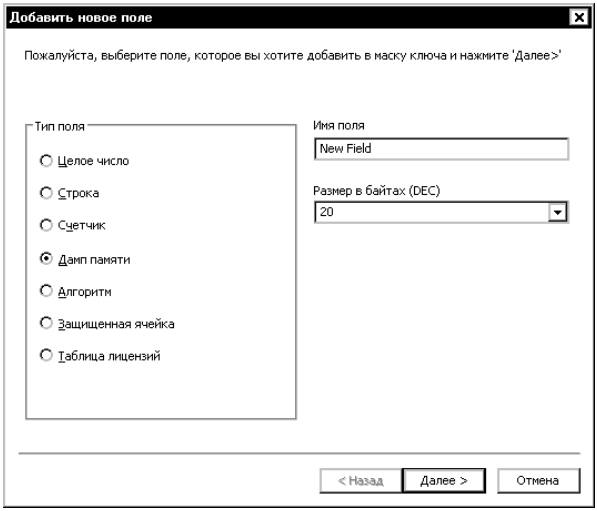
Кроме рассмотренных ранее защищенных ячеек, алгоритмов и таблиц лицензий, в GrdUtil.exe можно создавать поля следующих типов: дамп, целое число, строка и счетчик.

Диалог создания указанных полей выполнен в виде мастера, состоящего из страниц **Добавить новое поле** и **Свойства поля**. Переход в новое окно происходит по нажатию на кнопку **[Далее]** после выполнения текущего диалога.

Добавление нового поля

Чтобы создать поле одного из указанных типов, выполните команду **Редактировать | Добавить новое поле**.

В появившемся диалоге **Добавить новое поле** выберите тип, задайте имя и выберите размер поля:

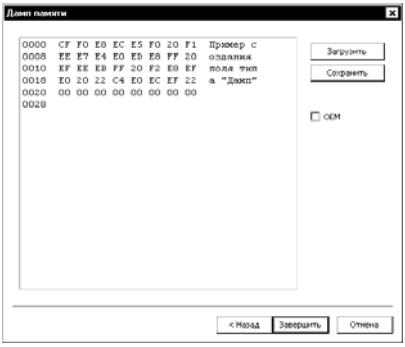


Размеры полей разных типов:

Тип поля	Возможный размер поля, байтов
Дамп	Произвольный размер
Строка	
Целое число	1, 2, 4, 8
Счетчик	

Свойства поля

Диалог **Свойства поля** служит для ввода данных и определения дополнительных параметров поля:



Элементы управления диалога **Свойства** для каждого типа поля:

Тип поля	Элемент интерфейса	Назначение
Дамп	Окно шестнадцатеричного редактора	Ввести значение дампа
	Флаг **OEM**	Выбрать Windows- / DOS-кодировку. По умолчанию используется Windows-кодировка (ANSI) – опция OEM отключена
	Кнопка **[Загрузить]**	Загрузить дамп из файла с расширением *.dmp
	Кнопка **[Сохранить]**	Сохранить дамп в файле с расширением *.dmp
Строка	Окно ввода	Ввести значение строки
	Переключатель **ANSI/Unicode**	Выбрать ANSI/Unicode-кодировку. По умолчанию используется ANSI-кодировка
Целое число	Флаг **Беззнаковое целое**	Выбрать подтип поля: со знаком/ без знака
	Список **Система счисления**	Выбрать систему счисления
	Поле **Значение**	Ввести число
Счетчик	Список **Система счисления**	Выбрать систему счисления
	Поле **Значение**	Ввести число

После ввода данных нажмите на кнопку **[Завершить]**, при этом диалог создания поля закрывается, и новое поле появляется в списке полей Редактора маски.

Проверка выполнения функций Guardant API

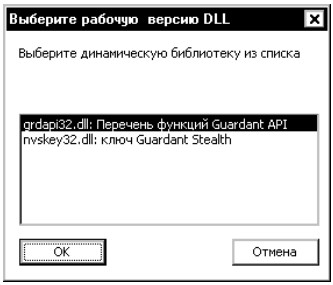
GrdUtil.exe предоставляет удобный сервис для проверки выполнения функций Guardant API с заданными параметрами. Такая возможность существенно экономит время при разработке системы защиты приложения: вызывать функции Guardant API и проверять результаты их выполнения можно без компиляции и отладки приложения.

Чтобы проверить корректность работы функции с заданными параметрами, выполните команду меню **Ключ | Выполнить функцию Guardant API**.

Важная информация

Описание операций Guardant API см. в разделе Интерфейс прикладного программирования Guardant.

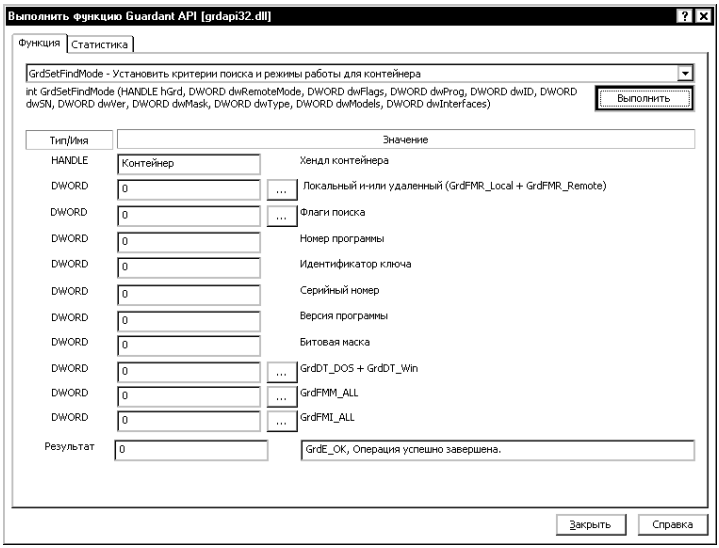
В появившемся диалоге выберите рабочую версию DLL из списка:



Поддерживаются следующие версии Guardant API:

- **GrdAPI32.dll** – библиотека функций Guardant API версии 5.x, интерфейс программирования для работы со всеми типами ключей Guardant
- **NvsKey32.dll** – библиотека Guardant API для ключей Guardant Stealth II / Stealth / Fidus, а также Guardant Net / Net II, если они используются в локальном варианте

После выбора библиотеки и нажатия на кнопку **[OK]** появляется диалог **Выполнить функцию Guardant API**; в заголовке которого отображается имя выбранной библиотеки:



Диалог состоит из двух вкладок: **Функция** и **Статистика**.

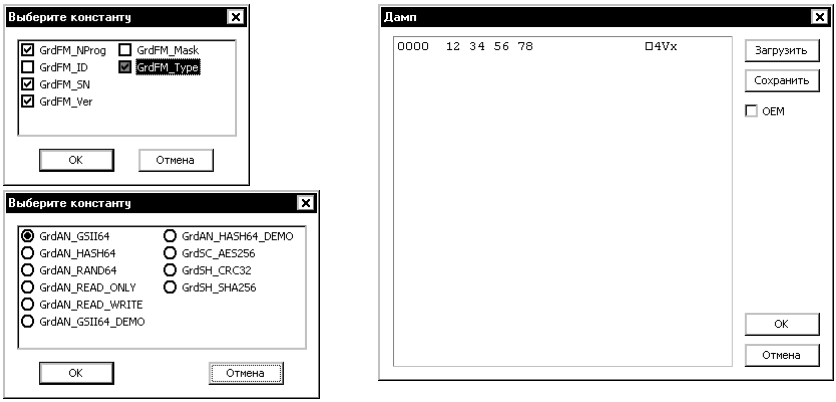
Функция

Вкладка позволяет выбрать из списка, задать параметры и выполнить функцию Guardant API, а также вывести в окне диалога результаты ее выполнения.

Элементы управления вкладки **Функция**:

Элемент интерфейса	Назначение
Разворачивающийся список функций, доступных для выполнения	Выбрать нужную функцию Guardant API
Прототип выбранной функции с перечислением параметров и типов в стандарте языка С	Прототип автоматически выводится ниже списка после выбора функции
Кнопка **[Выполнить]**	Выполнить выбранную функцию после задания параметров ее вызова
Столбец Тип/Имя	Отображение типа и названия параметров функции
Столбец Значение	Задать параметров функции
Поле Результат	Вывод кода ошибки, его расшифровка

Некоторые поля ввода параметров функции снабжены кнопкой **[...]**, которая служит для вызова дополнительных диалогов:



Вызов функций

Вызов функций производится следующим образом:

- Выберите нужную функцию из разворачивающегося списка
- Введите значения ее параметров в соответствующие поля
- Нажмите на кнопку **[Выполнить]**
- Проанализируйте код возврата функции

Важная информация

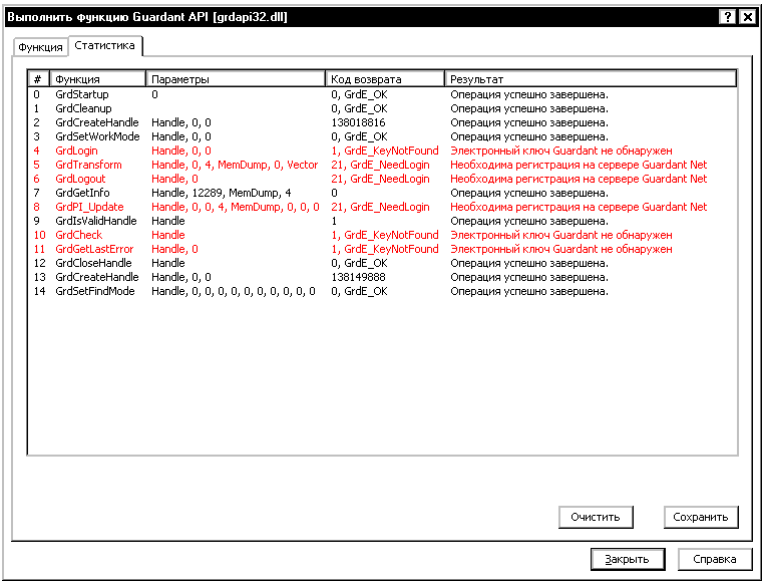
Существует определенный порядок вызова функций GrdAPI32.dll. Подробно см. раздел **Guardant API**.

Статистика

Результаты вызова функций сохраняются, их можно просмотреть во вкладке **Статистика**. Вкладка организована в виде таблицы, строки которой образует список функций, отсортированных в порядке вызова, а столбцы – характеристики функций:

Столбец	Описание
№	Порядковый номер вызванной функции
Функция	Название вызванной функции
Параметры	Перечень и значение заданных параметров функции
Код возврата	Код ошибки, полученный после вызова функции
Результат	Расшифровка кода ошибки

Вкладка **Статистика**:



#	Функция	Параметры	Код возврата	Результат
0	GrdStartup	0	0, GrdE_OK	Операция успешно завершена.
1	GrdCleanup		0, GrdE_OK	Операция успешно завершена.
2	GrdCreateHandle	Handle, 0, 0	138018816	Операция успешно завершена.
3	GrdSetWorkMode	Handle, 0, 0	0, GrdE_OK	Операция успешно завершена.
4	GrdLogin	Handle, 0, 0	1, GrdE_KeyNotFound	Электронный ключ Guardant не обнаружен
5	GrdTransform	Handle, 0, 4, MemDump, 0, Vector	21, GrdE_NeedLogin	Необходима регистрация на сервере Guardant Net
6	GrdLogout	Handle, 0	21, GrdE_NeedLogin	Необходима регистрация на сервере Guardant Net
7	GrdGetInfo	Handle, 12289, MemDump, 4	0	Операция успешно завершена.
8	GrdPI_Update	Handle, 0, 0, 4, MemDump, 0, 0, 0	21, GrdE_NeedLogin	Необходима регистрация на сервере Guardant Net
9	GrdIsValidHandle	Handle	1	Операция успешно завершена.
10	GrdCheck	Handle	1, GrdE_KeyNotFound	Электронный ключ Guardant не обнаружен
11	GrdGetLastError	Handle, 0	1, GrdE_KeyNotFound	Электронный ключ Guardant не обнаружен
12	GrdCloseHandle	Handle	0, GrdE_OK	Операция успешно завершена.
13	GrdCreateHandle	Handle, 0, 0	138149888	Операция успешно завершена.
14	GrdSetFindMode	Handle, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0	0, GrdE_OK	Операция успешно завершена.

Ниже таблицы расположены кнопки: **[Сохранить]** и **[Очистить]**. Первая служит для сохранения протокола вызова в файле формата *.cvs (*logfile.cvs* по умолчанию), а вторая – для сброса результатов.

Обновление памяти ключа

Обновление памяти ключа широко используется для изменения условий эксплуатации защищенного приложения или/ и его системы защиты, например:

- Продление срока использования программы, защищенной с ограничением времени использования или числа запусков
- Обновление версии программы
- Активация демо-версий
- Увеличение числа лицензий на использование сетевой версии программы
- Расширение функциональности программного комплекса
- Изменение данных или структуры памяти ключа

Возможны два варианта обновления памяти ключа:

- ***Удаленное (дистанционное)*** – обновление данных в ключе, находящемся у удаленного конечного пользователя
- ***Локальное*** – обновление данных в ключе, подсоединенном к компьютеру разработчика защищенного приложения

Зависимость обновления от подвида используемой маски

Процедура обновления памяти ключа зависит от того, в каком виде хранится маска обновляемого ключа – как прошивка, шаблон или файл *.nsd:

<table>
<tr><th rowspan="2">Вариант обновления</th><th colspan="3">Подвид маски</th><th rowspan="2"></th></tr>
<tr><th colspan="2">Файл маски | Шаблон маски</th><th>Прошивка</th></tr>
<tr><td rowspan="4">Удаленное</td><td colspan="2" rowspan="4">Полное обновление содержимого памяти удаленного ключа при помощи операции Ключ | Обновление ключа</td><td>Полное обновление памяти</td><td rowspan="8">Возможный тип обновления</td></tr>
<tr><td>Частичное обновление памяти</td></tr>
<tr><td>Обновление счетчика времени работы приложения</td></tr>
<tr><td>Операции активации / деактивации (только для ключей Stealth III/ Net III)</td></tr>
<tr><td rowspan="4">Локальное</td><td colspan="2" rowspan="4">Полная перезапись содержимого памяти локального ключа при помощи операции Ключ | Запись в ключ</td><td>Полное обновление памяти</td></tr>
<tr><td>Частичное обновление памяти</td></tr>
<tr><td>Обновление счетчика времени работы приложения</td></tr>
<tr><td>Операции активации / деактивации (только для ключей Stealth III/ Net III)</td></tr>
</table>

Удаленное обновление

Процесс удаленного (дистанционного) программирования ключей Guardant состоит из следующих этапов:

№	Этап	Кто выполняет
1	Создание запроса на обновление	Конечный пользователь по санкции разработчика
2	Создание дампа обновления	Разработчик после получения запроса на обновление от конечного пользователя
3	Обновление памяти ключа	Конечный пользователь после получения от разработчика дампа обновления
3а	Передача разработчику кода-подтверждения	Конечный пользователь после обновления памяти ключа
4	Завершение дистанционного обновления	Разработчик после получения кода-подтверждения от конечного пользователя

Далее детально рассматривается каждый этап.

1. Создание запроса на обновление

Конечный пользователь по санкции разработчика генерирует запрос на обновление (число-вопрос) при помощи специальной утилиты и передает его разработчику по любому каналу связи.

Важная информация

1. Начиная с версии ПО Guardant 5.1, для удаленного обновления ключей Guardant Stealth III/Net III и ключей Guardant предыдущих поколений используются РАЗНЫЕ клиентские утилиты.

2. Процедура дистанционного обновления ключа Guardant Stealth III/Net III требует наличия у конечного пользователя утилиты **GrdTRU.exe** и библиотеки **GrdVkc32.dll**, необходимой для работы утилиты. Эти файлы должны входить в комплект поставки защищенного приложения.

3. Процедура дистанционного обновления ключей Guardant Stealth II/Net II, Guardant Stealth/Net и Guardant Fidus требует наличия у конечного пользователя утилиты **GsRemote.exe** и библиотеки **GrdVkc32.dll**, необходимой для работы утилиты. Эти файлы должны входить в комплект поставки защищенного приложения.

4. Для успешного запуска клиентской утилиты удаленного обновления к порту компьютера должен быть подсоединен ключ с кодами доступа разработчика.

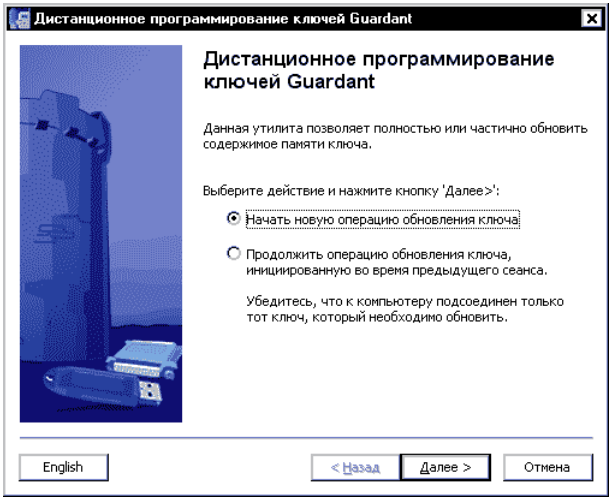
5. Если у конечного пользователя установлено несколько защищенных программ одного разработчика, то в порту необходимо оставить только тот ключ, который требует обновления.

6. Информацию, касающуюся работы с клиентской утилитой, рекомендуется включать в инструкцию для конечного пользователя.

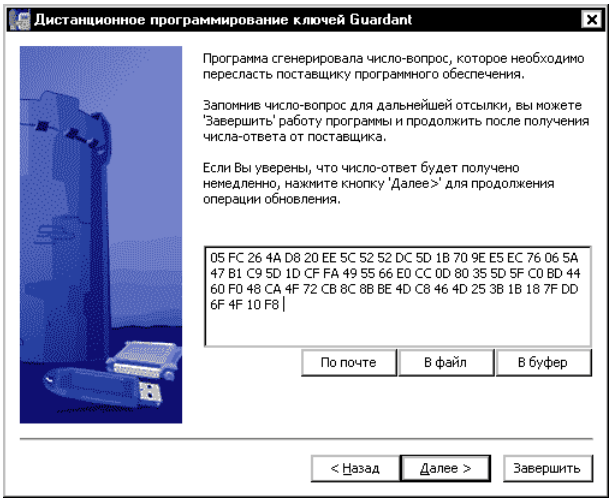
Утилита GsRemote.exe* выполнена в виде мастера, состоящего из нескольких страниц. Переход между страницами осуществляется при помощи кнопок **[Далее]** и **[Назад]**, расположенных в нижней части диалога.

* Для примера используется описание GsRemote.exe, работа с GrdTRU.exe **полностью аналогична**.

1. После первого запуска GsRemote.exe на экране появится страница, на которой необходимо выбрать пункт **Создать число-вопрос.**.:



2. На следующей странице отобразится сгенерированный ***запрос на обновление*** (число-вопрос) — последовательность шестнадцатеричных символов, содержащая информацию о ключе:



Запрос на обновление необходимо сохранить в файле (кнопка **[В файл]**), скопировать с помощью кнопки **[В буфер]** или сразу сформировать почтовое сообщение для разработчика приложения (кнопка **[По почте]**).

Работу утилиты GsRemote.exe можно завершить (кнопка **[Завершить]**) до получения дампа-ответа от разработчика.

2. Создание дампа обновления

После получения запроса на обновление разработчик формирует ***дамп обновления*** и отправляет его конечному пользователю.

Важная информация

1. Во время операции обновления к порту компьютера должен быть подсоединен ключ той же модели, что и обновляемый.

2. Обновление ключей Guardant Stealth III / Net III выполняется по технологии **Доверенного удаленного обновления** (см. пункт **Доверенное удаленное обновление**). Согласно этой технологии должен быть заранее задан пароль удаленного обновления, который вместе с данными маски записывается в ключ перед продажей конечному пользователю, а также сохраняется в прошивке.

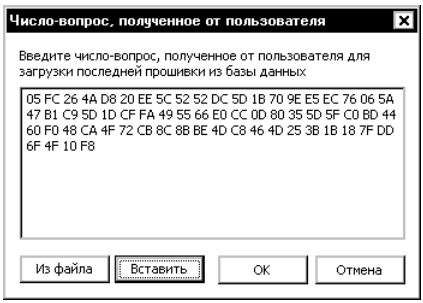
1. Перед началом процедуры получения дампа обновления загрузите в Редактор маску, которая будет использована для обновления удаленного ключа, и произведите в ней необходимые изменения.

Для обновления может использоваться шаблон или файл маски, а также прошивка.

Если удаленное обновление выполняется на основе прошивки, то при большом количестве прошивок бывает сложно найти нужную.

Чтобы автоматически загрузить в Редактор маски последнюю прошивку для удаленного ключа, выполните команду **База данных | Загрузить прошивку по числу-вопросу**.

Введите в окно появившегося диалога запрос на обновление, полученный от конечного пользователя (кнопка **[Из файла]** или **[Вставить]**):



После нажатия кнопки **[OK]** актуальная прошивка для удаленного ключа будет автоматически найдена по ID удаленного ключа и загружена в Редактор маски.

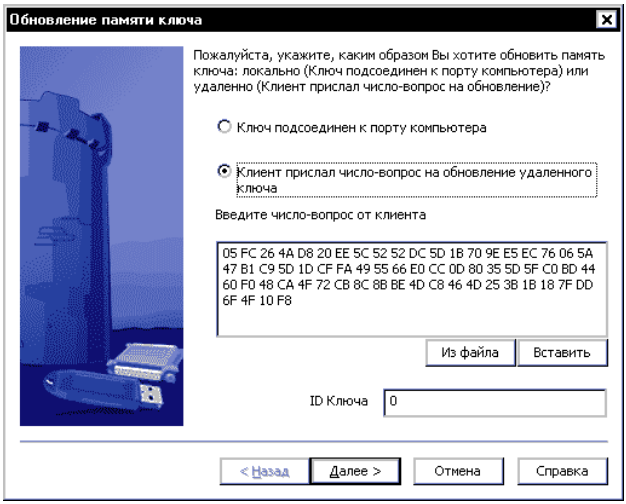
Внесите необходимые изменения в прошивку и начните процедуру создания дампа обновления.

Важная информация

Если в ходе редактирования шаблона маски или прошивки меняется их структура*, то для продолжения процесса обновления необходимо сохранить отредактированную маску в базе. Т. о. создается новый шаблон – с уникальным именем и/или версией, и обновление проводится уже на его основе.

2. Чтобы создать дамп обновления, выполните команду **Ключ | Обновить ключ**.

На экране появится диалог **Обновление памяти ключа**, который реализован в виде мастера, состоящего из нескольких страниц. Переход между страницами происходит при помощи кнопок **[Далее]** и **[Назад]**, расположенных в нижней части диалога:



* Под изменением структуры понимается удаление / добавление полей, а также редактирование содержимого полей, защищенных аппаратными запретами

а) Выберите с помощью переключателя пункт **Клиент прислал число-вопрос на обновление удаленного ключа*** и введите полученный запрос на обновление в поле. Для ввода воспользуйтесь кнопкой:

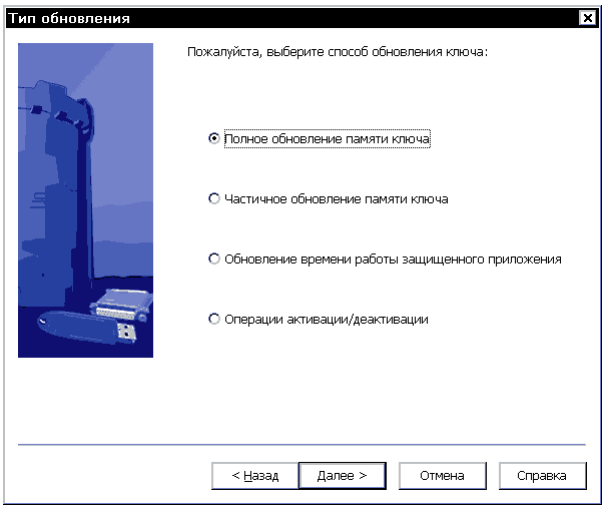
- **[Вставить]**, если число-вопрос находится в буфере обмена
- **[Из файла]**, если число-вопрос хранится в текстовом файле

После этого в поле ID ключа появится значение идентификатора ключа, извлеченное из запроса на обновление.

Важная информация

Если при обновлении от прошивки была успешно выполнена команда **База данных | Загрузить прошивку по числу-вопросу**, то запрос на обновление заносится в поле автоматически, а значение ID не выводится.

б) После нажатия кнопки **[Далее]** на экране появится страница **Тип обновления** с селектором выбора. Состояние селектора (доступные варианты обновления) зависит от разновидности маски, на основе которой выполняется перепрограммирование ключа (прошивка, шаблон или файл маски), и от типа ключа:



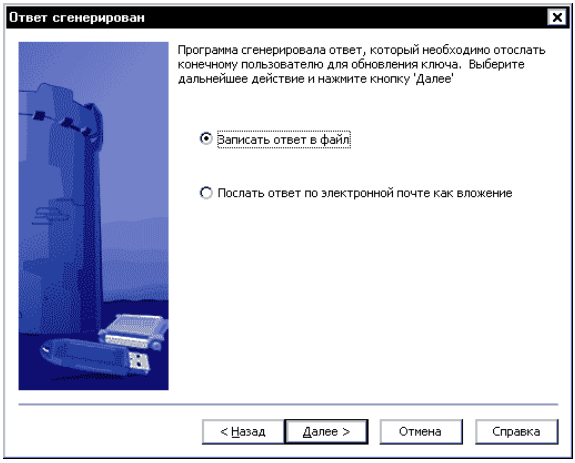
Выберите нужный пункт обновления и нажмите на кнопку **[Далее]**. Следующие несколько страниц служат для задания параметров обновления. Подробно о каждом из вариантов обновления см. далее в разделе **Тип обновления**.

