Guardant®

Система защиты от компьютерного пиратства

Руководство пользователя

Драйверы Guardant и HID-режим
Программирование ключей
Автоматическая защита
Сетевая защита

Издание 6.1

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:

Важно!

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. Он 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)925-77-90

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 аппаратное устройство, либо специальный файл (в случае софтверных ключей). 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
- 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 не совпадают с ожидаемыми, либо ключ отсутствует, то защищенное приложение прекращает свою работу

Семейство электронных ключей Guardant

Современная линейка Guardant состоит из электронных ключей следующих типов:

Guardant Sign

Высокоскоростной кроссплатформенный ключ с асимметричной криптографией на борту и возможностью работы без драйверов. Предназначен для защиты локального программного обеспечения. Инновационные технологии защиты, реализованные в Guardant Sign, применяются во всех современных моделях ключей Guardant.

Базируется на новой аппаратной платформе, в основе которой высокопроизводительные 32-разрядные RISC-микроконтроллеры.

Скорость работы многократно выше, чем у ключей предыдущих поколений. Объем памяти – 4096 байтов.

Поддерживает платформы Windows (включая Windows CE) и Linux.

Обладает возможностью работы без установки драйвера Guardant – HID-режим.

Содержит аппаратно реализованный асимметричный алгоритм выработки ЭЦП (ECC160) и алгоритмы симметричного шифрования (AES и GSII64).

Реализована аппаратная защита от запуска нескольких копий защищенного приложения в терминальном режиме.

Обладает встроенной защитой от анализа протокола обмена, основанной на взаимной аутентификации электронного ключа и Guardant API, использовании одноразовых сеансовых ключей при шифровании передаваемых данных.

Поддерживает основные возможности и технологии, реализованные в ключах предыдущего поколения: защищенные ячейки, аппаратные алгоритмы GSII64 (с дополнительными режимами работы), HASH64, RND64.

Guardant Time

Высокоскоростной кроссплатформенный ключ со встроенными часами реального времени и независимым источником питания. RTC-вариант Guardant Sign с сохранением всех его преимуществ. Используется для защиты и лицензирования по времени работы локального программного обеспечения.

Как и Guardant Sign, ключ базируется на высокопроизводительном 32-разрядном RISC-микроконтроллере. Скорость работы многократно выше, чем у ключей предыдущих поколений. Объем памяти – 4096 байтов.

Поддерживаются все возможности, реализованные в Guardant Sign: аппаратные асимметричные и симметричные алгоритмы (ECC, AES, GSII64), HID-режим, работа под Linux, аппаратная защита от запуска нескольких копий в терминальных сессиях, аппаратная защита протокола обмена.

Помимо этого, часы реального времени предоставляют расширенные сценарии лицензирования приложения по времени: блокирование/разблокирование аппаратного алгоритма в заданное время, задание промежутка времени, в течение которого алгоритм будет работоспособен, и многое другое.

Тесная интегрированность политик лицензирования с аппаратными алгоритмами ключа значительно повышают уровень и эффективность защиты.

Guardant Code

Высокоскоростной кроссплатформенный ключ, в котором передовые характеристики Guardant Sign гармонично сочетаются с возможностью загрузки и исполнения кода приложения внутри самого ключа. Предназначен для защиты дорогостоящего локального программного обеспечения.

32-разрядная аппаратная платформа, основанная на архитектуре Cortex-M3, обеспечивает новому ключу гарантированно высокую скорость передачи данных по зашифрованному каналу и вычислений, в том числе для чисел с плавающей точкой.

Технология Guardant Code поддерживает создание программ на языках высокого уровня, причем объем загружаемого кода может составлять порядка 20000 строк (на C).

Есть возможность вызывать аппаратные криптоалгоритмы и аппаратный датчик случайных чисел из кода, исполняемого в ключе.

В качестве средства разработки загружаемого кода можно использовать как компилятор GCC, так и любые другие версии, например RealView. Для MS Visual Studio существует специальный отладочный модуль.

Есть серийный вариант Guardant Code с RTC-модулем: Guardant Code Time.

Guardant Sign Net /Time Net

Подсемейство ключей Guardant, предназначенных для эффективной защиты и лицензирования сетевого программного обеспечения, в том числе по астрономическому времени использования. Включает сетевые варианты Guardant Sign и Guardant Time с сохранением и развитием всех достоинств и возможностей локальных модификаций.

Новая производительная аппаратная платформа, высокая скорость работы, асимметричные и симметричные криптоалгоритмы, HID-режим, работа под Linux, аппаратная защита протокола обмена.

Плюс к этому, Guardant Time Net может выполнять расширенные сценарии лицензирования приложения по времени: блокирование/разблокирование аппаратного алгоритма по указанной дате, задание промежутка времени, в течение которого алгоритм будет работоспособен.

Главным достоинством и отличием новых сетевых ключей от предшественников является туннельное шифрование сетевого трафика. То есть, передающаяся по сети информация зашифрована на сеансовых ключах, выработанных между защищенным приложением и электронным ключом без каких-либо посредников.

Guardant SP

Программный ключ для защиты тиражного локального ПО. Использование SP-ключей оправдано только в случаях: низкой стоимости приложения; когда бизнес-модель ПО не рассчитана на использование материальных носителей (распространение через Интернет); когда использование аппаратного ключа невозможно или нежелательно.

Принцип действия защиты на программных ключах основан на привязке к уникальным характеристикам компьютера, на котором работает защищенное приложение.

Размер области данных софтверного ключа составляет 4096 байт. Поддерживается платформа MS Windows.

SP-ключ поддерживает алгоритмы генерации ЭЦП (ECC160) и хэш-функции (SHA256) и симметричного шифрования (AES128).

Обладает встроенной защитой протокола обмена, основанной на взаимной аутентификации ключа и Guardant API и использовании псевдокода.

Сводная таблица характеристик ключей Guardant

Модель	Sign	Time	Code Time	Code	Sign Net	Time Net	SP
Тип	Локальный				Сетевой		Локальный
Платформа	Windows, Linux				Windows, Linux*		Windows
Объем памяти	4КБ		EEPROM: 4КБ, Flash: 128КБ		4КБ		
Аппаратные алгоритмы	ECC160, AES128, GSII64, SHA256, HASH64, RND64		ECC160, AES128, SHA256		ECC160, AES128, GSII64, SHA256, HASH64, RND64		ECC160, AES128, SHA256
HID-режим	Да						Нет
Защ.ячейки	Да						
TRU	Да						Нет
Часы RTC	Нет	Да		Нет		Да	Нет
Интерфейс	USB						Программ.

Комплект разработчика Guardant

Для начала работы с электронными ключами Guardant нужно загрузить с сайта компании «Актив» комплект разработчика и купить любой ключ, либо приобрести коробочную версию комплекта, куда уже входит один аппаратный ключ выбранной модели.

* Под Linux сервер Guardant Net может работать только в среде сборки Wine@etersoft.

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

Коробочная версия комплекта разработчика отличается от электронной только наличием одного ключа.

Коды доступа

Коды доступа – это четыре 32-битные числовые последовательности, служащие идентификаторами как разработчика, пользующегося ключами Guardant, так и самих электронных ключей.

Уникальный набор кодов доступа присваивается разработчику при первом заказе ключей Guardant и, далее, прошивается в каждый ключ перед продажей. Коды доступа однозначно идентифицируют каждого разработчика, использующего ключи Guardant. Электронные ключи с определенными кодами доступа может приобрести только лицо/организация-владелец этих кодов.

В процессе инсталляции программного обеспечения Guardant необходимо ввести коды доступа к ключам. Без ввода кодов доступа Комплект разработчика установить невозможно.

Демонстрационные коды доступа

Для оценки и изучения возможностей ключей Guardant можно использовать ключ и программное обеспечение Guardant с демонстрационными кодами доступа. Они опубликованы и общеизвестны, поэтому непригодны 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**

Электронный ключ (или коробочный комплект разработчика) с демо-кодами приобретается под залог стоимости. В дальнейшем, в течение 3-х месяцев можно получить собственные уникальные коды доступа и бесплатно заменить демо-ключ на ключ с рабочими кодами.

Глава 2
Приступая к работе

Системные требования

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

- IBM-совместимый компьютер
- Стандартный USB-порт
  (в случае использования аппаратных ключей)
- 32- или 64-битная ОС Windows 7/2008/Vista/2003/XP
- Свободное место на жестком диске ~ 150 Мб

Установка и обновление Комплекта разработчика

Установка программного обеспечения

1.1 Загрузка с сайта компании «Актив»:

Комплект разработчика Guardant доступен для свободной загрузки на странице сайта http://www.guardant.ru/support/download/software/

Загрузите и распакуйте архив, содержащий комплект, и запустите файл **Setup.exe**.

1.2 С оригинального компакт-диска:

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**.

1.3 С других дистрибутивных носителей:

Перейдите в каталог, содержащий комплект разработчика и запустите файл **Setup.exe**.

2) Руководствуясь указаниями программы-установщика, выполните следующие действия:

- Определите каталог для размещения ПО Guardant
- Введите коды доступа и контрольное число
  (если предполагается работа с демонстрационным ключом, то для автоматического ввода кодов достаточно установить флаг **Демо**)

Важно!

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

- Нажмите на кнопку **[Установить]** и следуйте указаниям программы установки
- После установки перезагрузите операционную систему

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.

Важно!

Ни в коем случае нельзя передавать файл **NvCodes.dat** посторонним лицам или организациям! **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.

Также в процессе работы программы **Setup.exe** в систему будут установлены драйверы Guardant.

Важно!

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

2. Пользователь должен обладать правами администратора системы, иначе установка драйверов будет невозможна.

По окончании процесса установки ПО в разделе **Программы** меню **Пуск** будет создана программная группа **Guardant Developers Kit № – "%Public Code%"**.

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

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.

Важно!

Комплект разработчика Guardant доступен для свободной загрузки на сайте компании «Актив» в разделе **Загрузить**.

Кроме того, последнюю версию комплекта можно получить в офисе **компании «Актив»** или ее дилеров.

Чтобы обновить ПО Guardant, выполните следующие действия:

- Распакуйте архив с новой версией комплекта разработчика
- Перейдите в каталог, содержащий обновление, и запустите файл **Setup.exe**
- В появившемся диалоге укажите каталог установки текущей версии ПО Guardant для обновления «поверх» или любой другой каталог для установки без «затирания» текущей версии
- Нажмите на кнопку **[Установить]** и следуйте указаниям программы установки
- После установки перезагрузите операционную систему

Установка электронного ключа

Установка USB-ключа

- Установите драйвер ключа (см. раздел **Драйверы Guardant**)
- Подсоедините ключ к USB-порту
- На ключе должен засветиться индикатор. Это показатель, что драйверы установлены корректно, ключ электрически исправлен и распознан операционной системой

Важно!

Для получения информации по установке LPT-ключей обратитесь к Руководству пользователя версий 5.31 и младше.

Устройство электронного ключа

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 современным локальным ключам относятся Guardant Sign/Time, Guardant Code / Code Time и программный ключ Guardant SP.

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 локальное, так и сетевое ПО. Современными сетевыми ключами являются Guardant Sign Net и Guardant Time Net.

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:

Код	Назначение
Личный код для чтения (Private Read Code)	Чтение памяти ключа и выполнение аппаратных алгоритмов
Личный код для записи (Private Write Code)	Запись данных в память ключа
Личный мастер-код (Private Master Code)	Выполнение специальных операций с ключом

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 Guardant хранится особое 4-байтовое число – уникальный идентификационный номер (ID) ключа. Он прошивается на этапе производства ключа и не может быть продублирован или изменен. ID можно использовать для «жесткой» привязки защищенного приложения к конкретному ключу. Посмотреть ID ключа можно утилитой диагностики, либо утилитой программирования ключей **GrdUtil.exe**.

Память и поля памяти

Аппаратные ключи Guardant имеют энергонезависимую память для хранения данных, которая обеспечивает сохранность данных в течение порядка 100 лет. 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.

В программных ключах Guardant SP память EEPROM эмулируется для единообразия и удобства работы.

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

Диагностика электронного ключа

В состав ПО Guardant входит утилита диагностики **GrdDem32.exe**, которая предназначена для обнаружения электронных ключей Guardant, проверки их работоспособности, сбора диагностической информации о системе, в которой они работают. На основе полученных данных утилита генерирует отчет для службы техподдержки.

GrdDem32.exe интегрирована в драйверы Guardant. В абсолютном большинстве ситуаций ее возможностей достаточно для полноценной диагностики.

Состав программного обеспечения

ПО Guardant функционально делится на следующие компоненты:

Утилиты

Guardant Integrator, файл **Guardant.exe.** Оболочка для быстрого доступа к часто используемым утилитам Комплекта разработчика.

Диагностика ключей, файл **GrdDem32.exe.** Служит для комплексной диагностики электронных ключей, сбора информации о системе, подготовки отчетов для службы техподдержки.

Автоматическая защита, файлы **NwKey32.exe**, **CodeObfuscator.exe** и **CodeProtect.exe.** Утилиты для автозащиты готовых Win32-приложений (включая .NET-сборки).

GUI-оболочки профилировщиков автозащиты, файлы **NativeProfiler-GUI.exe** и **DotNetProfilerGUI.exe**. Утилиты, выполняющие анализ приложения перед автозащитой, с целью оптимального выбора функций для защиты.

Мастер лицензирования и автозащиты, файл **LicenseWizard.exe.** Управляющая оболочка, позволяющая быстро произвести автоматическую защиту приложений в соответствии с выбранной политикой лицензирования.

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

Дистанционное программирование, файлы **GrdTRU.exe** и **GsRemote.exe**. Утилиты для удаленного обновления ключа на стороне конечного пользователя.

GrdAPI Explorer, файл **GAPIE_GUI_SE.exe**. Вызов функций GrdAPI с заданными параметрами из удобной GUI-оболочки с сохранением результатов в синтаксисе основных языков программирования.

Сервер Guardant Net, файл **Glds.exe.** Утилита, обеспечивающая работу сетевых электронных ключей Guardant в локальных сетях.

Мастер активации, файл **GuardantActivationWizard.exe**. Утилита для активации софтверных ключей Guardant SP.

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

Универсальные комплекты драйверов Guardant для 32- и 64-разрядных ОС Windows 7/2008/ Vista/ 2003/XP.

Служебные файлы

Служебные файлы, нужные для работы перечисленных утилит.

Защита программы

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

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. Такая возможность делает систему защиты уникальной.

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 множество режимов, которые позволяют настроить приложение на параметры

электронного ключа (привязать его к ID, серийному номеру и т. д.), ограничить число запусков или время работы приложения и т. д.

Основное преимущество метода – в малом времени установки защиты. На сам процесс требуется всего несколько минут. Кроме того, для установки защиты не нужны 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 устанавливается 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.
- Внешняя вакцина производит необходимые проверки и преобразования и запускает защищенное приложение

Более подробную информацию см. в гл. Автоматическая защита.

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 Guardant 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 функциями Guardant API при помощи **API Эксплорера** (файл **GAPIE_GUI_SE.exe**), интегрированного со справочным руководством (**GrdAPI.chm**)
- Изучите тестовые примеры программ на соответствующем языке программирования (см. каталог **"%Program Files%\Guardant\Guardant %#%\%Public Code%\Samples"**). В тестах содержатся примеры использования функций Guardant API
- Разработайте свою систему защиты, используя приобретенные знания и рекомендации, содержащиеся в приложении **Как повысить стойкость защиты при помощи API**

Какой метод выбрать?

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 автозащиту и защиту при помощи функций API
- Используйте аппаратные алгоритмы преобразования данных
- При автозащите 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 во 2-й части Руководства пользователя, глава **Повышение стойкости защиты. Рекомендации программисту**.

Глава 3
Подготовка электронного ключа к работе

Современные электронные ключи Guardant поддерживают платформы Windows и Linux. Подготовка ключа к работе на разных платформах имеет свои особенности, которые рассматриваются далее.

Установка ключа в среде Windows

Аппаратные ключи Guardant могут работать как через драйверы Guardant, так и без них (**HID-режим**, доступен для ключей моделей Guardant Sign/Time/Code и их сетевых версий).

Программные ключи Guardant SP и устаревшие аппаратные ключи Guardant Stealth III/Stealth II работают только через драйвер!

1. Работа ключа через драйвер Guardant

Важно!

1. Наличие установленных драйверов Guardant в Windows-системе обязательно для работы программных ключей SP и устаревших аппаратных ключей Guardant Stealth III/Stealth II и т. п. Поэтому драйверы Guardant должны входить в комплект поставки приложения, защищенного указанными ключами.

2. Новые модели ключей (Guardant Sign/Time/Code и их сетевые версии) могут работать под Windows как с драйверами, так и без установки драйверов (если ключ предварительно переведен разработчиком в HID-режим – см. описание HID-режима).

3. Драйверы Guardant универсальны для всех электронных ключей Guardant и операционных систем семейства Microsoft Windows одной разрядности.

4. Для 32- и 64-разрядных версий ОС Windows используются отдельные инсталляторы драйверов.

Установка драйверов

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

Чтобы переустановить драйверы (установить драйверы на другом компьютере), запустите файл **GrdDriversRU.msi** (или **Setup.exe**), который по умолчанию находится в каталоге:

ОС Windows	Каталог по умолчанию
32-разрядн.	%ProgramFiles%\Guardant\Guardant5\%PublicCode%\Drivers\x86\Windows\rus\
64-разрядн.	%ProgramFiles%\Guardant\Guardant5\%PublicCode%\Drivers\x64\Windows\rus\

После появления на экране мастера установки следуйте его указаниям. Инсталлятор произведет копирование и установку драйверов для всех типов электронных ключей Guardant независимо от интерфейса подключения.

Важно!

Во время установки драйверов ВСЕ приложения должны быть закрыты, в противном случае возможны ошибки разделения файлов.

Для Windows 7/2008/Vista/2003/XP также необходимо, чтобы пользователь, который работает с программой установки, обладал правами администратора системы.

Диагностика ключей Guardant

Для диагностики ключей Guardant воспользуйтесь апплетом **Драйверы Guardant** из **Панели управления** Windows.

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

Драйверы Guardant комплектуются утилитой диагностики **GrdDem32.exe**. Она помогает службе технической поддержки более оперативно проводить диагностику, анализируя сгенерированные утилитой отчеты.

Удаление драйверов

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 Guardant конечным пользователям

Разработчикам предоставляется несколько способов передачи драйверов конечным пользователям.

1. Распространение дистрибутива драйверов

Наиболее простым решением является распространение готового дистрибутива драйверов Guardant.

В состав ПО Guardant входят 32- и 64-разрядные версии драйверов, файлы которых имеют одинаковое название:

GrdDriversRU.msi	Дистрибутив x86 или x64 с русской версией драйверов Guardant
GrdDriversEN.msi	Дистрибутив x86 или x64 с английской версией драйверов Guardant

Необходимо просто включить в комплект поставки защищенного приложения русский и/или англиис вариант драйвера нужной разрядности.

2. Интеграция драйверов в дистрибутив защищенного приложения

Если предполагается устанавливать драйверы Guardant через инсталлятор защищенного приложения, то необходимо включить MSI-пакет с драйверами Guardant в свой комплект установки без изменений.

Драйверы Guardant работают с командной строкой через **msiexec**. Синтаксис команды должен быть следующим:

msiexec </обязат. параметр> <имя msi-пакета> [необязат. параметр]

К примеру, команды для скрытой установки и удаления драйверов должны выглядеть так:

msiexec /i GrdDriversRU.msi /quiet	Установить драйвер в «тихом» режиме
msiexec /x GrdDriversRU.msi /quiet	Удалить драйвер в «тихом» режиме

Другие опции Windows Installer см. с помощью команды **msiexec /?**

3. Использование драйверного Guardant API

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.

2. Работа ключа без драйвера. HID-режим

Аппаратные ключи, начиная с Guardant Sign, могут работать ОС семейства Windows без установки драйверов Guardant. Для этого ключи необходимо предварительно перевести в Human Interface Device (HID) режим при помощи утилиты GrdUtil.exe.

При подсоединении ключа в HID-режиме к USB-порту компьютера система распознает ключ как стандартное HID-совместимое устройство, после чего ключ сразу же готов к работе.

Режимы работы ключей Guardant в HID-режиме и со стандартным драйвером для пользователя ничем не отличаются.

Установка ключа в среде Linux

Аппаратные ключи, начиная с Guardant Sign, поддерживают работу в среде Linux, в том числе, в **HID-режиме**[*].

Также поддерживается работа защищенных Windows-приложений под WINE.

Для работы с ключами в ОС GNU/Linux необходимо добавить правило для штатного средства обработки **HotPlugging**. На большинстве современных дистрибутивов, использующих ядро 2.6.x, таким средством является **udev** (http://kernel.org/pub/linux/utils/kernel/hotplug/udev.html).

Правило для **udev** добавляется следующим образом:

Для ключей в драйверном режиме, и в случае использования файлов-устройств USB Device Filesystem

```
# cp etc/grdnt.udev /etc/udev/rules.d/95-grdnt.rules
```

Для ключей в HID-режиме

```
# cp etc/grdnt_hid.udev /etc/udev/rules.d/95-grdnt_hid.rules
```

Для записи в каталог **/etc/udev/rules.d** требуются права суперпользователя.

Указанные правила предписывают **udev** установить права на чтение и запись для файла-устройства, представляющего электронный ключ Guardant в системе. Это позволит обращаться к ключу с привилегиями любого пользователя системы.

Информацию по защите приложений под Linux см. во 2-й части Руководства пользователя.

[*] Для этого ключи необходимо предварительно перевести в HID-режим при помощи утилиты GrdUtil.exe

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 образом (маской) ключа:
  - 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 образа 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 образов
  - Ведение базы конечных пользователей
  - Хранение прошивок (всех фактов записи образа 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 моделями ключей Guardant.

Системные требования

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

- Операционные системы Windows 7/Vista/2003/XP
- Драйверы Microsoft Jet версии 4.0 или выше, необходимые для доступа к базе данных (как правило, эти драйверы уже установлены в операционной системе)
- 10 Мб 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 составляет его образ.

Образ ключа (образ) – содержимое памяти электронного ключа Guardant, сохраненное в базе данных GrdUtil или во внешнем файле *.nsd; совокупность полей памяти, их структуры и значений, представленная в удобной для восприятия форме в редакторе образа GrdUtil.exe.

Вся работа с данными ключа в GrdUtil.exe происходит на уровне образов: образ редактируется и сохраняется, измененные данные записываются из образа в память ключа.

Существует несколько разновидностей образов, различающихся по месту и типу хранения. Образ 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, прошивки или файла формата .nsd.

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

Прошивка – факт записи образа в память электронного ключа, а также совокупность данных записанного образа, 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 образа – файл формата *.nsd, в котором может быть сохранен образ ключа. Работа с образом, сохраненным в файле, имеет серьезные ограничения по сравнению с образами, хранящимися в базе данных GrdUtil.exe. Так, к примеру, не сохраняются факты прошивок файлом образа 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 образов, 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

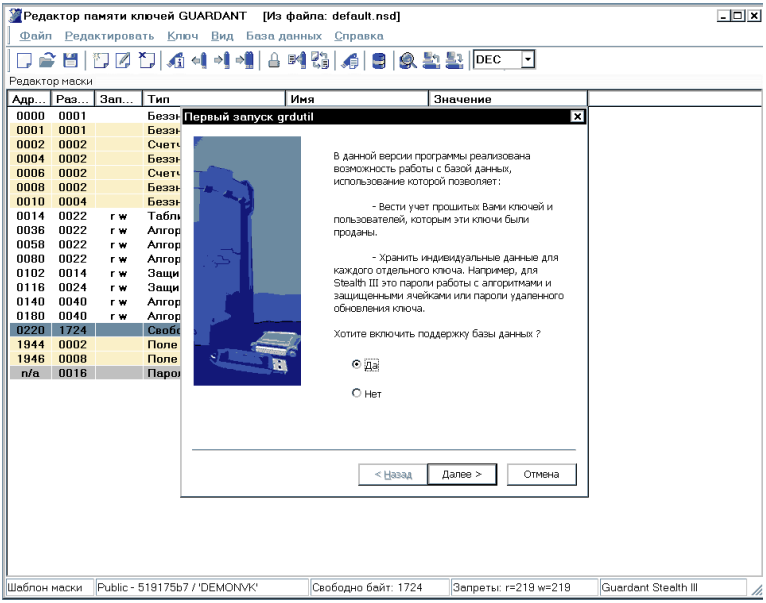
Важно!

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 %#%\%Public Code%\Bin"**
- Выберите элемент **Программирование ключей Guardant** в оболочке **Guardant Интегратор**
- Выберите элемент **Программирование ключей Guardant** из программной группы **Комплект разработчика Guardant %#% – "%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.

Важно!

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/ribbon-интерфейс
- Редактор образа
- Инструменты базы данных (в режиме базы данных)
- Прошивки (в режиме базы данных)
- Список ключей, подсоединенных к компьютеру
- Панель состояния

Меню и панель инструментов. Способы выполнения операций

Структура меню и панели инструментов GrdUtil.exe:

Раздел меню	Команда меню	Горячие клавиши	Панель инструментов	Описание команды
Файл	Создать образ	Ctrl+N		Создать образ
	Открыть образ	Ctrl+O		Загрузить образ из файла
	Сохранить образ	Ctrl+S		Сохранить образ в файле
	Сохранить как...	Ctrl+A	-	Сохранить образ в новом файле
	Выход	Alt+F4	-	Закрыть приложение
Ключ	Записать образ в ключ	Ctrl+W		Перенести данные из текущего образа в память ключа
	Пакетная запись ключей	-		Поочередно перенести данные из текущего образа в память нескольких ключей
	Информация о ключах	Ctrl+I		Получить информацию о ключе(-ах)
	Включить HID- режим	-		Включить/отключить режим работы без драйвера Guardant
	Единственный сессионный ключ GrdAPI	-		Вкл/откл защиту от запуска нескольких копий приложения, защищенных Guardant API
	Единственный сессион. ключ автозащиты	-		Вкл/откл защиту от запуска нескольких копий приложения, защищенных автозащитой

Раздел меню	Команда меню	Горячие клавиши	Панель инструментов	Описание команды
	Запретить изменение времени в ключе	-		Вкл (по умолч.)/выкл блокировку вызова функции GrdSetTime для ключей с таймером
	Шаблон Guardant SP			Создать неактивированный софтверный ключ
	Защищенный шаблон GrdSP			Создать неактивированный софтверный ключ, защищенный от записи
	Создать отладочный ключ Guardant SP			Создать и активировать SP-ключ для тестирования системы защиты
	Настроить параметры привязки			Задать параметры привязки к комплектующим компьютера
	Обновление ключа	Ctrl+M		Выполнить удаленное программирование памяти ключа
	Завершить обновление ключа	Ctrl+E		Завершить удаленное программирование памяти ключа по факту прошивки
Образ ключа	Добавить поле	Ins		Создать поле нужного типа
	Переименовать			Переименовать поле
	Дамп поля			Показать содержимое поля
	Свойства поля	Ctrl+P		Редактировать поле
	Удалить поле	Del		Удалить выбранное поле
	Установить запреты	Ctrl+L		Установить аппаратные запреты на чтение и/или запись участка памяти ключа
	Дамп образа			Показать содержимое образа в шестнадцатеричном редакторе
	Конвертировать образ	Alt+C	-	Перенести данные в образ для другого типа ключей
	Исходный код образа			Сохранить образ в виде исходного кода *.cpp
База данных	Включить / выключить базу данных	Ctrl+D		Перейти в режим/выйти из режима базы данных GrdUtil.exe
	Управление клиентами			Открыть диалог **Клиенты**
	Управление образами			Открыть диалог **Образы**
	Поиск записанных образов	Alt+L		Выполнить поиск прошивок по заданным параметрам поиска
	Найти последний образ, записанный в ключ	-		Загрузить в Редактор образа последнюю прошивку для текущего подсоединенного ключа

Раздел меню	Команда меню	Горячие клавиши	Панель инструментов	Описание команды
	Найти образ по числу-вопросу	Alt+F		Загрузить в Редактор образа прошивку, сведения о которой содержатся в запросе на обновление ключа
	Сохранить образ в БД	Alt+W		Сохранить редактируемую маску в базе данных GrdUtil.exe
	Сохранить образ в БД как..	Alt+S		Сохранить редактируемую маску в базе данных под новым именем и/или другой версией
	Настройка базы данных	Alt+D	-	Запустить мастер настройки базы данных
	Конвертация базы данных	-	-	Конвертировать БД после изменения ее формата
Разное	Создать отчет алгоритма	Ctrl+Q		Получить ответы аппаратного алгоритма
	Шифрование данных	Ctrl+T		Кодировать/декодировать данные выбранным аппаратным алгоритмом
	Обозреватель Guardant API	Ctrl+F		Проверить корректность выполнения функций Guardant API с заданными параметрами
	Справка по Guardant API	-		Вызвать справочный файл по функциям GrdAPI
Вид	Список ключей			Показать/скрыть окно **Ключи**
	Панель инструментов	Alt+B	-	Показать/скрыть панель инструментов
	Строка статуса	Alt+V	-	Показать/скрыть строку статуса
	Система счисления: BIN, DEC, HEX	Ctrl+2, Ctrl+0, Ctrl+6	-	Выбрать систему счисления: двоичную, десятичную и шестнадцатеричную соответственно
	Ribbon интерфейс	-	-	Выбрать отображение ленточного интерфейса или панели инструментов
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:

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

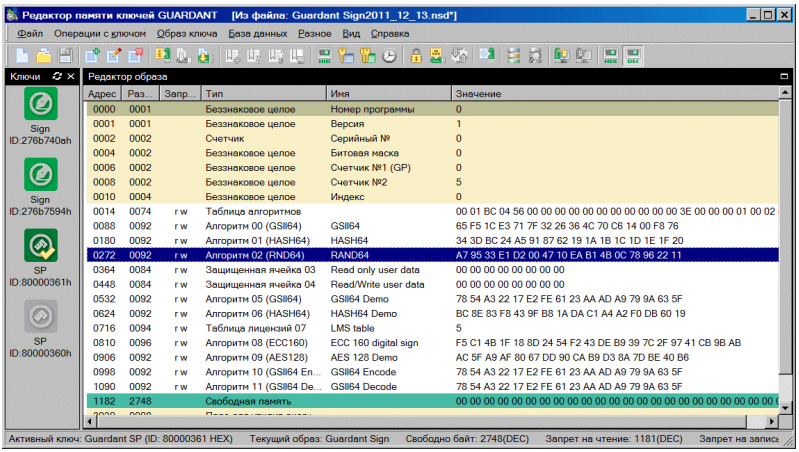
Выбор нужного поля памяти и просмотр его свойств происходит с помощью:

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

Редактор образа

В Редакторе образа происходит вся работа по программированию ключа, в частности:

- Загрузка образа в Редактор из базы данных или файла
- Редактирование структуры и содержимого образа
- Сохранение текущего образа в базе данных или файле
- Запись данных образа в подсоединенный ключ
- Процедура обновления ключа на основе загруженного образа



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

Редактор образа 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

Слева от Редактора образа находится вспомогательное окно **Ключи**, позволяющее оперативно выбрать активный*** электронный ключ из нескольких подсоединенных к компьютеру.

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 TRU

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 образа:

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

** Более подробно каждый тип поля будет рассмотрен далее

*** Т. е. ключ, с которым будут выполняться те или иные операции – прошивка, работа с аппаратными алгоритмами и проч.

- Состояние ***Образ***:
  если образ загружен из списка прошивок
- Состояние ***Шаблон образа***:
  если образ загружен 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

Образ ключа

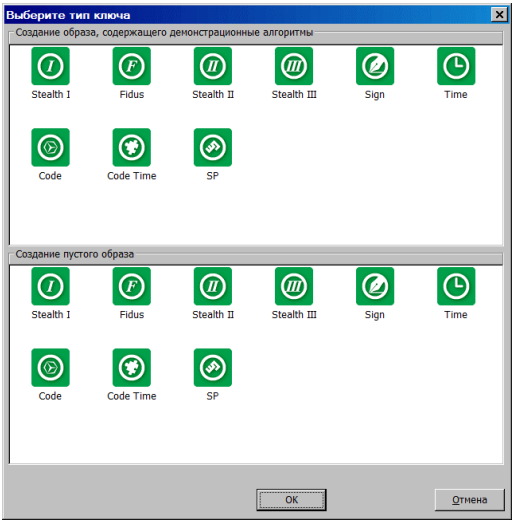
Совокупность всех полей памяти и их содержимого составляют так называемый ***образ*** – структуру, маску памяти ключа.

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

Образы различных ключей отличаются друг от друга. Поэтому важно следить, чтобы в ключ записывался подходящий по типу образ.

Создание образа

Чтобы создать новый образ, выполните команду меню **Файл | Создать образ**. В появившемся диалоге выберите тип образа для работы с определенным типом ключей:



При первом запуске утилиты появляется диалог создания нового образа.

Тип создаваемого образа должен соответствовать модели электронного ключа, который на тот момент подсоединен к порту.

Важно!

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

После нажатия кнопки **[OK]** новый образ загружается в **Редактор образа** и ему присваивается имя формата ***модель_ключа_год. месяц.число.nsd***, которое потом можно изменить. Имя образа отображается в заголовке главного окна утилиты.

Сохранение образа

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

Чтобы сохранить образ в базе данных GrdUtil.exe, выполните команду меню **База данных | Сохранить образ в БД**.

Чтобы сохранить образ в базе данных GrdUtil.exe под другим именем, выполните команду меню **База данных | Сохранить образ в БД как...**

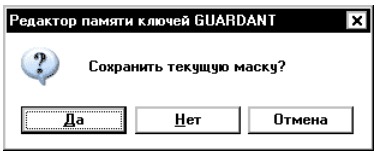
Режим базы данных является предпочтительным режимом работы с GrdUtil.exe. Подробное описание см. в разделе **База данных GrdUtil.exe**.

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

При работе в режиме базы данных образ, записываемый в ключ, автоматически сохраняется в базе в виде ***прошивки***. В отличие от шаблона образа 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 образа в файле

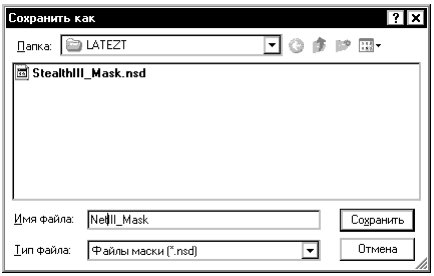
При завершении работы с текущим образом GrdUtil.exe запрашивает подтверждение на его сохранение, если изменения не были сохранены:



Внешним признаком того, что образ был изменен, и перед закрытием его необходимо сохранить, служит звездочка после названия образа в «шапке» главного окна GrdUtil.exe:

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

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



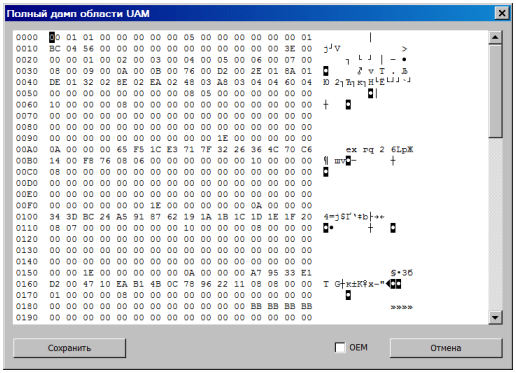
После операции сохранения в заголовке главного окна GrdUtil.exe будет отображаться имя сохраненного образа, а также указание на то, что образ сохранен в файле:

Получение содержимого образа в виде дампа

Редактор памяти дает возможность получать и редактировать содержимое, как полного образа ключа, так и его отдельных полей.

Для получения содержимого образа в виде дампа загрузите в Редактор нужный образ и выполните команду **Образ ключа | Показать дамп образа**.

Диалоговое окно **Полный дамп области UAM** представляет собой 16-ричный редактор и позволяет просматривать, редактировать и сохранять в файле содержимое образа ключа:



Сохранение образа в виде исходного кода

Редактор позволяет сохранять образ ключа в виде исходного кода на C++. Это облегчает разработку альтернативных утилит программирования ключей.

Чтобы экспортировать текущий образ ключа в исходный код, выполните команду меню **Образ ключа | Исходный код записи**.

На экране появится системный диалог сохранения файла (имя файла по умолчанию **DongleImageWrite.cpp**), который содержит массив содержимого образа и процедуру его записи в ключ из собственного приложения.

Загрузка образа

Во время запуска GrdUtil.exe происходит автоматическая загрузка образа, который редактировался во время последнего сеанса работы с утилитой.

Если же образ из последнего сеанса не сохранялся, то при старте утилита выводит диалог создания нового образа.

1. Загрузка шаблона образа из базы данных GrdUtil.exe

Чтобы загрузить маску из базы данных в Редактор образа, выполните команду меню **База данных | Управление образами**. Далее выберите и откройте нужный образ в появившемся диалоге.

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

2. Загрузка образа 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 образа.

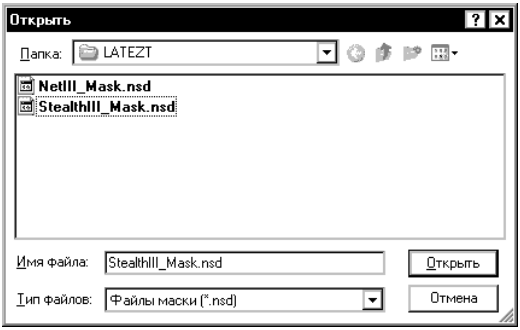
Чтобы загрузить прошивку в Редактор образа, используйте один из следующих вариантов выполнения операции:

- 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 образа из файла

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



Выбранный файл будет загружен в Редактор образа.

Поля памяти

Для наглядности и удобства работы память ключа в утилите GrdUtil.exe представлена в виде отдельных полей. Каждое поле – это 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 составляют его образ.

По своему назначению память ключа логически делится на несколько областей (перечислены 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 во 2-й части Руководства пользователя, раздел **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 образа.

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).

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

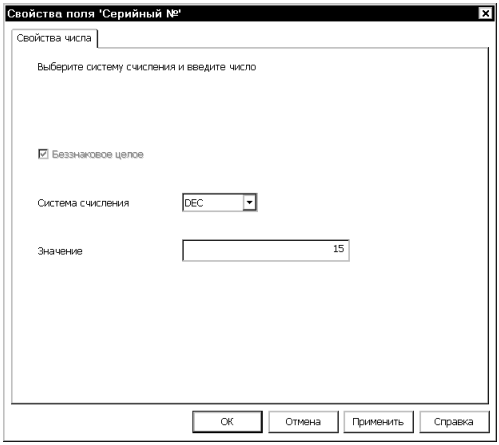
Таблица характеристик полей общего назначения:

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	Устаревшая технология. Теперь для ограничения числа запусков приложения необходимо использовать счетчики аппаратных алгоритмов	-	-
0008	Сетевой ресурс (счетчик №2), беззнаковое целое	0 - 65535	Отображение реального ресурса лицензий сетевых ключей (в современных сетевых ключах сетевой ресурс хранится в **таблице лицензий**, а в счетчике №2 только отображается)	/GN2, /GN3 /GN3S	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.

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 и проч.

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

Важно!

Ключи Guardant всех моделей, кроме Guardant Code, поступающие в продажу, по умолчанию содержат в области полей свободного назначения несколько дескрипторов стандартных аппаратных алгоритмов и защищенных ячеек.

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	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		В новых ключах произвольный размер. В старых: 1 – 255 Б + служебные поля	Дескриптор защищенной ячейки	Обязательные запреты на запись и чтение	GrdPI_Activate, GrdPI_Deactivate, GrdPI_Read, GrdPI_Update
Таблица лицензий	Guardant Sign Net/ Time Net/ Net III	Длина каждого модуля=1 или 2 Б Максимум - 254 байта +служебные поля	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
	Guardant Net II / Net	Длина каждого модуля=1 или 2 Б Максимум – 127 модулей	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 записи образа в память ключа	Запрет на запись (при необходимости)	GrdRead, GrdWrite
Дамп		Произвольный размер	Двоичный дамп	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 современных сетевых ключей. 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.

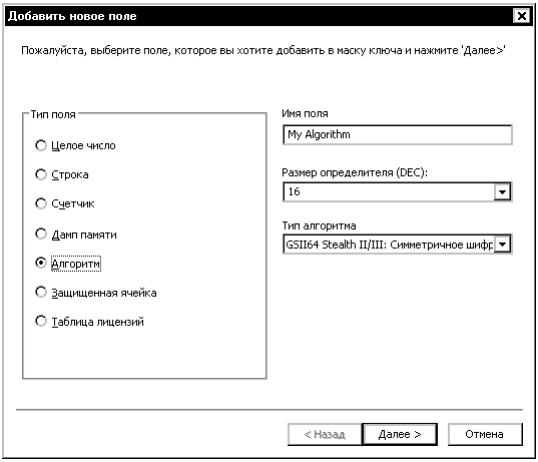
Создание полей

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

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

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

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 образа 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

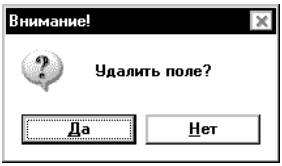
Чтобы изменить содержимое или отредактировать свойства поля, выделите поле в текущей маске и выполните команду **Редактировать | Свойства поля**.

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:

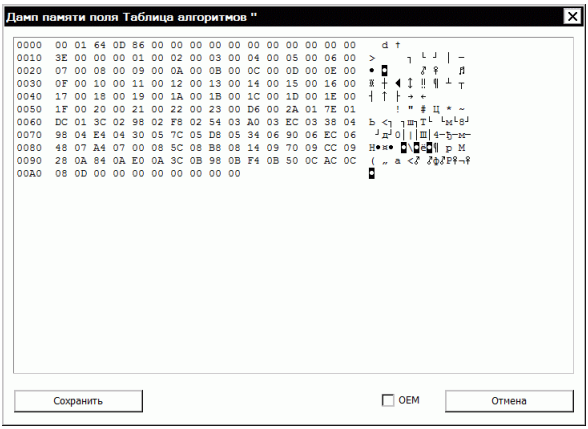


Важно!

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

Получение содержимого поля в виде дампа

Чтобы просмотреть и отредактировать дамп поля, а также сохранить его в файле, выделите нужное поле в маске, вызовите контекстное меню, нажав правую кнопку мыши и выберите команду **Показать дамп поля**:



База данных GrdUtil.exe

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

Важно!

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

Каждый факт записи шаблона образа 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 образ, который 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.

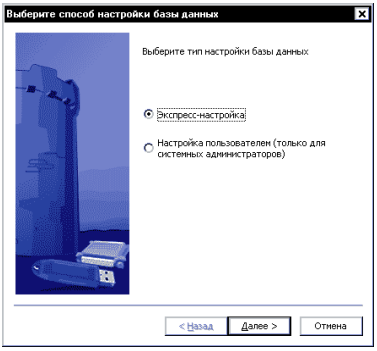
Настройка базы данных

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

Важно!

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.



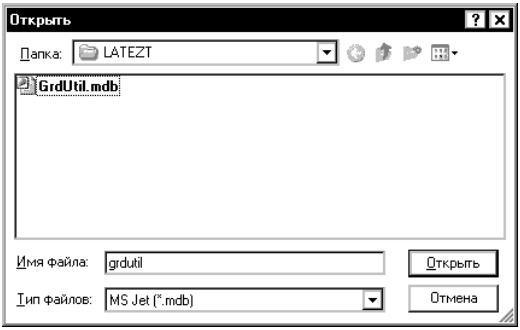
1. На первой странице Мастера выберите тип настройки базы данных **Экспресс-настройка**:

Важно!

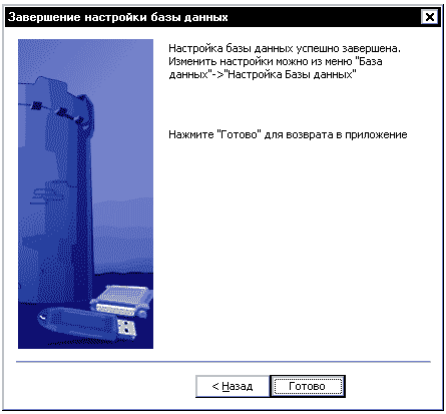
1. Вариант **Пользовательская настройка** в данном руководстве подробно не рассматривается. Он необходим только в ситуации, когда предполагается использование базы данных с какими-либо специфичными настройками, например, при использовании Microsoft SQL Server. В таком случае настройка базы данных должна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

2. В состав ПО Guardant входят SQL-скрипты для создания БД (см. **grdutil_access.sql** и **grdutil_ms_sql_server.sql** в поддиректории **Doc** каталога установки SDK).

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 %#%\%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

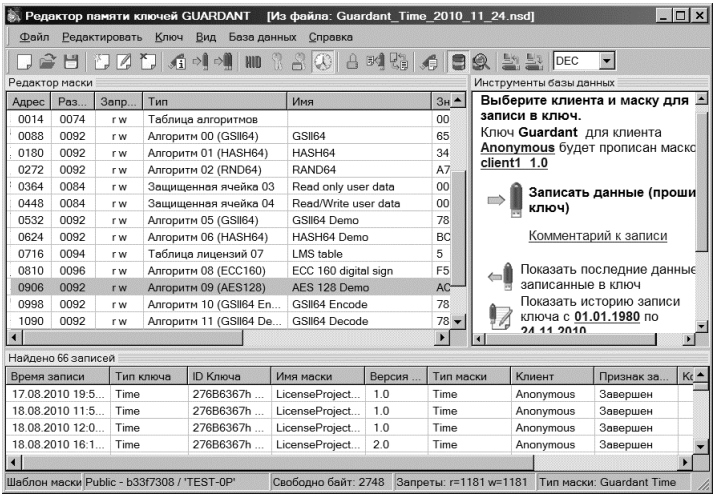
Работа в сетевом режиме является побочной возможностью базы GrdUtil.exe и не тестировалась досконально

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 аналогично (см. выше).

Включение базы данных

Чтобы включить режим базы данных, выполните команду **База данных | Режим работы с БД**.

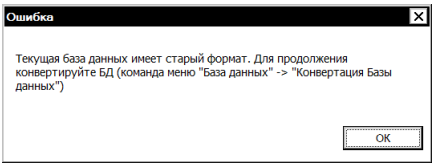
После выполнения команды к окну Редактора образа добавятся окна **Инструменты базы данных** (справа) и **Прошивки** (внизу):



Конвертация базы данных

Формат базы данных меняется от версии к версии GrdUtil, и в некоторых случаях после обновления комплекта разработчика может требоваться конвертация БД.

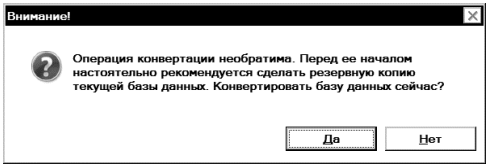
Признаком необходимости конвертации служит сообщение, которое выдается при запуске утилиты программирования:



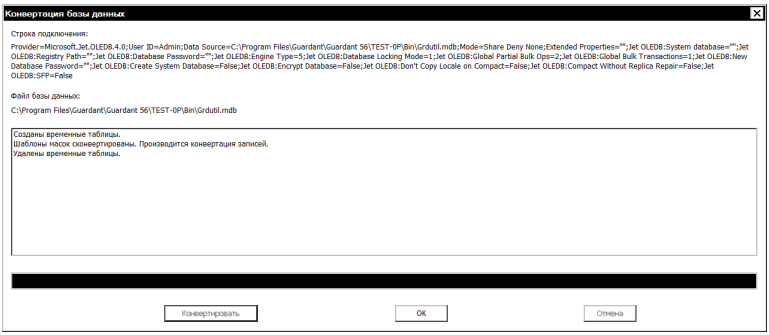
По нажатию на кнопку **OK** в диалоге сообщения происходит возврат в главное окно утилиты.

Чтобы конвертировать базу данных в актуальный формат, *сохраните копию текущей БД* и выполните команду меню **База данных | Включить базу данных**.

На экране появится предупреждение о необратимости операции:



По нажатию на кнопку **Да** появляется диалог конвертации:



Чтобы начать процесс конвертации, нажмите на кнопку **Конвертировать** в нижней части диалога. По успешном завершении конвертации будет выдано соответствующее сообщение.

Теперь можно включить базу данных и приступить к работе.

Важно!

Для конвертации базы данных GrdUtil из командной строки служит команда: **[путь]GrdUtil.exe -dbConvert**

Инструменты базы данных

Окно **Инструменты базы данных** состоит из набора кнопок-пиктограмм, посредством которых происходит управление базой шаблонов, прошивок и конечных пользователей:

Элемент интерфейса	Назначение
Запись в ключ	Выполнить команду меню **Ключ|Запись в ключ**, Ctrl+W
Текущее состояние ключа	Загрузить в Редактор образа последнюю прошивку для текущего ключа, подсоединенного к порту
История записи ключа	Вывести в окне **Прошивки** список прошивок за указанный период для ключа, подсоединенного к порту
Управление образами	Вызвать диалог **Образы** для работы с базой шаблонов образов

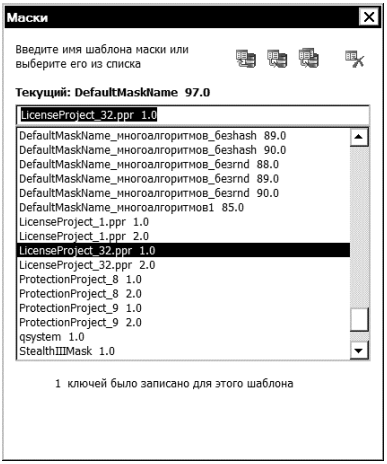
Элемент интерфейса	Назначение
Управление клиентами	Вызвать диалог **Клиенты** для работы базой конечных пользователей защищенного приложения
Поиск образов	Вызвать диалог **Поиск** для задания параметров поиска прошивок, хранящихся в базе данных

Кроме того, некоторые кнопки дублируются гиперссылками в верхней части окна, упрощающими работу с базой данных.

Образы

Диалог **Образы** вызывается по нажатию кнопки **Управление образами** и служит для работы с базой шаблонов образов, которые используются для программирования ключей.

Элементы управления диалога позволяют сохранять шаблоны в базе данных, удалять шаблоны и загружать их в Редактор образа:

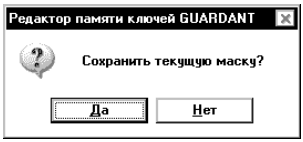


Элементы управления диалога **Образы**:

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

Сохранение шаблона образа в базе данных

При завершении работы с текущим образом GrdUtil.exe запрашивает подтверждение на его сохранение, если изменения не были сохранены:

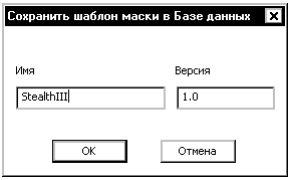


Внешним признаком того, что образ была изменен, и перед закрытием его необходимо сохранить, служит звездочка после названия образа в заголовке главного окна GrdUtil.exe:

Чтобы сохранить редактируемую маску в базе данных, выполните команду меню **База данных | Сохранить образ в БД**.

1. Если образ ранее уже сохранялся в базе данных, то сохранение образа происходит сразу после выполнения вышеуказанной команды, без выдачи диалога.

2. Если образ ранее не сохранялся в базе данных, то на экране появится диалоговое окно **Сохранить образ в базе данных**, содержащее поля ввода **Имя** и **Версия**. Первое служит для задания имени (по умолчанию *модель ключа*), второе – для задания версии образа (по умолчанию *1.0*).



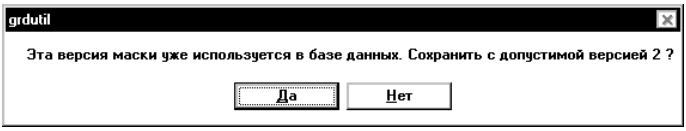
По нажатию кнопки **[OK]** происходит сохранение образа в базе данных. Сохраненный образ появляется в списке шаблонов в диалоге **Образы**.

Имя и версия сохраненного образа, а также указание на то, что образ сохранен в базе данных, отображаются в заголовке главного окна GrdUtil.exe:

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

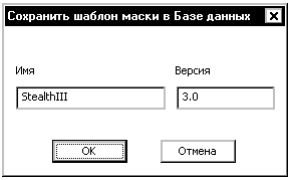
Сохранение шаблона образа в базе данных под другим именем

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 образа), второе – для задания версии образа (по умолчанию содержит увеличенное на 1.0 текущее значение версии образа).



По нажатию кнопки **[OK]** происходит сохранение образа в базе данных. Сохраненный образ появляется в списке шаблонов в окне вкладки **Образы**.

Имя и версия сохраненного образа, а также указание на то, что образ сохранен в базе данных, отображаются в заголовке главного окна GrdUtil.exe:

Загрузка шаблона образа из базы данных

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

Выбранный образ будет загружен в Редактор. В заголовке главного окна GrdUtil.exe отобразятся имя и версия образа, а также указание на то, что образ загружен в Редактор из базы данных:

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

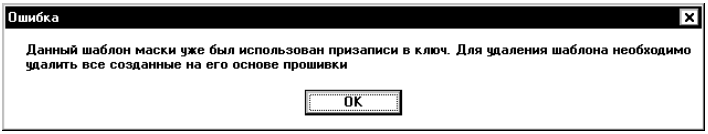
После загрузки образа Редактор показывает список полей и значения, записанные в них на момент последней модификации образа.

Удаление шаблона образа из базы данных

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

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

В этом случае шаблон образа 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:

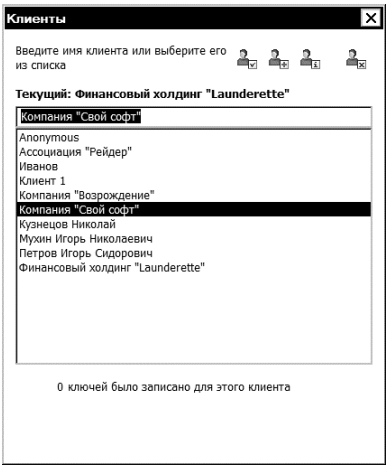
После нажатия кнопки **[Да]** в диалоге подтверждения шаблон образа будет удален из базы данных.

2. Если удаляемый шаблон образа не использовался для программирования ключей, то маску можно удалить сразу же, после вывода

на экран формального диалога подтверждения на удаление образа (см. выше).

Клиенты

Диалог **Клиенты** вызывается по нажатию кнопки **Управление записями клиентов** и служит для работы с базой конечных пользователей защищенного приложения. Элементы управления диалога позволяют добавлять и удалять конечных пользователей из списка, а также определять пользователя, на которого будет зарегистрирована прошивка:



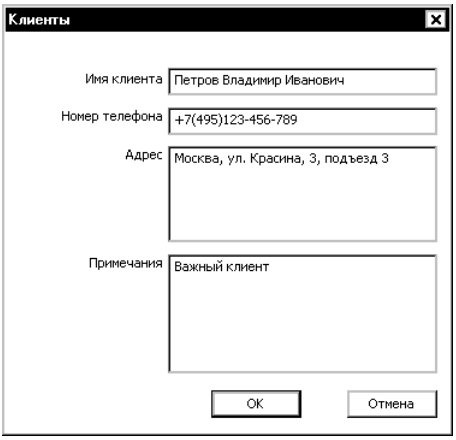
Элементы управления диалога **Клиенты**:

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 диалог **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 диалога **Клиенты**.

Важно!

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 меткой ***текущий***.

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

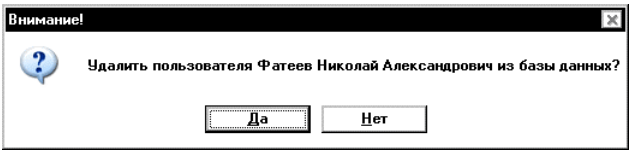
Удаление клиента из базы данных

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.

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

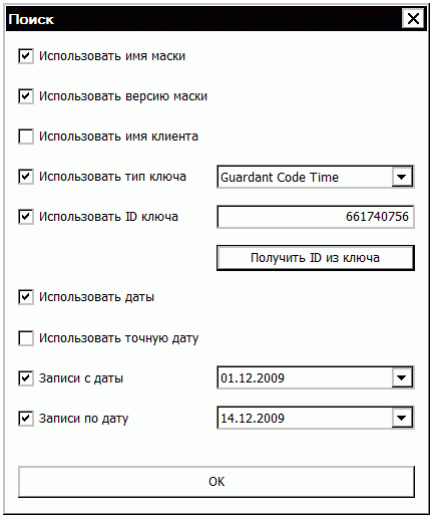


После нажатия кнопки **[Да]** в диалоге подтверждения клиент будет удален из базы данных.

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

Диалог **Поиск** вызывается по нажатию кнопки **Поиск прошивок ключей** и служит для задания параметров поиска прошивок, хранящихся в базе данных. Диалог 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***.

Т. о., к примеру, всегда можно узнать в какой именно ключ (ID, тип) записан тот или иной образ (имя, версия), и у какого конечного пользователя этот ключ должен находиться.

Важно!

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, зарегистрированные на данного пользователя.

Элементы управления вкладки **Поиск**:

Элемент интерфейса	Назначение
Флаг **Использовать имя образа**	Вывести в результатах поиска все прошивки для текущего образа. При поиске прошивок учитывается имя образа
Флаг **Использовать версию образа**	Вывести в результатах поиска все прошивки для текущего образа. При поиске прошивок учитывается версия образа
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 Sign/ Time /Code /CodeTime / Stealth III / Stealth II / Stealth / 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

Вывод результатов поиска

Чтобы найти прошивки согласно заданным критериям поиска, выполните команду меню **База данных | Поиск прошивок ключей**.

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

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

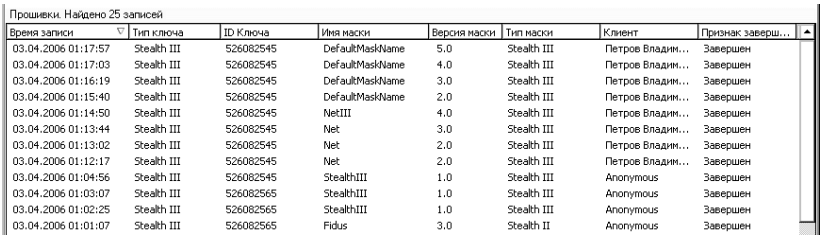
Сброс результатов поиска

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 образа, записанных в ключ, а также сам факт записи образа 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 завершенности
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 образа 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 образа	Название шаблона образа, записанного в ключ
Версия образа	Версия шаблона образа, которая была записана в ключ
Тип образа	Тип образа, данные из которого 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 образа
- Сортировать список прошивок
- Перерегистрировать прошивки на другого пользователя
- Удалять прошивки

Загрузка прошивки в Редактор образа

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 образа.

Чтобы загрузить прошивку в Редактор образа, нажмите на кнопку-пиктограмму **Поиск прошивок ключей** и 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 образа в нижнем левом углу статусной строки переключится в положение «образ», а в заголовке главного окна GrdUtil.exe появится название загруженного образа и признак того, что он взят из базы данных:

Сортировка списка прошивок

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

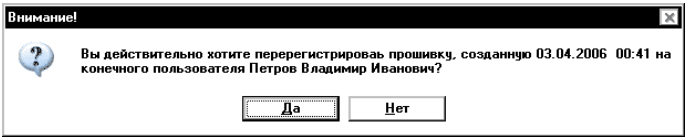
Сортировка допустима по любому из столбцов таблицы и может выполняться по убыванию или возрастанию значений параметров.

Направление сортировки указывается стрелкой-индикатором в заголовке выбранного столбца.

Перерегистрация прошивки на другого конечного пользователя

Чтобы перерегистрировать прошивку на другого пользователя, выберите его имя в диалоге **Клиенты** (кнопка **Управление записями клиентов** в **Инструменты базы данных**). 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:

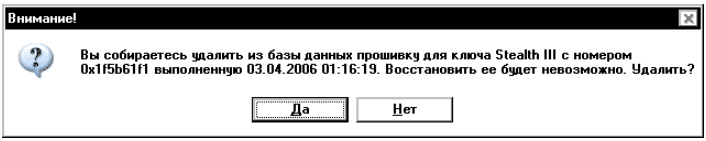


По нажатию кнопки **[Да]** прошивка будет зарегистрирована на выбранного конечного пользователя. Факт перерегистрации отразится в списке прошивок.

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.

HID-режим

Все аппаратные ключи Guardant, начиная с Guardant Sign, могут работать без установки драйверов. Для этого ключи необходимо предварительно перевести в Human Interface Device (HID) режим.

При подсоединении ключа в HID-режиме к USB-порту компьютера система распознает ключ как стандартное HID-совместимое устройство, и ключ сразу же готов к работе.

Работа с ключами Guardant в HID-режиме и со стандартным драйвером не имеет существенных отличий.

Важно!

Если ключ переведен в HID-режим, то при преобразовании больших блоков данных может наблюдаться снижение скорости работы ключа.