* При обновлении файлом или шаблоном маски этот пункт является единственно доступным

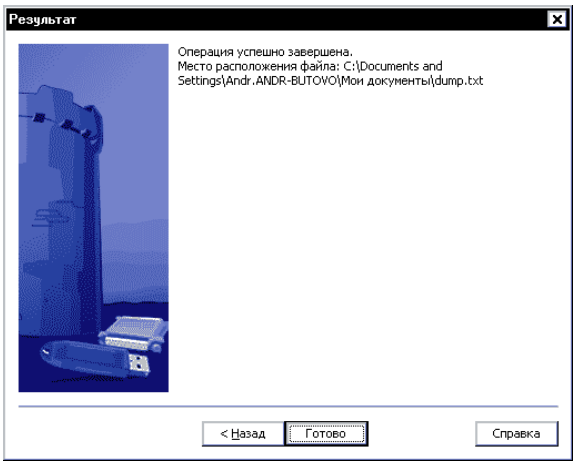
Важная информация

Во многих случаях удобно использовать следующую схему получения дампа обновления: заранее произвести в маске необходимые изменения, а затем, в ходе выполнения диалога **Дистанционное обновление** выбрать пункт **Полное обновление памяти ключа**.

в) После определения параметров обновления на экране появится страница **Ответ сгенерирован**, позволяющая выбрать способ дальнейшей обработки полученного дампа:



г) После нажатия кнопки **[Далее]** дамп сохраняется в файле или прикрепляется к почтовому сообщению в качестве вложения. На экране появляется **Результат** – последняя страница диалога обновления, на которой выдается сообщение об итоге операции:



Если используется база данных, то после создания дампа обновления GrdUtil.exe фиксирует факт прошивки удаленного ключа и присваивает этой операции статус незавершенной. Т. е. к списку прошивок маски с указанными параметрами добавляется новая прошивка, а в столбце **Признак завершенности** для нее указывается – ***В процессе*** (см. раздел **Прошивки**).

Сгенерированный дамп обновления необходимо передать конечному пользователю.

3. Обновление памяти ключа

После получения дампа обновления конечный пользователь может при помощи GsRemote.exe перепрограммировать память ключа.

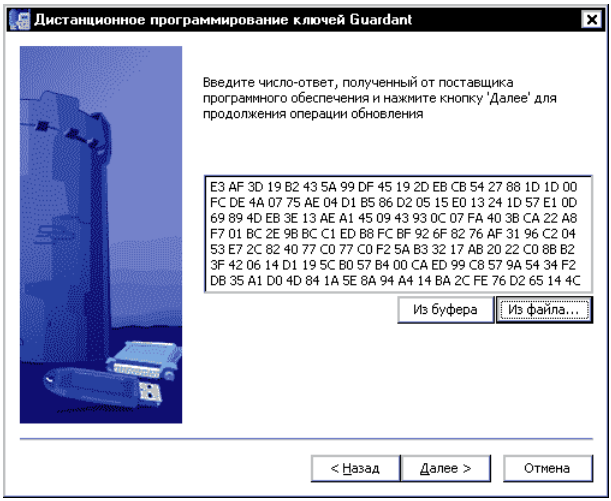
Важная информация

1. Данные обновления передаются в закодированном виде и могут быть использованы только один раз.

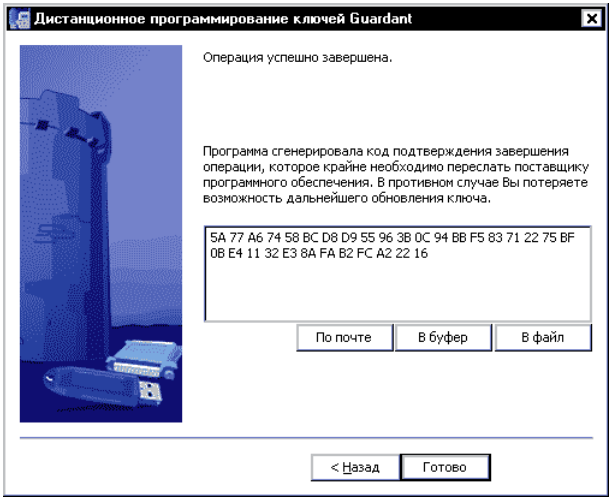
2. При генерировании данных обновления и прошивке их в ключ выполняется ряд проверок параметров ключа (ID, Общий код и некоторые другие). Этим исключается возможность подмены перепрограммируемого ключа.

Необходимо снова запустить утилиту GsRemote.exe, с помощью селектора-переключателя выбрать пункт **Обработать число-ответ..** и нажать на кнопку **[Далее]**.

На экране появится страница мастера, содержащая поле для ввода и отображения дампа обновления. Ввод дампа происходит при помощи одной из кнопок: **[Из буфера]** или **[Из файла...]**:



После ввода дампа и нажатия на кнопку **[Далее]** будет произведена операция по обновлению памяти ключа присланными данными. Затем на экране появится последняя страница мастера с итогами выполнения операции:



Важная информация

Последующие этапы необходимы в случае, когда обновление ключа происходит по факту прошивки или шаблона маски. Если же обновление выполнялось на основе данных файла маски, то шаг 3 является завершающим этапом обновления.

3а. Передача разработчику кода-подтверждения

В процессе обновления памяти ключа GsRemote.exe выдаст финальный ***код-подтверждение***, содержащий информацию о результате обновления (успешно/неудачно). Код-подтверждение необходимо сохранить в файле (кнопка **[В файл]**) или скопировать с помощью кнопки **[В буфер]** и передать разработчику приложения любым удобным способом.

4. Завершение удаленного обновления

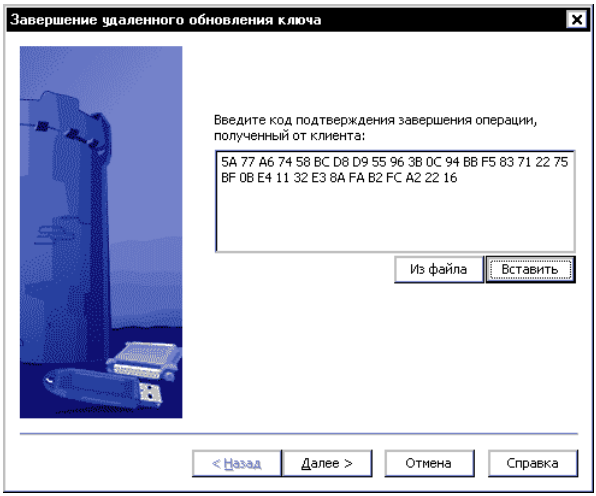
После получения кода-подтверждения разработчику необходимо завершить обновление: занести в базу данных информацию об итогах состоявшегося обновления.

Чтобы завершить процедуру удаленного обновления ключа, выполните команду **Ключ | Завершить удаленное обновление**.

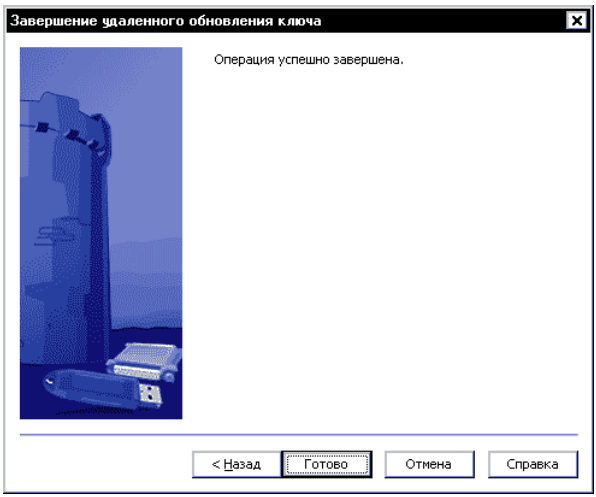
Важная информация

Завершение удаленного обновления не требует загрузки обрабатываемой прошивки в Редактор маски.

В появившемся диалоге **Завершение удаленного обновления ключа** введите код-подтверждение, полученный от конечного пользователя:



После ввода кода-подтверждения и нажатия на кнопку **[Далее]** будет произведена операция по завершению процедуры удаленного обновления. Затем на экране появится последняя страница мастера с итогами выполнения операции:



В случае успеха операции факт прошивки удаленного ключа получает статус завершенного. Признак завершенности для этой прошивки меняется на ***Завершен*** (См. раздел **Прошивки**).

Доверенное удаленное обновление

В ключах Guardant Stealth III / Net III используется ***Доверенное удаленное обновление*** (Trusted Remote Update). Основное достоинство этой технологии заключается в том, что вся информация, предназначенная для дистанционного обновления, расшифровывается и обрабатывается только внутри электронного ключа.

Доверенное удаленное обновление – технология безопасного удаленного обновления памяти электронного ключа, исключающая возможность компрометации и/или фальсификации данных. При обновлении используются алгоритмы шифрования GSII64 и Hash64. Идеология Доверенного удаленного обновления гарантирует, что однажды записанная в электронный ключ информация не может быть записана в него повторно, а данные, сгенерированные для одного ключа, не подойдут для другого.

Важная информация

Процедура Доверенного удаленного обновления предполагает использование уникальных данных, в частности, пароля удаленного обновления, для каждого ключа. Поэтому настоятельно рекомендуется при работе с Guardant Stealth III / Net III использовать режим базы данных для регистрации и сохранения уникальной информации. В противном случае, процедура удаленного обновления будет затруднена, или вовсе невозможна.

Пароль удаленного обновления

Пароль удаленного обновления – это 16-байтовая последовательность шестнадцатеричных символов, которая используется в процедуре Доверенного удаленного обновления для преобразования данных обновления. Пароль удаленного обновления может быть, как одинаковым для партии ключей, так и уникальным для каждого ключа.

Важная информация

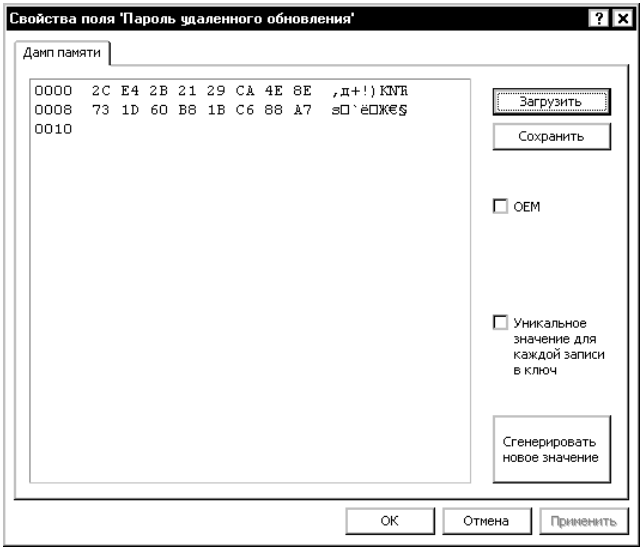
Для успешного выполнения Доверенного удаленного обновления пароль, содержащийся в удаленном ключе, должен совпадать с паролем, хранящимся в прошивке для этого ключа.

Пароль удаленного обновления содержится в неадресуемом, доступном для редактирования поле, которое расположено в маске типа Guardant Stealth III / Net III сразу после области полей специального назначения.

Для задания пароля удаленного обновления из приложения служит операция ***GrdTRU_SetKey***.

Чтобы просмотреть/ отредактировать пароль удаленного обновления, загрузите нужную маску, выделите поле **Пароль удаленного обновления** и выполните команду меню **Редактировать | Свойства поля**.

В появившемся диалоге **Свойства поля Пароль удаленного обновления**, который представляет собой шестнадцатеричный редактор, отредактируйте пароль:



По умолчанию GrdUtil.exe автоматически формирует пароль удаленного обновления. При необходимости его можно изменить, сформировав пароль самостоятельно, путем ввода нового значения непосредственно в окне редактора, или автоматически создать новый пароль (кнопка **[Сгенерировать новое значение]**).

Диалог также позволяет определить, будет ли использоваться уникальный или постоянный пароль. Если флаг **Уникальное значение для каждой записи в ключ** установлен, то при каждой операции прошивки в ключ значение пароля автоматически меняется на новую случайную последовательность. Т. о. каждый ключ получает уникальный пароль на обновление, хранящийся в памяти ключа и в его прошивке.

Элементы управления диалога **Пароль удаленного обновления**:

Элемент интерфейса	Назначение
Окно шестнадцатеричного редактора	Ввести значение пароля удаленного обновления
Кнопка **[Сгенерировать новое значение]**	Заменить текущее значение пароля новой случайной последовательностью
Флаг **Уникальное значение для каждой записи в ключ**	Использовать уникальный/ постоянный пароль. По умолчанию используется постоянный пароль – флаг не установлен.
Флаг **OEM**	Выбрать Windows- / DOS-кодировку. По умолчанию используется Windows-кодировка (ANSI) – опция OEM отключена
Кнопка **[Загрузить]**	Загрузить дамп из файла с расширением *.dmp
Кнопка **[Сохранить]**	Сохранить дамп в файле с расширением *.dmp

После ввода данных нажмите на кнопку **[Применить]** и закройте диалог.

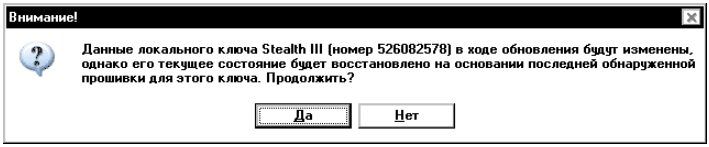
Особенности процедуры Доверенного удаленного обновления

Процедура Доверенного удаленного обновления при использовании GrdUtil.exe практически не отличается от обычного удаленного обновления (см. раздел **Удаленное обновление**), за исключением нескольких деталей:

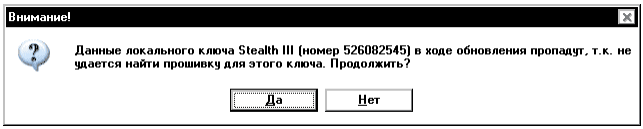
1) При выполнении команды **Ключ | Обновление ключа** GrdUtil.exe автоматически записывает в память «мастер»-ключа Guardant Stealth III / Net III, подсоединенного к порту, специальные алгоритмы GSII64 и Hash64. Эти алгоритмы участвуют в обработке запроса на обновление и кодировании данных обновления, используя пароль, хранящийся в прошивке обновляемого ключа.

После того, как дамп обновления сформирован, содержимое «мастер»-ключа автоматически восстанавливается путем записи в ключ последней прошивки для этого ключа[*].

Внешне эти действия проявляются только в виде предупреждения:

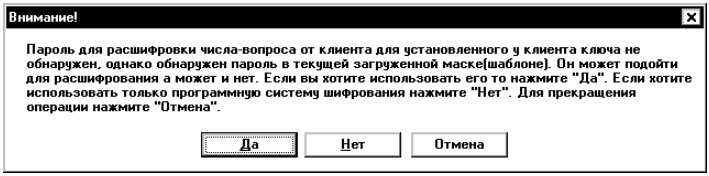


[*] Кроме, ситуации, когда обновление происходит не от прошивки. В этом случае содержимое «мастер»-ключа восстановлено не будет:



2) Если удаленное обновление выполняется без использования базы данных, на основе файла маски, то GrdUtil.exe предупреждает, что запрос на обновление не расшифруется при несовпадении паролей удаленного обновления в файле маски и в удаленном ключе.

Утилита предлагает в этом случае воспользоваться программным способом удаленного обновления (как для ключей предыдущих поколений) – см. кнопку **[Нет]** в окне предупреждения:



Локальное обновление

Локальное обновление памяти ключа подразумевает перепрограммирование памяти ключа, подсоединенного к порту компьютера, на котором установлен Комплект разработчика.

Чтобы обновить память локального ключа, выполните команду меню **Ключ | Обновление ключа**, либо **Ключ | Запись в ключ**. Причем первая команда будет доступна только в случае, когда используется прошивка ключа, а не шаблон или файл маски.

Особенности способов локального обновления:

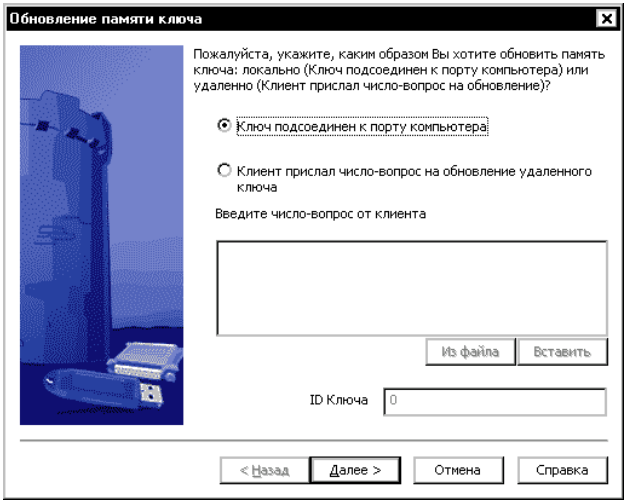
Способ обновления	Особенности
Команда **Ключ | Обновление ключа**	Способ доступен только при обновлении от прошивки. Позволяет выбрать тип обновления: полная перезапись или изменение содержимого отдельных полей
Команда **Ключ | Запись в ключ**	Происходит полная перезапись содержимого памяти ключа данными текущей маски

Локальное обновление командой Ключ | Обновление ключа

Практическая необходимость в локальном обновлении путем выполнения команды **Ключ | Обновление ключа** возникает, когда требуется сохранить в ключе «наработанные» данные (счетчики алгоритмов, к примеру), т. е. произвести изменение содержимого отдельных полей, не затрагивая остальные поля. В остальных случаях, проще отредактировать данные нужной прошивки или шаблона маски и выполнить команду **Ключ | Запись в ключ**.

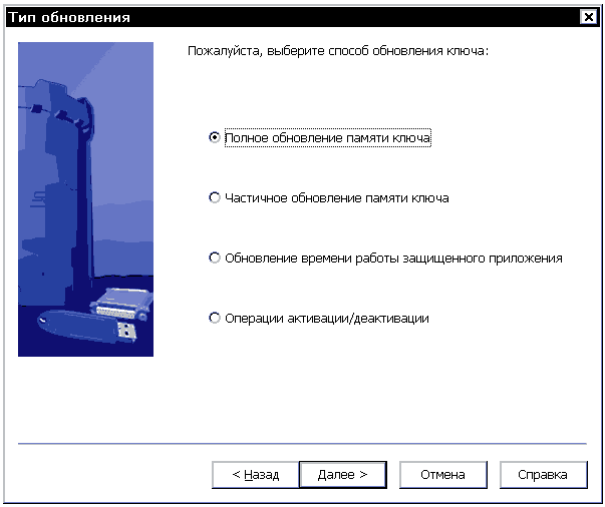
После выполнения команды **Ключ | Обновление ключа** на экране появляется диалог **Обновление памяти ключа**, который реализован в виде мастера, состоящего из нескольких страниц. Переход

между страницами происходит при помощи кнопок **[Далее]** и **[Назад]**, расположенных в нижней части диалогового окна.



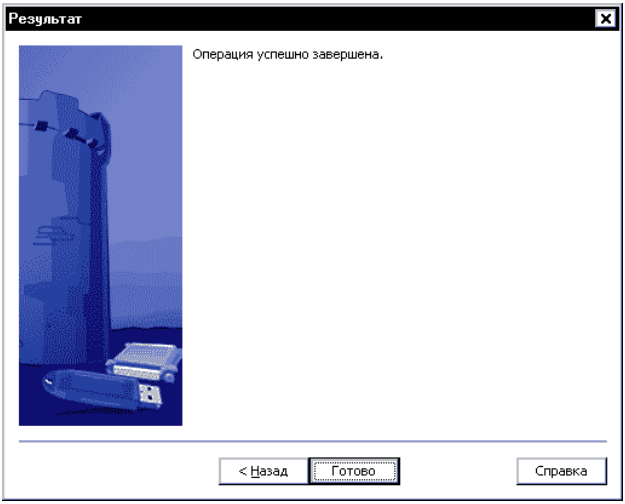
Выберите с помощью переключателя пункт **Ключ подсоединен к порту компьютера** и перейдите на следующую страницу.

На экране появится страница **Тип обновления** с селектором выбора. Состояние селектора (доступные варианты обновления) зависит от типа ключа:



Выберите нужный пункт обновления и нажмите на кнопку **[Далее]**. Следующие несколько страниц служат для задания параметров обновления и работы с редактором сценария. Подробно о каждом из вариантов обновления см. далее в разделе **Тип обновления**.

Последняя страница диалога локального обновления называется **Результат**, она отображает сообщение о результате операции:



После нажатия кнопки **[Готово]** обновление завершается, и GrdUtil.exe фиксирует в базе данных факт прошивки локального ключа.

Тип обновления памяти ключа

При удаленном или локальном обновлении памяти ключа по факту прошивки доступны следующие варианты обновления:

Тип обновления	Характеристика
Полное обновление памяти ключа	Перезапись всего содержимого памяти ключа данными текущей маски
Частичное обновление памяти ключа	Перезапись содержимого полей памяти ключа, не защищенных аппаратным запретом на запись. Для ключей Guardant Stealth III / Net III можно обновить только одно поле за процедуру обновления
Обновление времени работы защищенного приложения	Перезапись содержимого поля Счетчик №1 (GP)
Операции активации/ деактивации (только Guardant Stealth III/ Net III)	Активация, деактивация или перезапись выбранного участка защищенной ячейки, алгоритма или таблицы лицензий.

Полное обновление памяти ключа

При полном обновлении памяти ключа происходит перезапись всей памяти ключа данными выбранной маски.

Полное обновление выполняется в тех случаях, когда не требуется сохранения каких-либо «наработанных» данных в ключе, а также когда этот вариант обновления является единственно возможным – при обновлении от файла или шаблона маски.

Чтобы полностью обновить память ключа, выполните команду **Ключ | Обновление ключа**. На экране появится диалог мастера обновления.

Список страниц мастера обновления при полном обновлении памяти ключа:

№	Страница	Действия разработчика
1	Обновление памяти	Выберите удаленное или локальное обновление. При удаленном обновлении введите в поле запрос на обновление от конечного пользователя
2	Тип обновления	Выберите пункт **Полное обновление памяти ключа**
3	Результат	Проконтролируйте результаты работы мастера обновления

Частичное обновление памяти ключа

Вариант **Частичное обновление памяти ключа** служит для обновления участков памяти, которые не защищены аппаратными запретами на запись. Причем при удаленном обновлении Guardant Stealth III / Net III можно обновить только один участок памяти, во всех остальных случаях такое ограничение отсутствует.

Частичное обновление памяти актуально для тех ситуаций, когда необходимо сохранить «наработанную» в ключе информацию (поля счетчиков, счетчики алгоритмов), и полное обновление памяти неприемлемо. Частично обновить память можно только при обновлении от прошивки, сохраненной в базе данных.

Чтобы обновить участок памяти ключа, не защищенный аппаратным запретом на запись, выполните команду **Ключ | Обновление ключа**. На экране появится диалог мастера обновления.

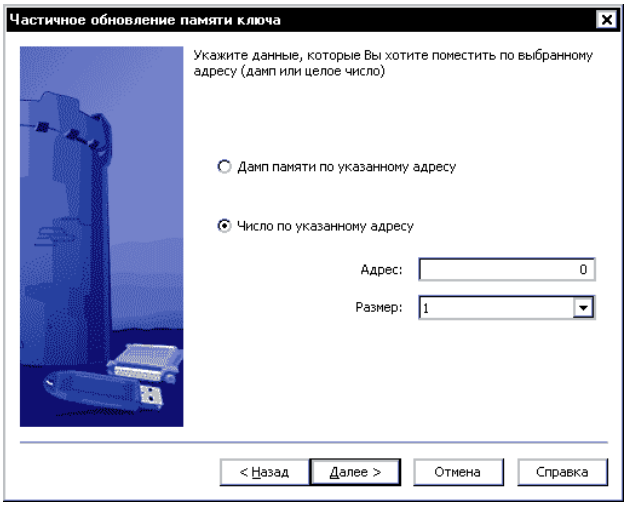
Список страниц мастера обновления при частичном обновлении памяти ключа:

№	Страница	Действия разработчика
1	Обновление памяти	Выберите удаленное или локальное обновление. При удаленном обновлении введите в поле запрос на обновление от конечного пользователя
2	Тип обновления	Выберите пункт **Частичное обновление памяти ключа**
3	Частичное обновление памяти ключа	Укажите тип данных, адрес и размер участка памяти, содержимое которого необходимо обновить (см. скриншот и описание после таблицы)

№	Страница	Действия разработчика
4	Частичное обновление памяти ключа	Введите данные обновления. Скриншот и описание интерфейса страницы совпадают с диалогом редактирования свойств соответствующих полей (см. раздел **Дампы, целые числа, строки и счетчики**)
5	Результат	Проконтролируйте результаты работы мастера обновления

Страница «Частичное обновление памяти ключа»

На странице **Частичное обновление памяти ключа** укажите тип данных, адрес и размер участка памяти, содержимое которого необходимо обновить:



Важная информация

Поля памяти, которые необходимо обновить, должны быть предварительно созданы в Редакторе маски.

Элементы управления страницы **Частичное обновление памяти ключа**:

Элемент интерфейса	Описание
Переключатель **Дамп памяти / Число**	Выбрать тип данных обновления: дамп или целое число
Поле **Адрес**	Указать адрес, начиная с которого будут записаны данные обновления. Этот адрес не должен быть защищен аппаратным запретом на запись
Поле/список **Размер**	Указать размер данных обновления, в байтах. Возможные значения: для дампа памяти – произвольный размер, для поля – 1, 2, 4

Обновление времени работы защищенного приложения

Если приложение защищено в режиме ограничения времени работы, возникает необходимость удаленно продлевать срок эксплуатации. Для этой цели в GrdUtil.exe предусмотрен специальный режим обновления.

Продлить время работы приложения без изменения всей памяти ключа можно только при обновлении от прошивки, сохраненной в базе данных.

Чтобы продлить срок работы приложения, выполните команду **Ключ | Обновление ключа**. На экране появится диалог мастера обновления.