Чтобы перевести ключ в HID-режим:

1. Загрузите прошивку из списка прошивок (или создайте новый файл образа) и выполните команду меню **Ключ | Включить HID-режим работы ключа**.

2. Запишите маску в ключ: выполните команду **Ключ |Запись в ключ**. При этом в меню напротив команды будет установлен флаг, а кнопка **[HID]** на панели инструментов перейдет в «нажатое» положение.

3. Переподсоедините ключ и протестируйте его работу в HID-режиме.

Чтобы вернуть ключ из HID-режима в режим работы с драйвером, повторите действия, указанные выше. Переподсоедините ключ и протестируйте его работу в драйверном режиме.

Защита от запуска нескольких копий приложения

В некоторых случаях у разработчика возникает необходимость препятствовать запуску более чем одной копии приложения одновременно. Наиболее яркий пример такой ситуации – работа приложения в терминальной сессии, когда возможен бесконтрольный запуск большого числа копий приложения, защищенного на локальный ключ.

Важно!

Только для локальных ключей!

Сетевые ключи не совместимы с данной технологией. Для достижения аналогичного эффекта используйте распределение сетевых лицензий по копиям приложения.

Локальные ключи Guardant обладают аппаратной защитой от запуска нескольких копий приложения. Ее принцип основан на том, что электронный ключ можно перевести в режим работы только с одним сессионным ключом одновременно[*].

В зависимости от того как защищается приложение, можно устанавливать следующие режимы управления сессионными ключами:

Способ защиты	Режим управления сессионными ключами
Автозащита	Единственный сессионный ключ для автоматической защиты
Guardant API	Единственный сессионный ключ для Guardant API
Комбинация автозащиты и GrdAPI	Любой из вышеперечисленных режимов, либо их комбинация

[*] Более подробно см. во 2-й части Руководства.

Чтобы включить защиту от запуска нескольких копий приложения:

1. Загрузите нужную прошивку из базы данных (или создайте новый файл образа) и выполните одну или обе из следующих команд меню (в зависимости от выбранной схемы защиты):
   - **Ключ | Единственный сессионный ключ для Guardant API**
   - **Ключ | Единственный сессионный ключ для автозащиты**

2. Запишите маску в ключ: выполните команду **Ключ |Запись в ключ**.

3. Защитите приложение при помощи Guardant API и/ или автозащиты. Протестируйте работу защищенного приложения.

Запрограммированный таким образом ключ позволяет одновременную работу не более одной копии приложения. После запуска 2-й копии приложения 1-я копия становится неработоспособной.

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.

Важно!

С помощью специальных операций Guardant API можно получить доступ к части содержимого защищенных ячеек – особой разновидности полей памяти, защищенных запретами.

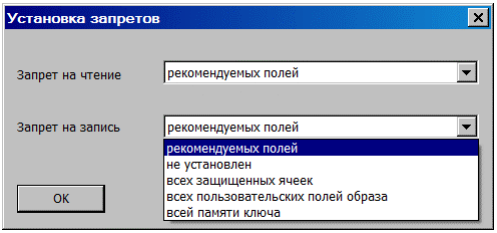
Запреты можно устанавливать на непрерывную область памяти свободного назначения, начиная с младших адресов (т. е. с адреса 14 UAM до области полей специальных операций).

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

Важно!

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**. Появившийся диалог **Установка запретов** состоит из разворачивающихся списков (**Запрет на чтение** и **Запрет на запись**):



Диалог позволяет выбрать следующие варианты установки запретов:

Запрет на чтение / запись	**Пояснение**
Не установлен	Запреты снимаются со всей памяти
Всех защищенных ячеек	Защищаются только ячейки и алгоритмы
Всех пользовательских полей образа	Защищаются все поля, созданные пользователем
Всей памяти ключа	Вся доступная память защищается запретами

Запреты можно выставлять как отдельно (только на чтение, только на запись), так и комбинировать их (и на чтение, и на запись).

Виды и обозначения аппаратных запретов в Редакторе образа:

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.

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

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

Запись в ключ

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

В процессе записи данных GrdUtil.exe автоматически переносит в ключ всю информацию из образа, загруженного 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 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 образа пишется в ключ (операция GrdWrite)
- На указанную область памяти устанавливаются аппаратные запреты (операция GrdProtect)

Соответствие типов образа и ключа

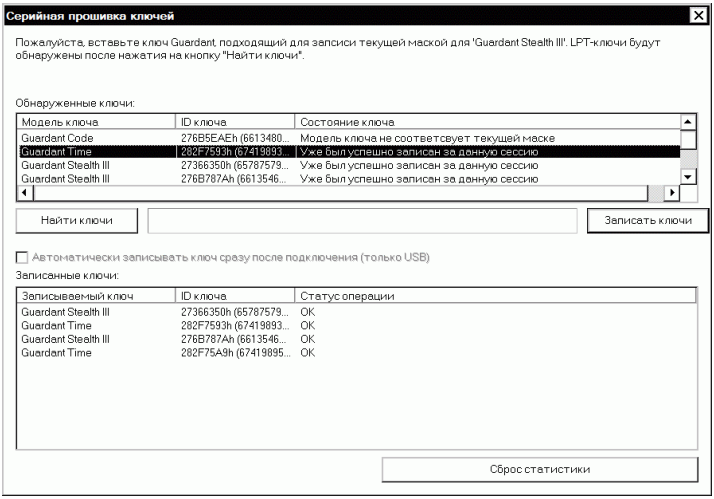
Образы для ключей разных типов отличаются друг от друга. Поэтому важно следить, чтобы в ключ записывался подходящий по типу образ (т. е. образ Guardant Sign/ Sign Net должен прошиваться в ключ Guardant Sign/ Sign Net). Это правило следует соблюдать в абсолютном большинстве ситуаций.

Однако в некоторых случаях может возникнуть необходимость записывать в ключ маску несоответствующего типа.

При этом следует иметь в виду, что после записи в ключ неподходящего образа, могут появиться серьезные ограничения и трудности при работе с таким ключом.

Пакетный режим записи

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



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 Руководство пользователя, Часть 2, глава **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
- Временные зависимости (только для ключей типа Time)
- Определитель алгоритма (кроме алгоритма ECC160)
- Ключ ECC160 (только для алгоритма ECC160)

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

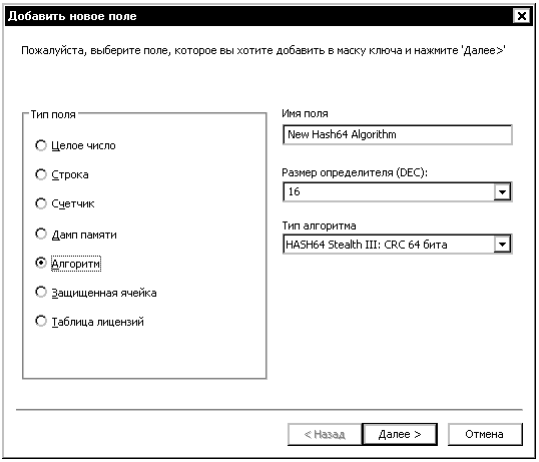
Создание алгоритма

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 образа ключа:

Тип образа	Тип алгоритма
Guardant Time/ Time Net	1. Симметричное шифрование: GSII64, в т. ч. особые режимы алгоритма только для кодирования и декодирования 2. Симметричное шифрование: AES128, в т. ч. особые режимы алгоритма только для кодирования и декодирования 3. Выработка ЭЦП: ECC160 4. Выработка хэш-функции: HASH64 5. Выработка хэш-функции: SHA256 6. Генерация случайных чисел: RND64
Guardant Sign/ Sign Net	
Guardant Code/ Code Time	1. Симметричное шифрование: AES128, в т. ч. особые режимы алгоритма только для кодирования и декодирования 2. Выработка ЭЦП: ECC160 3. Выработка хэш-функции: SHA256
Guardant Stealth III / Net III	1. Симметричное шифрование: GSII64 2. Выработка хэш-функции: HASH64 3. Генерация случайных чисел: RND64
Guardant Stealth II / Net II	1. Симметричное шифрование: GSII64 2. Однонаправленный алгоритм Stealth и его разновидности: Fast, Random, AutoProtect
Guardant Stealth / Net	Однонаправленный алгоритм Stealth и его разновидности: Fast, Random, AutoProtect
Guardant Fidus	-

Размер определителя

Определитель – основная составляющая дескриптора аппаратного алгоритма, которая задает конкретный вид функции преобразования. Определители алгоритмов в современных ключах Guardant имеют фиксированную четную длину, зависящую от типа алгоритма. 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:

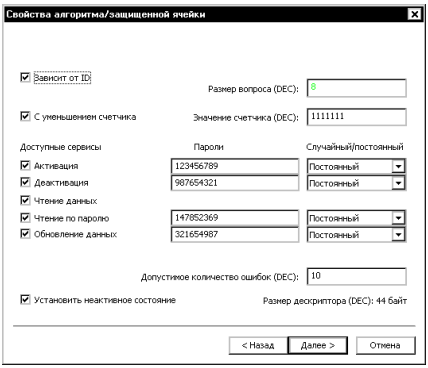
Тип алгоритма	Размер определителя, байтов
Симметричное шифрование: AES128	16
Выработка ЭЦП: ECC160	20
Симметричное шифрование: GSII64	16 или 32
Выработка хэш-функции: SHA256	-
Выработка хэш-функции: HASH64	16 или 32
Генерация случайных чисел: RND64	16 или 32
Однонаправленный Stealth – устаревш.	4 – 200 (оптимально - 32)

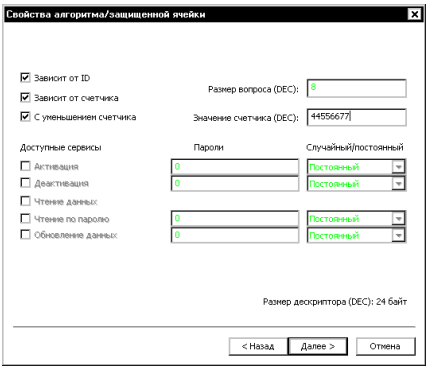
***

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.





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.).

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

Комбинация флагов	Режим работы алгоритма	Описание режима
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

Важно!

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 AES128	16
Асимметричный ECC160	20
Однонаправленный SHA256	32
Симметричный GSII64	8
Однонаправленные HASH64, RND64	8
Однонаправленный Stealth*	4 – 255 (желательно использовать четные числа)

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

Сервисы аппаратных алгоритмов

В группе **Доступные сервисы** расположены флаги, управляющие сервисами алгоритмов. Использование сервисов значительно расширяет функциональность аппаратных алгоритмов.

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.

Такие возможности стали доступны благодаря ***технологии защищенных ячеек***, частным случаем которых являются аппаратные алгоритмы.

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 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 образа 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 включить 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**.

* Если она уже не появилась после включения другого сервиса

Установка флага **Чтение по паролю** включает одноименный сервис, при этом становятся доступными опции (подробнее см. далее):

- 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 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 см. далее):

- 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 попыток 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.

* Если она уже не появилась после включения другого сервиса

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 образа, автоматически задается случайный пароль.

Важно!

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 образа 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.

Временные зависимости алгоритма. Технология Time

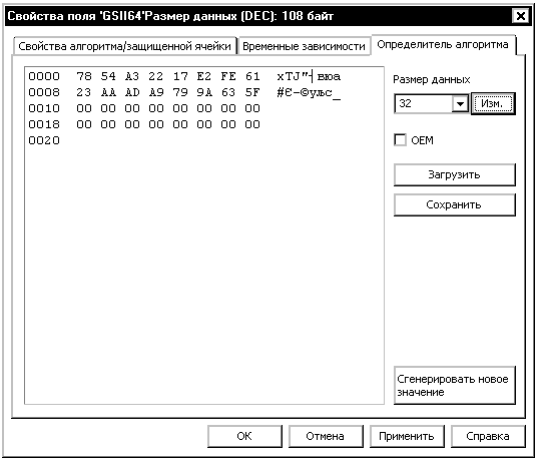
В ключах с часами реального времени за страницей **Свойства алгоритма** следует страница **Временные зависимости**, на которой расположены элементы, позволяющие ограничивать время работы защищенного приложения.

Смысл технологии ограничения времени заключается в том, что работоспособность алгоритма зависит от таймера (RTC), встроенного в ключи Guardant Time/ Time Net / Code Time.

Информацию по диалогу **Временные зависимости** см. в разделе **Ограничение астрономического времени работы приложения**.

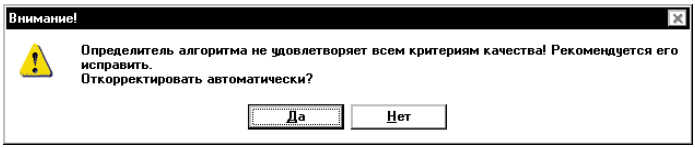
Редактирование определителя алгоритма

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

Элемент интерфейса	Назначение
Кнопка **[Сгенерировать новое значение]**	Автоматически создать новый определитель алгоритма
Дополнительные элементы управления для уже созданного алгоритма	
Список **Размер данных**	Изменить размер определителя созданного алгоритма
Кнопка **[Изменить]**	Отразить изменение размера определителя в окне редактора

Изменение размера определителя

GrdUtil.exe позволяет скорректировать размер определителя уже созданного алгоритма. Для этого выделите нужный алгоритм в списке полей, выполните команду меню **Редактировать | Свойства поля** и в появившемся диалоге перейдите на вкладку **Определитель алгоритма**.

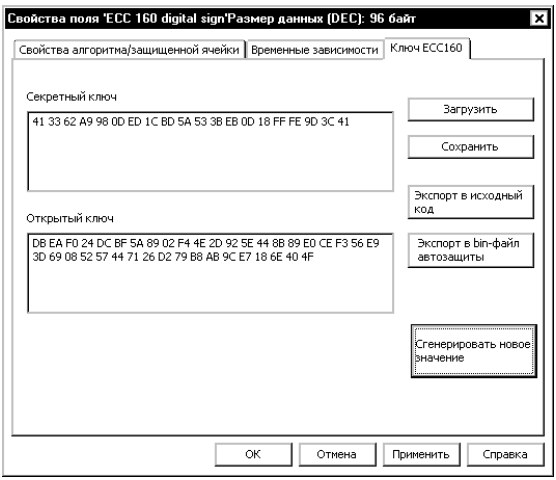
Для изменения размера определителя служит разворачивающийся список **Размер данных** в левой части диалога. После выбора нужного значения нажмите на кнопку **[Изменить]** рядом со списком и запишите маску в ключ.

Ключ ECC160

При создании (редактировании) асимметричного алгоритма цифровой подписи ECC160 место диалога **Редактирование определителя** занимает **Ключ ECC160**.

Диалог **Ключ ECC160** служит для генерации ключевой пары алгоритма цифровой подписи ECC160.

В верхней части диалога отображается секретный ключ алгоритма, в нижней – открытый. Справа располагаются кнопки, позволяющие выполнять сервисные действия с ключевой парой.



По умолчанию при создании алгоритма ECC160 генерируется ключевая пара из случайных чисел. Ключевую пару можно изменить, автоматически создав новую (кнопка **[Сгенерировать новое значение]**).

Кнопка **[Сохранить]** позволяет сохранить ключевую пару во внешнем файле (*.ecc). По нажатию кнопки **[Загрузить]** произойдет загрузка ранее сохраненной ключевой пары из файла (*.ecc).

Чтобы использовать алгоритм ECC160 при автозащите (опция /SIGN) необходимо указывать в качестве параметра опции файл с открытым ключом. Кнопка **[Экспорт в bin-файл автозащиты]** позволяет сохранить открытый ключ в файле (имя по умолчанию – PublicKey.bin).

Кнопка **[Экспорт в исходный код]** сохраняет ключевую пару в заголовочном файле на C++ (имя по умолчанию – EccKeySource.h).

***

После редактирования определителя (или ключа ECC160 для одноименного алгоритма) и нажатия на кнопку **[Завершить]** диалог создания алгоритма закрывается, и новый алгоритм появляется в списке полей образа. При этом GrdUtil.exe автоматически присваивает алгоритму числовое имя (порядковый номер)* и корректирует границу аппаратных запретов с учетом добавленного алгоритма.

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

Получение ответов симметричных алгоритмов

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

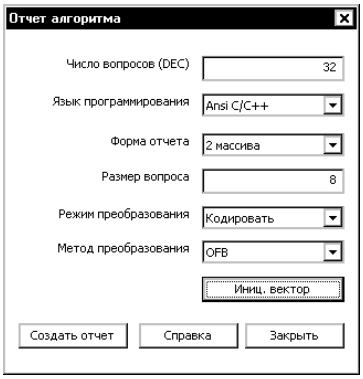
GrdUtil.exe предоставляет удобный сервис для получения ответов симметричных алгоритмов. Утилита обращается к выбранному алгоритму, получает его ответы и сохраняет результаты в специальном файле отчета.

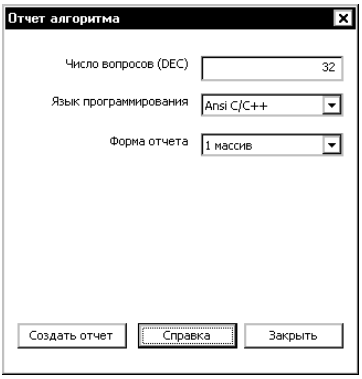
Важно!

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

* Для защищенных ячеек и аппаратных алгоритмов действует единая нумерация

Чтобы получить массив ответов, выделите алгоритм в списке и выполните команду меню **Разное | 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

Симметричные алгоритмы имеют 4 режима работы, которые отличаются по своим характеристикам и назначению. Подробнее об алгоритмах см. во 2-й части Руководства пользователя.

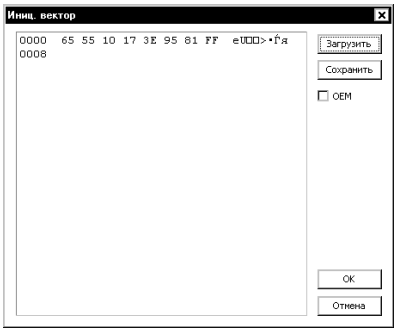
Выберите метод преобразования с помощью одноименного разворачивающегося списка.

Режим преобразования

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

Вектор инициализации

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



Элементы управления диалога **Вектор инициализации**:

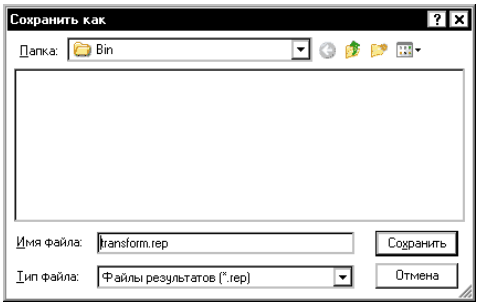
Элемент интерфейса	**Назначение**
Окно шестнадцатеричного редактора	Ввести значение вектора инициализации
Кнопка **[Загрузить]**	Загрузить дамп из файла с расширением *.dmp
Кнопка **[Сохранить]**	Сохранить дамп в файле с расширением *.dmp
Флаг **OEM**	Выбрать Windows- / DOS-кодировку. По умолчанию используется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

Зависимость режимов работы симметричного алгоритма от вектора инициализации:

Режимы работы алгоритма AES/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).

Шифрование данных симметричным алгоритмом

В электронных ключах Guardant реализованы симметричные аппаратные алгоритмы GSII64 и AES128. Они выполняют аппаратное преобразование данных. Подробную информацию о симметричных алгоритмах см. во 2-й части Руководства пользователя.

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

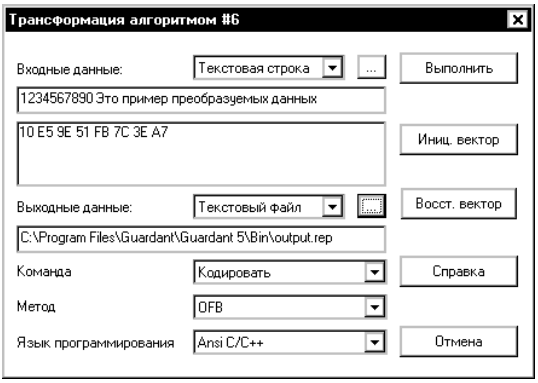
Для вызова алгоритма из приложения используется операция ***GrdTransform***. Подробную информацию об операциях Guardant API см. в разделе **Guardant API**.

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 образа алгоритм типа AES или GSII64 и выполните команду **Разное | Шифрование**. На экране появится диалоговое окно **Преобразование алгоритмом №N**[**].

[**] Где N – числовое имя (порядковый номер алгоритма)



В диалоге определите следующие параметры:

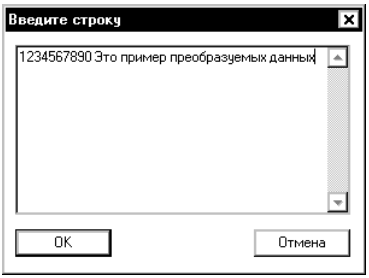
- Входные данные и их вид
- Вектор инициализации
- Выходные данные и их вид
- Направление и метод кодирования
- Язык программирования (если выходные данные представлены в виде исходного текста)

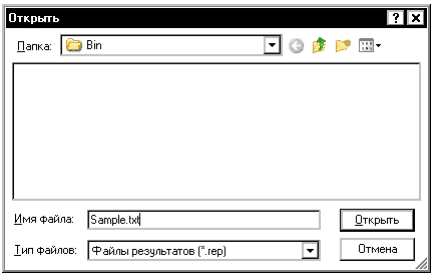
Входные данные

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**.

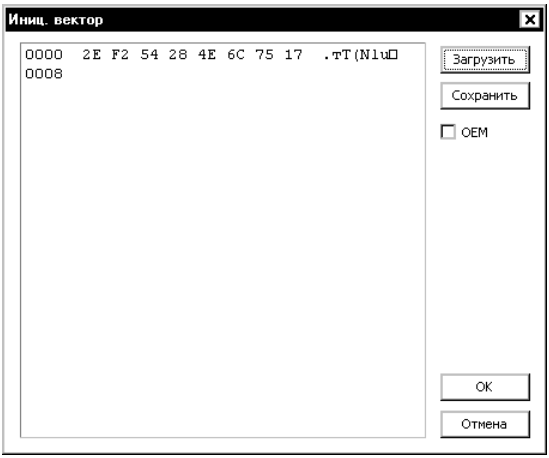
Вектор инициализации

Вектор инициализации – случайное число, которое используется для работы симметричного алгоритма в режимах работы CFB, CBC и OFB. Вектор инициализации для GSII64 равен 8 байтов, для AES128 – 16 байтов.

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.

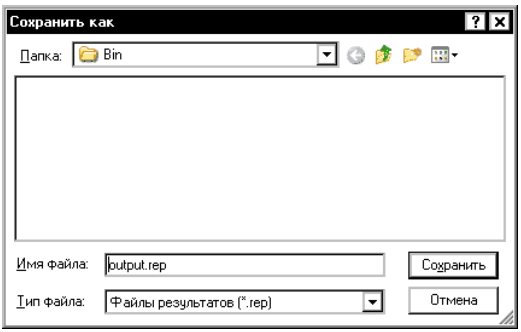
Выходные данные

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 и AES128 имеют 4 режима работы, которые отличаются по своим характеристикам и назначению. Подробнее см. во 2-й части Руководства пользователя.

Для выбора метода преобразования предназначен одноименный разворачивающийся список.

Выполнение преобразования

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

Запись данных образа в память ключа

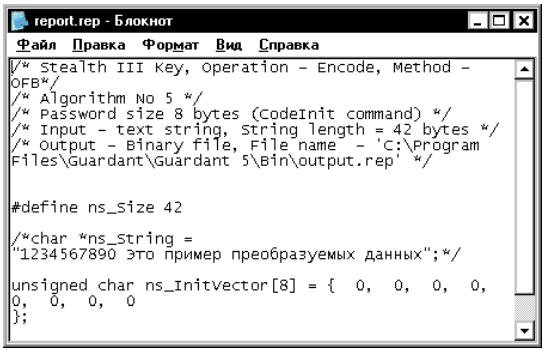
Перед выполнением преобразования утилита выдает запрос о необходимости записи данных образа в ключ:

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

Сохранение отчета

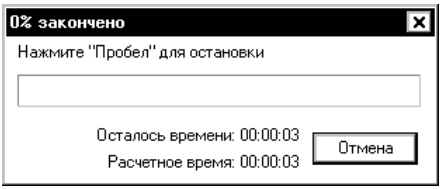
Далее на экране появится диалог сохранения отчета о преобразовании, в котором нужно указать имя (по умолчанию report.rep) и путь к файлу отчета.

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**).

Важно!

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

Защищенные ячейки

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

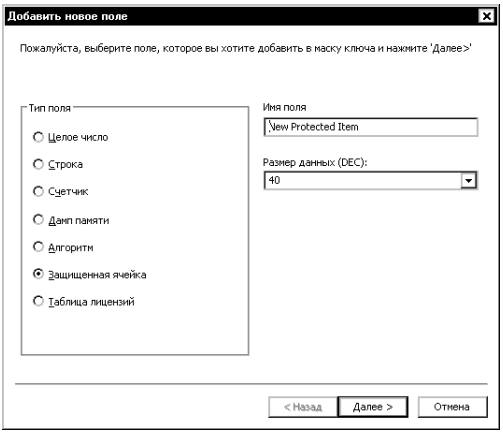
Переход к следующей странице происходит при помощи нажатия кнопки **[Далее]** после выполнения текущего диалога.

Создание защищенной ячейки

Размер защищенной ячейки в современных ключах не ограничен, а в Guardant Stealth III /Net III должен составлять от 1 до 255 байтов (+ служебные поля).

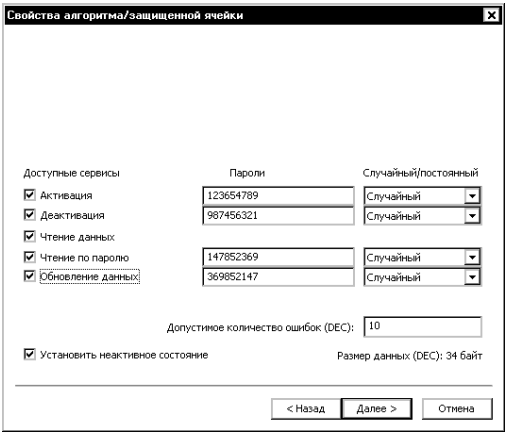
Чтобы создать защищенную ячейку, выполните команду меню **Редактировать | Добавить новое поле**.

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



Свойства защищенной ячейки

В окне **Свойства защищенной ячейки** расположены флаги, управляющие сервисами защищенных ячеек:



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.

Описание сервисов защищенной ячейки идентично описанию сервисов аппаратных алгоритмов (см. раздел **Сервисы аппаратных алгоритмов**).

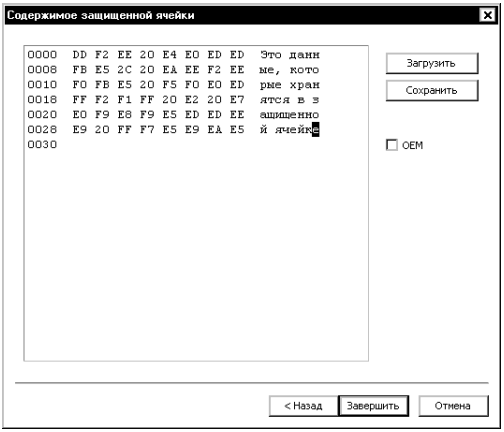
Размер данных защищенной ячейки

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.

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 образа. 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.

Таблица лицензий

Таблица лицензий – технология, применяющаяся в сетевых ключах Guardant. Ее назначение – хранение сетевого ресурса (кроме ключей Guardant Net II/Net) и контроль и управление лицензиями в программных комплексах, состоящих из нескольких модулей.

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
- Количество копий приложения, либо рабочих станций, использующих определенный модуль программы, ограничивается ресурсом этого модуля, записанным в таблице лицензий

Таблица лицензий в ключах Guardant Sign Net/ Time Net/ Net III отличается от таблицы лицензий в устаревших сетевых ключах:

Ключ	**Аппаратные запреты**	**Расположение в памяти ключа**	**Работа из приложения (Guardant API)**
Guardant Sign Net, Guardant Time Net, 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 II, Guardant Net	На запись	Строго после аппаратных алгоритмов и до начала полей других типов	GrdRead, GrdLogin, GrdLogout

Т. о., таблица лицензий в Guardant Sign Net/ Time Net/ Net III представляет собой частный случай **защищенной ячейки**.

* Для защищенных ячеек и алгоритмов действует единая нумерация

Создание таблицы лицензий

Важно!

Для Guardant Sign Net / Time Net / 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:

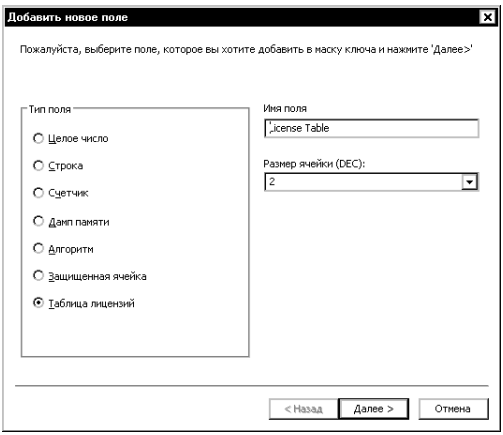
- Добавить таблицу лицензий
- Свойства таблицы лицензий
- Свойства защищенной ячейки (кроме Guardant Net II/Net)

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:

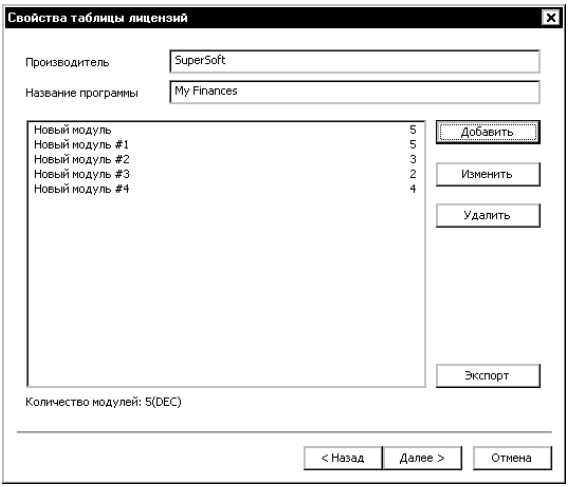
Размер ячейки, байтов	Максимальный ресурс лицензий
1	254
2	65534

Свойства таблицы лицензий

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

- **Реального сетевого ресурса ключа**
- 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:

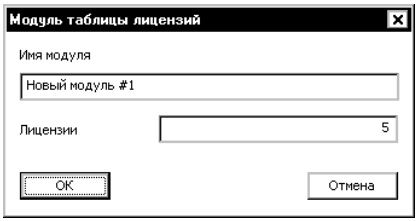
Кнопка	Назначение
[Добавить]	Вывести на экран диалог **Модуль таблицы лицензий** для создания нового модуля
[Изменить]	Вывести на экран диалог **Модуль таблицы лицензий** для редактирования параметров выбранного модуля
[Удалить]	Удалить выбранный модуль

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.

***

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

Диалог создания таблицы лицензий в маске современных сетевых ключей имеет дополнительное окно, в котором можно задействовать сервисы защищенных ячеек.

Свойства защищенной ячейки

Для таблицы лицензий современных сетевых ключей 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 алгоритмов. Подробную информацию см. в разделе **Сервисы аппаратных алгоритмов**.

После активации необходимых сервисов создание таблицы лицензий в маске современных сетевых ключей закончено. По нажатию на кнопку **[Завершить]** диалог создания нового поля закрывается, и таблица лицензий появляется в списке полей образа.

Загружаемый код

Ключи моделей Guardant Code / Code Time позволяют загружать и исполнять специально написанный код приложения внутри ключа. Подробно см. Руководство пользователя, Часть 3 «Guardant Code».

GrdUtil.exe предоставляет удобный сервис для работы с загружаемыми кодом, в том числе, его автоматическое преобразование из Bin-файла в формат, пригодный для записи в ключ (GCEXE), и, собственно, запись GCEXE во Flash-память.

Работа с полем типа **Загружаемый код** из приложения происходит при помощи специальных функций Guardant API: GrdCodeGetInfo, GrdCodeLoad, GrdCodeRun (подробности см. в GrdAPI.chm).

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

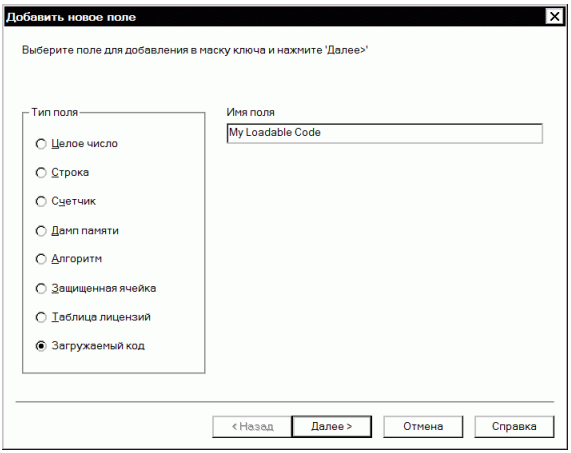
- Добавить загружаемый код
- Свойства загружаемого кода
- Временные зависимости (для Guardant Code Time)
- Настройки загружаемого кода

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

Добавление поля Загружаемый код

Чтобы создать поле типа **Загружаемый код**, выполните команду **Редактировать | Добавить новое поле**.

В появившемся диалоге выберите переключатель **Загружаемый код** и задайте имя поля:



Ячейка **Загружаемый код** имеет фиксированный размер 160 байт.

Свойства загружаемого кода

Поле типа **Загружаемый код** в Guardant Code / Code Time представляет собой частный случай **защищенной ячейки**, то есть, оно защищено аппаратными запретами на чтение и запись. И его состояние можно программировать и, в дальнейшем, управлять им из приложения.

Для загружаемого кода доступны только **сервисы активации и деактивации**, которые управляют состоянием защищенной ячейки и обеспечивают доступ к ее содержимому из приложения.

Работа с сервисами активации и деактивации загружаемого кода полностью идентична работе с сервисами защищенных ячеек и алгоритмов. Подробную информацию см. в разделе **Сервисы аппаратных алгоритмов**.

Временные зависимости загружаемого кода

В маске ключа Guardant Code Time, содержащего часы реального времени, за страницей **Свойства загружаемого кода** следует страница **Временные зависимости**, на которой расположены элементы, позволяющие ограничивать время работы защищенного приложения.

Смысл технологии ограничения времени заключается в том, что работоспособность ячейки **Загружаемый код** зависит от таймера (RTC), встроенного в ключ Guardant Code Time.

Информацию по диалогу **Временные зависимости** см. в разделе **Ограничение астрономического времени работы приложения**.

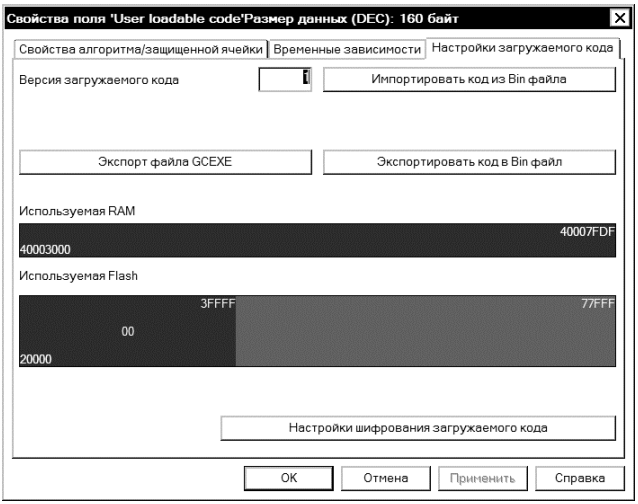
Настройки загружаемого кода

После компиляции кода и получения Bin-файла (подробно см. Руководство пользователя, Часть 2) разработчику необходимо передать бинарный код в GrdUtil.exe для его обработки и записи в ключ. Для этого служит диалог **Настройки загружаемого кода**.

Диалог позволяет:

- Импортировать предварительно скомпилированный код из Bin-файла
- Преобразовывать импортированный код в формат GCEXE, пригодный для записи в ключ Guardant Code / Code Time
- Записывать код в GCEXE-формате во Flash-память ключа или выгружать его во внешний файл.

Кроме того, диалог предоставляет дополнительные сервисы, упрощающие работу с загружаемым кодом.



Импорт загружаемого кода из Bin-файла

По нажатию кнопки **Импортировать код из Bin-файла**, находящейся в правой верхней части страницы **Настройки загружаемого кода**, появляется диалог выбора Bin-файла из нужного проекта.

При импорте GrdUtil.exe считывает из файла **имя_проекта.bmap** настройки, описывающие использование памяти ключа загружаемым кодом. После этого в диалоге отображаются:

Индикатор состояния	Назначение
Используемая RAM	Индикация выделенной для загружаемого кода оперативной памяти ключа, ее начального и конечного адреса
Используемая Flash	Индикация выделенной для загружаемого кода Flash-памяти ключа, ее начального и конечного адреса, а также номера ячейки, хранящей дескриптор загружаемого кода

Свободная память обозначается зеленым цветом, используемая – синим. Адресация дается в шестнадцатеричном формате.

Преобразование бинарного кода в формат GCEXE

По соображениям конфиденциальности загружаемый код не должен передаваться «наружу» в открытом виде.

Поэтому в GrdUtil.exe реализована эффективная схема подготовки кода для записи в электронный ключ и безопасной передачи обновлений загружаемого кода конечным пользователям.

GrdUtil.exe автоматически преобразует бинарный код в файл формата GCEXE, содержащий:

- Зашифрованный на AES исходный код
- Зашифрованный на *открытом* ключе ***ECC160 №#1*** сеансовый ключ AES, использовавшийся ранее для шифрования кода
- ЭЦП файла, полученную на закрытом ключе **ECC160 №#2**

При этом в дескрипторе (ячейке) загружаемого кода хранится «ответная часть» ключей ECC, используемых при преобразовании бинарного кода:

- *Закрытый* ключ ***ECC160 №#1*** для шифрования
- Открытый ключ **ECC160 №#2** для ЭЦП

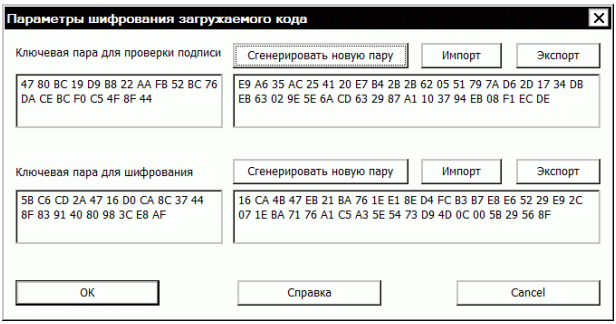
Что позволяет электронному ключу при обращении к загруженному коду успешно его проверять, расшифровывать и выполнять.

Важно!

Преобразование кода в формат GCEXE производится утилитой GrdUtil.exe **автоматически** при записи образа в ключ (либо нажатии на кнопку **Экспортировать GCEXE**), не требуя от разработчиков никаких действий для ее реализации, кроме настройки ключевых пар ECC160.

Настройки шифрования загружаемого кода

По нажатию кнопки **Настройки шифрования загружаемого кода** появляется диалог для работы с ключевыми парами:



Диалог предназначен для генерации, импорта и экспорта ключевых пар асимметричного алгоритма ECC160, которые используются при преобразовании бинарного кода в формат GCEXE (см. предыдущий пункт).

В верхней части диалога отображаются закрытый (слева) и открытый (справа) ключи **ECC160 №#2** для цифровой подписи зашифрованного кода.

В нижней части диалога находится ключевая пара (закрытый ключ – слева, открытый – справа) ***ECC160 №#1*** для шифрования бинарного кода.

Кроме того, диалог дополнен кнопками, позволяющими генерировать новые ключевые пары, экспортировать их во внешний файл для использования в приложении и импортировать ключевые файлы из других проектов.

Экспорт файла GCEXE. Обновление кода у пользователя

Экспорт GCEXE во внешний файл может потребоваться в случае, когда разработчику необходимо обновить загружаемый код в электронном ключе, находящемся у конечного пользователя.

В такой ситуации разработчику следует придерживаться следующей схемы действий:

0) На этапе разработки приложения должен быть предусмотрен механизм обновления загружаемого кода из приложения. Такой механизм реализуется при помощи функции Guardant API **GrdCodeLoad** (см. **GrdAPI.chm**).

1) После внесения необходимых изменений новая версия загружаемого кода компилируется в бинарный файл, который импортируется в **GrdUtil.exe**.

2) Для правильной работы обновленного загружаемого кода в удаленном ключе должны использоваться те же ключевые пары, которые применялись при программировании ключа в первый раз см. **Настройки шифрования загружаемого кода**.

3) При нажатии на кнопку **[Экспортировать GCEXE]** происходит формирование и выгрузка GCEXE во внешний файл.

4) Если необходимо сделать обновление кода зависимым от ключа (к примеру, при подготовке платных обновлений), то следует указать десятичный ID ключа конечного пользователя в диалоге, который возникает по нажатию кнопки **[Экспортировать GCEXE]**.

5) Сохраненный в формате GCEXE загружаемый код передается конечному пользователю, который производит обновление содержимого ключа способом, предусмотренным разработчиком на нулевом шаге.

****

Запись загружаемого кода в ключ

После выполнения настроек загружаемого кода остается завершить диалог и выполнить команду меню **Ключ | Запись в ключ**. При этом будут сформированы и записаны в ключ:

1) Прошивка, содержащая дескриптор загружаемого кода (в числе прочих полей), – в EEPROM-память.

2) Загружаемый код в формате GCEXE – во Flash-память ключа.

Софтверные ключи Guardant SP

Софтверный (программный) ключ – это средство защиты недорогих приложений.

Принцип работы Guardant SP основан на привязке ключа к уникальным характеристикам компьютерных комплектующих, состояние которых на момент активации фиксируется в SP-ключе и далее сверяется с текущими значениями при каждом обращении приложения к ключу.

Механизмы привязки, включая сбор и, в дальнейшем, контроль информации о компьютере, обмен данными с сервером активации, перешифрование файла-контейнера SP-ключа и др., реализованы в драйвере Guardant и являются внутренними. Они полностью прозрачны для пользователя и защищены от внешних воздействий.

Этот факт, а также вынос части логики (проверка серийного номера, выработка уникальных ключей шифрования для SP-ключа с участием контрольных значений комплектующих) на внешний защищенный ресурс – сервер активации – существенно повышает стойкость софтверных ключей и выделяет Guardant SP в ряду аналогичных решений.

В свою очередь, защищаемое приложение привязывается к софтверному ключу при помощи «обычных» технологий Guardant – автоматической защиты и/или Guardant API. Реализация этого участка системы защиты, а также подготовка шаблона возлагается на разработчика приложения.

Программный ключ представляет собой файл-контейнер ***.grdvd**, содержащий (в общем виде):

- ***Образ ключа***: защищенные ячейки, алгоритмы, и другие информацию для защиты приложения
- ***Лицензию***: серийный номер и «весовые» коэффициенты комплектующих компьютера на момент активации SP-ключа

- ***Служебные данные***, необходимые для действий с SP-ключом: активация, деактивация и т. п.

Содержимое SP-ключа зашифровано на алгоритме AES и защищено от модификации с использованием ECC. Хэши от вычисленных контрольных «весовых» значений комплектующих, к которым осуществляется привязка, участвуют в процессе шифрования файла-контейнера.

Для удобства работы в SP-ключе эмулируется EEPROM-память современных аппаратных ключей Guardant (размер области данных, доступных для использования составляет ~4Кбайт), а также обеспечивается совместимость со стандартными механизмами защиты. Т. о., программирование и обращение к софтверному ключу из приложения происходят точно также как и для любого другого ключа Guardant.

Это позволяет записывать в софтверный ключ те же типы полей, что и в аппаратные ключи, и после активации SP-ключа точно также работать с этим полями из защищенного приложения, как при помощи Guardant API, так и автозащиты.

Важно!

Далее рассматриваются только те аспекты работы с программными ключами, которые непосредственно связаны с созданием и программированием образа SP-ключа.

Комплексная информация по софтверным ключам содержится в **Руководстве по работе с софтверными ключами Guardant SP** и **Руководстве по работе с сервером активации**, доступными для загрузки на сайте www.guardant.ru

GrdUtil.exe предоставляет удобные сервисы для работы с SP-ключами, в том числе:

- Создание образа софтверного ключа
- Создание в образе полей различных типов, в том числе:
  - Защищенных ячеек и запись в них данных, необходимых для работы защищенного приложения
  - Алгоритмов шифрования, для последующего обращения к ним из приложения
- Создание шаблона, в т. ч. защищенного, а также активированного SP-ключа для отладки системы защиты
- Настройка параметров привязки ключа к компьютерным комплектующим

Порядок программирования софтверных ключей

В большинстве ситуаций удобно придерживаться следующего порядка работы с софтверными ключами из интерфейса GrdUtil:

1. Создать образ SP-ключа
2. Создать поля нужных типов, записать в них данные и сохранить образ.
3. Задать параметры привязки ключа/приложения к характеристикам компьютера.
4. Создать отладочный ключ
5. Выполнить привязку приложения к отладочному ключу при помощи автозащиты и/или Guardant API (см. **GrdAPI.chm**).
6. Протестировать работу защищенного приложения с отладочным ключом.
7. Используя ранее созданный образ, создать и растиражировать шаблоны SP-ключей для включения в дистрибутив защищенной программы вместе с серийным номером для активации и мастером активации **GuardantActivationWizard.exe**.

Настройка параметров привязки к компьютеру

Важным моментом при лицензировании приложения через привязку к компьютеру является соблюдение баланса между надежностью защиты и удобством для конечного пользователя.

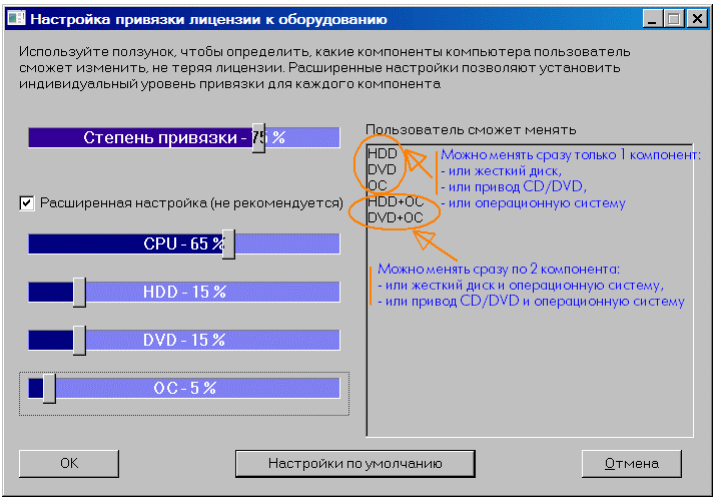
Необходимо сделать так, чтобы приложение было затруднительно использовать на другом компьютере без переактивации (покупки нового серийника), но при этом легальный пользователь имел бы возможность менять комплектующие, не теряя лицензии.

В случае с ключами Guardant SP такой баланс может достигаться как возможностью повторной активации на одном и том же серийном номере*, так и *гибкой политикой* привязки, при которой замена компонентов, которые традиционно обновляются чаще (к примеру, операционная система или жесткий диск), не вызывает нарушения лицензии.

Чтобы определить параметры привязки софтверного ключа к компьютеру, выполните команду **Ключ | Настроить параметры привязки**.

* По умолчанию серийные номера SP-ключей имеют заданный при продаже ресурс на несколько активаций, который может изменяться разработчиком приложения в сторону уменьшения.

На экране появится диалог **Настройка привязки лицензии к оборудованию**:



Элементы управления диалога (шкалы и ползунки) служат для задания степени привязки софтверного ключа к комплектующим компьютера.

Защита Guardant SP может контролировать неизменность 4-х основных компонентов: процессора, жесткого диска, DVD-привода и операционной системы.

Ползунки позволяют изменять степень привязки как в целом (шкала **Степень привязки**), так и по каждому компоненту в отдельности (флаг **Расширенная настройка**).

В правой части диалога отображается результат настройки, а именно, перечисляются компоненты, которые конечный пользователь может заменить, не переактивируя электронный ключ.

Результаты выводятся в виде допустимых вариантов замены. Каждый вариант выводится на отдельной строке.

Пример

Т. о., на приведенном выше скриншоте доступны 5 вариантов «легальной» замены комплектующих: 3 первых варианта позволяют менять на выбор или винчестер, или DVD, или OC, а последние варианты позволяют уже менять по два компонента*.

* Приведенный пример только иллюстрирует возможности настройки. Не рекомендуется использовать его в качестве окончательного варианта.

Создание отладочного софтверного ключа

Для установки защиты на приложение, а также для тестирования работы уже защищенного ПО, необходим ***активированный*** SP-ключ на компьютере разработчика.

Только активированный софтверный ключ «воспринимается» функциями Guardant API, утилитами комплекта разработчика (диагностика, автозащита и проч.) и защищенным приложением как полноценный электронный ключ, с которым возможно работать.

Поэтому после создания и программирования образа ключа, а также выполнения настройки параметров привязки к компьютеру, необходимо создать отладочный SP-ключ, т. е. программный ключ, активированный на компьютере разработчика, где выполняется защита и приложения и ее отладка.

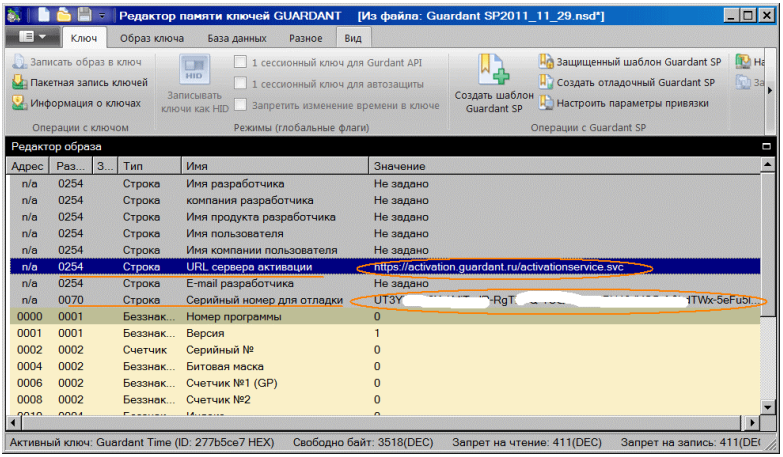
Важно!

Для успешного создания отладочного ключа Guardant SP должны быть выполнены следующие предварительные условия:

1. Произведена *регистрация* на сайте **sp.guardant.ru**, получены и переведены в статус **Передан конечному пользователю** серийные номера.
2. В директорию установки Guardant SDK помещен файл кодов доступа **nvcodes.dat** *с поддержкой софтверных ключей*, полученный после регистрации.
3. На компьютере, где будет выполняться активация, настроено и установлено Интернет-соединение.

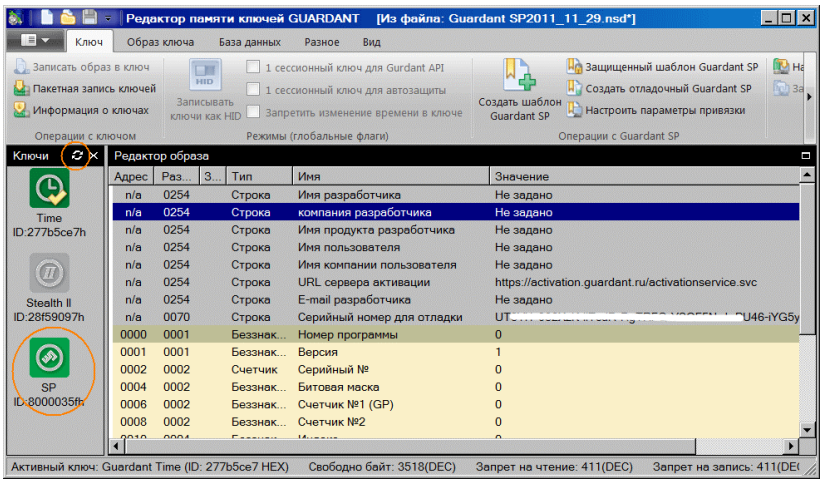
Чтобы создать отладочный SP-ключ, выполните следующие действия:

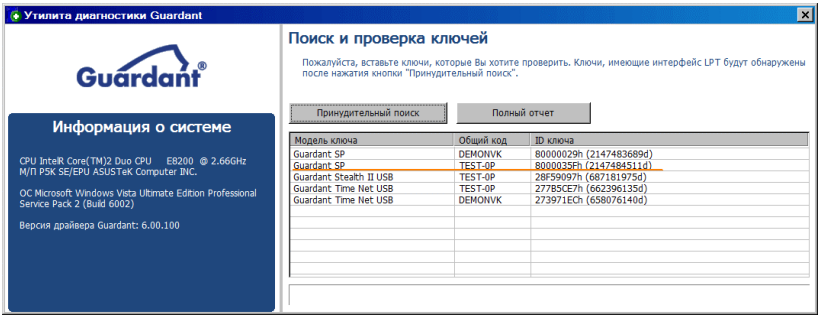
- Загрузите в Редактор нужный образ
- Отредактируйте образ при необходимости.
- Выберите в списке поле **Серийный номер для отладки** и выполните команду меню **Образ ключа | Свойства поля**
- Вставьте полученный серийный номер в поле ввода открывшегося диалога и завершите диалог нажатием на кнопку **OK**.
- Проверьте содержимое поля **URL сервера активации**, в нем должна содержаться строка: https://activation.guardant.ru/activationservice.svc . При необходимости отредактируйте поле.



- Выполните команду меню **Ключ | Создать отладочный Guardant SP** и следуйте инструкциям Мастера активации, появившегося на экране.

- После успешной активации *обновите список ключей* в окне **Ключи** утилиты GrdUtil.exe или вызовите утилиту диагностики Guardant из системной Панели управления (см. **Пуск/Панель управления/Драйверы Guardant/Диагностика**):





Т. о., при успешном выполнении команды **Создать отладочный Guardant SP** происходит создание и активация шаблона; отладочный софтверный ключ «привязывается» к характеристикам компьютера разработчика.

Полученный отладочный ключ используется для установки защиты при помощи Guardant API или автозащиты, а также тестирования защищенного ПО.

Работа с отладочным ключом ведется точно также как с любым аппаратным ключом Guardant.

Создание шаблона SP-ключа

Шаблон софтверного ключа представляет собой «заготовку», т. е. запрограммированный согласно выбраной схеме защиты, но не активированный, ключ.

Чтобы создать шаблон софтверного ключа, загрузите нужный образ в Редактор, при необходимости отредактируйте его и выполните команду меню **Операции с ключом | Создать шаблон**.

На экране появится системный диалог сохранения файла шаблона, который позволяет задать имя шаблона (по умолчанию **GrdVD_Template_год_месяц_число.grdvd**) и директории, где он будет расположен.

После этого сохраненный шаблон можно включить в комплект поставки защищенного приложения вместе с серийным номером и мастером активации **GuardantActivationWizard.exe**.

Создание защищенного шаблона SP-ключа

Защищенный шаблон отличается от обычного тем, что содержимое SP-ключа, активированного на таком шаблоне, невозможно изменить программным способом, включая вызовы функций Guardant API.

Это служит дополнительной защитой при использовании софтверных ключей.

Чтобы создать защищенный шаблон, выполните команду меню **Операции с ключом | Защищенный шаблон**.

В остальном работа с защищенным шаблоном полностью аналогична работе с обычным шаблоном, см. предыдущий пункт.

Мастер активации SP-ключей

Утилита **GuardantActivationWizard.exe** – это штатное средство активации софтверных ключей Guardant.

Утилита ориентирована на конечных пользователей и выполнена в виде мастера, состоящего из нескольких диалоговых окон.

Основными сценариями использования утилиты являются:

- Активация SP-ключа, переданного конечному пользователю вместе с защищенной программой
- Активация отладочного SP-ключа на компьютере разработчика

Важно!

Для успешной активации ключа Guardant SP необходимо выполнение следующих предварительных условий:

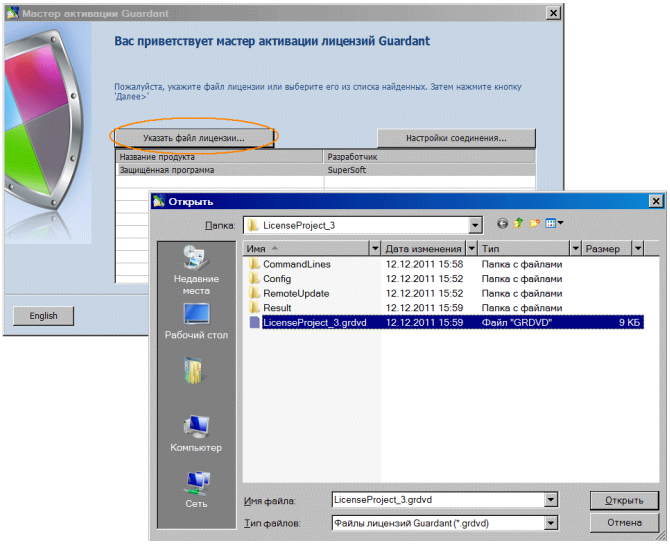
1. Комплект поставки защищенного приложения должен включать:
   - Драйвер Guardant версии не ниже 6.0,
   - Мастер активации **GuardantActivationWizard.exe**,
   - Запрограммированный шаблон ключа Guardant SP (файл вида ***.grdvd**)*, к которому привязано приложение,
   - Серийный номер для активации**
2. Драйвер Guardant 6.x должен быть установлен до активации.
3. Серийный номер должен иметь статус **Передан конечному пользователю** на сервере активации (сайт **sp.guardant.ru**).
4. На компьютере, где будет выполняться активация, должно быть настроено и установлено Интернет-соединение.

Чтобы активировать ключ Guardant SP, запустите мастер активации **GuardantActivationWizard.exe** и следуйте его указаниям:

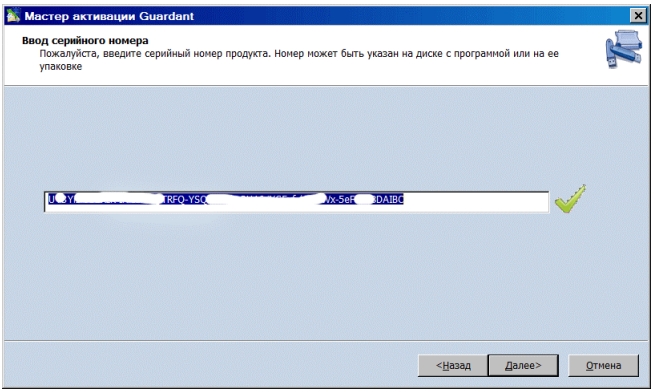
* В комплект поставки приложения защищенного SP-ключом также могут входить и другие файлы, состав которых зависит от варианта защиты. Полный список файлов защиты доступен в **Главе 8. Комплект поставки защищенного приложения**.

** В зависимости от бизнес-модели серийный номер может не входить в комплект поставки, а высылаться конкретному пользователю непосредственно перед активацией.

1. При помощи кнопки **Указать файл лицензии** выберите путь к файлу вида ***.grdvd**. Проверьте настройки Интернет-соединения и нажмите на кнопку **Далее**:



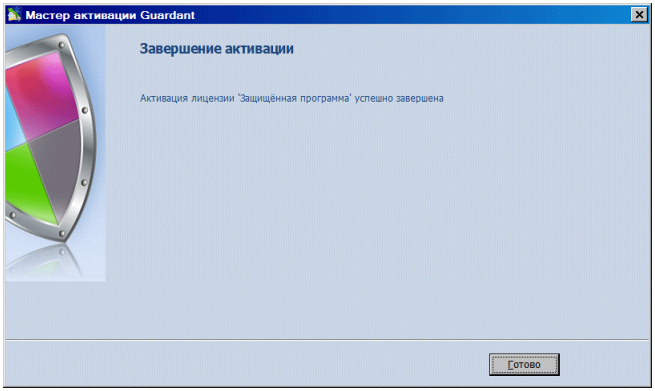
2. Укажите в поле ввода серийный номер для активации. Нажмите на кнопку **Далее**:



Мастер производит необходимый обмен информацией с драйвером ключа и сервером активации. При этом помимо прочего, происходит проверка введенного серийного номера, а также перешифрова-

ние файла софтверного ключа с использованием контрольных значений комплектующих компьютера.

3. Если активация прошла успешно, мастер выдает завершающее диалоговое окно:



После этого можно провести диагностику активированного ключа и начать работу с защищенным приложением.