Список страниц мастера обновления при обновлении работы защищенного приложения:

№	Страница	Действия разработчика
1	Обновление памяти	Выберите удаленное или локальное обновление. При удаленном обновлении введите в поле запрос на обновление от конечного пользователя
2	Тип обновления	Выберите пункт **Обновление времени работы**
3	Обновление времени работы	Определите новые значения для «чистого» времени работы приложения, а также продолжительности рабочего дня, если это необходимо. Скриншот и описание интерфейса страницы совпадают с диалогом редактирования свойств поля Счетчик GP (см. раздел «Ограничение времени работы приложения»)
4	Результат	Проконтролируйте результаты работы мастера обновления

Операции активации/ деактивации

Для ключей Guardant Stealth III/Net III существует специальный режим управления статусом защищенных ячеек и обновления их содержимого.

Его достоинство заключается в возможности управлять состоянием защищенной ячейки (активировать/деактивировать) и изменять ее содержимое (читать/перезаписывать), не затрагивая остальную память.

Режим изменения содержимого и статуса защищенной ячейки доступен только при обновлении от прошивки ключа Guardant Stealth III/Net III, сохраненной в базе данных.

Важная информация

1. Обновлять можно только те защищенные ячейки, для которых перед записью прошивки в ключ был включен хотя бы один из сервисов: активация, деактивация или запись данных.

2. Для обновления доступны только ранее включенные сервисы ячейки.

3. За сеанс удаленного обновления можно выполнить только по одной операции активации, деактивации и записи данных, если ранее эти сервисы были включены.

Чтобы обновить участок памяти ключа, защищенный аппаратным запретом на запись, выполните команду **Ключ | Обновление ключа**. На экране появится диалог мастера обновления.

Список страниц мастера обновления в режиме **Операции активации / деактивации**:

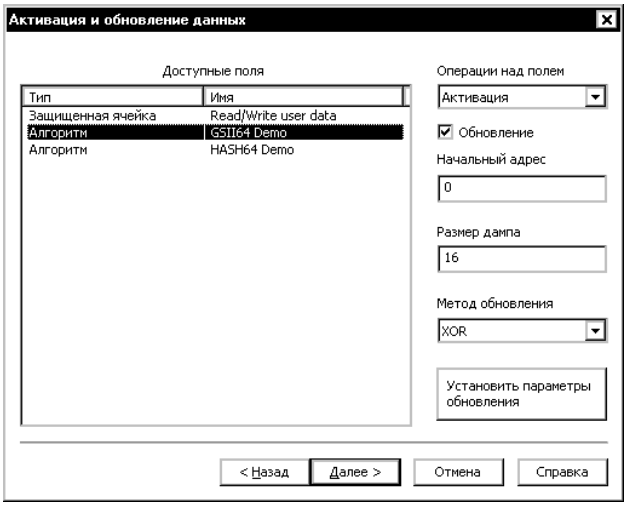
№	Страница	Действия разработчика
1	Обновление памяти	Выберите удаленное или локальное обновление. При удаленном обновлении введите в поле запрос на обновление от конечного пользователя
2	Тип обновления	Выберите пункт **Операции активации / деактивации**
3	Активация и обновление данных	Выберите защищенную ячейку, аппаратный алгоритм или таблицу лицензий и задайте для нее необходимые операции (см. скриншот и описание после таблицы)
4	Результат	Проконтролируйте результаты работы мастера обновления

Страница «Активация и обновление данных»

На странице **Активация и обновление данных** выберите защищенную ячейку, аппаратный алгоритм или таблицу лицензий и задайте для нее необходимые операции:

Возможные операции	Описание
Активация	Операция выполняется над ячейкой, находящейся в неактивном состоянии. Для успешного выполнения операции ячейка должна иметь ранее включенный сервис **Активация**.
Деактивация	Операция выполняется над ячейкой, находящейся в активном состоянии. Для успешного выполнения операции ячейка должна иметь ранее включенный сервис **Деактивация**.
Обновление участка защищенной ячейки	Операция выполняется над ячейкой, находящейся в активном состоянии. Для успешного выполнения операции ячейка должна иметь ранее включенный сервис **Обновление данных**. При выполнении операции происходит перезапись выбранного участка ячейки: 1) методом исключающего ИЛИ, 2) методом полного замещения содержимого

Страница **Активация и обновление данных**:



Элементы управления диалога **Активация и обновления данных**:

Элемент интерфейса		**Назначение**
Окно **Доступные поля**		Перечень защищенных ячеек, над которыми можно проводить операции обновления. Чтобы задать операции для поля, выделите его в списке
Список **Операции над полем**	Нет	Не использовать **Операции активации / деактивации**
	Активация	Активировать ячейку
	Деактивация	Деактивировать ячейку
Флаг **Обновление**		Установить флаг, чтобы использовать операцию обновления ячейки
Поле **Начальный адрес, байтов**		Задать смещение относительно начала обновляемой ячейки
Поле **Размер дампа, байтов**		Задать размер обновляемого участка ячейки
Список **Метод обновления**	**MOV**	Выполнить полное замещение содержимого участка ячейки на данные обновления
	XOR	Выполнить операцию исключающего ИЛИ над содержимым участка ячейки
Кнопка **[Установить параметры обновления]**		Ввести данные обновления в шестнадцатеричном виде

Программирование ключей из командной строки

Основные операции по программированию ключей можно выполнять из командной строки. Это особенно удобно в тех случаях, когда необходимо отделить обязанности по работе с Редактором маски и базой данных от обязанностей по прошивке и обновлению памяти ключа подготовленными данными.

Предполагается, что всю подготовку данных для прошивки ключа выполняет программист-разработчик системы защиты приложения, а операции по прошивке и обновлению памяти ключей возлагаются, например, на менеджера по продажам.

Для удобства работы и автоматизации процесса программирования ключей разработчик может создать GUI-утилиту или набор BAT-файлов на основе строчных команд GrdUtil.exe.

GrdUtil.exe предоставляет строчные команды для выполнения следующих действий:

- Запись маски в ключ
- Удаленное обновление ключа
- Завершение процедуры удаленного обновления ключа
- Локальное обновление ключа

Важная информация

1. Для успешного выполнения операций тип записываемой маски и тип программируемого ключа должны совпадать.

2. При операциях с локальным ключом необходимо учитывать, что запись производится в первый найденный ключ нужного типа.

3. Все числа, передаваемые в командную строку, должны быть в шестнадцатеричной системе счисления.

Запись маски в ключ

Для записи содержимого маски в ключ служит команда ***-write***.

В процессе выполнения команды происходит перезапись всей памяти ключа данными выбранной маски. В качестве источника данных может использоваться шаблон маски из базы данных или маска из файла *.nsd.

Команда ***-write*** имеет несколько опций:

Опция	Описание
-mask(name, version)	Записать в ключ содержимое шаблона маски. **name, version** – имя и версия шаблона маски из базы данных GrdUtil.exe

Опция	Описание
-user(user_name)	Зарегистрировать прошивку ключа на указанного конечного пользователя. **user_name** – имя конечного пользователя из базе данных GrdUtil.exe
-infile(file.nsd)	Записать в ключ содержимое файла маски. **file.nsd** – имя файла формата .nsd, который используется как источник обновления

Опции команды ***-write*** использовуются в следующих сочетаниях:

Сочетания опций	Описание
- write -mask(name, version)	Записать в ключ шаблон маски с указанным именем и версией. Прошивка будет зарегистрирована в базе данных на текущего конечного пользователя
- write -mask(name, version) -user(user_name)	Записать в ключ шаблон маски с указанным именем и версией. Прошивка будет зарегистрирована в базе данных на указанного конечного пользователя
- write -infile(file.nsd)	Записать в ключ маску из указанного файла

Удаленное обновление ключа

Для обновления памяти ключа, находящегося у конечного пользователя, служит команда ***-remote(query, outfile)***, где ***query*** - запрос на обновление, полученный от конечного пользователя, а ***outfile*** - имя текстового файла, в который будет помещен дамп обновления.

В качестве источника данных может использоваться шаблон маски из базы данных, маска из файла .nsd, а также дамп данных. В случае обновления от шаблона и файла маски происходит полная перезапись содержимого ключа. Обновление по дампу* позволяет перезаписать указанные участки памяти, не защищенные аппаратным запретом на запись, не затрагивая остальную память ключа.

Полученный дамп обновления необходимо передать конечному пользователю для завершения процесса удаленного обновления. Подробное описание процедуры удаленного обновления см. в разделе **Удаленное обновление**.

Важная информация

В текущей версии Комплекта разработчика удаленное обновление Guardant Stealth III / Net III из командной строки не поддерживается.

* Обновление по дампу возможно только при использовании базы данных GrdUtil.exe. Причем в базе должна храниться хотя бы одна прошивка для обновляемого ключа.

Команда ***-remote(query, outfile)*** имеет несколько опций:

Опция	Описание
-mask(name, version)	Создать дамп обновления на основе содержимого шаблона маски. **name, version** – имя и версия шаблона маски из базы данных GrdUtil.exe
-dump(address, hex_dump)	Создать дамп обновления на основе указанного дампа данных. **address** – адрес в памяти ключа, начиная с которого будет записан дамп обновления; **hex_dump** – данные, которые должны быть записаны в ключ; последовательность шестнадцатеричных символов
-infile(file.nsd)	Создать дамп обновления на основе содержимого файла маски. **file.nsd** – имя файла формата .nsd, который используется как источник обновления

Опции команды ***-remote(query, outfile)*** можно использовать в следующих сочетаниях:

Сочетания опций	Описание
-remote(query, outfile) **-mask(name, version)**	Создать дамп обновления на основе содержимого шаблона маски с указанным именем и версией. Прошивка будет зарегистрирована в базе данных на текущего конечного пользователя
-remote(query, outfile) **-dump(address, hex_dump)**	Создать дамп обновления на основе указанного дампа данных (нескольких дампов). Опция используется для обновления участков памяти, не защищенных аппаратными запретами на запись. Прошивка будет зарегистрирована в базе данных на текущего конечного пользователя
-remote(query, outfile) **-infile(file.nsd)**	Создать дамп обновления на основе содержимого указанного файла маски.

Завершение удаленного обновления

Для завершения обновления памяти ключа, находящегося у конечного пользователя, служит команда ***-complete (final_query)***, где ***final_query*** – код-подтверждение, полученный от конечного пользователя. Код-подтверждение содержит информацию об итоге обновления.

Завершение удаленного обновления необходимо только в случае работы в режиме базы данных GrdUtil.exe. Подробное описание процедуры завершения удаленного обновления см. в разделе **Удаленное обновление**.

Команда	Описание
-complete(final _query)	Обработать код-подтверждение, полученный от конечного пользователя. В случае успеха операции факт прошивки удаленного ключа получает статус завершенного (см. раздел Прошивки)

Локальное обновление

Для обновления памяти ключа, подсоединенного к порту компьютера разработчика, служит команда ***-local***.

В качестве источника данных может использоваться шаблон маски из базы данных, маска из файла .nsd, а также дамп данных. В случае обновления от шаблона и файла маски происходит полная перезапись содержимого ключа. Обновление по дампу* позволяет перезаписать указанные участки памяти, не защищенные аппаратным запретом на запись, не затрагивая остальную память ключа.

Команда ***-local*** имеет несколько опций:

Опция	Описание
-mask(name, version)	Записать в ключ содержимое шаблона маски. **name, version** – имя и версия шаблона маски из базы данных GrdUtil.exe
-dump(address, hex_dump)	Записать в ключ дамп данных **address** – адрес в памяти ключа, начиная с которого будет записан дамп **hex_dump** – данные, которые должны быть записаны в ключ; последовательность шестнадцатеричных символов
-infile(file.nsd)	Записать в ключ содержимое файла маски. **file.nsd** – имя файла формата .nsd, который используется как источник обновления

Опции команды ***-local*** используются в следующих сочетаниях:

Сочетания опций	Описание
-local -mask(name, version)	Записать в ключ содержимое шаблона маски с указанным именем и версией. Прошивка будет зарегистрирована в базе данных на текущего конечного пользователя
-local -dump(address, hex_dump)	Записать в память ключа указанный дамп обновления (несколько дампов). Опция используется для обновления участков памяти, не защищенных аппаратными запретами на запись. Прошивка будет зарегистрирована в базе данных на текущего конечного пользователя
-local -infile(file.nsd)	Записать в ключ содержимое указанного файла маски

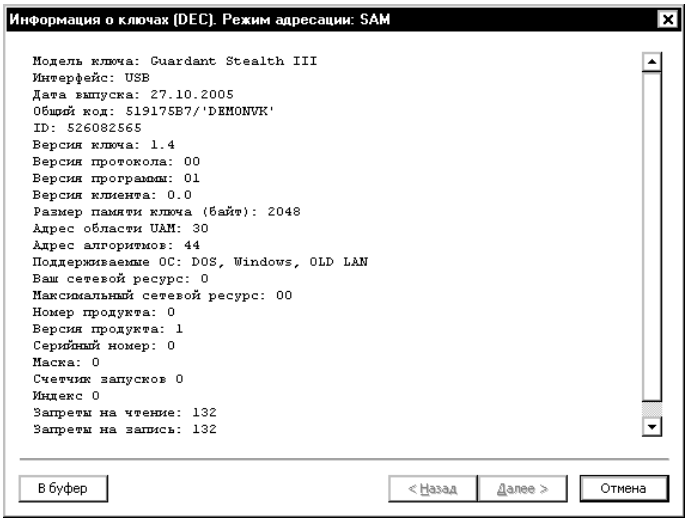
* Обновление по дампу возможно только при использовании базы данных GrdUtil.exe. Причем в базе данных должна храниться хотя бы одна прошивка для обновляемого ключа.

Вспомогательные операции

В этот раздел вынесено описание вспомогательных (**Получение информации о ключе**), а также редко используемых (**Конвертирование маски**) или устаревших возможностей утилиты, которые поддерживаются в GrdUtil.exe версии 5.x для совместимости с ключами Guardant Stealth / Net (**Программное преобразование данных при помощи GrdEncode и GrdDecode**).

Получение информации о ключе

Чтобы получить информацию о ключе, подсоединенном к порту компьютера, выполните команду меню **Ключ | Информация о ключе**.



Если к портам подсоединено несколько ключей, то информация по каждому из них выдается на отдельной странице диалога. Переключение между страницами происходит по нажатию кнопок **[Далее]** и **[Назад]**.

Кнопка **[В буфер]** служит для копирования информации о ключе в буфер обмена.

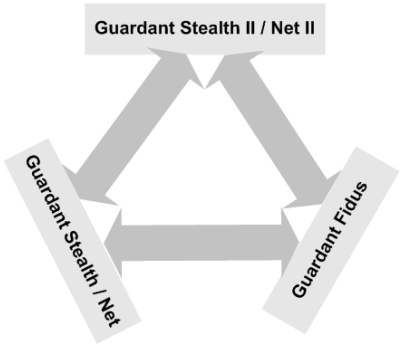
GrdUtil.exe выдает ту же информацию о ключе, что и утилита диагностики ключа: значения полей доступных только для чтения, полей специальных операций и полей общего назначения. Кроме того, показывается граница аппаратных запретов.

Информация о ключе выводится в системе адресации SAM и выбранной ранее системе счисления.

Конвертирование маски

При поддержке нескольких типов ключей может возникать необходимость переноса информации из маски для ключей одного типа в маску другого типа (например, для сохранения сложившейся адресации полей).

GrdUtil.exe предоставляет следующие варианты конвертирования масок:

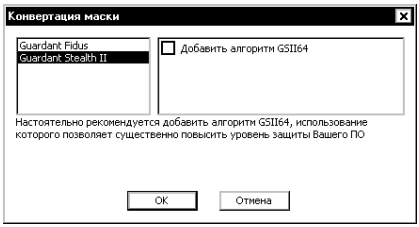


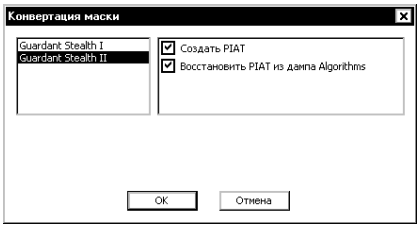
Важная информация

Нельзя конвертировать маски ключей Guardant Stealth III / Net III

Чтобы выполнить перенос данных в маску для другого типа ключей, выполните команду **Файл | Конвертировать маску**.

В появившемся диалоге выберите ***маску-приемник*** из списка доступных вариантов слева и установите необходимые опции конвертирования:





Набор опций конвертирования зависит от типа ***маски-передатчика*** и ***маски-приемника***:

Опция	**Маска-передатчик**	**Маска-приемник**	**Флаг установлен**	**Флаг не установлен**
Создать PIAT**	Fidus	Stealth / Net, Stealth II / Net II	В маске-приемнике будет создана таблица размещения алгоритмов и стандартные аппаратные алгоритмы	Поле, зарезервированное под таблицу алгоритмов, будет заполнено нулями, алгоритмы не будут созданы
Восстановить PIAT из дампа Algorithm*	Fidus	Stealth / Net, Stealth II / Net II	В маске-приемнике будут восстановлены таблица размещения алгоритмов и аппаратные алгоритмы	Поле, зарезервированное под таблицу алгоритмов, будет заполнено нулями, алгоритмы не будут восстановлены. Их данные будут перенесены в маску-приемник «как есть», т. е. в виде дампа
Добавить алгоритм GSII64	Stealth / Net	Stealth II / Net II	В маске-приемнике к алгоритмам Guardant Stealth будет добавлен алгоритм указанного типа	Содержимое маски-передатчика будет перенесено в маску-приемник без изменений
Сохранить PIAT в дампе	Stealth / Net, Stealth II / Net II	Fidus	В маске-приемнике будет создан дамп, содержащий данные таблицы размещения алгоритмов и самих аппаратных алгоритмов. Этот дамп можно использовать для восстановления данных аппаратных алгоритмов при обратном конвертировании	Данные алгоритмов не будут перенесены в маску Guardant Fidus

** PIAT – Protected Item Allocation Table, таблица размещения защищенных ячеек и аппаратных алгоритмов

* Опция доступна при обратном конвертировании данных Guardant Stealth / Net или Guardant Stealth II / Net II, сохраненных ранее в маске Guardant Fidus

Программное взаимообратное преобразование данных

Технические характеристики электронных ключей не позволяют выполнять кодирование / декодирование данных большого объема (~ от мегабайта и более) за приемлемое время. Для выполнения этой задачи приходится использовать программное взаимообратное преобразование информации.

В ключах Guardant Stealth / Net, Guardant Stealth II / Net II подобное преобразование выполняет программный алгоритм, реализуемый операциями GrdEncode и GrdDecode, а пароль для преобразования обрабатывается аппаратным однонаправленным алгоритмом типа Fast (реализуется операцией GrdCodeInit).

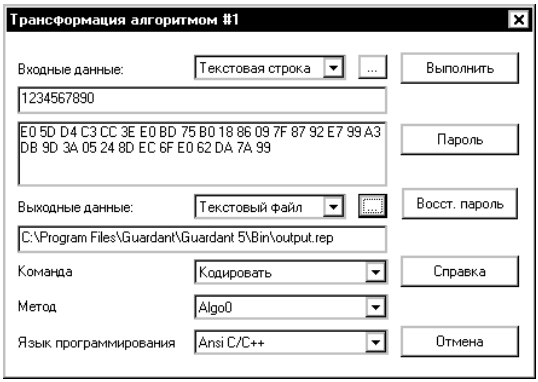
GrdUtil.exe предоставляет сервис подготовки данных для их программного преобразования. С помощью утилиты можно задать пароль взаимообратного преобразования и закодировать / декодировать информацию. Подготовленные данные в дальнейшем используются при защите приложения.

Важная информация

Алгоритм ключей Guardant Stealth / Net является морально устаревшим. В современной системе защиты приложения не рекомендуется использовать этот алгоритм. Сервис программного взаимообратного преобразования данных поддерживается GrdUtil.exe для совместимости с первым поколением ключей Guardant.

Подготовка данных для преобразования в GrdUtil.exe

Выделите в списке полей редактора маски алгоритм типа ***Fast*** и выполните команду **Ключ | Преобразование**.



В появившемся диалоге **Трансформация алгоритмом №N**[**] определите следующие параметры:

- Входные данные и их вид
- Пароль преобразования
- Выходные данные и их вид
- Направление и метод кодирования
- Язык программирования (если выходные данные представлены в виде исходного текста)

Входные данные

Данные, которые необходимо преобразовать, могут быть представлены в виде строки символов или файла любого формата.

Для выбора вида данных на входе служит раскрывающийся список в верхней части диалога.

По нажатию кнопки **[...]**, расположенной напротив списка, открывается, либо диалог **Введите строку для ввода строки символов**, либо стандартный системный диалог для указания имени файла данных и пути к нему:

Заданная строка или имя файла с данными и путь к нему отображаются в поле ввода **Входные данные**.

Пароль

Пароль для преобразования с фиксированной длиной 32 байта генерируется автоматически при открытии диалога **Трансформация алгоритмом №N** и отображается в соответствующем поле ввода.

При необходимости пароль по умолчанию можно изменить. По нажатию на кнопку **[Пароль]** на экране появляется шестнадцатеричный редактор, в окне которого можно скорректировать значение или ввести новый пароль.

Выходные данные

Преобразованные данные могут иметь следующий вид:

Данные на выходе	Описание
Исходный текст	Текстовый файл, содержащий закодированные данные в виде массива чисел и созданный по правилам синтаксиса одного из основных языков программирования: Assembler, C/C++, Pascal/Delphi
Двоичный файл	Закодированная последовательность байтов

Для выбора представления данных на выходе служит раскрывающийся список в средней части диалога.

[**] Где N – порядковый номер алгоритма

По нажатию кнопки **[...]**, расположенной напротив списка, открывается стандартный системный диалог для указания имени файла с преобразованными данными (по умолчанию - OutPut.rep) и пути к нему.

Имя файла с данными и путь к нему отображаются в поле ввода **Выходные данные**.

Кнопка [Выполнить]

По нажатию на кнопку **[Выполнить]** начинается процесс кодирования (декодирования) данных. Кнопка становится доступной после заполнения секций **Входные данные** и **Выходные данные**.

Язык программирования

Развертывающийся список **Язык программирования** расположен в нижней части диалога и становится доступным в том случае, если выбрано представление закодированных данных в виде исходного текста.

В списке представлены следующие языки программирования: Assembler, C/C++, Pascal/Delphi.

Кодирование и декодирование

Разворачивающийся список **Команда** служит для выбора операции, которая будет совершена над входными данными:

- Кодирование (выполняется операция GrdEncode)
- Декодирование (выполняется операция GrdDecode)

Выбор метода преобразования

Для выбора метода преобразования служит разворачивающийся список **Метод** в нижней части диалога.

Возможные методы преобразования информации:

Метод	Область применения	Описание
Базовый	Преобразование нестроковых данных	Преобразование выполняется блоками по 32 байта
Символьный	Преобразование строковых данных	Преобразование выполняется блоками по 32 байта. Строки преобразуются т. о., чтобы в них не было неотображаемых символов
Файловый	Преобразование файлов	Преобразование выполняется блоками произвольной длины (желательно не менее 32 байтов). При кодировании и декодировании необходимо сохранять один и тот же порядок следования блоков

***

После подготовительных действий можно приступать к выполнению преобразования

Выполнение преобразования

Преобразование начинается после нажатия на кнопку **[Выполнить]**, расположенную в верхней части диалога.

Запись данных маски в память ключа

Сначала утилита выдаст запрос о записи данных маски в ключ.

Предварительная запись маски в ключ необходима в случае, если в преобразовании пароля участвует новый алгоритм Fast, либо алгоритм Fast, у которого был изменен определитель.

Важная информация

После выполнения операции записи значение полей типа **счетчик** в маске автоматически увеличивается на единицу. Для восстановления предыдущих значений необходимо выполнить операцию чтения ключа.

Сохранение отчета

Далее на экране появится диалог сохранения отчета о преобразовании, в котором нужно указать имя (по умолчанию report.rep) и путь к файлу отчета.

Отчет о преобразовании представляет собой текстовый файл, созданный по правилам синтаксиса указанного ранее языка программирования. В отчете содержится статистическая информация о параметрах преобразования и заданный пароль в виде массива.

Процесс преобразования

После сохранения отчета на экране появится окно индикатора выполнения.

Процесс преобразования состоит из следующих этапов:

1) Операция GrdCodeInit вызывает выбранный аппаратный алгоритм Fast, который видоизменяет заданный пароль.

2) Операция GrdEncode (или GrdDecode, если выполняется декодирование) производит преобразование данных выбранным методом с использованием видоизмененного пароля

3) Преобразованные данные помещаются в заданный выходной файл в виде массива или последовательности байтов

Декодирование

Процесс декодирования выполняется аналогично процессу кодирования (см. выше).

Важная информация

Для корректного декодирования данных необходимо пользоваться тем же алгоритмом, паролем и методом преобразования, которые использовались для кодирования.

Глава 5
Guardant API

Интерфейс прикладного программирования Guardant представляет собой набор функций для работы с ключом, собранных в библиотечных файлах. Также в состав ПО Guardant входят исходные тексты примеров использования Guardant API для основных языков программирования и сред разработки.

Защита с использованием функций API заключается во встраивании в исходный код приложения вызовов к ключу согласно предварительно разработанной схеме (рекомендации по эффективному применению Guardant API содержатся в Приложении А).

Основные достоинства защиты при помощи API – высокий уровень стойкости защищенного приложения к взлому при правильном использовании функций API; возможность создания эффективной и нетривиальной защиты любой категории сложности; широкие возможности для использования аппаратных алгоритмов ключа – основного инструмента, с помощью которого усложняется логика работы защищенного приложения.

Для достижения большей защищенности приложения рекомендуется комбинировать автозащиту и защиту при помощи Guardant API.

Особенности Guardant API версии 5.x

Псевдокод

Псевдокод - это технология защиты исполняемых файлов от изучения логики их работы. Суть ее состоит в том, что определенные фрагменты исполняемых файлов дизассемблируются, анализируются и преобразуются в защищенный код виртуальной машины, которая генерируется тут же.

Анализировать логику работы защищенного подобным образом кода на порядок сложнее, чем инструкции Intel-совместимых процессоров, поскольку для псевдокода не существует никакого стандартного инструментария (отладчиков, дизассемблеров). Взломщику придется все делать вручную или создавать свои собственные инструменты.