Дампы, целые числа, строки и счетчики

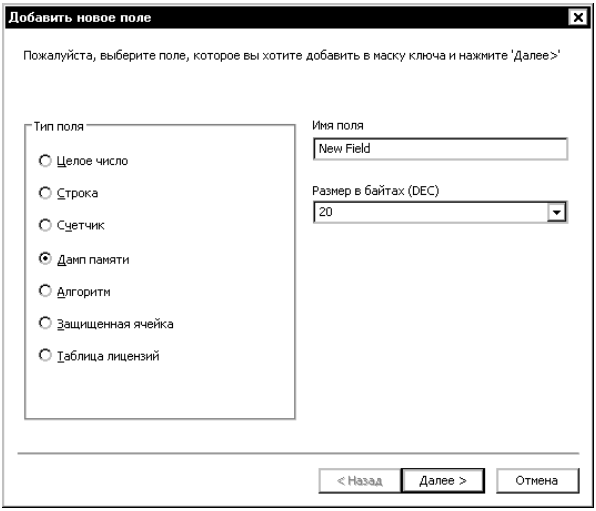
Кроме рассмотренных ранее, в GrdUtil.exe можно создавать поля следующих типов: дамп, целое число, строка и счетчик.

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:

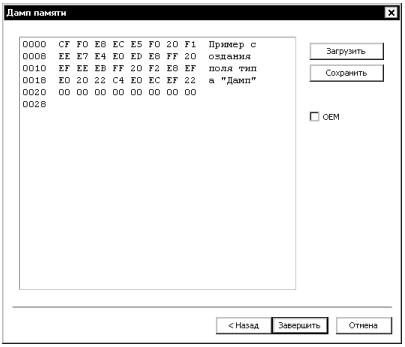


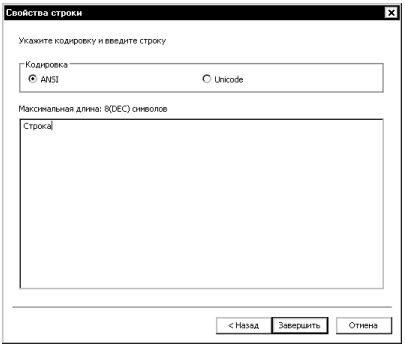
Размеры полей разных типов:

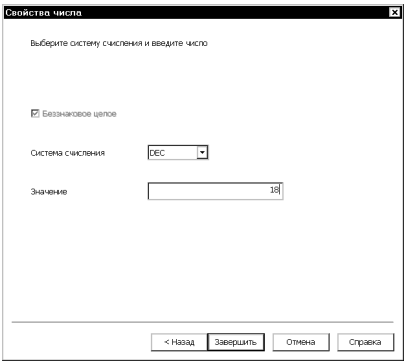
Тип поля	Возможный размер поля, байтов
Дамп	Произвольный размер
Строка	
Целое число	1, 2, 4, 8
Счетчик	

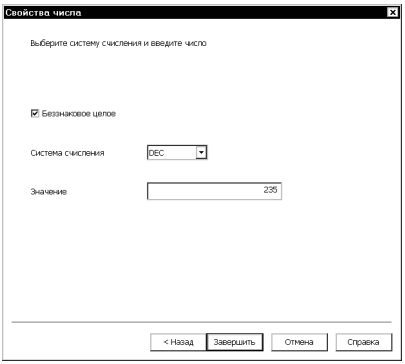
Свойства поля

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

Тип поля	Элемент интерфейса	Назначение
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 образа.

Ограничение времени работы и числа запусков приложения

Ограничение числа запусков приложения

Все современные модели локальных ключей Guardant позволяют лицензировать работу приложения, ограничивая количество его запусков. Это удобная технология, как для создания демо-версий, так и в других случаях.

Чтобы ограничить число запусков приложения:

1. Запустите утилиту программирования ключа GrdUtil.exe, загрузите нужный файл образа.

2. Выберите аппаратный алгоритм, который будет использоваться для защиты. Установите нужное число запусков программы с помощью 4-хбайтового счетчика алгоритма, как это описано ниже.

Важно! Автозащита

1. Для ограничения числа запусков приложения можно использовать только алгоритмы типа GSII64 или AES128.

2. В процессе автозащиты происходит множественный вызов аппаратного алгоритма, и если запрограммировать ключ до автозащиты, то указанное заранее значение счетчика уменьшится. По этой причине лучше **предварительно защитить исполняемый файл** с помощью **Мастера автозащиты** или **строчной утилиты**, и только потом прописать в ключ нужное значение счетчика.

3. Запишите маску в ключ.

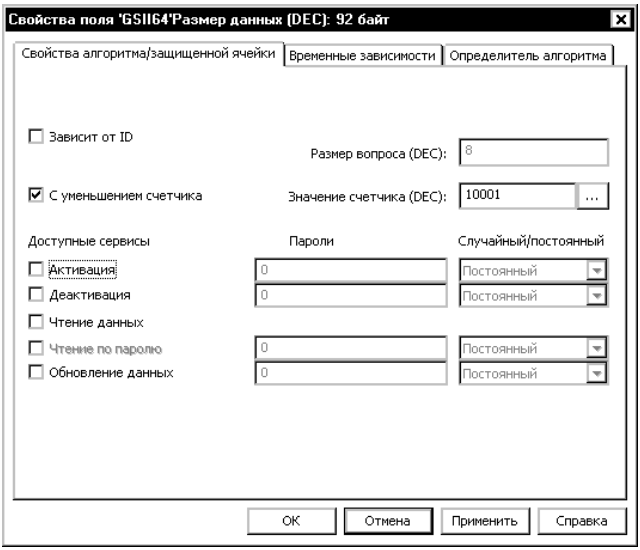
Теперь при каждом старте приложения счетчик алгоритма автоматически будет декрементироваться на единицу – в случае использования автозащиты, либо, в случае использования Guardant API, декрементировать счетчик нужно самостоятельно, производя вызов алгоритма.

После того, как значение счетчика обнулится, приложение перестанет запускаться.

Установка счетчика аппаратного алгоритма

1. Выберите в маске аппаратный алгоритм, который будет использоваться для защиты, и выполните команду меню **Редактировать | Свойства поля**.

2. В появившемся диалоге **Свойства поля**, вкладка **Свойства алгоритма/ защищенной ячейки**, установите флаг **С уменьшением счетчика** и задайте значение счетчика в появившемся поле:



3. Нажмите на кнопку **[Применить]** и закройте диалог.

Запишите маску в ключ.

Ограничение времени работы приложения

Электронные ключи Guardant Time/ Net Time / Code Time оснащены часами реального времени и позволяют ограничивать астрономическое время работы защищенного приложения.

Смысл технологии ограничения времени заключается в том, что работоспособность алгоритма зависит от таймера (RTC), встроенного в ключ.

С помощью ключей с часами реального времени можно реализовывать различные лицензионные политики, влияющие на время работы защищенного приложения:

Приложение сможет работать только после активации алгоритма, которая произойдет по наступлению указанной даты
Приложение перестанет работать после деактивации алгоритма, которая произойдет по наступлению указанной даты
Срок работы приложения ограничен заданным периодом работоспособности, т. н. «временем жизни», алгоритма. Алгоритм активируется при первом запуске приложения и деактивируется по истечению «времени жизни»

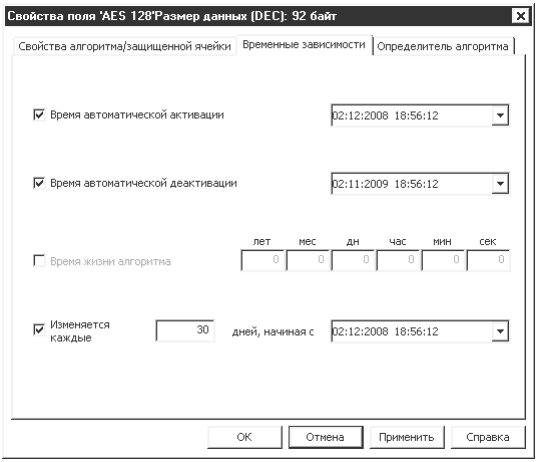
Установка временной зависимости алгоритма

1. Выберите в маске для ключа с RTC аппаратный алгоритм, который будет использоваться для защиты, и выполните команду меню **Редактировать | Свойства поля**.

Важно!

При автозащите установка временной зависимости имеет смысл только для алгоритма типа GSII64 и AES128, и именно того, который будет использоваться для защиты.

2. В появившемся диалоге **Свойства поля** перейдите на вкладку **Временные зависимости**.



3. С помощью флагов установите нужные временные зависимости (подробное описание см. в таблице ниже). Запишите маску в ключ.

Элементы управления диалога **Временные зависимости**:

Элемент интерфейса	Назначение
Флаг **Время автоматической активации**	Если флаг установлен, то алгоритм (а, следовательно, и защищенное приложение) станет работоспособным только после наступления указанной даты активации
Комбинированное поле ввода/календарь для установки даты активации	Установка календарной даты активации аппаратного алгоритма*

* Дату можно установить как непосредственно в поле ввода, так и используя календарь

Элемент интерфейса	Назначение
Флаг **Время автоматической деактивации**	Если флаг установлен, то аппаратный алгоритм (а, следовательно, и защищенное приложение) перестанет быть работоспособным сразу после наступления указанной даты деактивации
Комбинированное поле ввода/календарь для установки даты деактивации	Установка календарной даты деактивации аппаратного алгоритма
Флаг **Время жизни алгоритма**	Если флаг установлен, то время работы аппаратного алгоритма (а, следовательно, и защищенного приложения) будет ограничено указанным календарным периодом времени. Отсчет времени работы начинается после первого обращения к алгоритму (после первого старта приложения)
Поля ввода периода работоспособности алгоритма	Указание временного периода работоспособности алгоритма в формате **лет:мес:дн:час:мин:сек**
Флаг **Алгоритм изменяется каждые...**	Если флаг установлен, то определитель алгоритма будет постоянно изменяться через определенный временной промежуток, начиная с указанной даты (см. **FlipTime**)
Поле ввода, указывающее через сколько дней изменится алгоритм	Установка периодичности изменения определителя алгоритма для механизма **FlipTime** (в днях)
Комбинированное поле ввода/календарь для установки даты первого изменения алгоритма	Установка календарной даты активации механизма **FlipTime**

FlipTime. Изменение ответов алгоритма через указанный период времени

В ключах с RTC реализована технология **FlipTime**, позволяющая автоматически изменять значения, возвращаемые алгоритмом ключа по наступлению заданного временного значения.

Важно!

Технология FlipTime неприменима к ячейкам типа **Загружаемый код** в ключах Guardant Code Time!

FlipTime – это механизм, модифицирующий часть определителя алгоритма по достижению указанной при программировании ключа даты. Причем это изменение не однократное, определитель будет продолжать изменяться через заданный промежуток времени (в днях). Соответственно, всякий раз после изменения определителя, алгоритм будет возвращать другие значения в ответ на запросы.

Чтобы использовать механизм **FlipTime**, разработчик должен знать, какие ответы вернет алгоритм в каждом случае. Для решения этой задачи в комплект разработчика включена консольная утилита

FlipTime.exe, генерирующая массивы вопросов-ответов алгоритму для каждого факта изменения определителя.

Чтобы активировать механизм **FlipTime**:

1. Выделите в маске ключа с RTC нужный алгоритм, выполните команду меню **Редактировать | Свойства поля** и перейдите на вкладку **Временные зависимости**.

2. Установите флаг **Алгоритм изменяется каждые...** и в появившемся поле задайте период (в днях) изменения алгоритма.

3. С помощью комбинированного поля/календаря определите дату, по достижении которой механизм **FlipTime** будет задействован.

4. Сохраните маску и запишите ее в ключ (команда меню **Ключ | Запись в ключ**).

5. Запустите утилиту **FlipTime.exe** и, следуя ее указаниям, получите массивы вопросов-ответов алгоритма после каждого факта изменения определителя.

6. Используйте полученные массивы в приложении согласно заданным временным зависимостям.

Важно!

1. Технология FlipTime предназначена для использования только при защите с помощью Guardant API. Недопустимо ее использование при автозащите!

2. Пользоваться FlipTime следует с известной осмотрительностью, т. к. если в расчетах при проектировании защиты, программировании ключа или защите приложения будет допущена ошибка, то ее достаточно сложно будет диагностировать и исправлять в «боевых» условиях, когда ключ находится у конечного пользователя.

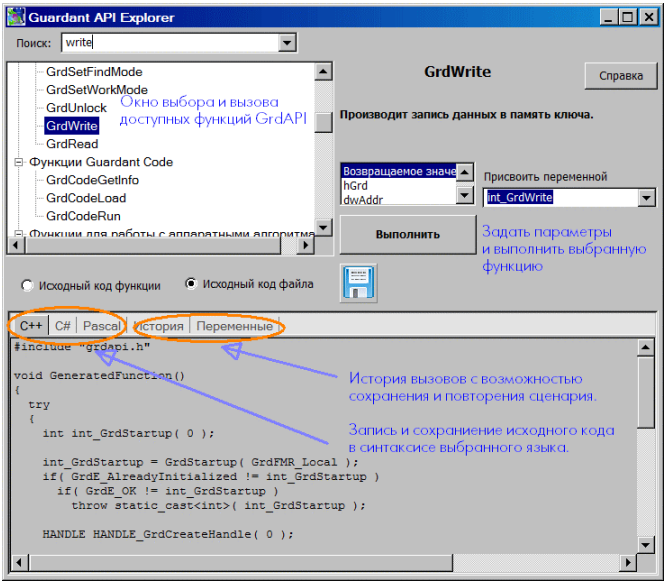
Технология **FlipTime** дает разработчику возможность реализовать изощренную стратегию защиты, при которой защитные механизмы будут видоизменяться через заданное время без дополнительного перепрограммирования ключа.

Обозреватель Guardant API. Выполнение функций с заданными параметрами

Утилита **Обозреватель Guardant API** предоставляет удобный GUI-сервис для изучения Guardant API и выполнения функций с заданными параметрами.

Это существенно экономит время при разработке защиты для приложения: ведь вызывать функции Guardant API, проверять и сохранять результаты их вызова в синтаксисе основных языков программирования можно без компиляции и отладки приложения.

Обозреватель доступен как из интерфейса GrdUtil.exe (команда меню **Разное | Обозреватель Guardant API**), так и в виде отдельной утилиты (см. файл **GAPIE_GUI_SE.exe** в каталоге установке комплекта разработчика).



Диалоговое окно утилиты разделено на три секции:

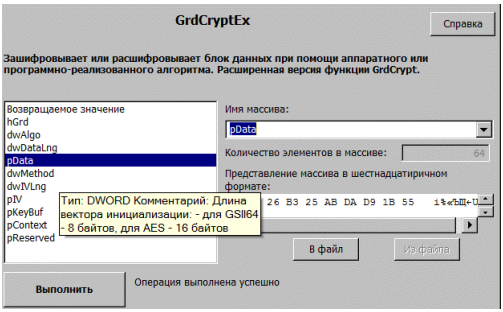
1) В левой верхней части происходит выбор функций, доступных для вызова.

Существует особая логика вызова функций Guardant API*. Поэтому обозреватель не показывает функции, которые в данный момент вызывать нельзя.

2) Справа отображается выбранная из списка функция.

Для ее вызова необходимо задать корректные значения параметров и нажать на кнопку **[Выполнить]**.

* Подробнее см. пример ниже, а также **GrdAPI.chm**, доступный из обозревателя по нажатию на кнопку **Справка**



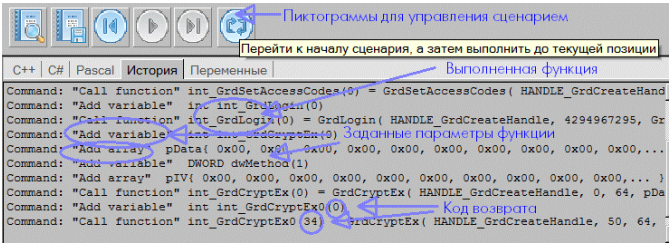
Параметры задаются последовательно путем выделения в списке курсором мыши и установки/выбора нужного значения. Если параметр не задан или задан неправильно, обозреватель выдает сообщение об ошибке.

Всплывающие по наведению курсора подсказки и контекстно-зависимая справочная система (см. кнопку **[Справка]**), позволяют получить информацию по текущей функции Guardant API.

3) Нижняя часть утилиты разделена на вкладки, в 3-х из которых выводится исходный код выполненной функции в синтаксисе **C++**, **C#** и **Delphi**, соответственно (см. скриншот на стр. 123).

Можно сохранять код как каждой функции отдельно, так и всех выполненных функций – в файле, готовом для компиляции.

Последние вкладки (**История** и **Переменные**) представляют собой историю вызовов функций и их результаты, а также список переменных и их значений.



История вызовов сохраняется в виде сценария (файл с расширением .sce). При необходимости можно возвращаться к любой из выполненных функций, используя для этого управляющие пиктограммы, расположенные над вкладками.

Начинаем работать с Обозревателем. Логин на локальный ключ

Регистрация на ключ – этой простейшая рутинная последовательность операций, которая выполняется каждый раз в начале работы с любым ключом. Успешный логин делает возможным всю дальнейшую работу с ключом.

Важно!

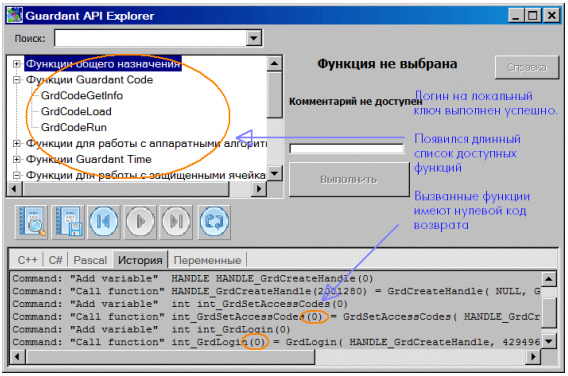
Для логина должен использоваться ключ с кодами доступа, которые были заданы при установке комплекта разработчика.

Сетевые ключи Guardant могут выступать в качестве локальных.

Чтобы при помощи Обозревателя Guardant API выполнить регистрацию на локальном ключе Guardant любой модели, выполните следующие действия:

1. Откройте Обозреватель любым из следующих способов:
   - Запустите файл **GAPIE_GUI_SE.exe** из каталога установки комплекта разработчика Guardant
   - Запустите утилиту GrdUtil.exe и выполните команду меню **Разное | Обозреватель GrdAPI**.
2. Из списка функций выберите **GrdStartup** и нажмите на кнопку **[Выполнить]** в правой части Обозревателя.
3. После ее успешного выполнения выберите появившуюся функцию **GrdGreateHandle** и выполните ее.
4. Выполните функцию **GrdSetAccessCodes**.
   Предварительно убедитесь, что в параметрах функции корректно указаны значения персональных кодов доступа к ключу.
5. Выполните функцию **GrdLogin**.

Об успешном выполнении логина на локальный ключ будут свидетельствовать нулевые коды возврата вызванных функций (включая **GrdLogin**), во вкладке **История**. Кроме того, в левой части диалога Обозревателя появится список с доступными функциями GrdAPI:



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 используемого образа

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

<table>
<tr><th rowspan="2">Вариант обновления</th><th colspan="3">Подвид образа</th><th rowspan="2">Возможный тип обновления</th></tr>
<tr><th colspan="2">Файл *.nsd</th><th>Шаблон из БД</th><th>Прошивка из БД GrdUtil</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"></td></tr>
<tr><td>Частичное обновление памяти</td></tr>
<tr><td>Операции активации/деактивации</td></tr>
<tr><td>Обновление времени деактивации алгоритма (для Guardant Time)</td></tr>
<tr><td rowspan="4">Локальное</td><td colspan="2" rowspan="4">Полная перезапись содержимого памяти ключа, подсоединенного к локальному компьютеру, при помощи операции Ключ | Запись в ключ</td><td>Полное обновление памяти</td></tr>
<tr><td>Частичное обновление памяти</td></tr>
<tr><td>Операции активации/деактивации</td></tr>
<tr><td>Обновление времени деактивации алгоритма (для ключей с RTC)</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.

Важно!

1. Процедура дистанционного обновления современных ключей Guardant требует наличия у конечного пользователя утилиты **GrdTRU.exe**. Этот файл должен входить в комплект поставки защищенного приложения.

2. Процедура дистанционного обновления устаревших ключей Guardant Stealth II/Net II, Guardant Stealth/Net и Guardant Fidus требует наличия у конечного пользователя утилиты **GsRemote.exe**. Этот файл должен входить в комплект поставки защищенного приложения.

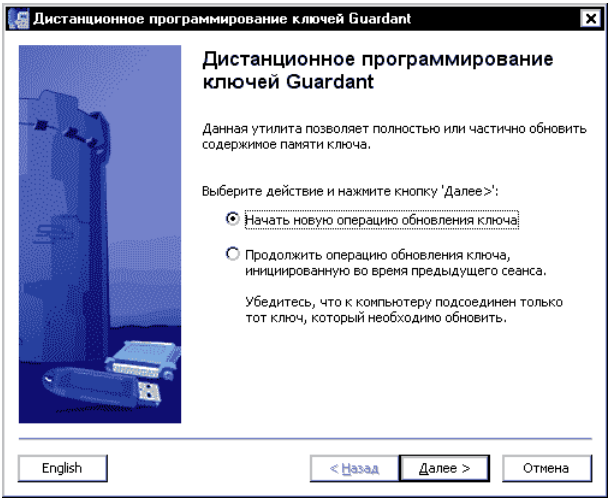
3. Для успешного запуска клиентской утилиты удаленного обновления к порту компьютера должен быть подсоединен ключ с кодами доступа разработчика.

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

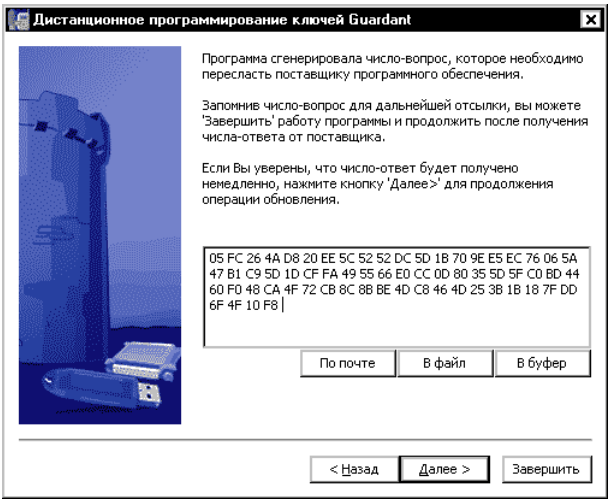
5. Информацию, касающуюся работы с клиентской утилитой, рекомендуется включать в инструкцию для конечного пользователя.

Клиентская утилита удаленного программирования выполнена в виде мастера, состоящего из нескольких страниц. Переход между страницами осуществляется при помощи кнопок **[Далее]** и **[Назад]**, расположенных в нижней части диалога.

1. После первого запуска утилиты на экране появится страница, на которой необходимо выбрать пункт **Начать новую операцию обновления ключа**:



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 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.

Важно!

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

2. Обновление современных ключей Guardant 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 образа записывается в ключ перед продажей конечному пользователю, а также сохраняется в прошивке.

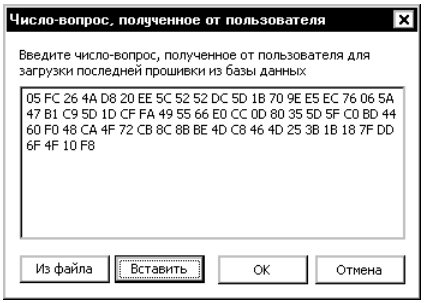
1. Перед началом процедуры получения дампа обновления загрузите в Редактор маску, которая будет использована для обновления удаленного ключа, и произведите в ней необходимые изменения.

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

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

Чтобы автоматически загрузить в Редактор образа 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 образа.

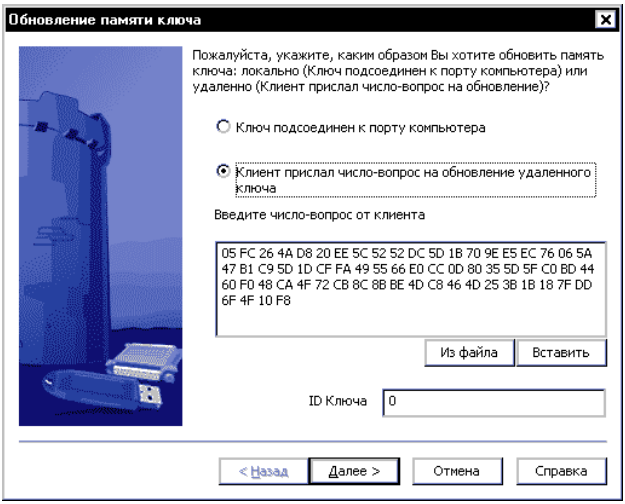
Внесите необходимые изменения в прошивку и начните процедуру создания дампа обновления.

Важно!

Если в ходе редактирования шаблона образа 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:

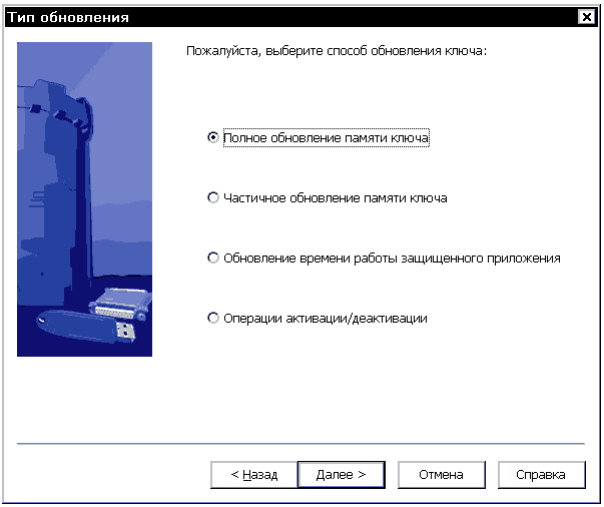
- **[Вставить]**, если число-вопрос находится в буфере обмена
- **[Из файла]**, если число-вопрос хранится в текстовом файле

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

Важно!

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 образа, на основе которого выполняется перепрограммирование ключа (прошивка, шаблон или файл образа), и от типа ключа:



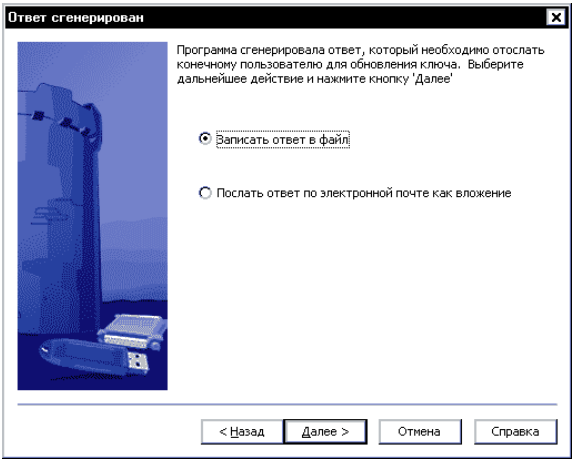
Выберите нужный пункт обновления и нажмите на кнопку **[Далее]**. Следующие несколько страниц служат для задания параметров обновления. Подробно о каждом из вариантов обновления см. далее в разделе **Тип обновления**.

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

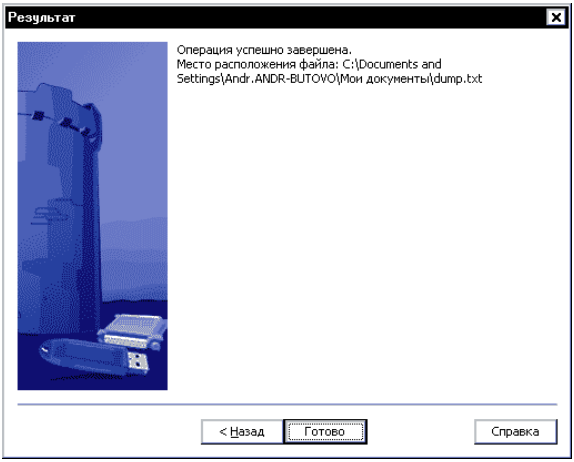
Важно!

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 образа 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 перепрограммировать память ключа.

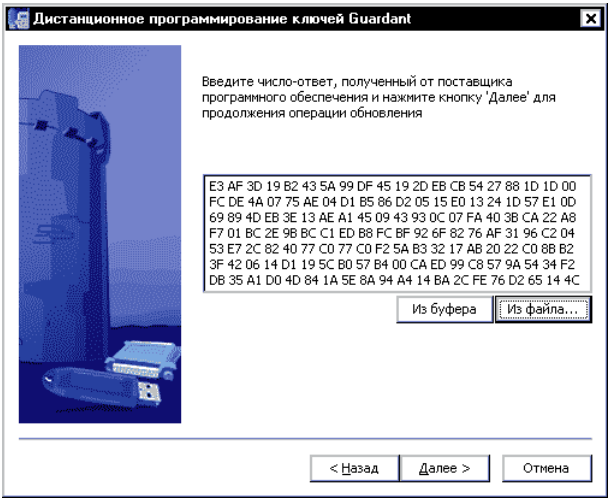
Важно!

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

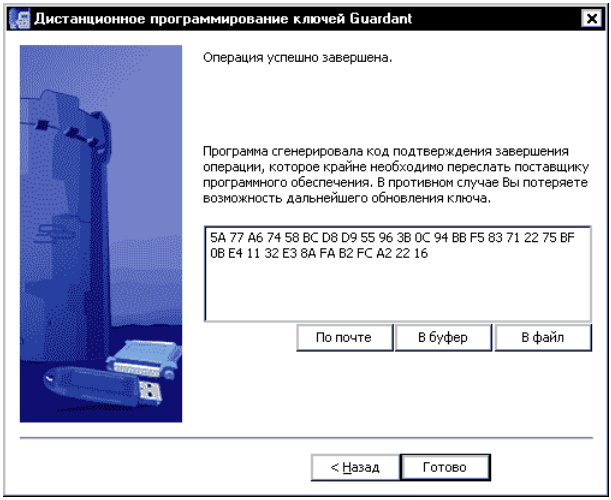
2. При генерировании данных обновления и прошивке их в ключ выполняется ряд проверок параметров ключа (ID, Общий код и некоторые другие). Этим исключается возможность подмены перепрограммируемого ключа

Необходимо снова запустить клиентскую утилиту, 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:



Важно!

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

3а. Передача разработчику кода-подтверждения

В процессе обновления памяти ключа клиентская утилита 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

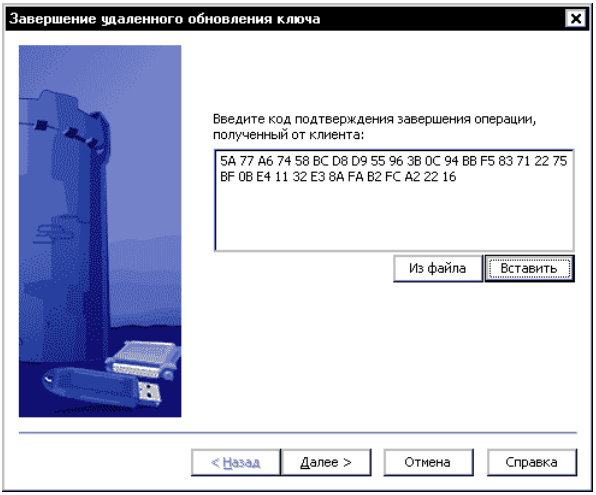
После получения кода-подтверждения разработчику необходимо завершить обновление: занести в базу данных информацию об итогах состоявшегося обновления.

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

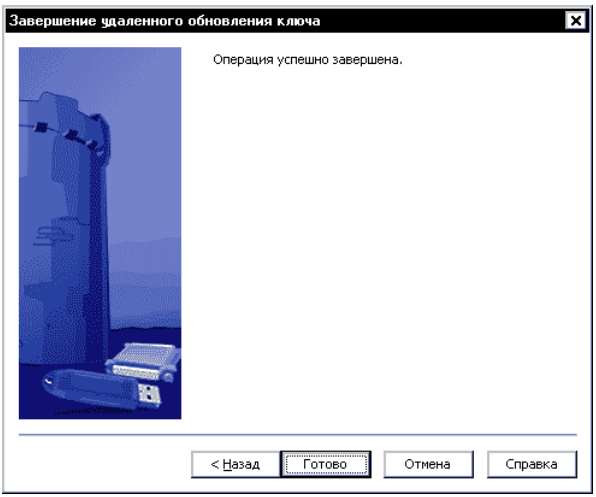
Важно!

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

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

Доверенное удаленное обновление – технология безопасного удаленного обновления памяти электронного ключа, исключающая возможность компрометации и/или фальсификации данных. 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.

Важно!

Процедура Доверенного удаленного обновления предполагает использование уникальных данных, в частности, пароля удаленного обновления, для каждого ключа. Поэтому настоятельно рекомендуется 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.

Важно!

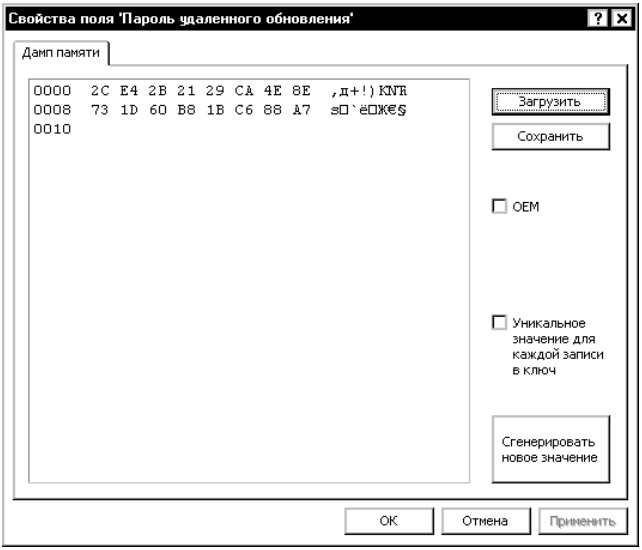
Для успешного выполнения Доверенного удаленного обновления пароль, содержащийся в удаленном ключе, должен совпадать с паролем, хранящимся в прошивке для этого ключа.

Пароль удаленного обновления содержится в неадресуемом, доступном для редактирования поле, которое расположено в образах современных ключей Guardant 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.

Элемент интерфейса	Назначение
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.

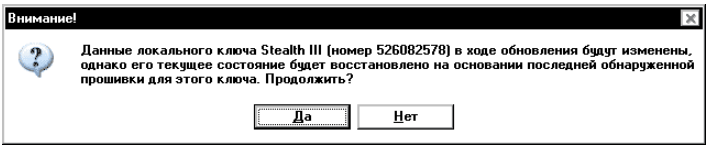
Особенности процедуры Доверенного удаленного обновления

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

При выполнении команды **Ключ | Обновление ключа** GrdUtil.exe автоматически записывает в память «мастер»-ключа, подсоединенного к порту, специальные алгоритмы. 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:



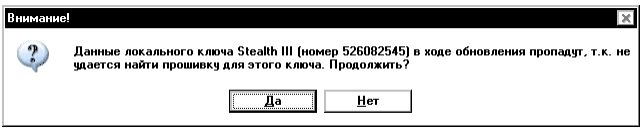
Если удаленное обновление выполняется без использования базы данных, на основе файла образа, то запрос на обновление не будет декодирован при несовпадении паролей удаленного обновления в файле образа и в удаленном ключе!!!

Локальное обновление

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

---

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



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 образа.

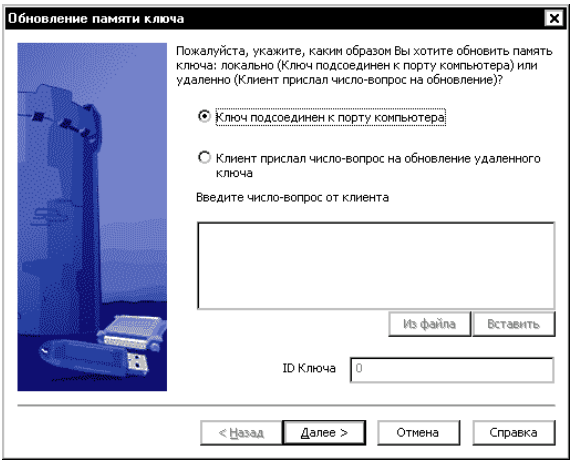
Особенности способов локального обновления:

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 текущего образа

Локальное обновление командой Ключ | Обновление ключа

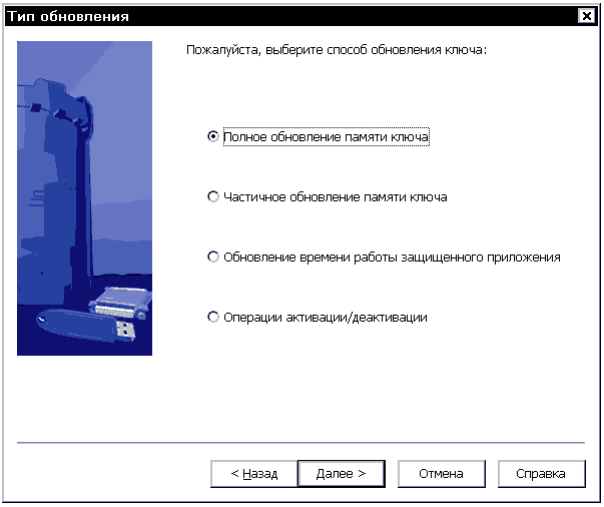
Практическая необходимость в локальном обновлении путем выполнения команды **Ключ | Обновление ключа** возникает, когда требуется сохранить в ключе «наработанные» данные (счетчики алгоритмов, к примеру), т. е. произвести изменение содержимого отдельных полей, не затрагивая остальные поля. В остальных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



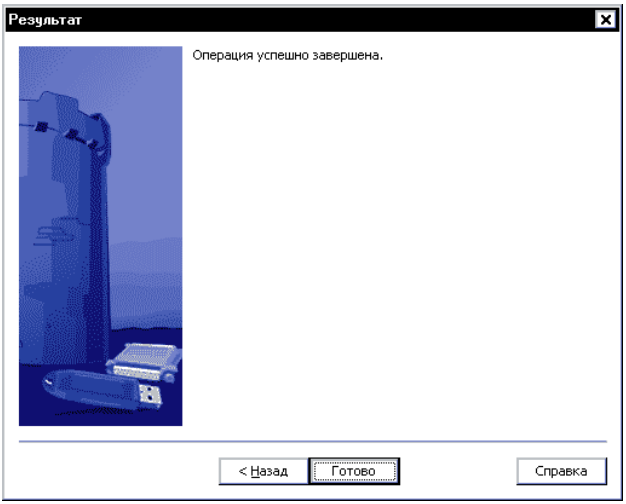
Выберите с помощью переключателя пункт **Ключ подсоединен к порту компьютера** и перейдите на следующую страницу.

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 текущего образа
Частичное обновление памяти ключа	Перезапись содержимого полей памяти ключа, не защищенных аппаратным запретом на запись. Можно обновить только одно поле за процедуру обновления
Операции активации/ деактивации	Активация, деактивация или перезапись выбранного участка защищенной ячейки, алгоритма или таблицы лицензий.
Изменение времени деактивации алгоритма (для ключей с RTC)	Продление времени работы приложения, защищенного ключом с часами реального времени в режиме лицензирования **Установить время деактивации**

Полное обновление памяти ключа

При полном обновлении памяти ключа происходит перезапись всей памяти ключа данными выбранного образа.

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 образа.

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 современных ключей Guardant можно обновить только один участок памяти.

Частичное обновление памяти актуально для тех ситуаций, когда необходимо сохранить «наработанную» в ключе информацию (поля счетчиков), и полное обновление памяти неприемлемо. Частично обновить память можно только при обновлении от прошивки, сохраненной в базе данных.

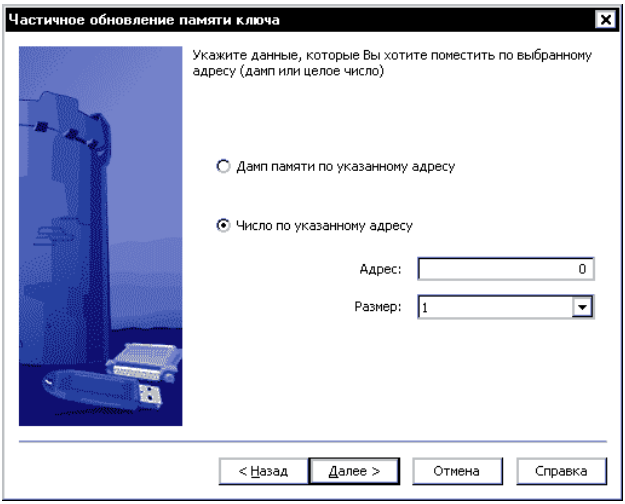
Чтобы обновить участок памяти ключа, не защищенный аппаратным запретом на запись, выполните команду **Ключ | Обновление ключа**. На экране появится диалог мастера обновления.

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)
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:



Важно!

Поля памяти, которые необходимо обновить, должны быть предварительно созданы в Редакторе образа.

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 счетчика GP (времени работы приложения)

Важно!

Устаревший режим обновления, изменяющий значение счетчика GP. Этот режим не подходит для обновления чистого времени работы приложений, использующих счетчик алгоритма (опция автозащиты /DTA).

Для обновления времени работы приложений, защищенных в указанном режиме (версия автозащиты 5.3 и выше) необходимо использовать **Полное обновление памяти ключа**.

Чтобы увеличить значение счетчика GP, выполните команду **Ключ | Обновление ключа**. На экране появится диалог мастера обновления.

Операции активации/ деактивации

Для современных ключей Guardant 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, сохраненной в базе данных.

Важно!

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**:

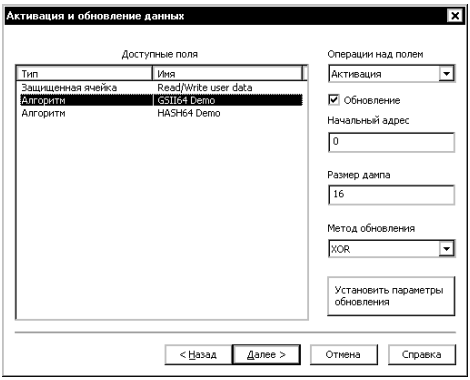
№	Страница	Действия разработчика
1	Обновление памяти	Выберите удаленное или локальное обновление. При удаленном обновлении введите в поле запрос на обновление от конечного пользователя
2	Тип обновления	Выберите пункт **Операции активации/ деактивации**
3	Активация и обновление данных	Выберите защищенную 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

Изменить время деактивации алгоритма. Технология Time

Данный вариант предназначен для ситуаций, когда необходимо продлить срок работы приложения, защищенного ключом Guardant Time в режиме лицензирования **Деактивировать алгоритм по наступлению указанной даты**.

Его достоинство заключается в возможности продлевать время работы приложения, не затрагивая остальную память.

Чтобы изменить время деактивации аппаратного алгоритма в ключе с RTC, выполните команду **Ключ | Обновление ключа**. На экране появится диалог мастера обновления.

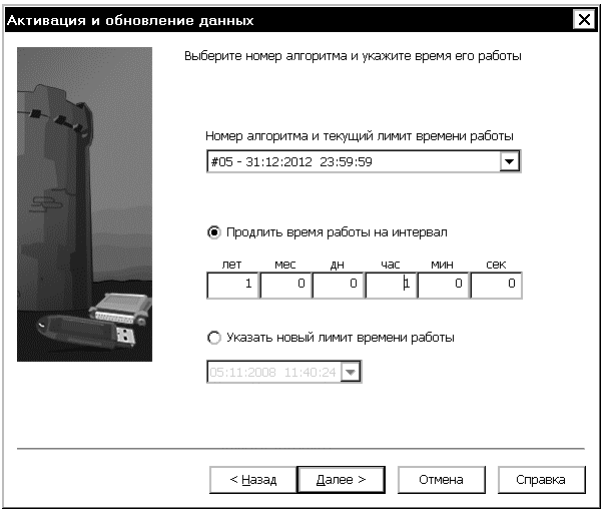
Список страниц мастера обновления в режиме **Изменить время деактивации алгоритма**:

№	**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 **Изменить время деактивации алгоритма**
3	Изменение времени деактивации	Выберите аппаратный алгоритм и задайте новое время деактивации (см. скриншот и описание после таблицы)
4	Ответ сгенерирован	Выберите вариант сохранения дампа обновления
5	Результат	Проконтролируйте результаты работы мастера обновления

Страница «Изменение времени деактивации»

На странице **Изменение времени деактивации** выберите аппаратный алгоритм, который нуждается в изменении срока деактивации, и укажите для него новый лимит времени работы или интервал работоспособности.

Диалог **Изменение времени деактивации**:



Элементы управления диалога **Изменение времени деактивации**:

Элемент интерфейса	**Назначение**
Разворачивающийся список **Номер алгоритма и текущий лимит времени работы**	Выбрать из списка аппаратный алгоритм, для которого необходимо изменить срок деактивации
Переключатель **Продлить время работы на интервал**	Выбрать вариант, при котором деактивация откладывается на заданный интервал
Поле для ввода нового интервала	Задать интервал работоспособности
Переключатель **Указать новый лимит времени работы**	Выбрать вариант, при котором задается календарная дата новой деактивации
Комбинированное поле/календарь	Задать новую дату деактивации

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 образа 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 образа в ключ
- Удаленное обновление ключа
- Завершение процедуры удаленного обновления ключа
- Локальное обновление ключа

Важно!

1. Для успешного выполнения операций тип записываемого образа и тип программируемого ключа должны совпадать.

2. При локальных операциях 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 образа в ключ

Для записи содержимого образа в ключ служит команда ***-write***.

В процессе выполнения команды происходит перезапись всей памяти ключа данными выбранного образа. В качестве источника данных может использоваться шаблон образа из базы данных или образ из файла *.nsd.

Команда ***-write*** имеет несколько опций:

Опция	Описание
-mask(name, version)	Записать в ключ содержимое шаблона образа из БД GrdUtil. **name, version** – имя и версия шаблона образа
-user(user_name)	Зарегистрировать прошивку ключа на указанного пользователя. **user_name** – имя конечного пользователя

Опция	Описание
-infile(file.nsd)	Записать в ключ содержимое файла образа. **file.nsd** – имя файла формата .nsd, который используется как источник обновления
-dongleID(hex)	Выполнить запись в ключ с указанным ID (в случаях, когда к портам подсоединены несколько ключей)

Опции команды ***-write*** используются в следующих сочетаниях:

Сочетания опций	Описание
-write –mask(name, version)	Записать в ключ шаблон образа с указанным именем и версией. Прошивка будет зарегистрирована в базе данных на конечного пользователя **Anonymous**
-write –mask(name, version) -user(user_name)	Записать в ключ шаблон образа 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
-write –infile(file.nsd) -mask(name, version)	Записать в ключ маску из указанного файла. Прошивка с указанным именем и версией будет зарегистрирована в базе данных на конечного пользователя **Anonymous**
-write –infile(file.nsd) -mask(name, version) -user(user_name)	Записать в ключ маску из указанного файла. Прошивка с указанным именем и версией будет зарегистрирована в базе данных на **указанного** конечного пользователя

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 образа из базы данных и образ из файла .nsd. При обновлении происходит полная перезапись содержимого ключа.

Полученный дамп обновления необходимо передать конечному пользователю для завершения процесса удаленного обновления. Подробное описание процедуры удаленного обновления см. в разделе **Удаленное обновление**.

Команда ***-remote(query, outfile)*** имеет несколько опций:

Опция	Описание
-mask(name, version)	Создать дамп обновления на основе содержимого шаблона образа. **name, version** – имя и версия шаблона образа из базы данных GrdUtil.exe
-infile(file.nsd)	Создать дамп обновления на основе содержимого файла образа. **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 образа** с указанным именем и версией. Прошивка будет зарегистрирована в базе данных на **текущего** конечного пользователя
-remote(query, outfile) -infile(file.nsd)	Создать дамп обновления на основе содержимого указанного **файла образа**.
-remote(query, outfile) -infile(file.nsd) -mask(name, version)	Создать дамп обновления на основе содержимого указанного **файла образа**. Прошивка с указанным именем и версией будет зарегистрирована в базе данных на **текущего** конечного пользователя

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 образа из базы данных и образ из файла .nsd. При обновлении происходит полная перезапись содержимого ключа.

Команда ***-local*** имеет несколько опций:

Опция	Описание
-mask(name, version)	Записать в ключ содержимое шаблона образа. **name, version** – имя и версия шаблона образа
-infile(file.nsd)	Записать в ключ содержимое файла образа. **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 образа с указанным именем и версией. Прошивка будет зарегистрирована в базе данных на текущего конечного пользователя
-local –infile(file.nsd)	Записать в ключ содержимое указанного файла образа

Получение кода возврата GrdUtil

Если при работе с командной строкой GrdUtil необходимо получать код возврата, то для этой цели следует использовать WinAPI.

Так, например, при создании дочернего процесса с помощью **CreateProcess()**, код ошибки может быть получен при помощи функции **GetExitCodeProcess()**.

Распространенные коды возврата:

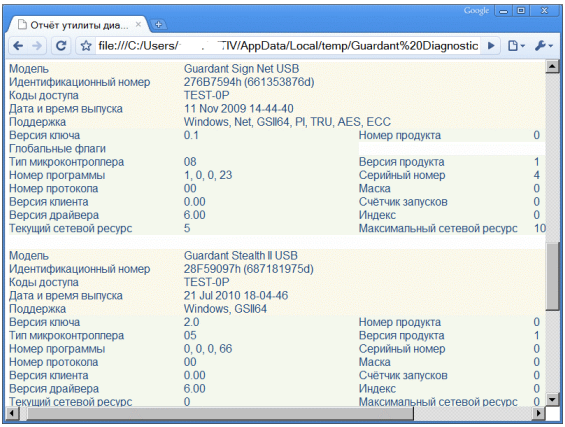
Код ошибки	Описание
0x00000000	Операция завершилась успешно
0x00000001	Ошибка в процессе записи ключа
0x00000002	Не найден файл
0x1000019F	Ошибка задания параметров командной строки
0x100001B4	При обновлении указано некорректное число-вопрос

Вспомогательные операции

В этот раздел вынесено описание вспомогательных (**Получение информации о ключе**), а также редко используемых (**Конвертирование образа**) возможностей утилиты.

Получение информации о ключе

Чтобы получить информацию о ключе, подсоединенном к порту компьютера, выполните команду **Ключ | Информация о ключах**:



GrdUtil.exe выдает ту же информацию о ключе, что и утилита диагностики ключа: значения полей доступных только для чтения, полей специальных операций, глобальных флагов, таблицы лицензий и полей общего назначения.

Запрет на изменение времени в ключах с таймером

В ключах Guardant Time/ Time Net/ Code Time можно задавать новое значение для встроенного таймера при помощи специальной функции Guardant API (см. описание **GrdSetTime** в **GrdAPI.chm**).

Однако если такая необходимость и возникает, то, как правило, на этапе программирования ключа (например, при реализации утилиты прошивки ключа, альтернативной **GrdUtil.exe**).

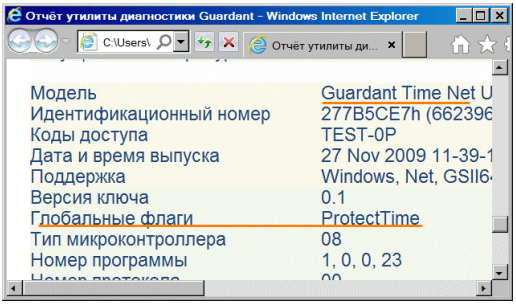
И, наоборот, для защиты приложения это несет угрозу, потому что при некоторых обстоятельствах (к примеру, компрометации кодов доступа), станет возможным перепрограммирование таймера ключа и незаконное продление лицензии.

Поэтому в ключах с таймером предусмотрена возможность блокировать на низком уровне вызов функции **GrdSetTime**.

Важно!

В RTC-ключах, поступающих с производства компании «Актив», а также в образах, создаваемых для RTC-моделей, блокировка времени уже выставлена по умолчанию (см. состояние флага **Запретить изменение времени в ключе** в Панели инструментов/ленточном интерфейсе GrdUtil.exe). Категорически не рекомендуется менять умолчательное значение без особой необходимости!

Чтобы проверить, блокирована ли возможность изменения времени в ключе, выполните его диагностику (см. предыдущий пункт или описание утилиты диагностики). Глобальный флаг **ProtectTime** должен быть установлен:



В случае отсутствия блокировки изменения времени в ключе загрузите в Редактор нужный образ RTC-ключа, проверьте состояние флага **Запретить изменение времени**, и если он не установлен,

выполните команду меню **Ключ | Запретить изменение времени в ключе**. После этого запишите образ в ключ.

В результате в ключе будет выставлен глобальный флаг **GrdGF_ProtectTime** (см. описание **GrdProtect** в **GrdAPI.chm**), и изменение состояния таймера станет невозможным без инициализации памяти ключа.

Конвертирование образа

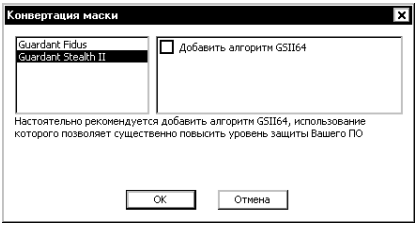
При поддержке нескольких типов ключей может возникать необходимость переноса информации из образа для ключей одного типа в маску другого типа (например, для сохранения сложившейся адресации полей).

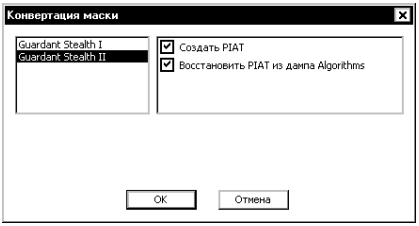
GrdUtil.exe предоставляет следующие варианты конвертирования образов:

		Образ-приемник							
		Sign/ Sign Net	Time/ Time Net	Code	Code Time	Stealth III /Net III	StealthII / Net II	Stealth / Net	Fidus
Образ передатчик	Sign/ Sign Net		+						
	Time/ Time Net	+							
	Code	+	+		+				
	Code Time	+	+	+					
	Stealth III / Net III	+	+	+	+				
	StealthII /Net II							+	+
	Stealth/Net						+		+
	Fidus						+	+	

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 ***образа-передатчика*** и ***образа-приемника***:

Опция	**Образ-передатчик**	**Образ-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 образа-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

Глава 5
Автоматическая защита

Автоматическая защита Guardant поддерживает 32-разрядные Windows-приложения и предназначена для обработки исполняемых файлов Native-приложений (*.exe), а также .NET-сборок (*.exe, *.dll).

Важно!

1. Автоматическая защита может быть установлена на приложение только при наличии в порту ключа нужной модели!

2. Несмотря на свои богатые возможности, автозащита не может гарантировать повышенной защищенности программного продукта. Настоятельно рекомендуем усилить защиту при помощи Guardant API!

Возможности автоматической защиты

Автоматическая защита Guardant предоставляет широкие возможности для защиты приложений:

Возможности автозащиты	Тип Win32-приложения	
	Исполняемые Native-приложения (*.exe)	.NET-сборки (*.exe, *.dll)
Поддержка локальных ключей, включая SP	+	+
Поддержка сетевых ключей	+	+
Использование для защиты произвольных алгоритмов типа GSII64 и AES	+	+
Защита без привязки к ключу	+	+
Привязка к уникальному параметру ключа (ID)	+	+
Привязка к серийному номеру и версии ключа	+	+
Защита нескольких продуктов или разных версий одного продукта	+	+
Защита многомодульных комплексов	+	+
Ограничение времени работы приложения	+	-
Ограничения числа запусков приложения	+	-
Проверка ключа через заданный промежуток	+	-
Контроль присутствия USB-ключа в порту	+	-
Защита импортируемых функций	+	-
Кодирование и упаковка приложения	+	-
Шифрование строк .NET-сборки	-	+
Обфускация .NET-сборки	-	+
Защита кода .NET-сборки	-	+

Инструментарий автозащиты

Инструментарий автоматической защиты Guardant состоит из нескольких консольных утилит (утилита для защиты Native-приложений, и две отдельные утилиты для обфускации и защиты кода .NET-сборок), а также утилит с графическим интерфейсом:

Утилиты автозащиты		Назначение
Название	**Тип**	
NwKey32.exe	Консольные	Защита обычных (Native) exe-файлов
CodeObfuscator.exe		Обфускация и шифрование строк .NET-сборок
CodeProtect.exe		Защита кода .NET-сборок
LicenseWizard.exe	GUI	Графическая оболочка для лицензирования и защиты исполняемых Native-файлов и .NET-сборок
NativeProfilerGUI.exe		Native-профилировщик
DotNetProfilerGUI.exe		.NET-профилировщик

Механизмы защиты Native-приложений и .NET-сборок имеют существенные отличия, поэтому далее рассматриваются отдельно:

- Автоматическая защита исполняемых файлов Native-приложений
- Автоматическая защита .NET-сборок

Автозащита исполняемых файлов Native-приложений

Принцип автозащиты

Основа автоматической защиты исполняемых Native-приложений – вакцина, выполненная в виде универсального внешнего модуля. Все функции защиты поддерживаются этим модулем, что позволяет унифицировать процесс защиты.

Автозащита исполняемых Native-приложений работает следующим образом:

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.

Кроме того, использование внешнего модуля защиты усиливает стойкость к изучению логики ее работы при помощи отладчиков.

Файл вакцины для исполняемых Native-приложений называется **GrdVkc32.dll** и входит в комплект автоматической защиты.

Важно!

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).

Ограничения автозащиты Native-приложений

Важно!

1. Автозащита должна выполняться на ключе той же модели, что будет поставляться с защищенной программой.
2. Для успешной установки и работы автозащиты в ключе, к которому привязывается приложение, должен содержаться алгоритм типа GSII64 или AES.
3. Определитель аппаратного алгоритма в ключе, используемом при защите, должен быть идентичен определителю этого же алгоритма в ключе из комплекта поставки защищенного приложения.

- 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

NwKey32.exe. Консольная утилита автозащиты Native-приложений

Защита исполняемых файлов Native-приложений (т. е. EXE-файлов PE-формата) производится строчной утилитой **NwKey32.exe**.

Утилиту автозащиты с выбранными опциями можно вызывать из командной строки или 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 процесса защиты

Для защиты исполняемого Native-приложения необходимо наличие следующих файлов в одной директории:

Файл	Описание
NwKey32.exe	Строчная утилита автоматической защиты
GrdVkc32.dll	Внешняя вакцина
NvCodes.dat	Служебный файл с информацией о кодах доступа к ключу
NwKey32.msg	Файл с сообщениями об ошибках
ImportWalker.dll	Реализация защиты таблицы импорта от восстановления (см. опцию /IMPORT_HOOK)
ProfilerManager.dll	Библиотека Native-профайлера

Важно!