В каждой копии виртуальной машины реализуется уникальным образом:

- Набор внутренних команд псевдокода
- Множественный взаимный контроль целостности (для затруднения модификации и установки точек останова)
- Обфускация кода виртуальной машины (замусоривание реального кода вторичным)
- Шифрование кода самой виртуальной машины
- Шифрование самого псевдокода
- Параметры многих команд рассчитываются только во время выполнения (защита от статического дизассемблирования)
- Отсутствие постоянных сигнатур в защищенном коде (затруднение поиска псевдокода в защищенном приложении)

Handle-ориентированность

Для работы с ключом создается контекст, который сохраняется в специальном защищенном контейнере. Доступ к контейнеру осуществляется через хэндл. Хэндл является идентификатором контекста, с которым и оперируют функции Guardant API.

Это позволяет упростить обмен с API за счет минимизации числа вызовов и параметров функций, а также сократить время работы функций.

Поддержка многопоточности

В защищенном контейнере сохраняются параметры поиска и прочие внутренние переменные. Это существенно расширяет возможности разработчиков приложений, поскольку наличие контекста позволяет работать с несколькими ключами из нескольких потоков одновременно.

Это дает возможность:

- Существенно повысить защищенность, поскольку отлаживать и анализировать многопоточные приложения гораздо сложнее
- Легче интегрировать API в сложные многопоточные приложения
- Работать с несколькими ключами одновременно

Защищенность, контроль целостности и аутентификация

Содержимое контейнера зашифровано. Осуществляется контроль целостности данных, хранящихся в контейнере. Коды доступа тоже хранятся в контейнере в зашифрованном виде и их не нужно указывать при каждом вызове функций Guardant API. В защищенном

контейнере хранятся только те коды доступа, которые действительно используются.

Guardant API контролирует целостность своего кода во избежание вмешательств извне. Код Guardant API защищен от статического и динамического анализа при помощи псевдокода.

Драйвер Guardant подписан при помощи асимметричной криптографии. Защищенное приложение проверяет подпись драйвера (автоматически при вызове функций Guardant API 5.x). Взломщик не сможет создать драйвер-эмулятор электронного ключа, чтобы у него сходилась подпись, поскольку закрытый ключ ему не известен.

Так же как и код Guardant API, драйвер защищен от анализа с использованием технологии псевдокода. Это подразумевает, в том числе, и контроль целостности кода драйвера.

Унификация

Для унификации и упрощения написания кода все функции Guardant API сделаны универсальными для работы, как с локальными, так и с сетевыми ключами. При инициализации библиотеки необходимо указать, с какими ключами будет работать приложение: локальными, сетевыми, или с обоими типами.

Это существенно упрощает процесс программирования систем защиты, рассчитанных на работу и с сетевыми и с локальными ключами, поскольку код защиты можно написать единый. Также максимально упрощается переход на сетевые ключи: по сути надо будет поменять всего один флаг в функции **GrdSetFindMode()**.

Список функций Guardant API

Функции Guardant API общего назначения

Функции общего назначения позволяют осуществлять сервисные операции, связанные с инициализацией, настройкой и деинициализацией Guardant API, а также выполнять основные опреации с электронными ключами:

- Искать электронный ключ, удовлетворяющий заданным условиям поиска
- Инициализировать память ключа
- Читать данные из памяти ключа
- Записывать данные в память ключа
- Устанавливать запреты на чтение/запись памяти ключа
- Преобразовывать информацию при помощи аппаратных алгоритмов ключа

Перечень функций Guardant API общего назначения:

Название функции	Код доступа	Краткое описание
GrdStartup	не нужен	Инициализировать Guardant API
GrdCleanup	не нужен	Деинициализировать Guardant API
GrdDllMain	не нужен	Инициализировать сетевые функции для использования в DLL
GrdGetLastError	не нужен	Получить информацию о последней ошибке из защищенного контейнера
GrdFormatMessage	не нужен	Преобразовать код ошибки в текстовое сообщение
GrdGetInfo	не нужен	Получить информацию из защищенного контейнера
GrdIsValidHandle	не нужен	Проверить валидность хэндла защищенного контейнера
GrdCreateHandle	не нужен	Создать хэнлд защищенного контейнера
GrdCloseHandle	не нужен	Закрыть хэндл защищенного контейнера
GrdSetAccessCodes	Public и Read обязательно, остальные по необходимости	Поместить коды доступа в защищенный контейнер
GrdSetWorkMode	не нужен	Установить режимы работы Guardant API
GrdSetFindMode	не нужен	Установить режимы поиска электронных ключей
GrdFind	Private Read	Найти электронный ключ, удовлетворяющий критериям поиска
GrdLogin	Private Read	Зарегистрироваться на электронном ключе
GrdLogout	Private Read	Снять регистрацию с электронного ключа
GrdLock	Private Read	Заблокировать ключ на время выполнения операций, требующих монольного режима работы с ключом
GrdUnlock	Private Read	Разблокировать электронный ключ
GrdCheck	Private Read	Проверить наличие электронного ключа
GrdDecGP	Private Read+ PrivateWrite	Декрементировать счетчик GP
GrdRead	Private Read	Прочитать данные из памяти электронного ключа
GrdWrite	PrivateWrite	Записать данные в память электронного ключа
GrdSeek	не нужен	Переместить указатель на текущий адрес памяти электронного ключа
GrdInit	Private master	Инициализировать память электронного ключа
GrdProtect	Private master	Установить аппаратные запреты чтения/записи и записать количество аппаратных алгоритмов и защищенных ячеек
GrdTransform	Private Read	Преобразовать данные аппаратно- или программно-реализованным алгоритмом
GrdCrypt	Private Read	Закодировать/раскодировать данные аппаратно или программно-реализованным алгоритмом
GrdHash	Private Read	Вычислить значение хэш-функции программно или аппаратно-реализованным алгоритмом

Название функции	Код доступа	Краткое описание
GrdCodeInit	PrivateRead	Инициализировать пароль быстрого взаимообратного преобразования
GrdEnCode	не нужен	Закодировать данные быстрым взаимообратным преобразованием
GrdDeCode	не нужен	Раскодировать данные быстрым взаимообратным преобразованием
GrdCRC	не нужен	Вычислить CRC

Функции Guardant API для работы с защищенными ячейками

Для работы с защищенными ячейками существуют специальные функции Guardant API:

Название функции	Код доступа	Краткое описание
GrdPI_Activate	Private Read и пароль на данную операцию, если он задан в ключе	Активировать аппаратный алгоритм / защищенную ячейку
GrdPI_Deactivate		Деактивировать аппаратный алгоритм или защищенную ячейку
GrdPI_Read		Прочитать данные из защищенной ячейки
GrdPI_Update		Обновить данные в защищенной ячейке

Функции Guardant API для работы с Trusted Remote Update

Технология Trusted Remote Update может быть реализована не только при помощи утилит, входящих в Комплект разработчика. При желании разработчики могут встраивать поддержку этой технологии непосредственно в свои приложения, используя набор предназначенных для этой цели функций:

Название функции	Код доступа	Краткое описание
GrdTRU_SetKey	Private master	Инициализировать память электронного ключа и записать секретный ключ TRU
GrdTRU_GenerateQuestion	Private Read	Сгенерировать число-вопрос
GrdTRU_DecryptQuestion	Private Read	Расшифровать и проверить подлинность числа-вопроса
GrdTRU_SetAnswerProperties	не нужен	Установить параметры числа-ответа
GrdTRU_EncryptAnswer	Private Read	Сгенерировать и зашифровать число-ответ
GrdTRU_ApplyAnswer	Private Read	Записать число-ответ в электронный ключ

Функции Guardant API для управления драйверами

Для того чтобы инсталлировать, деинсталлировать и настраивать драйверы Guardant из собственных приложений можно использовать функции управления драйверами, которые реализованы в виде динамически загружаемой библиотеки.

Название функции	Краткое описание
GrdDrvInstall	Инсталлировать драйверы Guardant
GrdDrvUnInstall	Деинсталлировать драйверы Guardant
GrdDrvIsInstalled	Проверить наличие драйверов Guardant в системе
GrdDrvGetVersion	Получить версию драйверов Guardant, установленных в системе
GrdDrvSetPortUseState	Установить режимы использования порта
GrdDrvGetPortUseState	Получить текущие установленные режимы использования порта
GrdDrvSetPrnPortTimeOut	Установить значение таймаута для LPT порта
GrdDrvGetPrnPortTimeOut	Получить текущее установленное значение таймаута для LPT порта
GrdDrvSetPortProperties	Установить параметры работы с LPT портом
GrdDrvGetPortProperties	Получить текущие установленные параметры работы с LPT портом
GrdDrvSetPortPropertiesAuto	Установить параметры работы с LPT портом по умолчанию
GrdDrvGetSystemInfo	Получить системную информацию о конфигурации LPT и USB-портов

Местонахождение файлов API

Категории файлов	Каталог с API
Заголовочные файлы	Guardant\Guardant 5\%PublicCode%\Include
Библиотеки и объектные модули	Guardant\Guardant 5\%PublicCode%\Lib
Примеры работы с Guardant API	Guardant\Guardant 5\%PublicCode%\Samples\
Функции совместимости с Guardant API 4.X	Samples\x86\Win32\Compatibility nXk API
Функции работы с Trusted Remote Update	Samples\x86\Win32\Dongle Trusted Remote Update
Функции Guardant API общего назначения	Samples\x86\Win32\General Guardant API
Функции управления драйверами Guardant	Guardant 5\Samples\x86\Win32\Driver Installation API

Коды ошибок Guardant API

Функции Guardant API, предназначенные для работы с электронными ключами, могут возвращать следующие коды ошибок:

Название ошибки	Код	Краткое описание
GrdE_Ok	0	Операция выполнена успешно
GrdE_DongleNotFound	1	Не найден ключ, отвечающий заданным условиям поиска
GrdE_AddressTooBig	3	Указанный адрес слишком велик
GrdE_GPis0	5	Счетчик запусков GP исчерпан (значение равно нулю)
GrdE_InvalidCommand	6	Неверная команда обращения к ключу
GrdE_VerifyError	8	Ошибка верификации при записи в память ключа
GrdE_NetProtocolNotFound	9	Сетевой протокол не найден
GrdE_NetResourceExhaust	10	Сетевой ресурс ключа Guardant Net исчерпан
GrdE_NetConnectionLost	11	Потеряно соединение с сервером Guardant Net
GrdE_NetDongleNotFound	12	Сервер Guardant Net не найден
GrdE_NetServerMemory	13	Ошибка распределения памяти сервера Guardant Net
GrdE_DPMI	14	Ошибка DPMI
GrdE_Internal	15	Внутренняя ошибка сервера Guardant Net
GrdE_NetServerReloaded	16	Сервер Guardant Net был перезагружен
GrdE_VersionTooOld	17	Данная команда не поддерживается данной версией ключа (ключ старой версии)
GrdE_BadDriver	18	Необходим драйвер Windows NT
GrdE_NetProtocol	19	Ошибка сетевого протокола
GrdE_NetPacket	20	Получен сетевой пакет недопустимого формата
GrdE_NeedLogin	21	Необходима регистрация на сервере Guardant Net
GrdE_NeedLogout	22	Необходимо снять регистрацию на сервере Guardant
GrdE_DongleLocked	23	Ключ Guardant Net занят другим приложением
GrdE_DriverBusy	24	Драйвер не может захватить порт
GrdE_CRCError	30	Ошибка CRC при обращении к ключу
GrdE_CRCErrorRead	31	Ошибка CRC при чтении данных из ключа
GrdE_CRCErrorWrite	32	Ошибка CRC при записи данных в ключ
GrdE_Overbound	33	Выход за границу памяти ключа
GrdE_AlgoNotFound	34	Аппаратный алгоритм с таким номером в ключе не найден
GrdE_CRCErrorFunc	35	Ошибка CRC аппаратного алгоритма
GrdE_AllDonglesFound	36	Все ключи перебраны
GrdE_ProtocolNotSup	37	Слишком старая версия Guardant API
GrdE_InvalidCnvType	38	Задан несуществующий метод взаимообратного преобразования
GrdE_UnknownError	39	Неизвестная ошибка при работе с алгоритмом/ячейкой, операция могла не завершиться
GrdE_AccessDenied	40	Неверный пароль доступа к защищенной ячейке

Название ошибки	Код	Краткое описание
GrdE_StatusUnchangeable	41	Статус защищенной ячейки изменить нельзя
GrdE_NoService	42	Для алгоритма/ячейки сервис не предусмотрен
GrdE_InactiveItem	43	Алгоритм/ячейка находятся в состоянии Inactive, команда не выполнена
GrdE_DongleServerTooOld	44	Попытка выполнить операцию, которую не поддерживает текущая версия сервера Guardant Net
GrdE_DongleBusy	45	В данный момент ключ не может выполнять никаких операций
GrdE_InvalidArg	46	Задано недопустимое значение одного из аргументов функции
GrdE_MemoryAllocation	47	Ошибка распределения памяти
GrdE_InvalidHandle	48	Недопустимый хендл
GrdE_ContainerInUse	49	Этот защищенный контейнер уже используется
GrdE_Reserved50	50	Зарезервировано
GrdE_Reserved51	51	Зарезервировано
GrdE_Reserved52	52	Зарезервировано
GrdE_SystemDataCorrupted	53	Нарушена целостность системных данных
GrdE_NoQuestion	54	Вопрос для удаленного обновления не был сгенерирован
GrdE_InvalidData	55	Недопустимый формат данных для удаленного обновления
GrdE_QuestionOK	56	Вопрос для удаленного обновления уже сгенерирован
GrdE_UpdateNotComplete	57	Процедура записи при удаленном обновлении не завершена
GrdE_InvalidHash	58	Неверное значение хеша данных удаленного обновления
GrdE_GenInternal	59	Внутренняя ошибка
GrdE_AlreadyInitialized	60	Данная копия Guardant API уже инициализирована
GrdE_LastError	61	Неизвестная ошибка

Специфика структуры программ, использующих Guardant API

Использование Guardant API требует соблюдения определенных условий, касающихся структуры программ. Это связано с тем, что для работы Guardant API требуется выполнение обязательных процедур, которые инициализируют API перед его использованием и деинициализируют после. Все функции, работающие непосредственно с электронным ключом, требуют, чтобы была выполнена регистрация на конкретном локальном или сетевом ключе.

Общая структура программы имеет вид:

Блок инициализации	**GrdStartup( )**	Инициализировать данную копию Guardant API
	GrdCreateHandle( )	Создать хэндл. Хэндлов может быть создано несколько, в том числе и для одного и того же ключа.
	GrdSetAccessCodes( )	Установить коды доступа в защищенный контейнер. Устанавливать следует только те коды, которые будут реально использоваться. Неиспользуемые коды можно заменить случайными числами.
	GrdSetFindMode( )	Установить режимы и критерии поиска ключа
	GrdFind( )	Выполнить поиск электронного ключа по заданным критериям. Следует учитывать, что критериям может соответствовать не один ключ, поэтому следует выполнять поиск до тех пор, пока не будут обнаружены все ключи, соответствующие критериям.
	GrdLogin( )	Из всех обнаруженных ключей следует выбрать один и выполнить процедуру регистрации. Без регистрации остальные функции не смогут обращаться к ключу.
Основной блок	Основная работа с электронным ключом. В этом блоке можно вызывать функции работы с электронным ключом: **GrdLock, GrdUnlock, GrdCheck, GrdDecGP, GrdRead, GrdWrite, GrdSeek, GrdInit, GrdProtect, GrdTransform, GrdCrypt, GrdHash, GrdPI_Activate, GrdPI_Deactivate, GrdPI_Read, GrdPI_Update**	
Блок деинициализации	**GrdLogout( )**	Снять регистрацию с текущего ключа.
	GrdCloseHandle( )	Закрыть хэндл
	GrdCleanup( )	Деинициализировать данную копиию Guardant API

Использование Guardant API в DLL

При использовании Guardant API в динамических библиотеках к блоку инициализации в обязательном порядке до вызова **GrdStartup( )** добавляется вызов функции **GrdDllMain( )**.

Подробное описание функций Guardant API

См. в справочном файле GrdAPI.chm, который находятся в директории:

%WinDir%\Program Files\Guardant\Guardant 5xx\%Public Code%\Doc

Глава 6
Автоматическая защита

Важная информация

1. Автоматическая защита Guardant, начиная с версии 5.x, поддерживает только ключи Guardant Stealth III / Net III и Guardant Stealth II / Net II.

2. Обратите особое внимание на принцип работы автоматической защиты Guardant, особенности совместного использования различных опций защиты и ограничения, связанные с установкой некоторых опций.

Возможности автоматической защиты

Автоматическая защита Guardant обладает широким набором возможностей. С ее помощью можно:

- Защищать любые готовые Win32-приложения, имеющие стандартный формат EXE-файла Windows PE (кроме .NET сборок)
- Защищать приложения от компьютерных вирусов, в том числе от вирусов-невидимок (stealth-вирусов)
- Использовать ID ключа для однозначной идентификации конкретного ключа
- Защищать несколько программных продуктов, пользуясь номером программы для идентификации конкретного продукта
- «Привязывать» защищаемые приложения к серийному номеру и версии ключа
- Используя поле битовой маски, производить защиту комплекса приложений с последующим разрешением/запрещением запуска определенных приложений из защищенного комплекса
- Производить опрос наличия ключа не только во время запуска защищенного приложения, но и в течение всего сеанса его работы через заданные вами промежутки времени
- Ограничивать количество запусков защищенного приложения, с возможностью последующего изменения счетчика запусков или снятия этого ограничения (в том числе дистанционно)
- Ограничивать время работы защищенного приложения с возможностью коррекции времени или снятия этого ограничения (в том числе дистанционно)
- Ограничивать число копий защищенного сетевого приложения, запущенных в локальной сети одновременно
- Создавать защиту, не зависящую от типа ключа

Предупреждение

Автоматическая защита, несмотря на свои богатые возможности, не может гарантировать надежной защищенности программного продукта. Настоятельно рекомендуем усилить защиту при помощи Guardant API!

Принцип работы автоматической защиты

Применяемый в системе Guardant принцип работы автоматической защиты отличается от используемых в других подобных системах. Основа защиты Guardant – вакцина, выполненная в виде универсального внешнего модуля. Все функции защиты поддерживаются этим модулем. Это позволяет полностью унифицировать процесс защиты, отказаться от различных неудобных для конечного пользователя и громоздких методов защиты нестандартного ПО.

Автоматическая защита Guardant работает так. В тело защищаемого приложения вписывается небольшой исполняемый модуль (внутренняя вакцина). В момент запуска приложения он загружает из отдельного файла внешнюю вакцину. И уже эта внешняя вакцина производит необходимые проверки и преобразования кода защищенного приложения и запускает его.

Подобный принцип работы автоматической защиты имеет большое преимущество, как для разработчика, так и для конечного пользователя защищенного продукта. Предположим, вышла новая версия операционной системы, для которой потребовалась доработка защиты. Для этого потребуется передать клиенту всего один небольшой файл – новую версию внешней вакцины. И нормальная работа защищенных приложений в новой версии ОС будет немедленно налажена.

Кроме того, использование внешнего модуля защиты усиливает стойкость к изучению логики ее работы при помощи отладчиков.

Файл вацкцины Guardant имеет имя **GrdVkc32.dll** и входит в комплект автоматической защиты.

Важная информация

В момент запуска защищенного приложения вакцина **GrdVkc32.dll** должна находиться там, где ее может найти функция **LoadLibrary** (это может быть текущий каталог, системные каталоги Windows, каталог, в котором находится само защищенное приложение, один из каталогов списка PATH).

Утилита автоматической защиты

Процесс защиты производится утилитой NwKey32.exe из состава ПО Guardant или мастером AutoProtectWizard.exe.

Утилиту автозащиты можно вызывать с выбранными опциями вручную или из BAT-файла.

Утилита автоматической защиты внедряет в код приложения набор команд (программный модуль), с помощью которых происходит загрузка внешней вакцины. Также утилита выполняет необходимые преобразования защищаемого приложения в соответствии с выбранным режимом защиты.

Файлы, необходимые для защиты

Для работы защищенного приложения необходимо наличие следующих файлов в одной директории:

Файл	Описание
NwKey32.exe	Утилита автоматической защиты
GrdVkc32.dll	Внешняя вакцина
NwKey32.msg	Файл с сообщениями об ошибках
NvCodes.dat	Служебный файл с информацией о кодах доступак ключу

Важная информация

1. Если файл NvCodes.dat отсутствует, приложения будут защищаться с демо-кодами.

2. Файл NvCodes.dat необходим только для утилиты автоматической защиты и программирования ключей. Сами защищенные приложения не нуждаются в этом файле. Ни в коем случае не передавайте его своим клиентам!

Файлы, необходимые для работы защищенного приложения

Для работы защищенного приложения необходимо наличие следующих файлов в одной директории:

Файл	Описание
GrdVkc32.dll	Внешняя вакцина
GrdNetAgent.exe	Внешний модуль, обеспечивающий сетевое взаимодействие. Только для приложений, защищенных сетевыми ключами.

Порядок защиты

Перед началом защиты подсоедините к компьютеру электронный ключ нужного типа.

Формат вызова утилиты автоматической защиты:

NwKey32.exe [опции] [путь]список_файлов

или

NwKey32.exe [опции] @[путь]filename.fil

Укажите в командной строке необходимые для защиты параметры и нажмите **Enter**. Утилита приступит к защите, выдавая по ходу работы необходимые сообщения:

- Значения полей памяти ключа, участвующих в защите. Это могут быть значения из реально подсоединенного ключа, либо значения, указанные в опциях
- Список указанных опций защиты (режимы привязки к ключу, кодирования приложений).
- Список сообщений, которые может выдавать внешняя вакцина при работе защищенных приложений
- Имя каталога, в который будут помещены защищенные приложения (выходной путь).
- Сообщения об ошибках, возникших в процессе защиты

Утилита завершит защиту текущего файла и закончит работу.

Процесс защиты можно прервать в любой момент, нажав **Esc.**

Краткое описание опций защиты

Опции автоматической защиты сгруппированы в таблицы по типам. В каждой таблице наряду с названием опции и ее кратким описанием указано, с какими ключами семейства Guardant она используется.

Опции установки типа электронного ключа

Опция	Описание	Тип ключа
/GS2[=N : L]	Установить привязку к ключу Guardant Stealth II	Guardant Stealth II /Net II
/GN2[=N : L]	Установить привязку к ключу Guardant Net II	Guardant Net II
/GS3[=N : L]	Установить привязку к ключу Guardant Stealth III	Guardant Stealth III /Net III
/GN3[=N : L]	Установить привязку к ключу Guardant Net III	Guardant Net III

Опции привязки к электронному ключу

Опция	Описание	Тип ключа
/UI[=[0x]...]	Проверять уникальность ID электронного ключа (используется заданное значение, либо значение из поля ID ключа)	Все
/US[=[0x]...]	Проверять уникальность серийного номера электронного ключа (используется заданное значение, либо значение из поля серийного номера ключа)	Все
/UV[=[0x]...]	Проверять версию (используется заданное значение, либо значение из поля версии ключа)	Все
/UM[=[0x]...]	Проверять маску (используется заданное значение, либо значение из поля маски ключа)	Все
/UN[=[0x]...]	Проверять номер программы (используется заданное значение, либо значение из поля номера программы ключа)	Все
/NOA	Не использовать аппаратные алгоритмы	Все
/DC	Ограничить количество запусков приложения (используется поле GP ключа; если в нем 65535 – не ограничивать, если 0 – не запускаться).	Все, кроме сетевых ключей
/DT	Ограничить время использования приложения (используется поле GP ключа; если в нем 65535 – не ограничивать, если 0 – не запускаться)	Все, кроме сетевых ключей
/T=xx	Проверять наличие электронного ключа периодически, через заданные интервалы времени	Все
/EXIT_DELAY	Отложить завершение приложения при отсутствии ключа	Все
/RC[=xx]	Если ключ не найден, проверять его наличие и выводить сообщение об отсутствии ключа заданное число раз	Все

Опции кодирования защищаемого приложения

Опция	Описание	Тип ключа
/CEN	Не кодировать загружаемую часть приложения	Все
/IDEN	Отключить шифрование инициализированных данных	Все

Дополнительные опции защиты приложения

Опция	Описание	Тип ключа
/V	Проверять целостность приложения	Все
/NOS	Не «очищать» приложение	Все
/ATR[=N]	Задать число таблиц вопросов-ответов к алгоритму	Все
/PACK	Упаковать секции исполняемого файла	Все
/CPA	Контролировать атрибуты страниц	Все
/IMPORT_HOOK=%:L	Защищать импортируемые функции	Все
/IMPLICIT_LINKING_SUPPORT	Поддержка неявного связывания DLL	Все

Сетевые опции

Опция	Описание	Тип глюча
/LOGIN_MODE= [H, S]	Выбрать режим лицензирования: по копиям приложения (H) или по рабочим станциям (S)	Сетевые ключи
/CLS	Проверять сетевой ключ на локальном компьютере	Сетевые ключи
/MN=xx	Использовать систему управления лицензиями	Сетевыеключи

Сервисные опции

Опция	Описание	Тип глюча
/MSG= [путь]*.msg	Брать сообщения вакцины из файла *.MSG (имя_утилиты.MSG – по умолчанию).	Все
/OUT= D:\PATH	Задать путь, по которому будут скопированы защищенные файлы (по умолчанию это каталог с исходными файлами)	Все
/Q	Запретить вывод сообщений утилиты защиты на экран.	Все
/SND [=file.ext]	Разрешить звуковые сообщения об ошибках. Сообщения берутся из файла FILE.EXT, либо из файла по умолчанию NWKEY32.SND.	Все

Коды ошибок

Утилита автоматической защиты может возвращать следующие коды ошибок:

Код ошибки	Описание
0	Процесс успешно завершен
1	Процесс прерван пользователем
2	Ошибка распределения памяти
3	Неверная опция
4	Файл *.MSG не найден
5	Файл имеет необрабатываемый формат
6	Ключ должен быть Windows-совместимым
7	Электронный ключ не найден
8	Файл уже защищен
9	Файл уже существует
10	Файл невозможно переименовать
11	Ошибка обмена с электронным ключом
32	Ошибка открытия файла
33	Невозможно изменить размер файла
34	Ошибка чтения из файла
35	Ошибка записи в файл
36	Переполнение. Выравнивание сегмента слишком велико
37	Невозможно защитить файл базы данных
38	Невозможно защить DLL файл
39	Слишком много сегментов в защищаемом файле (более 32)
40	Найден распределенный сегмент

Код ошибки	Описание
41	Невозможно изменить режим шифрования кода
42	Неподдерживаемый формат
43	Некорректный электронный ключ
44	Исчерпан ресурс
45	Для защиты используйте NWKEY.EXE
128	Некорректный файл параметров защиты
129	Счетчик инсталляций истек

Ограничения

Важная информация

1. Автоматическая защита Guardant, начиная с версии 5, поддерживает только ключи Guardant Stealth III / Net III и Guardant Stealth II / Net II.

2. Автозащита поддерживает только взаимообратные алгоритмы типа GSII64. Однонаправленные алгоритмы Stealth III и Stealth II использовать для защиты нельзя.

- Утилита автозащиты позволяет защищать все Win32 EXE-файлы, кроме .NET-сборок
- Не поддерживаются самораспаковывающиеся архивы ZIP, RAR и т. д.
- Не поддерживаются программы-мастера установки приложений, созданные в специализированных средах разработки: Wise Installer, Install Shield и других.
- Не гарантируется корректная защита или последующая работа приложения, которое перед защитой было упаковано специальным упаковщиком EXE-файлов: UPX, ASPACK и др.
- Не гарантируется корректная защита EXE- файлов, код которых был предварительно защищен от модификации или анализа

Опции автоматической защиты

Опции установки типа электронного ключа

Опции этой группы позволяют задать конкретный тип электронного ключа из семейства Guardant, к которому будет «привязано» защищенное приложение.

Установить привязку к типу ключа: /GS2[=N:L], /GN2[=N:L], /GS3[=N:L], /GN3[=N:L]

Описание:

Указание типа ключей, к которым будет привязано приложение.

N – номер аппаратного алгоритма ключа, используемого для защиты, **L** – длина вопроса алгоритму. Задание **N** и **L** является необязательным, если в ключи прописана стандартная маска соответствующего типа через редактор памяти ключей, поскольку по-умолчанию утилита автозащиты привязывает программу к наиболее подходящему алгоритму из стандартной маски. Значение длины вопроса алгоритма GSII64 должно быть кратно 8 и быть не более 256.

Если в ключ записана не маска по умолчанию, то необходимо указывать привязку к определенному номеру алгоритма. Нумерация алгоритмов начинается с нуля, при этом следует задавать правильную длину вопроса, которая может отличаться для каждого конкретного алгоритма.

Можно задавать одновременно несколько опций из этой группы – в любом сочетании. При этом защищенное приложение будет запущено, если хотя бы один из заданных типов электронных ключей Guardant будет подсоединен к компьютеру.

Поиск будет прекращен на любом этапе в случае, если будет найден ключ любого из заданных типов, удовлетворяющий условиям поиска (они задаются опциями привязки к ключу, описанными ниже). Если защищенное приложение будет опрашивать ключ в ходе дальнейшей работы, все обращения будут производиться только к тому типу ключа, который был найден при первом обращении. Это резко уменьшит время обмена с ключом, например, при периодической проверке его по таймеру.

Если при защите использовались опции этой группы, запуск защищенного приложения будет возможен только с электронным ключом: приложение при защите настраивается на код доступа к ключу заданного типа, подсоединенному к компьютеру на момент защиты.

В ключах, которые передаются клиентам вместе с защищенным приложением, должен быть создан алгоритм с таким же номером и длиной запроса, которые были указаны при защите.

Важная информация

Если опции этой группы не использовались, приложение не будет привязано к электронному ключу (т. е. оно будет запускаться и в случае, когда ни один из электронных ключей не подсоединен к компьютеру). Однако оно будет защищено **от отладчиков**. Вы можете использовать эту возможность, например, для защиты приложений, которые без электронного ключа работают в демо-режиме.

Пример:

```
nwkey32.exe /GS3 /GS2 myprog.exe
```

Защищенное Win32-приложение MYPROG.EXE будет запускаться в случае, если к компьютеру подсоединен любой из электронных ключей Guardant Stealth III или Guardant Stealth II. В ключи прошита маска по умолчанию. Дополнительные условия поиска ключа не заданы.

Опции привязки к электронному ключу

Опции этой группы позволяют настроить защищаемое приложение на параметры электронного ключа, с которым оно должно будет работать (задать дополнительные условия поиска ключа), а также задать особенности поведения защиты при работе этого приложения.

Опции из этой группы можно использовать совместно в любой комбинации, а также с опциями из других групп, за исключением случаев, особо оговоренных ниже.

Важная информация

Если ни одна из опций установки типа ключа не задана, то все опции из рассматриваемой группы будут недоступны (их использование бессмысленно, т. к. приложение вообще не будет привязано к какому-либо электронному ключу).

Проверять ID ключа: /UI[=[0x]...]

Тип электронного ключа:

```
Все
```

Описание:

Эта опция служит для «привязки» приложения к уникальному параметру ключа Guardant – его идентификационному номеру (ID). Защищенное с этой опцией приложение будет запускаться лишь в том случае, если к компьютеру подсоединен именно тот электронный ключ, с использованием которого была произведена защита программы (или ID которого был указан в опции).

Если опция указана в виде **/UI**, приложение будет привязано к ID того электронного ключа, который был подсоединен к компьютеру на момент защиты. Чтобы привязать приложение к ID какого-либо другого из ключей, присоединенных к компьютеру, укажите его значение в опции после знака «**=**». Для того чтобы написать ID в шестнадцатиричной системе счисления, укажите перед ним префикс «**0x**».

ID современных ключей Guardant можно узнать, запустив утилиту диагностики GrdDiag.exe, а также через команду **Ключ | Информация о ключе** утилиты GrdUtil.exe.

Важная информация

ID ключа – уникальная величина. Нельзя ни изменить ID ключа, ни изготовить ключ с нужным ID. Поэтому в случае выхода ключа из строя придется менять не только ключ, но и само защищенное приложение.

Примеры:

```
Nwkey32.exe /GN2 /UI myprog.exe
Nwkey32.exe /GS3 /UI=912459016 myprog.exe
```

В первом примере защищенное Win32-приложение MYPROG.EXE будет запускаться только с тем из ключей Guardant Net II (опция **/GN2**), который был подсоединен к компьютеру на момент защиты приложения (опция **/UI**, используется ID подсоединенного ключа). Во втором примере MYPROG.EXE будет запущен только с тем из ключей Guardant Stealth III (опция **/GS3**), который имеет ID=912459016 в десятичной системе счисления (опция **/UI=**912459016).

Проверять серийный номер ключа: /US [=[0x]sssss]

Тип электронного ключа:

Все

Описание:

Для привязки приложения к конкретному ключу также служит опция **/US**. В этом случае защищенное приложение будет запускаться только тогда, когда подсоединен электронный ключ, имеющий в поле серийного номера то значение, какое было задано при защите приложения.

Для задания серийного номера используется число от 0 до 65535.

Этот режим защиты более «мягкий», чем предыдущий: он использует параметр ключа, который можно записать в ключ. Следовательно, если ключ выйдет из строя, легко можно изготовить ключ с таким же серийным номером и заменить неисправный.

Важная информация

При привязке к ключу Guardant и одновременно при задании опции в виде **/US**, привязка будет произведена к серийному номеру подсоединенного ключа Guardant. Это правило действует и на опции привязки ко всем остальным параметрам ключей: к версии, маске, номеру программы и т. п. (они будут рассмотрены ниже).

Примеры:

```
Nwkey32.exe /GS2 /US myprog.exe
Nwkey32.exe /GS3 /US=0xFFFE myprog.exe
Nwkey32.exe /GN3 /GS2 /US myprog.exe
```

В первом примере защищенное приложение MYPROG.EXE будет запускаться только с тем из ключей Guardant Stealth II (опция **/GS2**), который был подсоединен к компьютеру на момент защиты приложения (опция **/US**, используется серийный номер подсоединенного ключа). Во втором примере MYPROG.EXE будет запущен только с тем из ключей Guardant Stealth III, который имеет серийный номер FFFE в шестнадцатиричной системе счисления (опция **/US=0xFFFE**). В третьем примере MYPROG.EXE будет запущен только с тем из ключей Guardant Stealth Net III или Guardant Stealth II /Net II (опции **/GS** и **/GN3**), который имеет серийный номер, как у подсоединенного на момент защиты ключа Guardant Net III и Guardant Stealth II/Net II (опция **/US**).

Проверять версию защищенного приложения: /UV[=[0x]vvv]

Тип электронного ключа:

Все

Описание:

Опция **/UV** служит для привязки приложений к значению, записанному в поле версии ключа. Этот режим удобен в случае, если часто выходят новые версии программного обеспечения. Он позволяет легко решать проблему обновления ПО.

Предположим, изначально программный продукт имеет версию 1.0. Тогда при его защите нужно задать опцию **/UV=10**, и записать это же значение в поле версии ключа, с которым будет работать версия продукта.

В дальнейшем, когда появится новая версия программного продукта (предположим, это версия 1.1), и будет необходимо обновить продукт старым клиентам, не нужно будет давать им новый электронный ключ. Вместо этого просто защитите новую версию с указанием опции **/UV=11**. Скорректировать поле версии в своем электронном ключе клиент сможет сам, получив от разработчика нужные данные и воспользовавшись утилитой дистанционного программирования ключей. После этого пользователь сможет работать как с новой, так и со старой версией программного продукта.

Дело в том, что защищенное с этой опцией приложение не только проверяет наличие ключа, но и анализирует содержимое его поля версии. Приложение запускается только в том случае, если версия в

поле ключа больше или равна версии, заданной при защите приложения. Значит, если пользователь где-либо достанет защищенную копию новой версии продукта, он не сможет с ней работать до тех пор, пока не скорректирует поле версии в своем ключе. А сделать этого сам он не сможет, т. к. не знает, какая информация для этого нужна.

Для задания версии в параметре **/UV** используется число в диапазоне от 0 до 255.

Пример:

```
Nwkey32.exe /GS2 /UV myprog.exe
Nwkey32.exe /GS3 /GS2 /UV=11 myprog.exe
```

В первом случае производится привязка приложения MYPROG.EXE к номеру версии, записанному в поле подсоединенного ключа (предположим, что в поле ключа было записано 10, т. е. версия 1.0). Защищенное приложение будет запущено в случае, если подсоединен ключ Guardant Stealth II со значением 10 или больше в его поле версии. Во втором случае происходит защита обновленного приложения MYPROG.EXE версии 1.1 (опция **/UV=11**). В этом случае приложение станет работоспособным, после того как конечный пользователь запишет значение 11 в поле версии своего ключа Guardant Stealth II или получит от вас новую улучшенную модель ключа Guardant Stealth III, с записанным значением 11 в поле версии. При этом он сможет работать как со старой (но без ключа Stealth III), так и с новой версией MYPROG.EXE.

Проверять маску: /UM[=[0x]mmmmm]

Тип электронного ключа:

Любой

Описание:

Если продукт, состоит из нескольких самостоятельных программных модулей, для его защиты можно воспользоваться опцией **/UM**. Эта опция позволяет выборочно разрешать или запрещать пользователю запуск определенных модулей из состава программного комплекса.

Например, электронный переводчик состоит из трех программ: MYPROG1.EXE – демо-версия с ограниченными возможностями, MYPROG2.EXE – англо-русский переводчик и MYPROG3.EXE – французско-русский переводчик.

Пусть сначала распространяется демо-версию, а затем, по мере надобности, клиенты докупают интересующие их программы-переводчики. Использование опции **/UM** облегчит задачу распространения такого продукта.

Защищенные в этом режиме приложения используют поле маски ключа в качестве набора семафоров, каждый из которых разрешает или запрещает запуск программы с номером, присвоенным ей при защите. При запуске программа проверяет значение соответствующего бита в поле маски ключа и, если он равен 1, приложение запускается.

При защите необходимо указать в опции **/UM** номер для каждого защищаемого приложения (его можно указывать в десятичной или – с использованием префикса «0x» – в шестнадцатиричной системе счисления). Например, MYPROG1.EXE номер 1, MYPROG2.EXE – номер 2 и MYPROG3.EXE – номер 4, т. к. числу 1 соответствует бит #0, числу 2 – бит #1, а числу 4 – бит #2.

В поле **Битовая маска** электронного ключа записывается число 1.

В этом случае можно включать в комплект поставки все три приложения. Однако клиент сможет запустить лишь MYPROG1.EXE, т. к. в поле маски его электронного ключа записано значение, соответствующее номеру только этого приложения (т. е. 1).

Если в дальнейшем клиент захочет использовать и MYPROG2.EXE, он при помощи переданной разработчиком информации утилитой дистанционного программирования скорректирует значение маски своего ключа, записав в нее число 3. Теперь он сможет запускать не только MYPROG1.EXE, но и MYPROG2.EXE, т. к. числу 3, теперь записанному в ключе, соответствуют биты #0 и #1.

Разрядность поля маски ключа – 16 бит. При защите желательно выбирать для программ такие номера, которые соответствуют единственному установленному биту, т. е. 1, 2, 4, 8, 16, 32, 64 и т. д. Присвоение других номеров может привести к возникновению некорректной ситуации при проверке маски.

Пример:

```
Nwkey32.exe /GS3 /UM=1 myprog1.exe
Nwkey32.exe /GS3 /UM=2 myprog2.exe
Nwkey32.exe /GS3 /UM=4 myprog3.exe
```

Показаны вызовы утилиты защиты приложений ключом Guardant Stealth III/Net III для примера, описанного выше по тексту.

Проверять номер программы: /UN[=[0x]nnn]

Тип электронного ключа:

Все

Описание:

Эта опция будет полезна в случае, если есть несколько различных программных продуктов (например, электронная бухгалтерия и программа складского учета), и все они защищены с помощью ключей Guardant. В этом случае можно привязывать их к одним и тем же параметрам ключа, сделав в то же время невозможным работу обоих продуктов с одним и тем же ключом.

Необходимо присвоить каждому продукту свой номер (пусть, например, электронная бухгалтерия будет иметь номер 0, а программа складского учета – номер 1) и защищать их с опцией **/UN**.

Тогда в поле номера программы ключей, предназначенных для работы с электронной бухгалтерией, нужно записать значение 0, а для программы складского учета – значение 1. Все остальные параметры защиты можно сделать одинаковыми. Защищенное в таком режиме приложение будет проверять значение поля номера программы, поэтому невозможно будет запустить электронную бухгалтерию с ключом, предназначенным для работы с программой складского учета, и наоборот.

Допустимые значения номера программы – от 0 до 255.

Все продаваемые ключи имеют в поле номера программы значение 0. Поэтому можно воспользоваться этой опцией, даже если в настоящий момент существует только один программный продукт. Ведь в будущем количество продуктов может увеличиться, – и тогда не нужно будет беспокоиться о том, как не допустить нелегального использования новой продукции старыми клиентами.

Пример:

```
nwkey32.exe /GS3 /UN=0 mybuh.exe
nwkey32.exe /GS3 /UN=1 mysklad.exe
```

Показаны вызовы утилиты защиты приложений ключом Guardant Stealth III для примера, описанного выше по тексту.

Не использовать аппаратные алгоритмы: /NOA

Тип электронного ключа:

Все

Описание:

По умолчанию приложения, защищенные автозащитой с ключом Guardant, используют аппаратный алгоритм типа GSII64, если он есть в памяти электронного ключа. Это значительно повышает уровень стойкости автоматической защиты, т. к. вводит в нее элемент преобразования данных.

Если же в памяти ключа нет алгоритмов GSII64, например, когда она используется для хранения других данных, автоматическая защита будет работать корректно лишь в том случае, когда она не использует аппаратный алгоритм в процессе защиты.

Чтобы отказаться от использования аппаратного алгоритма для автоматической защиты, нужно запустить утилиту NWKEY32.exe с опцией **/NOA**.

Пример:

```
nwkey32.exe /GS2 /NOA myprog.exe
```

Защищенное таким образом приложение не будет использовать аппаратные алгоритмы GSII64 при проверке ключа Guardant Stealth II.

Ограничить число запусков приложения: /DC

Тип электронного ключа:

Локальные ключи

Описание:

Воспользовавшись опцией **/DC** (Decrement of Counter), можно легко превратить рабочее приложение в демонстрационное, ограничив срок его «жизни».

В поле счетчика запусков электронного ключа (GP) занесите число запусков этого приложения.

Защищенное приложение при каждом запуске уменьшает на 1 значение счетчика, записанное в поле GP ключа. Когда оно станет равным 0, приложение перестанет запускаться.

Используя этот режим можно продавать свое программное обеспечение с ограничением ресурса его использования. Например, пользователь оплачивает первые сто сеансов работы с приложением; в этом случае приложение защищается с параметром **/DC**, а в поле

GP ключа записывается значение 100. Когда пользователь израсходует этот ресурс, разработчик, используя процедуру удаленного обновления, вписывает в него новое значение ресурса (например, еще 100 запусков). Таким образом, программное обеспечение как бы сдается в аренду.

Во-вторых, можно легко превратить демо-версию приложения в полноценную рабочую версию. В этом случае, после оплаты пользователем стоимости рабочей версии, при помощи удаленного обновления в поле GP ключа записывается значение 65535. Если поле GP содержит это значение, то защищенное приложение при запуске его не уменьшает, автоматически превращаясь в полноценную защищенную программу.

Таким образом, для того чтобы приложение имело ограниченный ресурс использования, его нужно защитить с параметром **/DC**, а поле GP ключа должно содержать значение от 1 до 65534.

Число-ответ является уникальной величиной и зависит от числа дополнительных запусков, запрошенных пользователем (а в последнем случае – от числа 65535).

Конечный пользователь всегда сможет определить, сколько раз он еще сможет запустить приложение, защищенное в этом режиме. Для этого ему нужно запустить утилиту диагностики ключа. Интересующая его информация будет отображаться в поле **Счетчик запусков**.

Важная информация

1. Для сетевого типа защиты эта опция недоступна.

2. Все защищенные приложения, привязанные к одному ключу в этом режиме, уменьшают один и тот же счетчик. Если, например, программный продукт состоит из двух защищенных приложений MYPROG1.EXE и MYPROG2.EXE, поле GP ключа содержит 5, и пользователь трижды запустил MYPROG1.EXE, то приложение MYPROG2.EXE он сможет запустить всего два раза.

Пример:

```
Nwkey32.exe /GS3 /GS2 /NOA /UV=10 /DC myprog.exe
```

Пример защиты приложения MYPROG.EXE версии 1.0 ключом Guardant Stealth III или Guardant Stealth II без использования аппаратного алгоритма (опция **/NOA**), с проверкой версии продукта (опция **/UV=10**) и с ограничением числа его запусков (опция **/DC**). MYPROG.EXE будет запущен, если подсоединен ключ Guardant Stealth II или Guardant Stealth III со значением в поле версии 10 или больше, и перестанет запускаться, когда значение счетчика GP ключа достигнет нуля.

Ограничить время использования приложения: /DT

Тип электронного ключа:

Локальные ключи

Описание:

Защита Guardant позволяет поставлять приложения с ограничением времени их работы.

Суть этого режима защиты в следующем. При защите приложения указывается время работы приложения (например, 5 часов 30 минут). Пользователь проработает с ним 5ч. 30м. «чистого» времени (ключ считает время только тогда, когда приложение находится в работе). По истечении этого срока оно перестанет работать. Однако при помощи удаленного обновления можно увеличить время использования приложения или получить право на его использование без ограничений по времени.

Для защиты в этом режиме служит опция **/DT** (Decrement of Time).

В поле GP ключа необходимо записать время использования защищенного приложения. Для записи времени в ключ предназначена утилита ***GrdUtil.exe***.

При работе защищенного приложения будет происходить периодическое (с постоянным периодом) уменьшение счетчика времени, записанное в поле GP ключа. Когда время использования приложения истечет, будет выдано соответствующее сообщение, приложение будет закрыто, а его повторный запуск станет невозможен.

Если же во время очередного обращения к ключу он не будет обнаружен, работа приложения будет блокирована до тех пор, пока пользователь не подсоединит ключ и не нажмет кнопку для повтора.

Важная информация

Для сетевого типа защиты эта опция недоступна.

Опции **/DT** и **/DC** (уменьшение счетчика запусков программы) используют одно и то же поле ключа (поле GP), одновременное их использование при защите программ недопустимо. Т. о., вызов утилиты защиты с опциями **/DC** и **/DT** является неверным.

Кроме того, являются несовместимыми опции **/DT** и **/T=xx** (проверка наличия ключа по таймеру, подробное описание см. ниже). Дело в том, что опция **/DT** включает в себя возможности опции **/T=xx**, являясь ее расширением. Поэтому использование обеих опций при защите приложений лишено смысла. Вызов утилиты защиты с опциями **/DT** и **/T=xx** одновременно является неверным.

Все защищенные приложения, привязанные к одному ключу в этом режиме, уменьшают один и тот же счетчик времени. Если, например, программный продукт состоит из двух защищенных приложений MYPROG1.EXE и MYPROG2.EXE, ключ запрограммирован на три часа и пользователь два часа проработал с MYPROG1.EXE, то с приложением MYPROG2.EXE он сможет работать только один час.

Пример:

```
nwkey32.exe /GS3 /DT myprog.exe
```

Пример защиты приложения MYPROG.EXE ключом Guardant Stealth III в режиме ограничения времени. С MYPROG.EXE можно будет работать до тех пор, пока значение счетчика GP ключа не станет равным нулю.

Проверять наличие электронного ключа периодически: /T=xx

Тип электронного ключа:

Все

Описание:

Можно проверять наличие электронного ключа не только при запуске, но и в течение всего сеанса работы защищенного приложения через заданные вами промежутки времени.

Такая возможность обеспечивается опцией **/T=xx**, где «xx» – значение промежутка между двумя проверками в минутах. Если эта опция не указана, то задается проверка через каждую минуту.

Можно задавать интервалы проверки от 1 до 60 минут (допустимы только целые числа).

Если при очередной проверке ключ не будет обнаружен, защищенное приложение выдаст на экран соответствующее сообщение, либо звуковой сигнал. Работа приложения будет блокирована до тех пор, пока пользователь не подсоединит ключ и не нажмет кнопку для повтора.

Важная информация

Опции **/T=xx** и **/DT** являются несовместимыми (подробное описание см. выше).

Пример:

```
Nswkey32.exe /GS3 /T=05 myprog.exe
Nswkey32.exe /GS3 /T=5 myprog.exe
```

Эквивалентные примеры защиты приложения MYPROG.EXE с ключом Guardant Stealth III. MYPROG.EXE будет опрашивать ключ в течение всего времени работы, с интервалами в пять минут.

Выводить сообщение об отсутствии ключа заданное число раз: /RC[=xx]

Тип электронного ключа:

Все

Описание:

Если в процессе работы защищенного приложения ключ не будет обнаружен, то можно проверять его наличие и выводить сообщение об отсутствии ключа заданное число раз. Для этого служит опция **/RC[=xx]**, где **xx** - число проверок ключа. Значение опции по умолчанию равно 50.

Применение этой опции дает возможность пользователю сохранить наработанные данные и корректно завершить работу с приложением, например, в случае выходе ключа из строя.

При отсутствии ключа на экран выводится сообщение «Guardant Dongle not Found or Printer is off!» с кнопками **[Retry]** и **[Cancel]**. После нажатия **[Retry]** программа продолжает работать ~ 10 – 30 сек, а затем проверяет ключ. Если ключ не найден, вновь выводится сообщение, если найден – приложение продолжает работать в штатном режиме.

Если после заданного числа проверок ключ не будет обнаружен, то выводится вышеописанное сообщение с кнопкой **[Ok]**, после нажатия которой приложение завершает работу.

Пример:

```
Nswkey32.exe /GS3 /RC=10 myprog.exe
```

При отсутствии ключа приложение myprog.exe будет проверять ключ Guardant Stealth III и сообщать об его отсутствии 10 раз.

Отложить завершение приложения при отсутствии ключа: /EXIT_DELAY[=xxx]

Тип электронного ключа:

Все

Описание:

Если в процессе работы защищенного приложения ключ не будет обнаружен, то, при использовании данной опции, приложение выведет на экран сообщение о принудительном завершении работы приложения через заданное время (значение по умолчанию – 120 секунд).

Это даст возможность пользователю сохранить наработанные данные и корректно завершить работу с приложением, например, в случае выходе ключа из строя.

Опции кодирования защищаемого приложения

Отменить кодирование приложения:
/CEN

Тип электронного ключа:

Все

Описание:

Данная опция отключает шифрование секций защищаемого приложения.

Пример:

```
nwkey32.exe /GS2 /T=10 /CEN myprog.exe
```

Пример защиты приложения MYPROG.EXE с ключом Guardant Stealth II. MYPROG.EXE ***не будет закодирован*** (опция **/CEN**). Проверка наличия ключа Guardant Stealth II будет производиться приложением в течение всего времени его работы с интервалами в 10 минут.

Отключить шифрование инициализированных данных:
/IDEN

Тип электронного ключа:

Все

Описание:

Рекомендуется ставить в случаях, когда загрузчику ОС необходимо использовать эти данные до точки входа в программу. Т. е., если защищенное приложение запускается некорректно, то можно выставить эту опцию. Следует заметить, что защищенность приложения при установке этой опции уменьшается. Т. о., по умолчанию, опция отключена, и инициализированные данные шифруются (за исключением ряда секций, которые всегда не шифруются).

Сетевые опции автозащиты

Выбрать режим лицензирования:
/LOGIN_MODE=[H, S]

Тип электронного ключа:

Сетевые ключи

Описание:

Опция позволяет выбрать режим лицензирования сетевого приложения. При указании параметра H (**/LOGIN_MODE=H**) лицензии будут распределяться по копиям приложения (хэндлам). Т. е. каждая запущенная копия приложения будет занимать одну лицензию.