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 Native-приложения

Для работы защищенного приложения необходимо наличие следующих файлов в одной директории:

Файл	Описание
GrdVkc32.dll	Внешняя вакцина

Порядок защиты Native-приложения

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 на кнопку **[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.**

Коды ошибок NwKey32.exe

Утилита NwKey32.exe 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	Некорректно заданы алгоритмы для привязки к ключу; алгоритм, указанный в опции SIGN[=N:FileName.Ext], не является ECC
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

Код ошибки	Описание
34	Ошибка чтения из файла
35	Ошибка записи в файл
36	Переполнение. Выравнивание сегмента слишком велико
37	Невозможно защитить файл базы данных
38	Невозможно защитить DLL-файл
39	Слишком много сегментов в защищаемом файле (более 32)
40	Найден распределенный сегмент
41	Невозможно изменить режим преобразования 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

Сводная таблица опций защиты Native-приложений

Опции автоматической защиты Native-приложений 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	Модель
/GS3S[=[N]:[L]:[ID]:[S[<FileName.bin>]]	Привязать к Guardant Time/Sign	Time/Sign
/GN3S[=[N]:[L]:[ID]:[S]:[<FileName.bin>]]	Привязать к Guardant Time/SignNet	Time/Sign Net
/GC=N:L[:[ID]:[S]:[<FileName.bin>]]	Привязать к Guardant Code/CodeTime	Code/Code Time
/GS3[=[N]:[L]:[ID]]	Привязать к Guardant Stealth III	StealthIII/Net III
/GN3[=[N]:[L]:[ID]]	Привязать к Guardant Net III	Net III
/GS2[=[N]:[L]:[ID]]	Привязать к Guardant Stealth II	Stealth II /Net II
/GN2[=[N]:[L]:[ID]]	Привязать к Guardant Net II	Net II
/GSP[=[N]:[L]:[ID]:[S[<FileName.bin>]]	Привязать к софтверному ключу Guardant SP	SP

Опции привязки к электронному ключу

Опция	Описание	Модель
/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

Опция	Описание	Модель
/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 образа ключа)	Все
/UN[=[0x]...]	Проверять номер программы (используется заданное значение, либо значение из поля номера программы ключа)	Все
/NOA	Не использовать аппаратные алгоритмы	Все
/T=xx	Проверять наличие электронного ключа периодически, через заданные интервалы времени	Все
/EXIT_DELAY	Отложить завершение приложения при отсутствии ключа	Все
/USB_DONGLE_CONTROL	Контролировать извлечение USB-ключа из порта компьютера	Все с USB-интерфейсом
/RC[=xx]	Если ключ не найден, проверять его наличие и выводить сообщение об отсутствии ключа заданное число раз	Все

Опции, влияющие на защищенность приложения

Опция	Описание	Модель
/CEN	Не кодировать загружаемую часть приложения	Все
/IDEN	Отключить кодирование инициализированных данных	Все
/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
/IMPLICIT_LINKING_SUPPORT	Поддержка неявного связывания DLL	Все
/IMPORT_HOOK [=%:L]	Защищать импортируемые функции	Все
/IMPORT_HOOK_LIST	Защищать импортируемые функции по списку	Все
/RIP_CODE [=%[:FileName.Ext]]	Извлечь инструкции из тела приложения	Все
/RIP_CODE_LIST	Извлечь инструкции из тела приложения по списку	Все

Опции, касающиеся ограничений работы приложения

Опция	Описание	Модель
/LICENSE_TIME [=limit]	Вывести предупреждение об оставшемся времени работы	Для ключей с RTC
/DCA	Ограничить число запусков приложения	Локальные
/LICENSE_COUNTER [=limit]	Вывести предупреждение об оставшемся числе запусков	Только для Sign и выше
/LICENSE_URL= string	Вывести ссылку на сайт разработчика приложения	Только для Sign и выше

Сетевые опции

Опция	Описание	Модель
/LOGIN_MODE= H | S | P	Выбрать режим лицензирования: по хэндлам (H), рабочим станциям (S) или процессам (P)	Сетевые ключи
/MN=xx	Использовать систему управления лицензиями	Сетевые ключи

Сервисные опции

Опция	Описание	Тип ключа
/MSG= [путь]*.msg	Брать сообщения вакцины из файла *.MSG (имя_утилиты.MSG – по умолчанию)	Все
/SPLASH [=Filename.bmp]	Показать заставку при старте защищенного приложения	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	Все
/SILENT	Запретить вывод сообщений защищенного приложения	Все

Опции автоматической защиты

Опции установки типа электронного ключа

Опции этой группы позволяют задать модель электронного ключа Guardant, к которой будет «привязано» защищенное приложение.

Привязать к типу ключа:
/GS3S[=[N]:[L]:[ID]:[S]:[<FileName.bin>]],
/GN3S[=[N]:[L]:[ID]:[S]:[<FileName.bin>]],
/GC N:L[:[ID]:[S]:[<FileName.bin>]],
/GS3[=[N]:[L]:[ID]],
/GN3[=[N]:[L]:[ID]],
/GS2[=[N]:[L]:[ID]],
/GN2[=[N]:[L]:[ID]],
/GSP[=[N]:[L]:[ID]:[S]:[<FileName.bin>]]

Описание:

Указание модели ключей, к которым будет привязано приложение:

GS3S	GN3S	GC	GS3	GN3	GS2	GN2	GSP
Time/Sign	Time/Sign Net	Code	Stealth III	Net III	Stealth II	Net II	SP

Дополнительные параметры:

N	Номер симметричного алгоритма шифрования GSII64 или AES, который будет использован при автозащите. Для Guardant Code обязательный параметр
L	Длина вопроса алгоритму, 8 <= L<= 256, где L - число, кратное 8
ID	ID электронного ключа, к-й будет использован при установке защиты
S	Номер алгоритма электронной цифровой подписи ECC160

FileName.bin	Файл, содержащий открытый ключ ЭЦП для алгоритма ECC160. По умолчанию - PUBKEY_08.BIN, расположенный в текущей директории

Если при защите использовались опции этой группы, запуск защищенного приложения будет возможен только при наличии электронного ключа, т. к. приложение при защите настраивается на код доступа и привязывается к ключу заданного типа, подсоединенному к компьютеру на момент защиты.

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.

1. Нотация

Дополнительные параметры указываются через символ-разделитель – двоеточие. Необязательные параметры можно пропускать, при этом если за пропущенным следуют другие параметры, то символ **:** требуется печатать.

Пример:

/GS3S=:::2

Привязка к Guardant Sign с умолчательным алгоритмом шифрования и алгоритмом типа ECC160 под номером 2 (с открытым ключом по умолчанию); ID не указывается.

2. Алгоритм шифрования

При задании дополнительных параметров **N** и **L** в процессе защиты будут использоваться симметричные алгоритмы шифрования GSII64 или AES с указанным номером и длиной вопроса.

Если параметр **N** не задан, то процесс защиты будет выполнен с алгоритмом по умолчанию.

Параметры алгоритма по умолчанию зависят от типа ключа:

Модель	**Умолчательные параметры симметричного алгоритма шифрования**
Sign/Time	GS3S=0:8
Sign/Time Net	GN3S=0:8
Code	Умолчательный алгоритм отсутствует. Параметр **N** обязателен
Stealth III	GS3=0:8
Net III	GN3=0:8
Stealth II	GS2=4:8
Net II	GN2=4:8
SP	GSP=3:8

Если алгоритм по умолчанию для указанного типа ключа отсутствует в прошивке ключа, будет выдана соответствующая ошибка.

Если параметр **L** не задан, алгоритм будет вызван с длиной вопроса по умолчанию. При использовании неверного значения **L** будет выдана соответствующая ошибка.

3. Выбор ключа для защиты из нескольких подсоединненых к портам

Если к компьютеру подсоединены несколько ключей одной модели, то чтобы выбрать для проведения защиты определенный ключ, следует указать его ID при помощи одноименного параметра. **ID** задается в десятичном (ID=1234), или в шестнадцатеричном (0xABCD) виде.

4. Проверка цифровой подписи

При установленном параметре **S** защищенное приложение будет автоматически, наряду с регулярными вызовами **GrdTransform**, вызывать последовательность функций **GrdSign** – **GrdVerifySign** для выработки и проверки ЭЦП случайного числа.

Модель	**Умолчательные параметры алгоритма ЭЦП**
Sign/Time	GS3S=:::8:PUBKEY_08.BIN
Sign/TimeNet	GN3S=:::8:PUBKEY_08.BIN
Code	-
Stealth III	Не поддерживается
Net III	Не поддерживается
Stealth II	Не поддерживается
Net II	Не поддерживается
SP	GN3S=:::2:PUBKEY_08.BIN

Если при работе с алгоритмом ECC160 (параметр **S** установлен) не задан параметр **FileName.bin**, то будет использован умолчательный открытый ключ **PUBKEY_08.BIN** из текущего каталога.

Важно!

1. В ключах, которые передаются клиентам вместе с защищенным приложением, должны быть созданы алгоритмы с таким же номером, определителем и длиной запроса, какие были указаны при защите.

2. 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 /GS3S=5:::2 /GS3::12345678 MyProg.exe
```

Защищенное Win32-приложение MyProg.exe будет запускаться в случае, если к компьютеру подсоединен ключ Guardant Sign с симметричным алгоритмом #5 (длина вопроса по умолчанию) и ECC-

алгоритмом #2 (открытый ключ по умолчанию) или Guardant Stealth III с умолчательными параметрами и ID=12345678.

Причем для ключа Guardant Sign в процессе работы будет вырабатываться и проверяться цифровая подпись случайного числа, генерируемого вакциной.

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 можно использовать совместно в любой комбинации, а также с опциями из других групп, за исключением случаев, особо оговоренных ниже.

Важно!

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 шестнадцатеричной 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.

Важно!

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 /GN3S /UI myprog.exe

NwKey32.exe /GS2 /UI=912459016 myprog.exe

В первом примере защищенное Win32-приложение MyProg.exe будет запускаться только с тем из ключей Guardant Sign Net (опция **/GN3S**), 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 II (опция **/GS2**), который имеет ID=912459016 в десятичной системе счисления (опция **/UI=912459016**).

Проверять серийный номер ключа:
/US [=[0x]s]

Тип электронного ключа:

Все

Значение параметра:

0<=s<=65535

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.

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.

Важно!

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 /GS3S /US MyProg.exe

NwKey32.exe /GC /US=0xFFFE MyProg.exe

NwKey32.exe /GN3S /GS3 /US MyProg.exe

В первом примере защищенное приложение MyProg.exe будет запускаться только с тем из ключей Guardant Sign (опция **/GS3S**), 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 Code, который имеет серийный номер FFFE в шестнадцатеричной системе счисления (опция **/US=0xFFFE**).

В третьем примере MyProg.exe будет запущен только с тем из ключей Guardant Sign Net или Guardant Stealth III (опции **/GS3** и **/GN3S**), который имеет серийный номер, как у подсоединенного на момент защиты ключа Guardant Sign Net и Guardant Stealth III (опция **/US**).

Проверять версию защищенного приложения: /UV[=[0x]v]

Тип электронного ключа:

Все

Значение параметра:

0<=v<=255

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 про-

граммирования 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.

Пример:

NwKey32.exe /GS3S /UV MyProg.exe

NwKey32.exe /GS3S /GS3 /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 Sign 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 III или получит от вас новую улучшенную модель ключа Guardant Sign, с записанным значением 11 в поле версии. При этом он сможет работать как со старой, так и с новой версией MyProg.exe.

Проверять маску:
/UM[=[0x]m]

Тип электронного ключа:

Все

Значение параметра:

Единица, степени числа 2 в диапазоне 1<=m<65535

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 битовой маски 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 битовой маски 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 шестнадцатеричной 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 MyProg3.exe, т. к. числу 3, теперь записанному в ключе, соответствуют биты #0 и #1.

Разрядность поля битовой маски – 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.

Если указать опцию без параметра, то при защите будет использовано значение поля «Битовая маска», содержащееся в памяти ключа, подсоединенного к порту во время защиты.

Пример:

```
NwKey32.exe /GS3S /UM=1 MyProg1.exe
NwKey32.exe /GS3S /UM=2 MyProg2.exe
NwKey32.exe /GS3S /UM=4 MyProg3.exe
```

Показаны вызовы утилиты защиты приложений ключом Guardant Sign для примера, описанного выше по тексту.

Проверять номер программы:
/UN[=[0x]n]

Тип электронного ключа:

Все

Значение параметра:

0<=n<=255

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.

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 /GS3S /UN=0 mybuh.exe
NwKey32.exe /GS3S /UN=1 mysklad.exe
```

Показаны вызовы утилиты защиты приложений ключом Guardant Sign для примера, описанного выше по тексту.

Не использовать аппаратные алгоритмы: /NOA

Тип электронного ключа:

Все

Описание:

По умолчанию приложения, защищенные автозащитой, активно используют аппаратный алгоритм электронного ключа. Это значительно повышает уровень стойкости автоматической защиты, т. к. вводит в нее элемент преобразования данных.

Если же в памяти ключа по каким-то причинам нет алгоритмов GSII64 (для Guardant Code – AES), например, когда она используется для хранения других данных, автоматическая защита будет работать корректно лишь в том случае, когда она станет не обращаться к аппаратному алгоритму в процессе защиты.

Чтобы отказаться от использования аппаратного алгоритма для автоматической защиты, нужно запустить утилиту NwKey32.exe с опцией **/NOA**.

Пример:

```
NwKey32.exe /GS3S /NOA MyProg.exe
```

Защищенное таким образом приложение не будет использовать аппаратные алгоритмы GSII64 при проверке ключа Guardant Sign.

Контролировать извлечение USB-ключа из порта компьютера: /USB_DONGLE_CONTROL

Тип электронного ключа:

Локальные USB-ключи

Описание:

Опция позволяет защищенному приложению отслеживать присутствие локального USB-ключа Guardant в порту компьютера. Если в процессе работы приложения ключ извлекается из порта, то выводится соответствующее сообщение, и приложение реагирует, как это указано в описании опции **/RC**.

Пример:

NwKey32.exe /GS3S /USB_DONGLE_CONTROL MyProg.exe

Приложение MyProg.exe привязано к ключу Guardant Sign, причем происходит контроль присутствия ключа в порту. Если ключ вынуть во время работы приложения, то будет выдано сообщение типа «Не найден электронный ключ».

Проверять наличие электронного ключа периодически: /T=x

Тип электронного ключа:

Все

Значение параметра:

1<=x<=60 минут, только целые числа

Описание:

Можно проверять наличие электронного ключа не только при запуске, но и в течение всего сеанса работы защищенного приложения через заданные промежутки времени.

Такая возможность обеспечивается опцией **/T=x**, где «x» – значение промежутка между двумя проверками в минутах.

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.

Пример:

NwKey32.exe /GS3S /T=05 MyProg.exe

NwKey32.exe /GS3S /T=5 MyProg.exe

Эквивалентные примеры защиты приложения MyProg.exe с ключом Guardant Sign. MyProg.exe будет опрашивать ключ в течение всего времени работы, с интервалами в пять минут.

Выводить сообщение об отсутствии ключа заданное число раз: /RC[=x]

Тип электронного ключа:

Все

Значение параметра:

0<=x<=255

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[=x]**, где **x** - число проверок ключа.

Применение этой опции дает возможность пользователю сохранить наработанные данные и корректно завершить работу с приложением, например, в случае выхода ключа из строя.

Если опция не указана, сообщение будет выводиться 50 раз.

При отсутствии ключа на экран выводится диалог с сообщением и 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:

```
Nwkey32.exe /GC /RC=10 MyProg.exe
```

Приложение MyProg.exe будет проверять наличие ключа Guardant Code и при его отсутствии 10 раз выводить сообщение.

Отложить завершение приложения при отсутствии ключа: /EXIT_DELAY[=x]

Тип электронного ключа:

Все

Значение параметра:

1<=x<=600 секунд, по умолчанию 120

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.

Это даст возможность пользователю сохранить наработанные данные и корректно завершить работу с приложением, например, в случае выходе ключа из строя.

Пример:

```
Nwkey32.exe /GS3 /EXIT_DELAY=60 MyProg.exe
```

При отсутствии ключа Guardant Sign приложение MyProg.exe выведет сообщение о завершении работы приложения через 1 минуту.

Опции, влияющие на защищенность приложения

Отменить кодирование приложения: /CEN

Тип электронного ключа:

Все

Описание:

Данная опция отключает кодирование секций защищаемого приложения.

Пример:

```
NwKey32.exe /GS3S /T=10 /CEN MyProg.exe
```

Пример защиты приложения MyProg.exe с ключом Guardant Sign. MyProg.exe ***не будет закодирован*** (опция **/CEN**). Проверка наличия ключа Guardant Sign будет производиться приложением в течение всего времени его работы с интервалами в 10 минут.

Отключить кодирование 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 кодируются (за исключением ряда секций, которые всегда не кодируются).

Пример:

NwKey32.exe /GS3S /IDEN MyProg.exe

Инициализированные данные приложения MyProg.exe, защищенного ключом Guardant Sign, не будут закодированы.

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

Размер защищенного приложения MyProg.exe будет контролироваться. Приложение не привязано к ключу.

Не «очищать» приложение:
/NOS

Тип электронного ключа:

Все

Описание:

Обычно автозащита удаляет из Native-приложений данные, не входящие в состав сегментов. 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 FoxPro-приложениях, 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 /GS3S /NOS MyProg.exe
```

Пример защиты Native-приложения с ключом Guardant Sign. 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:

Все

Значение параметра:

1<=N<=20

Описание:

Опция задает число независимых таблиц вопросов-ответов к аппаратному или программному алгоритму.

Если опция не указана, автозащита будет использовать 2 таблицы.

В целях повышения стойкости защиты можно установить **/ATR** и увеличить число таблиц. 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.

Пример:

```
NwKey32.exe /GS3S /ATR=4 MyProg.exe
```

Пример задает 4 таблицы вопросов-ответов приложения к алгоритму GSII64 ключа Guardant Sign.

Упаковать секции исполняемого файла:
/PACK

Тип электронного ключа:

Все

Описание:

Опция включает режим сжатия секций защищаемого Native-файла. Это позволяет уменьшить (при наличии такой возможности) размер приложения. Распаковка требуемых секций исполняемого файла будет происходить во время выполнения.

Пример:

```
NwKey32.exe /GS3S /PACK MyProg.exe
```

Приложение MyProg.exe будет привязано к электронному ключу Guardant Sign и сжато.

Контролировать атрибуты страниц:
/CPA

Тип электронного ключа:

Все

Описание:

При использовании опции защищенное Native-приложение будет контролировать атрибуты страниц памяти в процессе работы и восстанавливать исходные значения в случае их изменения.

Злоумышленник, исследуя защищенное приложение, может пытаться изменить атрибуты страниц памяти в своих целях. Опция **/CPA** препятствует этой практике.

Важно!

Категорически запрещается использовать опцию **/CPA**, если приложение защищено Guardant API!

Пример:

```
NwKey32.exe /GS3 /CPA MyProg.exe
```

Приложение MyProg.exe, привязанное к ключу Guardant Sign, будет контролировать атрибуты страниц памяти. Приложение не использует Guardant API.

Включить поддержку неявного связывания:
/IMPLICIT_LINKING_SUPPORT

Тип электронного ключа:

Все

Описание:

Некоторые библиотеки, импортируемые защищаемым Native-приложением, могут содержать секцию **.tls**. При использовании таких библиотек, необходимо включать эту опцию, в противном случае могут возникать непредвиденные ошибки.

Пример:

NwKey32.exe /GS3S /IMPLICIT_LINKING_SUPPORT /PACK MyProg.exe

Приложение MyProg.exe будет привязано к электронному ключу Guardant Sign с поддержкой неявного связывания и сжатием.

Защитить импортируемые функции:
/IMPORT_HOOK[=%:L]

Тип электронного ключа:

Все

Значение параметра:

1<=%<=100, по умолчанию 30

1<=L<=100, по умолчанию 5

Описание:

Во избежание установки точек останова на начало импортируемых функций, рекомендуется включать эту опцию.

Запрещается устанавливать /IMPORT_HOOK совместно с /IMPORT_HOOK_LIST

Параметр **%** определяет тот процент импортируемых функций, которые будут защищены таким образом.

Параметр **L** определяет количество инструкций из каждой импортируемой функции, которые должны быть защищены.

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 Native-приложения.

Защитить импортируемые функции по списку:
/IMPORT_HOOK_LIST[=MyApp.exe_IH.piw]

Тип электронного ключа:

Все

Значение параметра:

MyApp.exe_IH.piw	Путь к файлу с результатами профилирования. Необязательный параметр. Если неуказан, то файл должен находиться в текущей директории

Описание:

/IMPORT_HOOK_LIST позволяет выбрать из списка и защитить импортируемые функции.

Запрещается устанавливать /IMPORT_HOOK_LIST совместно с /IMPORT_HOOK

При установке опции в одной директории с защищаемым файлом должен находиться конфигурационный файл с именем типа **MyApp.exe_IH.piw**, в котором содержится список защищаемых/незащищаемых функций импорта.

Дизассемблирование и анализ функций импорта приложения, а также генерирование конфигурационного файла, выполняется автоматически при помощи профилировщика, который можно использовать как отдельно (**NativeProfilerGUI.exe**), так и вызывать из **Мастера лицензирования и автозащиты**.

В результате работы профилировщика создается ini-файл, в котором знаком «+» отмечены функции для защиты, а знаком «-» – те функции импорта, которые защищать не следует.

Сгеренированный Мастером конфигурационный файл можно редактировать и в дальнейшем использовать при защите функций импорта из командной строки.

Пример:

NwKey32.exe /GS3S /IMPORT_HOOK_LIST MyProg.exe

Все импортируемые приложением MyProg.exe функции, отмеченные «+» в файле **MyProg.exe_IH.piw**, будут защищены на ключ Guardant Sign. На момент защиты файл **MyProg.exe_IH.piw**, первоначально созданный профилировщиком Мастера лицензирования и автозащиты, должен находиться в одной директории с защищаемым приложением MyProg.exe.

Извлечь инструкции из тела приложения /RIP_CODE[=%[:Filename.Ext]]

Тип электронного ключа:

Все

Значение параметра:

1<=%<=100	Вероятность переноса инструкции в тело виртуальной машины, по умолчанию 50
FileName.Ext	Путь к map-файлу. Необязательный параметр

Описание:

Опция разрешает утилите автозащиты извлечь ряд инструкций из тела защищаемого Native-приложения и перенести их в тело виртуальной машины.

Запрещается устанавливать **/RIP_CODE** совместно с **/RIP_CODE_LIST**

Если защищается несколько приложений, то параметр **FileName.Ext** должен указывать путь к папке, содержащий map-файлы приложений с именами, идентичными именам защищаемых приложений. Путь должен оканчиваться символом «\». Второй параметр обязательным не является.

Примеры:

NwKey32.exe /GS3S /RIP_CODE=10 MyProg.exe

Приложение будет привязано к Guardant Sign. Для инструкций в теле приложения существует 10-процентная вероятность переноса в виртуальную машину.

NwKey32.exe /GS3S /RIP_CODE=40:D:\Mypath\MyApp.map MyProg.exe

Приложение будет привязано к Guardant Sign. Для инструкций в теле приложения существует 40-процентная вероятность переноса в виртуальную машину. Используется map-файл MyApp.map.

NwKey32.exe /GS3S /RIP_CODE=50:D:\Mypath\ MyProg1.exe MyProg2.exe

Приложения MyProg1.exe и MyProg2.exe будут привязаны к Guardant Sign. Для инструкций в теле приложений существует 50-процентная вероятность переноса в виртуальную машину. Используются map-файлы из каталога **D:\Mypath**.

Извлечь инструкции из тела приложения по списку /RIP_CODE_LIST[=MyApp.exe_RC.prc]

Тип электронного ключа:

Все

Значение параметра:

MyApp.exe _RC.prc	Путь к файлу с результатами профилирования. Необязательный параметр. Если неуказан, то файл должен находиться в текущей директории

Описание:

Опция разрешает утилите автозащиты извлечь заранее намеченные инструкции из тела защищаемого Native-приложения и перенести их в тело виртуальной машины.

Запрещается устанавливать /RIP_CODE_LIST совместно с /RIP_CODE

При установке опции в одной директории с защищаемым файлом должен находиться конфигурационный файл с именем типа **MyApp.exe_RC.prc**, в котором содержится список защищаемых/незащищаемых инструкций.

Дизассемблирование и анализ инструкций приложения, а также генерирование конфигурационного файла, выполняется автоматически при помощи профилировщика, который можно использовать как отдельно – (**NativeProfilerGUI.exe**), так и вызывать из **Мастера лицензирования и автозащиты**.

Для успешной работы профилировщика требуется наличие в одной директории с защищаемым приложением его файла сопоставления (**имя_файла.map**), который создается при компиляции приложения

В результате работы профилировщика создается ini-файл, в котором знаком «+» отмечены инструкции для защиты, а знаком «-» – те инструкции, которые извлекать не следует.

Сгеренированный Мастером конфигурационный файл можно редактировать и в дальнейшем использовать его при защите из командной строки.

Пример:

NwKey32.exe /GS3S /RIP_CODE_LIST MyProg.exe

Приложение будет привязано к Guardant Sign. Выбранные инструкции будут перенесены в виртуальную машину, согласно списку, содержащемуся в файле **MyProg.exe_RC.prc**. На момент защиты файл **MyProg.exe_RC.prc**, первоначально созданный профилировщиком **Мастера лицензирования и автозащиты**, должен находиться в одной директории с защищаемым приложением MyProg.exe.

Сетевые опции автозащиты

Выбрать режим лицензирования:
/LOGIN_MODE=H|S|P

Тип электронного ключа:

Guardant Sign Net/ Time Net/ Net III/ Net II

Описание:

Опция позволяет выбрать режим лицензирования сетевого приложения.

Важно

Подробную информацию по режимам лицензирования см. в главе **Защита сетевых приложений**

Установка параметра **P** задает режим лицензирования по процессам. В этом случае, каждый процесс (по сути, копия защищенного приложения) будет получать отдельную лицензию.

При указании параметра **H** (**/LOGIN_MODE=H**) лицензии будут выделяться на каждый создаваемый хэндл.

Так как автозащита оперирует единственным хэндлом, то в случае, если приложение не использует Guardant API, распределение по хэндлам не будет отличаться от распределения по процессам (**P**).

Однако при комбинации GrdAPI и автозащиты хэндлов будет уже 2 и может произойти перерасход лицензий.

При указании параметра S (**/LOGIN_MODE=S**) лицензии будут распределяться по рабочим станциям. Т. е. будет ограничиваться число рабочих станций, на которых одновременно запущены копии защищенного приложения.

Пример:

```
NwKey32.exe /GN3S /LOGIN_MODE=P MyProg.exe
```

Сетевое приложение MyProg.exe будет привязано к ключу Guardant Sign Net, причем лицензии будут распределяться по процессам: каждая запущенная копия приложения получит отдельную лицензии.

Использовать систему управления лицензиями:
/MN=x

Тип электронного ключа:

Guardant Sign Net/ Time Net/ Net III/ Net II

Значение параметра:

0<=x<=127

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**. 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**)

Важно!

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 /GN3S /MN=00 MyProg1.exe
NwKey32.exe /GN3S /MN=01 MyProg2.exe
NwKey32.exe /GN3S /MN=02 MyProg3.exe
NwKey32.exe /GN3S /MN=03 MyProg4.exe
```

Показаны вызовы утилиты автозащиты с сетевым ключом Guardant Sign Net для примера, описанного выше по тексту.

Опции, касающиеся ограничений работы приложения

Вывести предупреждение об оставшемся времени использования /LICENSE_TIME[=limit]

Тип электронного ключа:

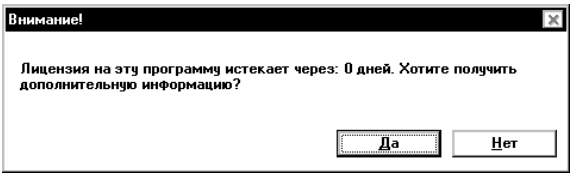
Guardant Time/Net Time/Code Time

Значение параметра:

1<=limit<=365 дней, по умолчанию 14 дней

Описание:

Для приложений, защищенных современными ключами Guardant в режиме ограничения времени, можно выводить на экран предупреждение об оставшемся сроке использования.



Дополнительный параметр опции (limit) задает период до окончания заданного срока работы приложения (в днях), по достижении которого при каждом запуске будет появляться предупреждение.

Пример:

NwKey32.exe /GS3S /LICENSE_TIME MyProg.exe

Приложение MyProg.exe будет привязано к электронному ключу Guardant Time. Время работы приложения будет ограничено на некий заданный период, указанный во временной зависимотси алгоритма GSII64 с номером 0, причем за 2 недели до окончания заданного срока начнет выводиться предупреждение об этом.

Ограничить число запусков приложения /DCA

Тип электронного ключа:

Локальные

Описание:

Процедура ограничения числа запусков приложения состоит из последовательных обязательных этапов:

1. Выполните автозащиту приложения с опцией **/DCA** и другими необходимыми опциями.

2. Запустите утилиту программирования ключа GrdUtil.exe, загрузите нужный файл образа, установите требуемое число запусков программы с помощью счетчика алгоритма GSII64, как это описано в разделе **Программирование времени работы приложения**. Запишите маску в ключ.

Теперь при каждом запуске приложения счетчик алгоритма будет автоматически декрементироваться на единицу.

После обнуления счетчика приложение станет неработоспособно. Для увеличения числа запусков приложения, находящегося у конечного пользователя, используйте процедуру удаленного обновления памяти ключа.

Важно!

1. В процессе автозащиты происходит множественный вызов алгоритма ключа, и если запрограммировать ключ перед автозащитой, то выставленное значение числа запусков уменьшится. По этой причине лучше придерживаться указанной выше последовательности действий при ограничении числа запусков.

2. Если в режиме ограничения числа запусков на один ключ защищается несколько приложений, причем для автозащиты используется один и тот же алгоритм, то каждое приложение будет декрементировать один и тот же счетчик алгоритма. К примеру, если программный продукт состоит из двух защищенных приложений MyProg1.exe и MyProg2.exe, и в счетчике алгоритма содержится значение 5, то пользователь, трижды запустив приложение MyProg1.exe, сможет запустить MyProg2.exe только два раза.

3. Опции /T и /DCA несовместимы, т. к. обе используют один и тот же счетчик алгоритма.

Пример:

```
NwKey32.exe /GS3S=3:8 /DCA MyProg.exe
```

Приложение MyProg.exe будет привязано к электронному ключу Guardant Sign. Число запусков приложения будет ограничено на период, указанный в счетчике алгоритма GSII64 с номером 3.

Вывести предупреждение об оставшемся числе запусков /LICENSE_COUNTER[=limit]

Тип электронного ключа:

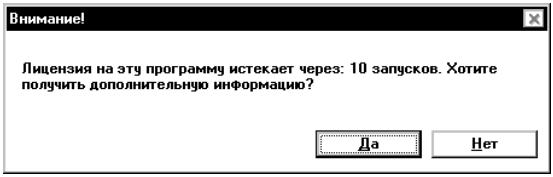
Современные, начиная с Guardant Sign

Значение параметра:

1<=limit<=1000, по умолчанию 10

Описание:

Для приложений, защищенных современными ключами Guardant в режиме ограничения количества запусков, можно выводить на экран предупреждение об оставшемся числе запусков.



Необязательный параметр опции (**limit**) задает число оставшихся запусков приложения, по достижении которого при каждом старте программы будет выводиться предупреждение.

Пример:

NwKey32.exe /GS3S /DCA /LICENSE_COUNTER MyProg.exe

Приложение MyProg.exe будет привязано к электронному ключу Guardant Sign и ограничено по числу запусков. За 10 запусков до окончания заданного количества начнет выводиться предупреждение об этом.

Выдать ссылку на сайт разработчика приложения /LICENSE_URL=string

Тип электронного ключа:

Современные, начиная с Guardant Sign

Значение параметра:

Сообщение разработчика, как правило, адрес сайта

Описание:

Для приложений, защищенных современными ключами Guardant в режимах ограничения времени работы или количества запусков, можно выдавать ссылку на сайт разработчика приложения, где пользователь может получить информацию об условиях продления работы программы.

Опция используется только совместно с **/LICENSE_COUNTER** или **/LICENSE_TIME**. Ссылка на сайт задается при помощи обязательного параметра **string**.

Вместе с появлением предупреждения об ограничении работы приложения на экран будет выводиться предложение посетить сайт разработчиков:

В случае согласия пользователя (нажатие на кнопку **[Да]**), будет запущен браузер и открыт указанный при защите сайт.