При указании параметра H (**/LOGIN_MODE=S**) лицензии будут распределяться по рабочим станциям. Т. е. будет ограничиваться число рабочих станций, на которых одновременно запущены копии защищенного приложения.

Использовать систему управления лицензиями: /MN=xx

Тип электронного ключа:

Сетевые ключи

Описание:

Эта опция позволяет учитывать сетевой ресурс каждого модуля, входящего в многомодульное приложение.

Допустим, защищенный программный комплекс состоит из 4-х модулей:
MYPROG1.EXE – Бухгалтерия, MYPROG2.EXE – Зарплата, MYPROG3.EXE – Кадры, MYPROG4.EXE – Канцелярия.

При помощи данной опции можно контролировать использование любого модуля. Для этого необходимо последовательно защитить каждый из модулей с опцией **/MN=xx**, где xx – номер модуля (возможные значения – 0 – 127). Если будет использован номер, превышающий количество модулей в таблице лицензий, то при попытке регистрации приложения на сервере Guardant Net будет возвращен код ошибки 10 – **Сетевой ресурс программы исчерпан** (**License counter of Guardant Net exhausted**)

Важная информация

Для использования системы управления лицензиями необходимо создать в памяти ключа специальное поле **Таблица лицензий**, в котором прописать количество модулей и ресурс лицензий каждого из них.

Защищенные в этом режиме приложения при запуске регистрируются на сервере Guardant Net и занимают определенный ресурс из таблицы лицензий

Теперь конечный пользователь сможет запускать модули защищенного приложения только на определенном разработчиком количестве рабочих станций.

Пример:

```
nwkey32.exe /GN2 /MN=00 myprog1.exe
nwkey32.exe /GN2 /MN=01 myprog2.exe
nwkey32.exe /GN2 /MN=02 myprog3.exe
nwkey32.exe /GN2 /MN=03 myprog4.exe
```

Показаны вызовы утилиты защиты приложений с сетевым ключом Guardant Net II для примера, описанного выше по тексту.

Проверять сетевой ключ на локальном компьютере: /CLS

Тип электронного ключа:

```
Сетевые ключи
```

Описание:

При использовании этой опции защищенное приложение будет обращаться к Guardant Net III (Net II) как к локальному ключу, если отсутствует связь с сервером ключа.

/CLS – устаревшая опция, использующаяся для совместимости. Ее современным аналогом является сочетание опций /GN3 /GS3 (или /GN2 /GS2).

Дополнительные опции защиты приложения

Опции из этой группы позволяют задавать некоторые дополнительные (сервисные) режимы защиты приложений. Они могут быть заданы совместно с любыми другими опциями автоматической защиты (в том числе и при защите без привязки к ключу), за исключением случаев, особо оговоренных ниже.

Проверять целостность приложения: /V

Тип электронного ключа:

```
Все
```

Описание:

Защищенные Windows-приложения из-за особенностей своего строения не обладают способностью самовосстановления. Однако они при запуске проверяют свою целостность и сигнализируют о ее нарушении. В этом случае рекомендуется восстановить приложение с дистрибутива или с резервной копии.

Пример:

```
Nwkey32.exe /V myprog.exe
```

Не «очищать» приложение: /NOS

Тип электронного ключа:

Любой

Описание:

Обычно утилиты защиты Windows-приложений удаляют из них данные, не входящие в состав сегментов приложений. Это позволяет в большинстве случаев удалить забытую отладочную информацию и даже зачастую сэкономить на размере защищенного приложения: файл станет меньше, чем был до защиты.

Однако в некоторых случаях подобное «очищение» негативно сказывается на дальнейшей работе приложений, т. к. некоторые из них могут хранить свои важные данные нестандартным образом в произвольном месте EXE-файла. Например, в Windows-приложениях, написанных на FoxPro, интерпретируемый байт-код хранится нестандартно – не в соответствующем сегменте, а просто в конце EXE-файла.

Опция **/ NOS** (No Strip, не очищать приложение) позволяет отменить операцию «очистки». Если защищенное Windows-приложение работает некорректно, защитите его с опцией **/NOS**.

Пример:

```
Nwkey32.exe /GS2 /NOS myprog.exe
```

Пример защиты Win32-приложения с ключом Stealth II. При защите из приложения не будут удалены данные, не входящие в состав его сегментов (т. е. приложение не будет «очищено»).

Задать число таблиц вопросов-ответов к алгоритму: /ATR[=N]

Тип электронного ключа:

Любой

Описание:

Опция задает число независимых таблиц вопросов-ответов к алгоритму. По умолчанию значение этой опции равно 2.

В целях повышения стойкости защиты его можно увеличить (максимальное значение /ATR=200). При этом увеличивается количество вопросов от защищенного приложения к электронному ключу, что дает большие гарантии в защите приложения от специальных утилит-фильтров, которые может использовать хакер.

Необходимо помнить, что большое значение /ATR ведет к существенному увеличению времени работы автозащиты.

Упаковать секции исполняемого файла: /PACK

Тип электронного ключа:

Все

Описание:

Опция включает режим сжатия секций защищаемого файла. Это позволяет уменьшить (при наличии такой возможности) размер приложения. Распаковка требуемых секций исполняемого файла будет происходить во время выполнения.

Пример:

```
Nwkey32.exe /GS3 /PACK myprog.exe
```

Приложение MYPROG.EXE будет привязано к электронному ключу Guardant Stealth III и сжато.

Контролировать атрибуты страниц: /CPA

Тип электронного ключа:

Все

Описание:

При использовании опции защищенное приложение будет контролировать атрибуты страниц памяти в процессе работы и восстанавливать исходные значения в случае их изменения.

Хакер, исследуя защищенное приложение, может изменять атрибуты страниц памяти в своих целях. Опция /CPA препятствует этой практике.

Важная информация

Категорически запрещается использовать опцию /CPA, если приложение защищено Guardant API!

Включить поддержку неявного связывания: /IMPLICIT_LINKING_SUPPORT

Тип электронного ключа:

Любой

Описание:

Некоторые библиотеки, импортируемые защищаемым приложением, могут содержать секцию .tls. При использовании таких библиотек, необходимо включать эту опцию, в противном случае могут возникать непредвиденные ошибки.

Пример:

```
Nwkey32.exe /GS2 /IMPLICIT_LINKING_SUPPORT /PACK
myprog.exe
```

Приложение MYPROG.EXE будет привязано к электронному ключу Guardant Stealth II с поддержкой неявного связывания и сжатием.

Защитить импортируемые функции: /IMPORT_HOOK=%:L

Тип электронного ключа:

```
Все
```

Описание:

Во избежание установки точек останова на начало импортируемых функций, рекомендуется включать эту опцию.

Параметр **%** определяет тот процент импортируемых функций, которые будут защищены таким образом.

Параметр **L** определяет количество инструкций из каждой импортируемой функции, которые должны быть защищены. Максимальное значение **L**=100.

Пример:

```
Nwkey32.exe /GS3 /IMPORT_HOOK=100:10 /PACK
myprog.exe
```

Все импортируемые приложением MYPROG.EXE функции будут защищены. Также будет по возможности уменьшен размер защищенного приложения.

Сервисные опции автозащиты

Задать собственные сообщения вакцины: /MSG=[путь]имя_файла

Тип электронного ключа:

```
Все
```

Описание:

Если при запуске защищенного приложения происходит какая-либо ошибка, вакцина выдает соответствующее сообщение, и приложение прекращает работу. Сообщения вакцины можно разделить на основные и дополнительные. Основные сообщения выдаются, если не найден электронный ключ, счетчик в ключе исчерпан, запущена нелегальная копия приложения и в других подобных случаях, а дополнительные, например, при возникновении внутренней ошибки защиты.

Основные сообщения при защите ПО берутся из файла .MSG и вписываются в вакцину. По умолчанию используется файл **<имя_утилиты_защиты>.MSG**, находящийся в каталоге, который содержит соответствующую утилиту защиты.

Файл .MSG представляет собой обычный текстовый файл, состоящий из четырех строк. В каждой строке записано одно из основных сообщений, которые может выдавать вакцина при запуске защищенной программы, если возникнут следующие ситуации:

- 1-е сообщение, заданное в файле .MSG выдается, когда:
  - Аппаратный алгоритм ключа Guardant возвращает неверный ответ
- 2-е сообщение выдается, когда
  - Не подсоединен электронный ключ с заданными ID, номером программы, серийным номером, версией, маской (при задании соответствующих условий поиска)
  - В ключе отсутствует аппаратный алгоритм, который использовался при защите
- 3-е сообщение выдается, когда:
  - Исчерпан счетчик запусков (режим ограничения количества запусков)
  - Исчерпано время использования приложения (режим ограничения времени использования приложения)
  - Превышено заданное число пользователей, одновременно работающих с защищенным сетевым приложением в локальной сети (только для сетевых ключей)
- 4-е сообщение выдается, когда:
  - При запуске защищенного приложения возникла системная ошибка (недостаточно памяти, ошибка распределения системных ресурсов и т. п.)
  - В защищенном приложении обнаружен вирус

Можно задавать свои сообщения для вакцины. Для этого нужно написать файл, содержащий четыре сообщения на русском или другом языке. Длина одного сообщения не должна превышать 59 символов. Каждое сообщение надо писать в отдельной строке, одно сообщение может состоять только из одной строки, пустые строки в файле .MSG недопустимы.

При защите необходимо указать имя нового файла с сообщениями и, если это нужно, путь к нему в параметре **/MSG**. Если путь к файлу не указан, то утилита будет искать файл NwKey32.exe в текущем каталоге.

Примеры:

```
Nwkey32.exe /GS2 /MSG=c:\mydir\mymes1.msg
myprog.exe
Nwkey32.exe /GS3 /MSG=mymes2.msg myprog.exe
```

В первом случае сообщения вакцины будут считаны из файла MYMES1.MSG, находящегося в C:\MYDIR, а во втором случае – из файла MYMES2.MSG, находящегося в каталоге утилитой NwKey32.exe.

Задать собственные звуковые сообщения об ошибках: /SND[=filename.ext]

Тип электронного ключа:

Все

Описание:

При возникновении ошибки защищенное приложение может не только выдавать на экран сообщение, но и проигрывать соответствующий звуковой файл. Звуковые сообщения задаются при помощи опции **/SND[=FileName.Ext]**.

По умолчанию (опция **/SND**) используются звуковые сообщения, указанные в файле NWKEY32.SND, однако можно задать свой файл .SND следующим образом, например:

```
/SND=myfile.snd
```

Файл .SND представляет собой обычный текстовый файл, состоящий из 4 строк. В каждой строке записывается имя (без указания пути и расширения) звукового файла *.WAV, который будет проигран при возникновении соответствующей ошибки. *.WAV-файлы записываются в файле .SND в том же порядке, в котором текстовые сообщения записываются в файле *.MSG. Длина имени файла *.WAV – не больше 8 символов, каждое имя пишется на новой строке.

Файлы *.WAV должны быть включены в комплект поставки защищенного программного продукта и при его инсталляции скопированы туда, где их может найти соответствующая функция Windows – это может быть текущий каталог, каталог, в котором находится защищенная программа, один из каталогов списка PATH.

Пример:

```
nwkey32.exe /GS2 /SND=mysnd.snd myprog.exe
```

Пример защиты приложения MYPROG.EXE с ключом Guardant Stealth II. При возникновении одной из основных ошибок будет проигран соответствующий .WAV-файл, заданный в MYSND.SND (MYSND.SND находится в том же каталоге, что и утилита защиты).

Задать выходной путь: /OUT=путь

Тип электронного ключа:

Все

Описание:

По умолчанию защищенные файлы помещаются в тот же каталог, в котором находятся исходные файлы (последние при этом переименовываются в файлы с расширением .OLD). При помощи опции **/OUT** можно задать иной путь, по которому утилита защиты будет помещать защищенные файлы.

Примеры:

```
Nwkey32.exe /GS2 /T=2 /OUT=c:\protect
myprog.exe
nwkey32.exe /GS3 /T=2 /LBL=myprog.vks
/OUT=c:\protect  myprog.exe
```

В обоих случаях приложение MYPROG.EXE при защите будет скопировано в каталог C:\PROTECT.

Запретить вывод сообщений утилиты на экран: /Q

Тип электронного ключа:

Все

Описание:

Обычно в процессе работы утилита защиты выводит на экран различные сообщения, которые позволяют ориентироваться в ходе выполнения процесса защиты: информация, считанная из ключа, режимы защиты программ и данных, сообщения об ошибках и т. п.

Однако утилита может быть запущена из какой-то внешней программы, имеющей оконный интерфейс. В этом случае сообщения утилиты могут испортить информацию, выдаваемую внешней программой.

Для запрета вывода сообщений на экран надо запустить утилиту защиты с опцией **/Q**.

В случае возникновения ошибки утилита прервет сеанс защиты, а тип ошибки можно определить по коду возврата.

Пример:

```
Nwkey32.exe /GS2 /Q myprog.exe
```

Задание списка файлов для защиты

Утилита защиты может за один сеанс работы защитить несколько файлов. Список файлов для защиты должен задаваться в командной строке после указания всех опций защиты, при этом файлы должны разделяться пробелами, например:

```
Nwkey32.exe /GN2 /PACK /V /Q myprog.exe
myprog2.exe myprog3.exe
```

В этом случае утилита защитит файлы myprog.exe, myprog2.exe и myprog3.exe, находящиеся в текущем каталоге сжимая при этом их секции, выполняя установку защиты от вирусов. Также будут отключены сообщения утилиты автозащиты.

Список файлов может содержать и полные пути к ним, а также символы "*****" и "**?**", например:

```
Nwkey32.exe /GS3 c:\mydir\*.exe
Nwkey32.exe /GS2 myexec.exe
«c:\My distrib\*.exe» c:\delphi_projects\*.exe
```

В первом случае утилита защитит все .EXE-файлы в текущем каталоге, а во втором - файл MYEXEC.EXE и все исполняемые файлы в каталогах **C:\My Distrib** и **C:\delphi_projects**. Обратите внимание на то, что вы обязательно должны указывать расширения файлов, составляющих список (исключения составляют файлы данных без расширений), либо маску, заданную символами "*****" или "**?**".

Использование спискового файла .FIL

Может возникнуть ситуация, когда из-за обилия параметров и (или) файлов будет превышен максимально допустимый размер командной строки. Чтобы избежать этого, можно использовать так называемый списковый файл.

Списковый файл – это обычный текстовый файл, имеющий расширение .FIL и содержащий список файлов для защиты (кодирования). Каждый файл, составляющий список, должен быть записан в отдельной строке, при необходимости можно указывать полные пути и символы "*****" и "**?**" в файлах. Пример содержимого спискового файла:

```
"My prog.exe"
c:\new_project\*.exe
c:\project-buh\*.exe
c:\new_base\Module*.exe
```

Чтобы указать утилите, что она должна взять список файлов из спискового файла, его имя нужно вписать в командной строке после указания всех опций защиты (т. е. вместо списка файлов, рассмотренного в предыдущем пункте).

Чтобы утилита смогла отличить списковый файл от обычного, этот параметр командной строки должен начинаться с символа "@":

```
Nwkey32.exe /GS2  @myfil.fil
Nwkey32.exe /GS3 /IMPLICIT_LINKING_SUPPORT
@c:\mydir\myfil.fil
```

В первом случае файлы для защиты будут взяты из спискового файла MYFIL.FIL, находящегося в текущем каталоге. Во втором случае утилита будет использовать списковый файл MYFIL.FIL, находящийся в каталоге C:\MYDIR.

В командной строке можно задать несколько списковых файлов. Это удобно для группировки защищаемых файлов по типам:

```
Nwkey32.exe /GN2 /GS3 /DEP=1234 @sklad_progs.fil
@others_utils.fil
```

В этом случае удобно имена всех исполняемых файлов складского приложения, подлежащих защите, поместить в SKLAD_PROGS.FIL, а имена всех прочих утилит, входящих в поставку – в OTHERS_UTILS.FIL.

Глава 7
Guardant Net: защита сетевых приложений

Концепция Guardant Net

Guardant Net – это технология защиты сетевых приложений с помощью электронных ключей Guardant.

Основными компонентами Guardant Net являются:

- Сетевой ключ, с помощью которого происходит защита и лицензирование сетевого приложения
- Сервер Guardant Net – утилита, которая обрабатывает и передает запросы от клиента к ключу и обратно
- Клиент – копия защищенного приложения, которая обращается к серверу ключа с удаленного компьютера
- Сетевой протокол, по которому происходит обмен между сервером и клиентом

К важным понятиям Guardant Net также относятся сетевой ресурс ключей и его распределение (в том числе, при работе с многомодульными программными комплексами).

Сетевые ключи

Сетевые ключи Guardant предназначены для защиты и лицензирования сетевых приложений. Под лицензированием подразумевается ограничение количества одновременно работающих в ЛВС клиентских приложений. Цель лицензирования – запретить запуск клиентов сверх разрешенного разработчиком количества. Ресурс лицензий записывается разработчиком в память сетевого ключа Guardant.

Для защиты и лицензирования сетевого продукта достаточно использовать один сетевой ключ Guardant на всю локальную сеть. Он может быть установлен на любую рабочую станцию или сервер.

К сетевым ключам относятся следующие модели:

Модель ключа	Основные характеристики					
	Объем памяти	Интерфейс	Аппаратные алгоритмы	LMS	Protected Items	TRU
Guardant Net III	2048 Кб	USB	GSII64; HASH64, RND64	+**	+	+
Guardant Net II	256 Кб	USB, LPT	GSII64; Stealth I	+	-	-
Guardant Net*	256 Кб	USB, LPT	Stealth I	+	-	-

Сетевой ресурс ключей и его распределение

Каждый сетевой ключ обладает определенным сетевым ресурсом, который позволяет ограничивать число одновременно запущенных клиентов. Сетевой ресурс ключа может распределяться по рабочим станциям или копиям приложения (начиная с версии ПО Guardant 5.1), в зависимости от решаемой задачи.

Для многомодульных приложений используется система управления лицензиями, когда каждому модулю приложения дополнительно присваивается отдельный сетевой ресурс лицензий.

Сетевой протокол

Сетевые ключи Guardant могут работать в любых локальных сетях с интерфейсами TCP/IP и NetBIOS. Однако ключ и сервер ключа должны быть установлены на компьютере под управлением ОС семейства Windows.

Сервер Guardant Net

Защищенные сетевые приложения не могут обращаться непосредственно к сетевому ключу. Связующим звеном между защищенным приложением (клиентом) и сетевым ключом выступает специальная утилита – программный сервер Guardant Net (файл GrdSrv.exe). Сервер обеспечивает прохождение запросов от клиента непосредственно к ключу и обратно по правилам сетевых протоколов TCP/IP или NetBIOS.

Важная информация

Сервер Guardant Net должен быть загружен на том же компьютере, к которому подсоединен сетевой электронный ключ.

В одной директории с GrdSrv.exe должен находиться конфигурационный файл GrdSrv.ini, в котором хранятся настройки сервера ключа.

** Поле типа «Защищенная ячейка», в котором, кроме ресурсов модулей многомодульного приложения хранится реальный сетевой ресурс ключа

* Не выпускается с 2005 года

Клиент Guardant Net

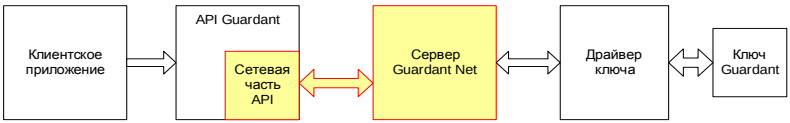
Защищенное автозащитой или при помощи функций Guardant API приложение (клиент Guardant Net) при запуске должно найти сервер ключа и зарегистрироваться на нем, чтобы продолжить работу. В качестве клиента может выступать как копия приложения, так и рабочая станция (в зависимости от схемы распределения сетевого ресурса ключа). Каждому клиенту после регистрации на сервере ключа выделяется единица сетевого ресурса.

Для работы клиента Guardant Net не требуется установка драйвера Guardant, т. к. клиент не обращается непосредственно к ключу. В одной директории с клиентским приложением должен находиться конфигурационный файл GnClient.ini, в котором хранятся настройки клиента.

Принцип работы сетевой защиты Guardant

Для работы защищенного приложения в локальной сети необходимо и достаточно установить один сетевой электронный ключ на любую рабочую станцию или сервер.

Работу с электронным ключом по сети обеспечивает клиентская (Guardant Net API и/или «вакцина» автоматической защиты) и серверная (сервер Guardant Net) части ПО Guardant. Для связи клиентской и серверной частей ПО Guardant Net необходимо настроить конфигурационные файлы клиента (GnClient.ini) и сервера ключа (GrdSrv.ini).



При запуске сервер Guardant Net считывает и запоминает сетевые ресурсы и другие параметры ключей, подсоединенных к данному компьютеру. Защищенный клиент, чтобы начать работу с ключом, должен зарегистрироваться на сервере (выполнить функцию GrdLogin). В процессе регистрации клиента сервер проверяет, подсоединен ли к компьютеру ключ с запрашиваемыми параметрами, и уменьшает на 1 значение его сетевого ресурса. В противном случае он возвращает клиенту ошибку «Электронный ключ не найден». После успешной регистрации приложение может выполнять с ключом все доступные операции. При завершении своей работы приложение снимает свою регистрацию с сервера (выполняет функцию GrdLogout). В процессе снятия регистрации производится возврат (увеличение на 1) сетевого ресурса соответствующего ключа.

Если клиент запрашивает регистрацию на сервере Guardant Net в тот момент, когда сетевой ресурс ключа уже исчерпан (равен 0), сервер вернет соответствующую ошибку, и данная копия приложения не будет запущена.

Важная информация

Сетевые ресурсы корректируются не в памяти ключей, а в памяти сервера. Это дает гарантию сохранности сетевого ресурса ключа при аппаратных сбоях в сети, «подвисании» рабочих станций и т. п.

Сетевой ресурс ключа

Число одновременно работающих клиентских приложений ограничивается сетевым ресурсом ключей Guardant Net III/ Net II/ Net.

Необходимо различать ***максимальный*** и ***реальный*** сетевой ресурс ключа.

Максимальный сетевой ресурс программируется компанией «Актив» на этапе предпродажной подготовки сетевого ключа. Значение максимального сетевого ресурса содержится в памяти ключа по адресу 19 (SAM) и не может быть изменено. Возможные значения максимального сетевого ресурса: 5, 10, 20, 50, 100, без ограничений.

Реальный сетевой ресурс программируется разработчиком перед передачей защищенного приложения и ключа конечному пользователю. Значение реального сетевого ресурса содержится в памяти ключа:

Модель сетевого ключа	**Местоположение реального сетевого ресурса**
Guardant Net III	Первый модуль таблицы лицензий. Значение ресурса дублируется в счетчике №2 (38 SAM (8 UAM)) после создания таблицы лицензий.
Guardant Net II/ Net	Счетчик №2 (38 SAM (8 UAM))

Реальный сетевой ресурс равен числу оплаченных конечным пользователям лицензий на использование приложения и не может быть больше максимального сетевого ресурса ключа.

Изменить значение реального сетевого ресурса можно:

Способ		**Порядок действий**
С помощью GrdUtil.exe	Guardant Net III	Выполните команду меню **Редактировать | Добавить поле**, выберите тип поля «Таблица лицензий», выделите модуль «Общий ресурс ключа» и задайте его значение (по умолчанию 5). После создания таблицы лицензий значение реального сетевого ресурса будет автоматически продублировано в поле «Счетчик №2»
	Guardant Net II/ Net	Выделите поле «Счетчик №2», установите новое значение и выполните команду меню **Ключ|Запись**

Способ		Порядок действий
Из приложения	Guardant Net III	Создайте в памяти ключа таблицу лицензий (см. формат таблицы лицензий) и задайте значение первого модуля таблицы (т. н. «Общий ресурс ключа»). Продублируйте это значение по адресу 38 SAM (8 UAM) с помощью команды GrdRead.
	Guardant Net II/ Net	Запишите командой GrdRead новое значение по адресу 38 SAM (8 UAM)

Распределение сетевых ресурсов

В программном обеспечении Guardant версий 5.x сетевые лицензии могут распределяться по рабочим станциям или по копиям приложения. Оба способа имеют свои особенности и применяются для решения различных задач. К примеру, распределение лицензий по копиям приложения (хэндлам) хорошо подходит для использования на терминальном сервере.

Распределение сетевых ресурсов по рабочим станциям

При распределении сетевых ресурсов ключа по рабочим станциям:

1. Сетевой ресурс ключа уменьшается на 1 только при запуске первой копии защищенного приложения. Если с этого же компьютера запускать новые копии того же приложения (либо другие приложения, привязанные к тому же ключу), сетевой ресурс ключа уже не будет уменьшаться

2. Сетевой ресурс ключа возвращается (увеличивается на 1) только по завершении работы последней копии защищенного приложения, запущенной на данном компьютере. При этом порядок, в котором были запущены копии, не имеет значения.

3. Если компьютер с запущенным на нем защищенным приложением «завис», то лицензия, выделенная компьютеру, на 15 минут останется закрепленным за ним. С этого компьютера приложение по-прежнему можно будет запускать, однако это уже не повлечет за собой изменения сетевого ресурса ключа.

Распределение сетевых ресурсов по копиям приложения

При распределении сетевых ресурсов ключа по копиям приложения (хэндлам):

1. Сетевой ресурс ключа уменьшается на 1 при запуске каждой копии защищенного приложения, независимо от того, запущены ли несколько копий на одном или на нескольких компьютерах.

2. Сетевой ресурс ключа возвращается (увеличивается на 1) по завершении работы любой копии защищенного приложения, запущенной на любом компьютере в сети. При этом порядок, в котором были запущены эти копии, не имеет значения.

3. Если защищенная копия приложения «зависла», то лицензия, захваченная этой копией, на 15 минут останется закрепленным за ней. С этой рабочей станции приложение по-прежнему можно будет запускать, причем это влечет за собой уменьшение сетевого ресурса ключа (каждая запущенная копия захватывает 1 лицензию).

Важная информация

При распределении сетевых ресурсов ключа по хэндлам необходимо иметь в виду, что запуск нескольких копий одного приложения на одном компьютере может привести к неоправданному расходованию сетевых ресурсов.

Высвобождение зависших лицензий

Зависание лицензий на сервере ключа – это распространенная ситуация, при которой единица сетевого ресурса на время остается закрепленной за рабочей станцией или копией приложения после их зависания или некорректного завершения работы. В этом случае в результате сбоя не выполняется операция GrdLogout, и сеанс связи с сервером не прекращается после фактического завершения работы приложения.

В сетевом ПО Guardant реализован механизм высвобождения зависших лицензий. Клиентское приложение производит регулярный, как минимум, один раз в 5 минут, опрос сервера Guardant Net. Сеанс связи с приложением, не отправившим данные в течение 3-х интервалов между опросами, прекращается, а данный клиент помечается в консоли сервера как «неактивный» (выделяется серым цветом). Т. о. лицензия становится свободной уже через 15 минут после зависания, но при этом информация о неактивном подключении остается в списке клиентов.

Лицензия, использовавшаяся неактивным клиентом, либо возвращается прежнему клиенту при повторной регистрации на сервере, либо передаётся новому клиенту, если тому не хватает свободной лицензии.

Важная информация

1. Необходимо учитывать, что в окне сервера помимо текущих активных подключений также отображаются все неактивные клиенты с момента запуска сервера. Это не означает, что каждый неактивный клиент занимает в данный момент единицу сетевого ресурса. В данном случае, это всего лишь своеобразная история подключений к серверу.

2. Информация о зависших соединениях удаляется автоматически через 1 сутки или больше (см. параметр BadClientKillTimeout в файле GrdSrv.ini).

Таблица лицензий

Для лицензирования многомодульных приложений предназначена специальная технология Guardant – *система управления лицензиями (LMS – License Management System).*

В этом случае, помимо общего (реального) сетевого ресурса ключа отдельный сетевой ресурс назначается каждому модулю программного комплекса. В памяти ключа должно быть создано специальное поле типа «Таблица лицензий», которое содержит значение сетевого ресурса по каждому модулю.

Важная информация

Для ключей Guardant Net III наличие таблицы лицензий обязательно в любом случае, т. к. в ней хранится значение реального сетевого ресурса ключа.

При использовании таблицы лицензий происходит двухуровневый контроль лицензий:

1. Общее количество рабочих станций или число одновременно запущенных копий приложения (в зависимости от способа распределения), ограничивается реальным сетевым ресурсом ключа
2. Количество рабочих станций, на которых одновременно используется определенный модуль программы, или одновременно запущенные копии определенного модуля (в зависимости от способа распределения сетевого ресурса), ограничивается ресурсом этого модуля (значение соответствующего байта таблицы лицензий)

При этом значение реального сетевого ресурса может не совпадать с суммой ресурсов лицензий всех модулей.

Пример:

Защищенное приложение состоит из 4-х модулей: Бухгалтерия, Персонал, Маркетинг, База продаж.

Реальный сетевой ресурс ключа равен 15.

Ресурс каждого модуля указан в таблице:

Модуль	Ресурс модуля
Бухгалтерия	5
Персонал	6
Маркетинг	3
База продаж	10

По рабочим станциям

Модули приложения могут работать на 15 рабочих станциях одновременно в любом сочетании, но количество компьютеров, на которых работает какой-либо один модуль, не может превышать ресурс этого модуля (т. е. не более 5 компьютеров для Бухгалтерии, не более 10 – для Баз продаж и т. п.)

Если на одном компьютере запущено несколько модулей, например, Бухгалтерия, Маркетинг и База продаж, то реальный сетевой ресурс ключа уменьшается на 1, также на 1 уменьшается ресурс каждого из этих модулей.

По копиям приложения

Одновременно могут работать 15 копий модулей приложения в любом сочетании, но число запущенных копий определенного модуля не может превышать его ресурс (т. е. не более 5 копий Бухгалтерии, не более 3 – Маркетинга, не более 10 – Баз продаж и т. п.).

Если на компьютере (или нескольких компьютерах) будет запущено, например, 4 копии Бухгалтерии, 3 – Маркетинга и 2 – Базы продаж, то реальный сетевой ресурс ключа уменьшится на 9, а ресурс каждого модуля будет уменьшен по числу запущенных копий для данного модуля.

Создание и использование таблицы лицензий

Для использования системы управления лицензиями необходимо:

1) Создать в памяти ключа поле «Таблица лицензий», в котором определить количество модулей, их ресурсы лицензий, а также дополнительные параметры

2) Экспортировать параметры таблицы лицензий в файл настроек GrdSrv.ini

3) При использовании Guardant API:

- Указать номер модуля таблицы лицензий в параметре dwModuleLMS функции GrdLogin
- Указать следующее значения флага dwLoginFlags функции GrdLogin:
  - GrdLM_PerStation – при распределении сетевых ресурсов по рабочим станциями
  - GrdLM_PerHandle – при распределении сетевых ресурсов по копиям приложения

4) При автоматической защите:

- Задействовать опцию **/MN=xx**, где xx – номер модуля таблицы лицензий (или параметр Мастера автозащиты «Использовать систему управления лицензиями»)
- Задействовать опцию **LOGIN_MODE=[H, S]**, где H – распределение сетевых ресурсов по хэндлам (копиям приложения), S – распределение сетевых ресурсов по рабочим станциям

Формат таблицы лицензий

Для создания таблицы лицензий с помощью Guardant API необходимо знание ее формата.

Адрес таблицы лицензий в памяти ключа определяется по значению, указанному в поле **kmTableLMS (29 SAM)**.

Заголовок таблицы имеет размер 2 байта.

Формат таблицы лицензий (LMS) Guardant Net III

Для Guardant Net III наличие таблицы лицензий необходимо в любом случае, т. к. в ней хранится реальный сетевой ресурс ключа. Реальный ресурс ключа лежит в первой по счету записи LMS. За ним следуют записи с ресурсами модулей.

Таблица лицензий в Guardant Net III представляет собой частный случай защищенной ячейки (см. описание формата дескриптора защищенной ячейки в части 2 Руководства пользователя).

Формат таблицы лицензий* Guardant Net III:

Адрес	Размер	Значение	Комментарии
0000h	2 байта	'LT'	Сигнатура таблицы
0002h	2 байта	0x0510	Версия таблицы
0004h	4 байта	CRC	Контрольная сумма таблицы, подсчитанная при помощи GrdCRC от -1
0008h	2 байта	LMS_FLAG_LICENCE_SIZE_2BYTES	Флаги таблицы. LMS_FLAG_LICENCE_SIZE_2BYTES установлен, если размер записи – 2 байта, иначе -1 байт
0010h	1 байт	Количество записей таблицы	Ограничено максимальным размером данных в защищенной ячейке, общий размер LMS не может превышать 254 байта.
0011h	5 байтов	Зарезервировано	Не используется
0016h	1 или 2 байта	0-255 или 0-65535	Реальный сетевой ресурс ключа
0017h или 0018h	1 или 2 байта	0-255 или 0-65535	0-й модуль LMS

* Содержимое поля rs_K[] защищенной ячейки (см. Руководство пользователя, часть 2, глава 10, «Дескриптор защищенной ячейки»).

При программировании ключа в поле «Счетчик №2» необходимо прописать значение, равное значению основного ресурса в таблице лицензий, т. к. «Счетчик №2» нужен утилитам диагностики для отображения информации о ключе.

Формат таблицы лицензий Guardant Net II/ Net

Для ключей Guardant Net II/ Net наличие таблицы лицензий обязательно только в том случае, если приложение использует многомодульное лицензирование.

Здесь таблица лицензий отвечает только за ресурсы модулей, основной ресурс сетевого ключа лежит в счетчике GP2.

Формат таблицы Guardant Net II/ Net:

Адрес	Размер	Значение	Комментарии
0000h	1 байт	Размер модуля + Количество записей в таблице	Если старший бит байта установлен в 1, то ресурсы модулей имеют размер – 2 байта, иначе – 1 байт. Остальные 7 бит содержат количество записей от 1 до 127 или насколько хватит памяти ключа. Для старых клиентов в новом сервере можно использовать лицензирования по копиям приложения, если в старшем бите модуля указать 1. Так 128 – лицензирования по основному ресурсу в режиме лицензирования по копиям. 129 – модуль №0 по копиям и т.д.
0001h	1 байт	Зарезервировано	Не используется
0002h	1 или 2 байта	Ресурс 0-го модуля	Размер может быть 1 или 2 байта (максимальное значение 255 и 63355 соотв.), следующая запись лежит прямо за данной записью. Если указан максимальный ресурс, то ресурс модуля считается неограниченным.
0003h или 0004h	1 или 2 байта	Ресурс 1-го модуля	Адрес зависит от размера записи

Поддерживаемые сети и протоколы

Сетевые ключи Guardant могут работать в любых локальных сетях с интерфейсами NetBIOS и TCP/IP. Однако следует учитывать, что сервер Guardant Net является Win32-приложением, поэтому он должен быть загружен на сервере или рабочей станции, работающей под управлением ОС семейства Windows.

Важная информация

В большинстве случаев можно рекомендовать отключение протокола NetBIOS в конфигурационных файлах клиента и сервера и работу только по протоколу TCP/IP.

Сетевое программное обеспечение Guardant Net поддерживает сетевые протоколы TCP/IP и NetBIOS (или любой его эмулятор):

Тип приложения	Поддерживаемый сетевой протокол	
	TCP/IP	NetBIOS
Win32	+	+
Win16	+	+
DOS	-	+

В локальной сети должен быть настроен хотя бы один из указанных протоколов, в противном случае сервер ключа выдаст ошибку типа «Протокол не найден».

Для работы с TCP/IP и создания общего интерфейса для обоих типов протоколов используется эмуляция команд NetBIOS посредством WinSock. На более высоком уровне TCP/IP представляется как отдельный NetBIOS LANA (т. н. TCP-LANA), который можно использовать для вызова команд с семантикой NetBIOS.

Работа по протоколу TCP/IP

Сетевое программное обеспечение Guardant не налагает ограничений на работу по протоколу TCP/IP. Доступности клиента по сети могут препятствовать только настройки сетевых экранов, маршрутизаторов и т. п. оборудования, а также настройки сетевых политик безопасности.

Таким образом, допускается работа с Guardant Net по Интернет (удаленные филиалы, корпоративная VPN), однако эта возможность является побочной, она не тестировалась досконально.

Работа по протоколу NetBIOS

В тех ситуациях, когда использование NetBIOS является единственной возможностью (например, при работе с сетевым DOS-приложением), следует учитывать особенности этого сетевого интерфейса.

По умолчанию, клиент, работающий по протоколу NetBIOS, «видит» сервер Guardant Net только в пределах одного сегмента сети, поскольку этот протокол является немаршрутизируемым. Когда возникает необходимость организовать работу защищенного приложения в сложной сети, состоящей из нескольких сегментов, можно воспользоваться одной из следующих рекомендаций:

1. Установить ключ и сервер Guardant Net на компьютер (например, Windows-сервер), доступный сразу из нескольких сегментов сети.
2. Разрешить межсегментный обмен пакетами. Для этого нужно установить или переконфигурировать маршрутизатор и/или коммутатор.

3. Установить в каждом сегменте сетевой ключ и сервер Guardant Net.
4. Использовать на компьютерах, входящих в другие сегменты, локальные ключи.

Поддержка нескольких адаптеров и сетевых интерфейсов

Все сетевое ПО Guardant Net может работать одновременно с несколькими сетевыми интерфейсами и сетевыми адаптерами. Каждая такая пара «сетевой интерфейс – адаптер» однозначно нумеруется и называется LANA (LAN Adapter).

Сервер Guardant Net

Сервер Guardant Net (файл GrdSrv.exe) обеспечивает связь между защищенным сетевым приложением и ключом сетевым Guardant в ЛВС. Один сервер GrdSrv.exe может обслуживать запросы к нескольким сетевым электронным ключам Guardant.

Загрузка сервера

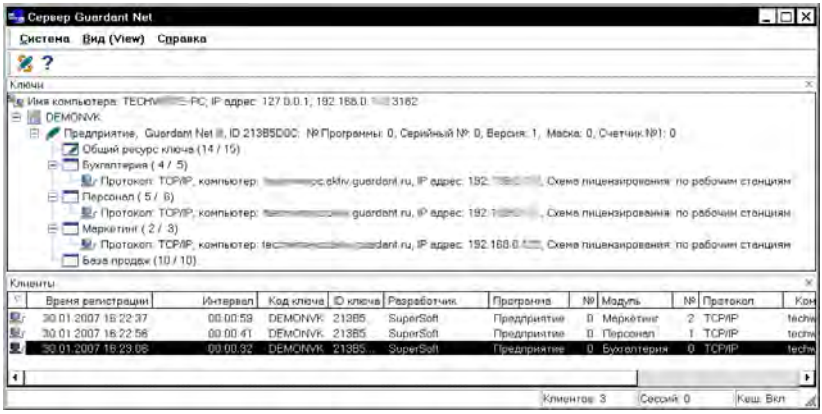
Сервер Guardant Net должен быть загружен на том же компьютере, к которому подсоединен сетевой электронный ключ.

В пределах локальной сети можно запускать несколько серверов Guardant Net. Они должны находиться на разных компьютерах и обладать уникальными NetBIOS-именами (NetBIOS-имя и другие параметры сервера задаются в GrdSrv.ini). Запуск двух серверов (сервера и сервиса, двух сервисов) на одном компьютере невозможен.

Чтобы загрузить сервер Guardant Net выполните одно из следующих действий:

- Откройте файл **GrdSrv.exe**, находящийся в каталоге **"%Program Files%\Guardant\Guardant 5\%Public Code%\Bin"**
- Выберите элемент **Сервер Guardant Net** на вкладке **Сетевая защита** утилиты **Guardant Интегратор**
- Выберите элемент **Сервер Guardant Net** из программной группы **Комплект разработчика Guardant 5 – Public Code** в меню **Пуск**

После загрузки сервера на экране появится его главное окно:



Главное окно сервера

Сервер Guardant Net сочетает функции собственно сервера и монитора. Т. о. непосредственно в окне сервера можно наблюдать текущее состояние подключений.

В главном окне сервера ключа отображаются следующие элементы управления:

- Меню
- Ключи
- Клиенты
- Строка статуса

Меню

Меню сервера ключа GrdSrv.exe имеет следующую структуру:

Раздел меню	Команда меню	Описание команды
Система	Настройки	Вывести на экран окно настроек сервера
	Перечитать информацию о ключах	Обновить информацию о ключах, подсоединенных к компьютеру. Сервер ключа автоматически выполняет эту операцию (случайным образом в диапазоне от 10-ти до 15 минут)
	Остановить	Завершить работу сервера ключа

Раздел меню	Команда меню	Описание команды
Вид	Ключи	Показать/скрыть окно с информацией об электронных ключах
	Клиенты	Показать/скрыть окно с информацией о защищенных клиентских приложениях
	Панель инструментов	Показать/скрыть панель инструментов
	Строка статуса	Показать/скрыть строку статуса
	Switch to English/Russian	Выбрать английский/русский язык
Справка	О программе	Получить информацию о версии GrdSrv.exe

Ключи

В окне **Ключи** с помощью древовидной структуры представлена информация о компьютере, на котором установлен ключ (ключи) и сервер, а также основные данные о клиентах:

Уровень вложенности	Что отображается	Описание
Основание древа	Информация о компьютере	Имя компьютера
		IP-адрес
		NetBIOS-имя
1-й	Общий код ключа	Идентификатор разработчика
2-й	Информация о ключе	Модель ключа
		Название приложения
		Битовая маска
		ID ключа
		Номер программы
		Серийный номер
		Версия программы
		Значение счетчика GP
3-й	Информация о реальном сетевом ресурсе и модулях защищенного приложения	Значение реального сетевого ресурса ключа: текущий / максимальный
		Название модуля таблицы лицензий, его ресурс лицензий: текущий/ максимальный
4-й	Информация о клиенте	Сетевой протокол, по которому выполнено соединение
		Имя и IP-адрес компьютера, на котором запущен клиент
		Схема распределения сетевых ресурсов

Зарегистрированный на сервере сетевой ключ Guardant может быть доступным, заблокированным или недоступным.

Статус ключа отражают специальные символы рядом с пиктограммой ключа:

Пиктограмма	Статус ключа
	Ключ заблокирован операцией GrdLock
	Ключ недоступен (отсоединен)
	Отсутствие пометок означает, что ключ доступен, т. е. подсоединен к порту компьютера и не заблокирован.

Клиенты

Окно Клиенты служит для получения оперативной информации о зарегистрированных на сервере ключа клиентах. Окно организовано в виде таблицы, строки которой образуют характеристики клиентов, а столбцы – перечень клиентов:

Столбец	Характеристика клиентского приложения
Время регистрации	Время регистрации клиентского приложения на сервере ключа в формате день.месяц.год часы:минуты:секунды
Интервал	Время, прошедшее с момента последнего обмена данными с клиентом в формате часы:минуты:секунды
Код ключа	Общий код (Public Code) ключа, используемого клиентом
ID ключа	Идентификатор ключа, используемого клиентом
Разработчик	Название фирмы-разработчика (если используется таблица лицензий)
Программа	Название клиентского приложения (если используется таблица лицензий)
№	Порядковый номер клиентского приложения (если используется таблица лицензий)
Модуль	Название модуля из состава многомодульного клиентского приложения (если используется таблица лицензий)
№	Номер модуля из состава многомодульного клиентского приложения (если используется таблица лицензий)
Протокол	Протокол, используемый клиентом для связи с сервером
Компьютер	hostname или NetBIOS-имя компьютера, на котором запущено клиентское приложение
IP-адрес	IP-адрес компьютера, на котором запущено клиентское приложение
Версия	Версия клиента Guardant
Платформа	Платформа, для которой написано клиентское приложение: Win32, Win16, DOS

Данные о клиентах можно сортировать по возрастанию или убыванию по любому из параметров. Направление сортировки указывается стрелкой-индикатором в заголовке выбранного столбца

Зарегистрированный на сервере клиент может быть активным, неактивным и заблокированным.

Статус клиента отражают специальные символы:

Пиктограмма	Статус клиента
	Активный клиент
	Неактивный, зависший клиент
	Клиент заблокирован операцией GrdLock

Строка статуса

В строке статуса сервера отображаются следующая статистическая информация о его работе:

Параметр	Описание
Клиенты: xx	Число клиентов, которых обслуживает сервер в данный момент
Сессии: xx	Количество сессий (сеансов связи), открытых с клиентами
Кэш: Вкл. /Выкл.	Состояние кэша сервера

Сервис Windows Vista/2003/XP/2000/NT

Сервер Guardant Net может работать не только как обычное оконное приложение, но и как сервис (служба) Windows Vista/2003/XP/2000/NT.

Преимущества сервиса состоит в том, что он запускается самой операционной системой во время ее загрузки, для его запуска не нужно выполнять процедуру регистрации на компьютере, и пользователю доступны специальные средства Windows по управлению сервисом.

Установка и запуск сервиса Guardant Net

Для установки сервиса запустите сервер Guardant Net из командной строки с опцией /i: GrdSrv.exe /i.

Для установки и запуска сервиса запустите сервер Guardant Net из командной строки с опцией /s: GrdSrv.exe /s.

Это действие нужно произвести только один раз. После того как сервис Guardant Net будет успешно запущен, защищенные приложения получат доступ к сетевым ключам Guardant. Сервис будет запускаться автоматически при каждом старте ОС.

Остановка сервера Guardant Net

Работу сервиса Guardant Net можно временно приостановить. Для этого нужно зайти в **Панель управления|Службы** (для Windows NT) или **Панель управления|Администрирование|Службы** (для Windows Vista/2003/XP/2000) и щелкнуть правой кнопкой мыши на строке «Guardant Net service». В появившемся меню нужно выбрать «Стоп». Сервис останется инсталлированным в систему, однако

перестанет обрабатывать запросы к сетевым ключам. Для возобновления его работы нужно выполнить старт сервиса из «Панели управления» или командой GrdSrv.exe /s.

Важная информация

Остановка сервиса не предполагает его выгрузки из списка сервисов, т. е. при перезагрузке компьютера сервис Guardant Net будет снова загружен.

Удаление сервиса Guardant Net из системы

Для удаления сервиса Guardant Net нужно запустить GrdSrv.exe с опцией **/r**: **GrdSrv.exe /r**

Опции командной строки

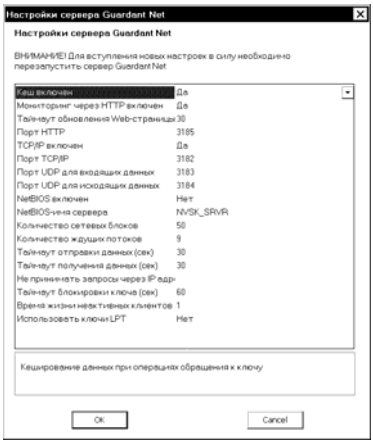
Сервис Guardant Net поддерживает следующие опции командной строки:

GrdSrv.exe /i [q]	Установить сервис (службу) Guardant Net. Взаимодействие с рабочим столом отключено
GrdSrv.exe /s [q]	Стартовать службу Guardant Net, предварительно установив ее, если до этого она не была установлена
GrdSrv.exe /e [q]	Остановить службу Guardant Net
GrdSrv.exe /r [q] **(или GrdSrv.exe /u)**	Остановить и удалить службу Guardant Net
GrdSrv.exe /?	Краткая справка

Дополнительная опция **/q** отключает вывод диагностических сообщений. Код ошибки помещается в LASTERROR.

Настройка сервера ключа. Файл GrdSrv.ini

Менять настройки сервера Guardant Net можно непосредственно из главного окна сервера с помощью команды **Система | Настройки** или путем редактирования файла настроек **GrdSrv.ini**.



Файл **GrdSrv.ini** должен находиться в том же каталоге, что и сервер ключа. Если этот файл не будет найден, всем параметрам сервера ключа будут присвоены значения по умолчанию.

На момент написания Руководства сервер имеет следующие конфигурируемые параметры, записанные в секциях [NCBs], [CACHE], [TIMEOUT], [SYSTEM], [PROTOCOLS] и [SERVER]:

Секция	Параметр и его значение по умолчанию	Возможные значения параметра	Описание
[NCBs]	TotalNCB=50	1 – 256	Максимальное количество сетевых пакетов, которое может получать/ передавать сервер.
	NCBInLANA=3	1 – 9	Число новых клиентов, которое может быть зарегистрировано сервером на каждом LANA в один момент времени. Для нормальной работы сервера должно выполняться условие: TotalNCB > NCBInLANA*LANAs + 2, где LANAs – количество LANA на данной рабочей станции.
[CACHE]	CacheMode=On	On/Off	Кэш уменьшает время ответа GrdSrv.exe при выполнении чтения памяти ключа. Наиболее эффективно влияет на работу сервера с большим количеством клиентов и повышает устойчивость работы сервера в моменты пиковых нагрузок.
[TIMEOUT] *	TO_SEND=30	1 – 120 [сек]	Таймаут на посылку данных клиентом серверу ключа.
	TO_RECEIVE=30	1 – 120 [сек]	Таймаут на прием данных клиента сервером ключа.
	TO_SEARCH=5	1 – 120 [сек]	Таймаут на широковещательный поиск сервера ключа.
	BadClientKillTimeout =1	0 – 365 [суток]	Таймаут, по которому сервер Guardant Net удаляет информацию о неактивных клиентах. 0 – не очищать
[PROTOCOLS]	TCP_IP=1	0, 1, 2	Используемые сетевые протоколы: 1, 2 – протокол используется, 0 – протокол не используется
	NETBIOS=2		

* Увеличение таймаутов необходимо при работе по медленной линии или большой загрузке сервера ключа, чтобы клиент, ожидающий ответ, не отключался по истечении таймаута.

Секция	Параметр и его значение по умолчанию	Возможные значения параметра	Описание
[SERVER]	TCP_PORT=3182	Любой доступный порт	Номер TCP/IP-порта, по которому происходит обмен между сервером и клиентом Guardant
	UDP_PORT_CLIENT= 3183		Номер UDP-порта, через который клиент рассылает датаграммы серверам
	UDP_PORT_SERVER= 3184		Номер UDP-порта, через который сервер отвечает клиенту
	NB_NAME= NVSK_SRVR	Любое	NetBIOS-имя сервера Guardant Net. Не должно совпадать с host name компьютера
	IGNORED_IP= XXX.XXX.XXX.XXX	Список игнорируемых IP-адресов	Блокирование запросов нежелательных сетевых интерфейсов, при использовании нескольких сетевых карт. IP-адреса в списке разделяются запятыми
	UseLPTDongles=Off	On/Off	Использовать / не использовать LPT-ключи
	Dongles=x	По числу описаний	Число описаний ключей (только если используется система управления лицензиями)
[HTTP]	HttpService=On	On/Off	Мониторинг состояния службы Guardant Net через web-браузер
	HttpRefresh Timeout=30	Любое допустимое	Интервал обновления страницы web-браузера
	HttpPort=3185	Любой	Номер порта для приема http-запросов

Секция [KEY_xx]

При использовании системы управления лицензиями (LMS) в **GrdSrv.ini** с помощью утилиты GrdUtil.exe (или вручную в любом текстовом редакторе) добавляются секции вида **[KEY_xx]**, где xx – номер секции. В этих секциях находятся описания ключей для различных многомодульных приложений – данные таблицы лицензий и параметры поиска ключа.

Возможные параметры поиска ключа расположены в секции по убыванию приоритета, т. е. ID ключа имеет абсолютный приоритет, а битовая маска – самый низкий. Уровень более приоритетного параметра выше суммы всех менее приоритетных.

При запуске сервер Guardant Net считывает информацию из подсоединенных ключей и подбирает для каждого ключа наиболее подходящее из описаний, содержащихся в ini-файле. Подходящим описанием считается такое, в котором суммарный приоритет параметров поиска, удовлетворяющих данному ключу, больше суммарного приоритета других описаний. Если в ini-файле есть несколько секций описаний с одинаковым суммарным приоритетом, то будет использоваться первая из них.

Параметры секции **[KEY_xx]**:

Параметр	Описание
PublicCode=xx	Общий код ключа
ID=xx	Идентификационный номер ключа. Параметр имеет абсолютный приоритет. Если указан ID ключа, то описание будет присвоено только данному ключу
VendorName=xx	Название компании-разработчика. Данные таблицы лицензий
ProgramName=xx	Название защищенного программного комплекса. Данные таблицы лицензий
ProgramNumber=xx	Номер программы. Дополнительный параметр поиска описания для ключа
Version=xx	Версия. Дополнительный параметр поиска описания для ключа
Mask=xx	Битовая маска. Дополнительный параметр поиска описания для ключа
SerialNumber=xx	Серийный номер. Дополнительный параметр поиска описания для ключа
Module0=xx	Имя 1-го модуля программного комплекса. Данные таблицы лицензий
ModuleN=xx	Имя N-го модуля программного комплекса. Данные таблицы лицензий

Клиент Guardant Net

Чтобы начать работу, клиентское приложение должно зарегистрироваться на сервере ключа. При старте сетевой клиент запрашивает сервер Guardant Net и продолжает работать только после успешной регистрации на сервере.

Клиент ищет сервер, как с помощью встроенных механизмов Guardant API, так и используя настройки файла конфигурации **GnClient.ini**: широковещательный поиск и указание прямого IP-адреса.

Широковещательный поиск включен в конфигурационном файле GnClient.ini по умолчанию. В этом случае при запуске клиент рассылает по сети периодические широковещательные запросы (датаграммы). На запросы откликается сервер Guardant Net, удовлетворяющий заданным параметрам поиска ключа. Широковещательный поиск может значительно сократить время регистрации клиента на сервере Guardant Net, особенно в Win32-приложениях.

Однако в некоторых случаях требуется указывать IP-адрес компьютера, на котором установлен ключ и запущен сервер.

В процессе прихода ответов от сервера ключей, клиент формирует список с его координатами (имя и IP-адрес хоста, NetBIOS-имя сервера).

Поиск сервера у Win32-клиента происходит быстрее, чем у DOS- и Win16-клиентов:

- По истечению времени рассылки запроса клиенты DOS и Win16 пытаются поочередно установить связь с каждым найденным сервером, пока один из них не примет клиента. Если зарегистрироваться на серверах, найденных через широковещание, не удалось, или такие не найдены, клиент устанавливает связь с сервером, параметры которого указаны в файле GNCLENT.ini (IP_NAME, NB_NAME).
- У Win32-приложений процесс добавления NetBIOS-имени по LANA, широковещательный поиск по датаграммам и прямой поиск по настройкам файла GnClient.ini происходят параллельно. Как только найден первый сервер Guardant Net, клиент пытается установить с ним связь, и, в случае успеха, прерывает работу всех параллельных процессов, за исключением процесса добавления NetBIOS-имени, которое нельзя остановить. Затем клиент получает лицензию и завершает регистрацию на сервере.

При большом количестве клиентов сильно возрастает нагрузка на сеть и сервер ключа. В этом случае, рекомендуется устанавливать несколько ключей и серверов Guardant Net. Время отклика сервера зависит от его загрузки: раньше отвечает менее загруженный на данный момент сервер. Таким образом, автоматически происходит балансировка загрузки сервера ключа.

Настройка клиента Guardant Net

Конфигурируемые параметры клиента Guardant Net собраны в секциях [PROTOCOLS], [TIMEOUT] и [SERVER] файла GnClient.ini. Этот файл должен находиться в одной директории с копией защищенного приложения. Для сетевого клиента JAVA конфигурационный файл может также располагаться в системном каталоге.

При отсутствии GnClient.ini всем параметрам сервера Guardant Net присваиваются значения по умолчанию. В этом случае клиент будет искать сервер со стандартным именем (NVSK_SRVR) и только по протоколу NetBIOS.

Секция	Параметр и его значение по умолчанию	Возможные значения параметра	Описание
[PROTOCOLS]	TCP_IP=1	0, 1, 2	Используемые сетевые протоколы: 1, 2 – протокол используется, 0 – протокол не используется
	NETBIOS=2		

Секция	Параметр и его значение по умолчанию	Возможные значения параметра	Описание
[TIMEOUT] **	TO_SEND=30	1 – 120 [сек]	Таймаут на посылку данных клиентом серверу ключа.
	TO_RECEIVE=30	1 – 120 [сек]	Таймаут на прием данных клиента сервером ключа.
	TO_SEARCH=5	1 – 120 [сек]	Таймаут на широковещательный поиск сервера ключа.
[SERVER]	TCP_PORT=3182	Любой доступный порт	Номер TCP/IP-порта, по которому происходит обмен между сервером и клиентом Guardant Net
	UDP_PORT_CLIENT =3183		Номер UDP-порта, через который клиент рассылает датаграммы серверам
	UDP_PORT_SERVER =3184		Номер UDP-порта, через который сервер отвечает клиенту
	BC_ADDR =255.255.255.255	-	Адрес широковещания. Константу 255.255.255.255 рекомендуется использовать при рассылке широковещательных пакетов. Фактически это означает, что при любой маске подсети все станции получат пакет.
	IP_NAME=127.0.0.1	Реальный IP-адрес или host name	IP-адрес компьютера, на котором установлены сетевой ключ и сервер Guardant Net. Если в сети используются динамические IP-адреса (DHCP-сервер), то вместо IP-адреса следует указывать host name компьютера.
	NB_NAME =NVSK_SRVR	Любое	NetBIOS-имя сервера Guardant Net. Не должно совпадать с host name компьютера.
	SEARCH=On	On/Off	Включение/выключение широковещательного поиска

Мониторинг состояния службы Guardant Net

Для мониторинга состояния сервера (или службы) Guardant Net можно использовать, как специальную утилиту **GrdMon.exe**, так и обычный web-браузер.

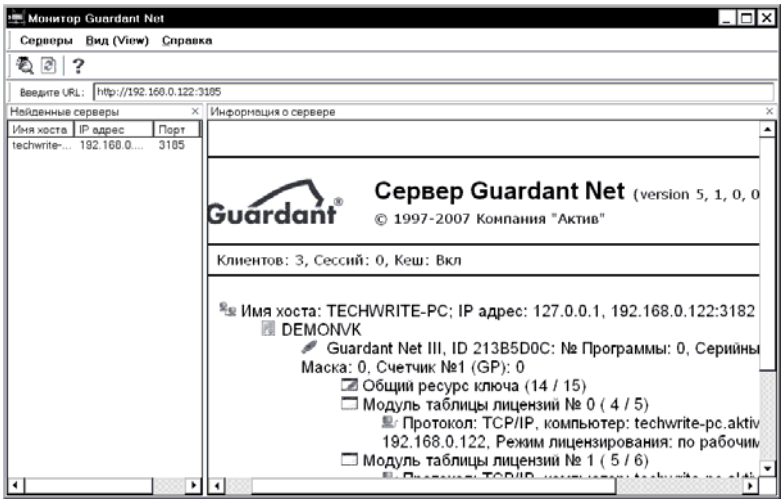
И сетевой монитор и web-браузер предоставляют одинаковую информацию по серверу ключа. Единственным отличием между ними является то, что сетевой монитор выдает данные по всем серверам Guardant Net, запущенным в локальной сети. Тогда как web-браузер отображает окно только одного сервера Guardant Net, IP-адрес которого был указан в строке поиска.

** Увеличение таймаутов необходимо при работе по медленной линии или большой загрузке сервера ключа, чтобы клиент, ожидающий ответ, не отключался по истечении таймаута.

Сетевой монитор GrdMon.exe

Сетевой монитор GrdMon.exe предназначен для получения по сети оперативной информации обо всех доступных серверах и сетевых электронных ключах Guardant в пределах локальной сети. Сетевой монитор можно запускать с любой рабочей станции.

При запуске монитор находит по сети все доступные серверы Guardant Net, устанавливает с ними связь и выдает на экран полученную информацию:



Окно монитора GrdMon.exe разделено на 2 секции.

В левой в виде алфавитного списка указываются URL и IP-адреса всех компьютеров ЛВС, на которых запущены серверы Guardant Net. Переключение между строками списка (серверами Guardant Net) происходит по двойному щелчку мыши на нужной записи.

В правой секции монитора выдается информация о параметрах и состоянии текущих сервера и сетевого ключа Guardant. Эта информация идентична той, которую выдает в своем окне сам сервер (см. описание сервера).

Мониторинг через web-браузер

Для получения о сервере можно также воспользоваться стандартным web-интерфейсом, набрав в строке поиска браузере реальный IP-адрес (или host hame) компьютера, на котором установлен сервер и сетевой ключ Guardant, и указав порт, по которому сервер ключа принимает http-запросы:

Например, **http://192.168.0.x:3185/**

Необходимо также удостовериться, что в файле настроек сервера GrdSrv.ini включена поддержка http-запросов (параметр **HttpService=On**).

Повышение надежности сетевого обмена

Защита от перезагрузки сервера Guardant Net

Конечные пользователи могут попытаться запустить в сети большее, чем положено, число копий защищенного приложения. Для этого после запуска максимально допустимого числа его копий можно принудительно перезагрузить сервер ключа и получить возможность запустить еще такое же число копий приложения.

Защититься от этого достаточно просто. Защищенному сетевому приложению следует периодически проверять наличие электронного ключа. Используйте опцию проверки ключа по таймеру при автоматической защите приложения и/или производите опрос ключа при помощи функций API периодически из различных мест приложения.

После перезагрузки сервер Guardant Net уже не будет обрабатывать запросы приложений, запущенных до его перезагрузки. Первый же опрос ключа, произведенный «старой» копией приложения после перезагрузки сервера, приведет к возникновению ошибки. Т. о., через небольшой интервал времени все копии приложения, запущенные до перезагрузки сервера, станут неработоспособными.

Оптимизация работы сервера Guardant Net

1. Не рекомендуется обращаться к ключу слишком часто. Минимальное время отклика сетевого ключа Guardant составляет порядка 150-200 мс, т. о. сервер может провести реальный обмен с ключом не более 5-6 раз в секунду. А при выполнении операций с аппаратными алгоритмами время обмена может возрасти еще в несколько раз. Поэтому, скажем, пять приложений, опрашивающих ключ раз в секунду каждое, могут легко перегрузить сервер Guardant Net, так как от скорости его работы здесь уже практически ничего не зависит. Оптимальный интервал опроса ключа должен быть случайным и находиться в диапазоне от 5 до 30 минут. При этом нежелательно выполнять много тестов подряд – при этом резко возрастает вероятность пиковой перегрузки сервера. Если следовать этим рекомендациям, то это должно позволить серверу обслуживать в пределах порядка 100 защищенных приложений, запущенных одновременно. Также следует учитывать, что на одном сервере может быть зарегистрировано несколько ключей (каж-

дый со своим сетевым ресурсом), и о возможности пиковых перегрузок.

2. Не рекомендуется включать защищенное сетевое приложение на автозапуск при загрузке компьютера – при этом резко возрастает вероятность пиковой перегрузки сервера. Типичный пример – когда с началом рабочего дня одновременно включается сотня терминалов, и с них практически одновременно поступают запросы к ключу.
3. Не рекомендуется, во избежание пиковых перегрузок, производить сложных проверок ключа при загрузке защищенного приложения. Лучше ограничиться простой проверкой наличия ключа, а более сложные тесты оставить «на потом», сделав моменты их проведения случайными, привязанными к каким-то определенным событиям.
4. Не устанавливайте больших значений конфигурируемых параметров в ini-файле сервера Guardant Net. Это не принесет ожидаемого адекватного эффекта (увеличения быстродействия, стабильности работы при пиковых нагрузках на сервер и т. п.). Вместо этого сервер станет потреблять неоправданно много системных ресурсов. Значения параметров по умолчанию являются оптимальными для сетей с малым и средним количеством рабочих станций, увеличивать их имеет смысл лишь в каких-то серьезных случаях (например, при работе сервера с большим числом ключей в крупных сетях). Если ресурсов, задаваемых конфигурируемыми параметрами, станет не хватать, сервер сам сообщит об этом, выдав на экран соответствующее сообщение.

Совместное использование данных в сети

1. При опросе ключа нельзя ограничиваться «привязкой» только к кодам доступа, необходимо производить более углубленную проверку с использованием полей общего назначения. Это даст гарантию того, что защищенное приложение будет использовать сетевой ресурс именно того ключа, к которому оно «привязано», в случае, если на одном сервере зарегистрировано несколько ключей одного разработчика.
2. Не рекомендуется использовать для защиты сетевых приложений аппаратные алгоритмы, зависящие от уменьшающегося счетчика запусков. Есть вероятность того, что данная копия защищенного приложения получит неверные ответы от такого алгоритма: ведь в нем используется один счетчик для работы со всеми копиями защищенного приложения.

Глава 8
Комплект поставки защищенного приложения

Конечным пользователям должны быть переданы утилиты и файлы, которые обеспечивают работоспособность защищенного приложения и электронного ключа.

Поэтому в комплект поставки защищенного программного продукта, необходимо включить следующие файлы:

Вид защиты	**Тип электронного ключа**	
	Guardant Stealth xx, Fidus	**Guardant Net xx**
Драйверы Guardant		
Любой	Установка драйверов при помощи Microsoft Installer. Файлы из директории Guardant\Guardant 5\Drivers\x86\Windows: **GrdDriversRU.msi, Setup.Exe, Setup.Ini**	
	Установка драйверов из инсталлятора приложения при использовании драйверного Guardant API: **GrdDriversRU.msi и grddrv.dll**	
Общие файлы защиты		
Любой	1) Драйверы Guardant, 2) GrdDiag.exe	1) Драйверы Guardant, 2) GrdSrv.exe, GrdSrv.ini, GnClient.ini, GrdDiag.exe
Файлы для защиты Win32-приложений		
Автозащита	1) Общие файлы защиты 2) GrdVkc32.dll	1) Общие файлы защиты 2) GrdVkc32.dll 3) GrdNetAgent.exe
Защита Guardant API, использующая динамические библиотеки	Соответствующий файл динамической библиотеки	
Удаленное программирование ключа	1) Общие файлы; 2) GsRemote.exe (GrdTRU.exe для Stealth III/Net III) 3) GrdVkc32.dll	

Важная информация

Файл вакцины GrdVkc32.dll необходим при использовании утилит GsRemote.exe и GrdTRU.exe, поскольку они «накрыты» автозащитой.

Приложение А Повышение стойкости защиты. Рекомендации программисту

Электронные ключи Guardant – высокоэффективное средство программно-аппаратной защиты ПО, которое позволяет создать защиту практически любого уровня сложности и стойкости к взлому. Для этого необходимо придерживаться следующих рекомендаций:

Автоматическая защита

Для защиты готовых приложений от изучения логики их работы пользуйтесь утилитами автоматической защиты. При этом целесообразно использовать следующие опции:

Защита от отладчика. Она всегда включена по умолчанию и предназначена для защиты от изучения логики работы приложения при помощи отладчиков.

Защита от вирусов. Эта опция позволяет защитить приложение не только от файловых вирусов, но и от нелегальной модификации файла приложения (дописывания каких-то программных модулей, изменения копирайтов и т. п.).

Кодирование загружаемой части и внутренних оверлеев приложения. Это предохранит приложение от попыток использования другого метода изучения – дизассемблирования.

Если приложение использует хранящиеся во внешних файлах данные – целесообразно **кодировать файлы данных**. В этом случае, даже если хакеру удастся отделить защищенное приложение от вакцины автоматической защиты, оно не будет работать с кодированными данными, так как именно вакцина занимается их дешифровкой.

Используйте опции **проверки электронных ключей по времени**. В этом случае даже снятие полного дампа памяти приложения окажется для хакера пустым развлечением.

Защита при помощи API

Общие рекомендации

Под надежным прикрытием автоматической защиты вы можете строить свою внутреннюю защиту на основе Guardant API. При этом целесообразнее следовать следующим советам:

1. **Не храните в явном виде коды доступа к ключу**. Не следует хранить в «чистом» виде Личные коды, по которым производится доступ к электронному ключу. Для того чтобы осложнить хакерам их поиск в теле приложения, Личные коды можно закодировать (например, при помощи операции XOR). Непосредственно перед вызовом функции API раскодируйте их, а после того как функция отработала – сразу же удаляйте из памяти раскодированный вариант Личного кода. Также можно хранить Личные коды частями в разных переменных.

   При каждом вызове функция GetPrivateReadCode() будет декодировать демонстрационный Личный код для чтения, хранящийся в закодированном виде.

2. **Располагайте вызовы функций API по всему телу приложения**. Это резко увеличивает объем кода, который надо изучить хакеру для локализации кода проверок и его модификации.

3. Если приложение содержит оверлеи, то **поместите вызовы функций API и в оверлейную часть**. В этом случае на пути хакера встанет еще и системный оверлей-менеджер.

4. Очень **хорошим** приемом являются **«вероятностные» вызовы**, когда функция API вызывается не всегда, а с какой-то вероятностью – например 1/7. В разных местах приложения можно использовать различные вероятности вызовов (причем в жизненно важных местах приложения вероятность может быть больше, в менее важных местах – меньше). В результате хакер может попросту не найти те вызовы, которые происходят с малой вероятностью – и тогда якобы «взломанное» приложение рано или поздно преподнесет сюрприз нелегальным пользователям.

5. **Усложните логику обработки кодов возврата функций API**. Если вы будете проверять код возврата функции API простым **сравнением**, хакеру не составит труда уничтожить это сравнение прямо в теле приложения – и таким образом снять защиту. Разработайте более сложную логику. Например, используйте коды возврата в качестве индексов каких-то массивов данных, стартовых значений для генераторов псевдослучайных чисел, используйте их в алгоритмах математических вычислений, реализованных в приложении и т. п. Таким образом, точка, в

которой защищенное приложение принимает решение о своей дальнейшей работе, в чистом виде будет отсутствовать. Хакеру придется разбираться не только в логике работы защиты, но и в хитросплетениях процессов, происходящих в самом приложении. И защита станет полноправной частью приложения, без которой станут неверными его вычисления, будут использоваться не те данные и т.д.

6. **Откладывайте момент реакции приложения на коды возврата функций API**. Хорошо зарекомендовал себя прием, когда приложение принимает решение о своей дальнейшей работе не сразу по получении кодов возврата функции API, а значительно позже. Например, функция API вызывается при запуске приложения, а анализ возвращенных ею значений производится при выполнении пользователем каких-либо действий (открытие или сохранение файла, вызов меню настройки приложения и т. п.). В таком случае хакеру будет сложно проследить причинно-следственную связь поведения приложения. Кроме того, этот прием заставит хакера исследовать в отладчике слишком большие фрагменты кода приложения.
7. **Ограничивайте возможности приложения, если электронный ключ не найден**. Этот прием можно использовать в дополнение к описанному выше. Если функция API возвращает ошибку, можно не принимать кардинальных решений относительно дальнейшей работы приложения, а лишь ограничить его возможности. Например, перестать давать возможность экспортировать файлы в другие форматы, сохранять настройки, распечатывать отчеты и т. п. Причем в ответ на неудачу при вызове API из разных мест приложения можно «лишать» его все новых и новых возможностей, все более превращая в демонстрационное. Помимо важных, можно блокировать и несколько второстепенных возможностей – это увеличит вероятность того, что хакер попросту не заметит их отсутствия, и приложение в таком виде попадет на «черный рынок». Пользователей же такой «демо-версии» отсутствие части ее возможностей подтолкнет на приобретение легальной копии приложения.
8. **Подсчитывайте контрольные суммы важных участков кода приложения**. Это мощное средство защиты от модификации хакером кода приложения. Предварительно вы подсчитываете контрольную сумму участков кода, в которых производятся вызовы функций API, анализируются выходные данные функций и принимаются решения о дальнейшей работе приложения. В процессе работы приложения вы еще раз производите подсчет этих участков – и получаете достоверную информацию о том, были ли эти участки нелегально модифицирова-

ны. Производить подсчет контрольных сумм лучше в фоновом режиме (в 32-битных Windows-приложениях это можно делать в отдельном процессе), причем делать это следует значительно раньше или значительно позже того, как управление попадет в контролируемый участок кода. Один и тот же участок целесообразно контролировать из нескольких мест приложения. А реакция на факт модификации программного кода должна быть примерно такой же, как и на ошибку при вызове функции API. В результате хакер будет вынужден дополнительно искать по всему телу приложения места, в которых вы проверяете контрольные суммы. Для получения контрольной суммы можно воспользоваться функцией Guardant API, подсчитывающей CRC интересующего вас блока. А для повышения надежности метода можно, наряду с этой функцией, использовать и алгоритмы подсчета контрольной суммы, написанные вами. Их программная реализация вовсе не сложна – качественный алгоритм подсчета CRC можно встретить в любом учебнике по прикладному программированию.

9. **Меняйте логику работы приложения с модулями защиты**. Очень полезно бывает время от времени (оптимальный вариант – в каждой новой версии защищенного продукта) менять логику его работы с модулями защиты. Используйте новые приемы, подобные описанным здесь, по-иному комбинируйте их и т. п. – и тогда при выходе новой версии продукта хакер будет вынужден начинать работу по ее исследованию, что называется, «с нуля». Все его прежние наработки мгновенно теряют актуальность – ведь теперь приложение взаимодействует с модулями защиты совершенно по-иному.

Использование аппаратных алгоритмов ключей Guardant

Если вы остановили свой выбор на электронных ключах Guardant, в вашем распоряжении есть целый ряд эффективных приемов повышения стойкости защиты к взлому.

1. **Задействуйте аппаратные алгоритмы**. Это обязательное условие, если вы хотите сконструировать по-настоящему стойкую защиту! Правильное использование ответов аппаратных алгоритмов не только делает практически невозможным написание универсальных эмуляторов ключей Guardant. Удаление из защищенного приложения вызовов функций API также теряет смысл – ведь если аппаратный алгоритм не был запущен, то и важные для приложения данные не были декодированы.

2. **Задавайте ключу больше вопросов**. Если для защиты используется преобразование только одного массива данных (т. е. если приложение задает ключу всегда один и тот же вопрос), есть вероятность, что хакер все же подсмотрит верный ответ ключа и создаст подпрограмму-«заглушку», которая вместо функции API «подсовывает» приложению этот ответ. Поскольку в данном случае ключ возвращает всегда один и тот же ответ, такая подпрограмма-«заглушка» может свести роль аппаратного алгоритма в защите данного приложения на нет. Чтобы избежать этого, нужно задавать аппаратному алгоритму большое количество разных вопросов. Создайте массив различных вопросов (т. е. блоков кодированных данных), и в разных местах приложения декодируйте аппаратным алгоритмом разные кодированные блоки. Кстати, наряду с действительно важными для приложения данными в состав такого массива могут входить и «лишние» данные, на самом деле не нужные приложению — это только дезориентирует хакера. Очень хорошо было бы организовать процесс случайной выборки вопроса — тогда в одном и том же месте приложения, но в разные моменты времени будут обрабатываться разные вопросы. Это сделает практически невозможным для хакера создание как подпрограммы-«заглушки», так и эмулятора ключа.
3. **Используйте разные вопросы в разных версиях приложения**. Если в разных программных продуктах или разных версиях одного и того же продукта будут использоваться разные вопросы (т. е. разные блоки кодированных данных), это даст вам гарантию того, что хакер не сможет создать универсальный инструмент для взлома всех ваших продуктов (или всех их версий). Даже если хакер исхитрится-таки создать эмулятор для приложения, выход его новой версии заставит его производить свою работу заново — ведь новая версия приложения работает уже с другими кодированными данными.
4. **Задействуйте разные аппаратные алгоритмы в разных версиях приложения**. В электронном ключе создайте сразу несколько различных алгоритмов (например, 4). Затем, в первой версии программного продукта кодируйте данные, скажем, 1-м и 3-м алгоритмом, во второй — 2-м и 4-м, в третьей — 1-м и 2-м, и т. п. Эффект будет аналогичен рассмотренному выше, да и в самом электронном ключе ничего перепрограммировать будет не нужно — все необходимые аппаратные алгоритмы будут в нем присутствовать с самого начала.

5. **Вносите изменения в определители алгоритмов.** По своей сути определители алгоритмов аналогичны паролям или цифровым сертификатам, поэтому их тоже полезно время от времени менять. Это очень хорошая и распространенная во всем мире практика, повышающая защищенность системы.
6. **Не храните ответы аппаратных алгоритмов в защищенном приложении**. Сам принцип использования аппаратных алгоритмов основан на предпосылке, что хакер не знает заранее ответов алгоритма. В противном случае алгоритмы не имеют особого смысла: хакер, зная все его ответы, может создать и эмулятор ключа, и подпрограмму-«заглушку». Поэтому ни в коем случае не следует хранить ответы ключа где бы то ни было в самом приложении или в файлах данных! Просто используйте ответы ключа по назначению, без предварительной проверки правильности их декодирования алгоритмом, а сразу после использования – по возможности удаляйте из памяти. Данные могут быть декодированы неверно лишь в одном случае – если либо с защищенным приложением, либо с электронным ключом производились какие-то сомнительные манипуляции. Но в таком случае неверная работа (и даже «подвисание») приложения – событие весьма естественное. В самом деле, чего еще можно ожидать от приложения, «ломаемого» хакером, особенно если защита «надета» на приложение качественно? Но если проверка правильности ответов все же необходима, используйте для этого функцию подсчета CRC. На этапе установки защиты, перед тем как закодировать данные, вычислите их CRC, а в процессе работы приложения подсчитайте CRC этих же данных после их декодирования алгоритмом ключа. Если эти два значения совпали, то данные декодированы верно.
7. **Не храните «лишние» коды доступа в защищенном приложении**. Например, если приложение использует только чтение из ключа и запуск его аппаратных алгоритмов, нужно хранить в приложении только Личный код для чтения (Private Read Code). Остальные Личные коды не будут нужны, а их присутствие в теле приложения может сильно облегчить хакеру задачу получения доступа к ключу.

Важная информация

В связи с постоянно растущими вычислительными возможностями современных компьютеров настоятельно рекомендуется задавать размер вопроса/ответа и определителя аппаратных алгоритмов ключей Guardant не менее, чем 8 байт. Это позволит минимизировать шансы атаки методом brute force (полный перебор).