Пример:

NwKey32.exe /GS3S /DCA /LICENSE_COUNTER /LICENSE_URL= www.guardant.ru MyProg.exe

Приложение MyProg.exe будет привязано к электронному ключу Guardant Sign и ограничено по числу запусков. За 10 запусков до окончания заданного количества начнет выводиться предупреждение об этом, а также предложение посетить сайт разработчиков.

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 при защите ПО берутся из файла **.MSG** и вписываются в вакцину. По умолчанию используется файл со стандартными сообщениями **<имя_утилиты_защиты>.MSG**, находящийся в каталоге, который содержит соответствующую утилиту защиты.

Файл **.MSG** представляет собой обычный текстовый файл, состоящий из 16-сообщений, которые может выдавать вакцина при запуске и работе защищенной программы (см. содержимое файла **NwKey32.msg**).

Можно задавать свои сообщения для вакцины. Для этого нужно отредактировать умолчательный файл. Каждое новое 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.msg в текущем каталоге.

Примеры:

```
NwKey32.exe /GS3S /MSG=c:\mydir\MyMes1.msg MyProg.exe
Nwkey32.exe /GC /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.

Показать заставку при старте защищенного приложения:
/SPLASH[=FileName.bmp]

Тип электронного ключа:

Все

Значение параметра:

Графический файл формата bmp

Описание:

При старте защищенного приложения в некоторых случаях может наблюдаться небольшая задержка. Это связано с тем, что автозащита выполняет необходимые проверки и преобразования защищенного файла. Чтобы пользователь не был введен в заблуждение в ожидании старта приложения, можно в момент запуска выводить специальную заставку – стандартную или собственную.

Если опция указана без дополнительного параметра, то будет выведена стандартная заставка из ресурсов внешней вакцины GrdVkc32.dll.

Пример:

```
NwKey32.exe /GS3S /SPLASH=MyPicture.bmp MyProg.exe
```

При старте приложения MyProg.exe, защищенного на ключ Guardant Sign, будет выводиться заставка из файла MyPicture.bmp.

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 /GS3S /T=2 /OUT=c:\protect MyProg.exe
NwKey32.exe /GC /T=2 /OUT=c:\protect MyProg.exe
```

В обоих случаях приложение MyProg.exe при защите будет скопировано в каталог C:\Protect.

Запретить вывод сообщений NwKey32.exe в процессе защиты: /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 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 /GS3S /Q MyProg.exe
```

Запретить вывод сообщений защищенного приложения: /SILENT

Тип электронного ключа:

Все

Описание:

В некоторых случаях защищенное приложение не должно выводить на экран никаких сообщений (например, при работе сервисов Windows).

Для запрета вывода сообщений защищенного приложения на экран предназначена опция **/SILENT**.

Пример:

```
NwKey32.exe /GS3S /SILENT 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 /GN3S /PACK /V /Q MyProg1.exe MyProg2.exe MyProg3.exe
```

В этом случае утилита защитит файлы MyProg1.exe, MyProg2.exe и MyProg3.exe, находящиеся в текущем каталоге сжимая при этом их секции и выполняя проверку целостности. Также будут отключены сообщения утилиты автозащиты.

Список файлов может содержать и полные пути к ним, а также символы "*****" и "**?**", например:

```
NwKey32.exe /GS3S C:\MyDir\*.exe
NwKey32.exe /GN3S 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:

```
"MyProg.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 /GS3S @MyFil.fil
NwKey32.exe /GC /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 /GN3S /GS3S @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.

Автоматическая защита .NET-сборок

Принцип автозащиты .NET-приложений

Для автоматической защиты .NET-приложений используется подход, основанный на двухуровневой защите, где каждый уровень выполняет свои задачи в общем процессе:

- символьная обфускация защищаемого MSIL-кода
- перенос части MSIL-кода исполняемых файлов и динамических библиотек в защищенное хранилище

Такая концепция позволяет существенно повысить уровень защищенности приложения в целом, так как распространенный инструментарий обратного анализа приложений .NET (ildasm, reflector.net и т. п.) становится бессильным.

Это обусловлено тем, что б*о*льшая часть кода приложения будет храниться в защищенном Native-контейнере, который в свою очередь защищен псевдокодом и использует функционал электронного ключа.

Когда происходит вызов защищенного кода, который зашифрован на аппаратном алгоритме, то сначала происходит обращение непосредственно к самому ключу, и только после этого начинается выполнение кода.

Ограничения .NET-автозащиты

1. Не поддерживаются сборки со смешанным кодом и мультимодульные сборки.

2. См. ограничения для автозащиты исполняемых Native-файлов.

Порядок защиты .NET-приложений

Для автоматической защиты .NET-сборок используются 2 консольные утилиты:

Утилита	Назначение
CodeObfuscator.exe	Символьная обфускация MSIL-кода и шифрование строк
CodeProtect.exe	Перенос части MSIL-кода исполняемых файлов и динамических библиотек в защищенное хранилище

Утилиты **CodeObfuscator.exe** и **CodeProtect.exe** можно применять как совместно, так и ограничиться использованием одной из них. Однако следует всегда соблюдать определенный порядок использования утилит – сначала приложение необходимо обфусцировать, и только потом провести защиту кода:

- 1-й этап защиты: обфускация и шифрование строк .NET-сборки

- 2-й этап защиты: защита кода .NET-сборки

1. Обфускация кода и шифрование строк CodeObfuscator.exe

Символьная обфускация позволяет усложнить жизнь злоумышленнику, так как все осмысленные имена внутри защищаемого приложения превращаются в набор случайных символов, и обратный анализ такого кода становится гораздо сложнее. Если осмысленные имена больше не используются, то поиск по ключевым словам становится невозможным. Например, «SerialNumberCheck(uint licenseID, byte[] serialNumber) в виде обфусцированного имени будет выглядеть, как “abb(uint zxd, byte[] skf)”.

Технология перегрузки имен, также использованная в обфускаторе, позволяет присвоить одно и то же имя больш*о*му количеству сущностей внутри защищаемого приложения, что позволяет еще сильнее усложнить его обратный анализ.

Еще одним элементом защиты, который реализуется посредством обфускации, является шифрование строковых констант с использованием аппаратного алгоритма электронного ключа. Этот механизм позволяет обращаться к ключу в процессе выполнения, причем для каждого защищаемого приложения такая схема работы с ним будет уникальной, так как обращение к строковым константам происходит в разных местах обфусцированного кода.

Возможности обфускатора

Обфускатор Guardant предоставляет следующие возможности для защиты .NET-приложений:

- Обфускация исполняемых файлов (*.exe) и библиотек (*.dll)

- Символьная обфускация имен (типы, методы, поля, свойства, события, параметры методов и обобщений)
- Перегрузка полей, методов (присваивание одного и того же имени объектам с разными сигнатурами)
- Поддержка нескольких алфавитов (символы, цифры, символы и цифры, непечатные символы)
- Тонкая настройка параметров обфускации при помощи конфигурационного файла
- Сохранение в файл отображения обфусцированных имен в оригинальные
- Сохранение статистики обфускации (покрытие оригинального кода обфусцированным)
- Шифрование строк с использованием электронного ключа
- Обфускация графа потока управления

Файлы, необходимые для процесса защиты

Для защиты приложения необходимо наличие следующих файлов в одной директории:

Файл	Описание
CodeObfuscator.exe	Строчная утилита для обфускации .NET-приложений
GuardantDotNetApi.dll	Библиотека функций Guardant API .NET
NvCodes.dat	Файл с кодами доступа
LexicalObfuscator.dll	Основные компоненты символьного интерфейса обфускатора
LexicalObfuscator.Kernel.dll	Слой взаимодействия с Mono.Cecil, построение промежуточного представления
LexicalObfuscator.Types.dll	Библиотека типов промежуточного представления
Mono.Cecil.dll	Mono.Cecil с некоторыми специфичными изменениями
NameGenerator.dll	Генераторы неповторяющихся имен
Obfuscator.dll	Библиотека обфускатора. Выполняет загрузку и запись сборок, обработку ошибок и исключительных ситуаций, вызов символьного обфускатора и шифратора строк
Obfuscator.CodeUtility.dll	
StringEncryption.dll	Основные компоненты для шифрования строк
Templates.dll	Шаблоны логики работы с ключом для инжекта

Важно!

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 обфусцированной .NET-сборки

Для работы защищенного приложения необходимо наличие следующих файлов в одной директории:

Файл	Описание
CodeStorage32.dll	1. Индивидуальные защищенные хранилища зашифрованных строковых констант и MSIL-кода, а также функций Guardant API, для 32- и 64-разрядных ОС Windows. 2. Хранилища создаются при помощи опции **/INIT** и **уникальны** для каждого проекта защиты! 3. **CodeObfuscator.exe** и **CodeProtect.exe** в рамках одного проекта используют одно и те же хранилище!
CodeStorage64.dll	

Порядок защиты

Перед началом защиты подсоедините к компьютеру электронный ключ нужного типа.

Формат вызова утилиты **CodeObfuscator.exe**:

CodeObfuscator.exe [<опции обфускации>] [<модель ключа>[=<параметры привязки>]] [<опции поиска ключа>] [<опции защиты сборок>] [<файл 1>] ... [<файл N>]

Или:

CodeObfuscator.exe [<опции обфускации>] [<модель ключа>[=<параметры привязки>]] [<опции поиска ключа>] [Файл1.ext] [Файл2.ext]

...

[<опции обфускации>] [<модель ключа>[=<параметры привязки>]] [<опции поиска ключа>] [ФайлN.ext]

Укажите в командной строке необходимые для защиты параметры и нажмите на кнопку **[Enter]**. Утилита приступит к защите, выдавая по ходу работы необходимые сообщения.

После завершения защиты утилита закончит работу.

Процесс защиты можно прервать в любой момент, нажав **Esc**.

Общие и индивидуальные опции обфускации

Опции обфускатора условно делятся на **общие** и **индивидуальные**.

Действие общих опций распространяются на все защищаемые .NET-сборки из одного проекта. К общим опциям относятся /SO, /OP, /INIT, /SE и некоторые другие.

Но есть ряд опций, которые *могут быть* индивидуальными для каждой защищаемой сборки проекта. К ним относятся опции установки типа ключа, опции привязки к параметрам ключа и сетевые опции защиты. Более подробно см. консольную справку.

Сводные таблицы опций обфускатора

1. Опции установки типа ключа, привязки к параметрам ключа, а также сервисные опции одинаковы для всех строчных утилит автозащиты. Поэтому для пересекающихся опций будут даны гиперссылки на их подробное описание в разделе **Консольная автозащита исполняемых Native-файлов. NwKey32.exe**.

2. Так называемые индивидуальные опции обфускатора имеют смысл только при использовании совместно с опцией шифрования строк - **/SE**., которая осуществляет взаимодействие с электронным ключом.

3. Дополнительная информация об особенностях работы с обфускатором содержится во 2-й части Руководства пользователя (глава Рекомендации программисту).

Опции обфускатора 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	Модель
/GS3S[=[N]:[L]:[ID]:[S[<FileName.bin>]]	Привязать к Guardant Time/Sign	Time/Sign
/GN3S[=[N]:[L]:[ID]:[S]:[<FileName.bin>]]	Привязать к Guardant Time/SignNet	Time/Sign Net
/GC=N:L[:[ID]:[S]:[<FileName.bin>]]	Привязать к Guardant Code/CodeTime	Code/Code Time
/GS3[=[N]:[L]:[ID]]	Привязать к Guardant Stealth III	StealthIII/Net III
/GN3[=[N]:[L]:[ID]]	Привязать к Guardant Net III	Net III
/GS2[=[N]:[L]:[ID]]	Привязать к Guardant Stealth II	Stealth II /Net II
/GN2[=[N]:[L]:[ID]]	Привязать к Guardant Net II	Net II
/GSP[=[N]:[L]:[ID]:[S[<FileName.bin>]]	Привязать к софтверному ключу Guardant SP	SP

Опции привязки .NET-сборки 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 образа ключа)	Все
/UN[=[0x]...]	Проверять номер программы (используется заданное значение, либо значение из поля номера программы ключа)	Все
/RC[=xx]	Если ключ не найден, проверять его наличие и выводить сообщение об отсутствии ключа заданное число раз	Все

Опции обфускации .NET-сборки

Опция	Описание	Тип ключа
/SO	Выполнить символьную обфускацию .NET-сборки	Все
/XML=config.gpp	Использовать результаты работы Exclusion Utility	Все
/OP	Обфусцировать публичные интерфейсы .NET-сборки	Все

Опция	Описание	Тип ключа
/SE	Зашифровать строковые константы .NET-сборки	Все
/INIT	Создать защищенное хранилище	Все
/EXCEPT	Генерировать исключение при проблемах с ключом	Все
/MAP=FileName.map	Генерировать xml-файл с результатами обфускации	Все

Опции, ограничения времени работы и числа запусков приложения

Опция	Описание	Тип ключа
/LICENSE_TIME [=limit]	Вывести предупреждение об оставшемся времени	Только для Sign и выше
/LICENSE_URL= string	Вывести ссылку на сайт разработчика приложения	Только для Sign и выше

Сетевые опции

Опция	Описание	Тип ключа
/LOGIN_MODE= H | S | P	Выбрать режим лицензирования: по хэндлам (H), рабочим станциям (S) или процессам (P)	Сетевые ключи
/MN=xx	Использовать систему управления лицензиями	Сетевые ключи

Опции, влияющие на защищенность приложения

Опция	Описание	Тип ключа
/ATR=N	Задать число таблиц вопросов-ответов к алгоритму	Все
/AES_COUNT=N	Задать число ключей шифрования	Все

Сервисные опции автозащиты .NET-сборки

Опция	Описание	Тип ключа
/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
/SILENT	Запретить вывод сообщений защищенного приложения	Все
/Q	Запретить вывод сообщений утилиты защиты на экран	Все

Опции обфускатора

Создать защищенное хранилище /INIT

Тип электронного ключа:

Все

Описание:

Опция создает 32- и 64-разрядные защищенные контейнеры **CodeStorage32.dll** и **CodeStorage64.dll**, в которых хранятся зашифро-

ванные строковые константы и инструкции MSIL-кода (для **CodeProtect.exe**).

Опция обязательна для работы **CodeObfuscator.exe**.

Важно!

Защищенное хранилище едино для обфускатора и протектора кода и отличается только разрядностью. Поэтому опцию /INIT при использовании обеих утилит .NET-автозащиты можно вызывать только один раз, в начале работы. В противном случае, если, к примеру, при защите кода с помощью CodeProtect.exe /INIT вызвать повторно, то хранилище будет пересоздано, а результаты работы обфускатора уничтожены.

Пример:

CodeObfuscator.exe /GS3S /SE /INIT MyProg.dll

.NET-библиотека MyProg.dll будет привязана к ключу Guardant Sign, и ее строки будут зашифрованы и перенесены в защищенные хранилища **CodeStorage32.dll** и **CodeStorage64.dll**.

Выполнить символьную обфускацию .NET-сборки /SO

Тип электронного ключа:

Все

Описание:

Опция производит символьную обфускацию .NET-сборки. В процессе обфускации происходит переименование типов, методов, полей, свойств, событий, параметров методов и обобщений. Всем обфусцированным элементам присваиваются короткие схожие имена, что позволяет скрывать логику работы приложения за счет значительного снижения читаемости кода. Также в ходе работы обфускатор может перегружать (присваивать одно и то же имя объектам с разными сигнатурами) поля, методы и удалять свойства и события там, где это возможно.

Лексический обфускатор позволяет значительно усложнить процесс исследования .NET-приложений, сократить их размер, а также скрыть код, который был добавлен для работы с ключом.

Для повышения гибкости обфускации предусмотрена возможность задания исключений, а также алфавита обфускации (буквы, цифры, непечатаемые символы), при помощи утилиты **ExclusionUtility.exe** , см. опцию **/XML**.

Пример:

CodeObfuscator.exe /GS3S /SE /INIT /SO MyProg.dll

.NET-библиотека MyProg.dll будет привязана к ключу Guardant Sign и обфусцирована, а ее строки зашифрованы.

Задать процент обфускации графа потока управления
/CFO=percent

Тип электронного ключа:

Все

Значение параметра:

0<=%<=100

Описание:

При обфускация графа потока управления код разбивается на несколько блоков, к которым добавляются псевдо-блоки, и скрывается от анализа последовательность их исполнения. Последовательность исполнения блоков шифруется. Этот механизм существенно затрудняет статический анализ кода.

Опция **/CFO** задает процент шифрования кода графа потока управления.

Для повышения гибкости обфускации рекомендуется устанавливать **/CFO** совместно вместе с опцией **/XML**, это позволит использовать включения и/или исключения обфускации графа потока.

Пример:

CodeProtect.exe /GS3S /CFO=50 /XML=config.gpp MyProg.dll

.NET-библиотека MyProg.dll будет привязана к ключу Guardant Sign, 50 процентов ее методов будут зашифрованы и размещены в защищенном контейнере, при этом будут учтены исключения и включения обфускации графа потока из файла **config.gpp**.

Использовать результаты работы Exclusion Utility
/XML=FileName.gpp

Тип электронного ключа:

Все

Описание:

При выполнении символьной обфускации, обфускации графа потока управления, а также шифрования MSIL-кода нередко возникает необходимость тонкой настройки указанных защитных механизмов.

Для этого предназначена утилита **ExclusionUtility.exe**, которая расположена в папке **Bin** комплекта разработчика.

Утилита позволяет задавать:

- Исключения и алфавит для символьной обфускации

- Исключения и включения для обфускации графа потока управления
- Исключения и включения для защиты MSIL-кода

Пример:

CodeObfuscator.exe /GS3S /SE /INIT /SO /XML=MyProg.gpp MyProg.dll

.NET-библиотека MyProg.dll будет привязана к ключу Guardant Sign и обфусцирована, а ее строки зашифрованы. Исключения обфускации указываются в файле MyProg.gpp.

Обфусцировать публичные интерфейсы .NET-сборки /OP

Тип электронного ключа:

Все

Описание:

По умолчанию, при символьной обфускации игнорируются все элементы со спецификатором доступа **public**, т. к. эти методы, свойства и так далее могут быть вызваны из сторонних библиотек и приложений. Если включить опцию **/OP**, то будут обфусцированы и все публичные интерфейсы. Это можно делать только в том случае, если в группу обфусцируемых файлов включаются все сборки, входящие в приложение, и ни одна из этих сборок не используется извне. Опция значительно повышает эффективность обфускации, за счет того, что в коде приложения практически не остается оригинальных лексических элементов и понять назначение функций по их названию становится невозможно.

Важно!

Опция **/OP** используется только совместно с опцией символьной обфускации **/SO**

Как правило, опцию **/OP** можно включать для большинства exe-сборок, т. к. они редко содержат экспортируемые элементы или рефлексируются другими сборками. Однако для dll-сборок использование опции возможно лишь при условии замкнутости приложения, которое описано выше.

Пример:

CodeObfuscator.exe /GS3S /SE /INIT /SO /OP MyProg.dll

.NET-библиотека MyProg.dll будет привязана к ключу Guardant Sign, ее символьные и публичные интерфейсы будут обфусцированы, а строки зашифрованы.

Зашифровать строковые константы .NET-сборки /SE

Тип электронного ключа:

Все

Описание:

Опция выполняет шифрование строк с использованием электронного ключа.

В процессе выполнения приложения строки будут успешно дешифрованы только при наличии в порту ключа с нужными параметрами.

Кроме того, автоматически контролируется правильность расшифрования строк, и в случае ошибки автоматически повторяются попытки залогиниться к ключу.

Важно!

Все индивидуальные опции (привязки к типу и параметрам ключа) имеют смысл только при установленной опции /SE.

Пример:

```
CodeObfuscator.exe /GS3S /SE /INIT MyProg.dll
```

.NET-библиотека MyProg.dll будет привязана к ключу Guardant Sign, и ее строки будут зашифрованы.

Генерировать исключение при проблемах с ключом /EXCEPT

Тип электронного ключа:

Все

Описание:

Иногда возможны ситуации, когда в результате действия автозащиты, запущенное приложение не может продолжать работу.

К примеру, может получиться, что счетчик аппаратного алгоритма (если ключ запрограммирован с уменьшением счетчика) будет исчерпан в неподходящий момент. При этом пользователь может потерять наработанные данные, если приложение завершится принудительно.

Чтобы иметь возможность обрабатывать подобные ситуации, рекомендуется устанавливать данную опцию.

Задать число ключей шифрования
/AES_COUNT=N

Тип электронного ключа:

Все

Значение параметра:

2<=N<=200

Описание:

В утилитах .NET-автозащиты реализован механизм автоматической генерации, расчета количества и смены ключей шифрования для алгоритма AES, использующегося при шифровании строковых констант и MSIL-кода.

На определенный объем шифруемых данных выделяется некий «пул» ключей шифрования (действует логарифмическая зависимость). В процессе работы приложения эти ключи меняются. Т. о. минимизируется возможность дешифрования данных при компрометации одного из ключей шифрования.

Механизм генерации и смены ключей является внутренним и прозрачным для пользователей. Влиять на него можно посредством установки опции **/AES_COUNT**, задающей число генерируемых ключей шифрования.

Необходимо отметить, что использование неоптимальных значений опции может привести к существенному замедлению как процесса защиты, так и падению производительности защищенного приложения.

Ускорить генерацию таблиц вопросов в процессе защиты
/FAST

Тип электронного ключа:

Все

Описание:

В некоторых случаях защита больших .NET-проектов может занимать значительное время, основная часть которого уходит на множественные обращения к алгоритму ключа при подготовке таблиц вопросов/ответов для защиты строковых констант (см. /SE).

Причем на время защиты влияют как особенности защищаемых сборок (например, число шифруемых строк), так используемые параметры защиты (например, значения опций /ATR и /AES_COUNT).

Опция /FAST призвана ускорить процесс защиты .NET-приложений за счет параллельных вычислений таблиц вопросов на нескольких аппаратных ключах Guardant одновременно.

Для успешного использования этой опции необходимо, чтобы к портам компьютера были подсоединены несколько современных (рекомендуется использовать ключи на современной аппаратной платформе) ключей Guardant (оптимальное число – 3 – 4), с одинаковой прошивкой, содержащей аппаратный алгоритм для защиты.

Результатом использования /FAST будет существенное ускорение работы утилиты .NET-автозащиты, которое может составлять от 30 до 100% (значения относительные и зависят от сопутствующих факторов, в т. ч. от вычислительной мощности компьютера).

Генерировать файл результатов обфускации
/MAP=FileName.map

Тип электронного ключа:

Все

Описание:

После символьной обфускации все элементы защищаемой сборки получают новое случайное имя. При задании опции /MAP генерируется MAP-файл в формате XML, где описываются результаты работы обфускатора - какие элементы сборки были переименованы, какое имя получили.

Данный файл необходим, если совместно с обфускатором используется утилита автозащиты MSIL-кода **CodeProtect.exe**, в которой задается файл исключений. Т. к. в файле исключений прописываются необфусцированные имена, то утилите необходимо дополнительно задавать MAP-файл для нахождения в защищаемом коде соответствующего обфусцированного метода.

Пример:

```
CodeObfuscator.exe /GS3S /SE /INIT /SO /MAP=MyProg.map MyProg.dll
```

.NET-библиотека MyProg.dll будет привязана к ключу Guardant Sign и обфусцирована, а ее строки зашифрованы. Результаты обфускации будут сохранены в файле MyProg.map.

2. CodeProtect.exe. Защита кода .NET-сборки

Утилита CodeProtect.exe выполняет защиту кода .NET-приложений (исполняемых файлов и библиотек) путем переноса MSIL-кода защищаемого приложения в Native-контейнер, который осуществляет подгрузку необходимого кода в процессе выполнения защищенного приложения.

Принцип работы CodeProtect.exe

Защита кода позволяет автоматически убрать его часть из защищенного приложения и обращаться к нему только по требованию в момент выполнения. Любое обращение к защищенному хранилищу кода инициирует сначала проверку наличия электронного ключа, и лишь при его соответствии тому, на котором была произведена защита приложения, производится дальнейшее расшифрование защищенного кода.

Код каждого метода из защищенного приложения имеет свою уникальную и независимую от других участков кода привязку к электронному ключу. Такой подход существенно затрудняет обратный анализ и реализует уникальную последовательность обращений, опирающуюся на граф потока выполнения защищаемого приложения.

Защита EXE

Защищенный исполняемый файл остается Managed-сборкой, но большая часть его кода переносится в Native-DLL специального вида, которая представляет собой защищенное хранилище кода:

1. В процессе выполнения происходит вызов кода защищенного метода, после чего вызывается код заглушки
2. Заглушка инициирует выполнение кода из защищенного хранилища относительно вызванного метода
3. Происходит выполнение защищенного кода

Защита DLL

При защите динамических DLL-библиотек для платформы .NET использован аналогичный подход, что и при защите исполняемых файлов, но с учетом специфики использования библиотечного кода. В процессе защиты из оригинальной DLL-библиотеки удаляется MSIL-код, и на его место вставляются вызовы специальных методов-заглушек. Т. о., после защиты от оригинальной DLL-библиотеки остается только код с вызовом методов-заглушек. Данный подход к защите позволяет сохранить работоспособность принципов взаимодействия с защищенной библиотекой: на нее по-прежнему можно добавлять ссылки (references) из других .NET сборок, и она по-прежнему имеет статус Managed-сборки. После защиты DLL сборки нет необходимости вносить какие-либо изменения в код, который уже использует защищенную библиотеку, так как методы-заглушки возьмут на себя всю работу по перенаправлению вызовов к защищенному хранилищу кода.

1. Исполняемый файл обращается к методу из защищенной библиотеки
2. После обращения защищенному методу вызывается код заглушки
3. Заглушка инициирует выполнение кода из защищенного хранилища относительно вызванного метода
4. Происходит выполнение защищенного кода.

Возможности CodeProtect.exe

- Защита кода исполняемых .NET-файлов
- Защита кода динамических .NET-библиотек
- Возможность тонкой настройки параметров защиты при помощи конфигурационного файла

Файлы, необходимые CodeProtect.exe в процессе защиты

Для защиты приложения необходимо наличие следующих файлов в одной директории:

Файл
Protect.dll
AutoProtDotNet.dll
Mono.Cecil.dll
DllProtector.dll
Sec.dll

Файлы, необходимые для работы защищенной .NET-сборки

Для работы защищенного приложения необходимо наличие следующих файлов в одной директории:

Файл	**Описание**
CodeStorage32.dll	1. Индивидуальные защищенные хранилища зашифрованных строковых констант и MSIL-кода, а также функций Guardant API, для 32- и 64-разрядных ОС Windows. 2. Хранилища создаются при помощи опции **/INIT** и **уникальны** для каждого проекта защиты! 3. **CodeObfuscator.exe** и **CodeProtect.exe** в рамках одного проекта используют одно и те же хранилище!
CodeStorage64.dll	

Порядок защиты

Перед началом защиты подсоедините к компьютеру электронный ключ нужного типа.

Формат вызова утилиты **CodeProtect.exe**:

CodeProtect.exe [Общие опции] [Индивидуальные опции] [File1.Ext] [File2.Ext] ... [FileN.Ext]

Или

CodeProtect.exe [Общие опции] [Индивид. опции для File1.Ext] [File1.Ext] [Индивид. опции для File2.Ext] [File2.Ext] ... [Индивид/ опции для FileN.Ext] [FileN.Ext]

Укажите в командной строке необходимые для защиты параметры и нажмите на кнопку **[Enter]**. Утилита приступит к защите, выдавая по ходу работы необходимые сообщения.

После завершения защиты утилита закончит работу.

Процесс защиты можно прервать в любой момент, нажав **Esc.**

Общие и индивидуальные опции CodeProtect.exe

Опции CodeProtect.exe, также как и опции обфускатора условно делятся на **общие** и **индивидуальные**. Подробно см. **Общие и индивидуальные опции обфускации**, а также консольную справку по CodeProtect.exe.

Сводные таблицы опций CodeProtect.exe

Важно!

Опции установки типа ключа, привязки к параметрам ключа, а также сервисные опции одинаковы для всех строчных утилит автозащиты. Поэтому для пересекающихся опций будут даны гиперссылки на их подробное описание в разделе **Консольная автозащита исполняемых Native-файлов. NwKey32.exe**.

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	Модель
/GS3S[=[N]:[L]:[ID]:[S[<File-Name.bin>]]	Привязать к Guardant Time/Sign	Time/Sign
/GN3S[=[N]:[L]:[ID]:[S]:[<File-Name.bin>]]	Привязать к Guardant Time/SignNet	Time/Sign Net
/GC=N:L[:[ID]:[S]:[<FileName.bin>]]	Привязать к Guardant Code/CodeTime	Code/Code Time
/GS3[=[N]:[L]:[ID]]	Привязать к Guardant Stealth III	StealthIII/Net III

/GN3[=[N]:[L]:[ID]]	Привязать к Guardant Net III	Net III
/GS2[=[N]:[L]:[ID]]	Привязать к Guardant Stealth II	Stealth II /Net II
/GN2[=[N]:[L]:[ID]]	Привязать к Guardant Net II	Net II
/GSP[=[N]:[L]:[ID]:[S[<File-Name.bin>]]	Привязать к софтверному ключу Guardant SP	SP

Опции привязки .NET-сборки 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 битовой маски)	Все
/UN[=[0x]...]	Проверять номер программы (используется заданное значение, либо значение из поля номера программы ключа)	Все
/RC[=xx]	Если ключ не найден, проверять его наличие и выводить сообщение об отсутствии ключа заданное число раз	Все

Опции защиты кода .NET-сборки

Опция	Описание	Тип ключа
/INIT	Создать защищенное хранилище	Все
/PER=percent	Задать процент защищаемых методов .NET-сборки	Все
/XML=config.gpp	Использовать результаты работы Exclusion Utility	Все
/MAP	Использовать результаты обфускации	Все
/EXCEPT	Генерировать исключение при проблемах с ключом	Все

Опции, ограничения времени работы и числа запусков приложения

Опция	Описание	Тип ключа
/LICENSE_TIME [=limit]	Вывести предупреждение об оставшемся времени	Только для Sign и выше
/LICENSE_URL= string	Вывести ссылку на сайт разработчика приложения	Только для Sign и выше

Сетевые опции

Опция	Описание	Тип ключа
/LOGIN_MODE= [H, S]	Выбрать режим лицензирования: по копиям приложения (H) или по рабочим станциям (S)	Сетевые ключи
/MN=xx	Использовать систему управления лицензиями	Сетевые ключи

Опции, влияющие на защищенность приложения

Опция	Описание	Тип ключа
/ATR=N	Задать число таблиц вопросов-ответов к алгоритму	Все
/AES_COUNT=N	Задать число ключей шифрования	Все

Сервисные опции автозащиты .NET-сборки

Опция	Описание	Тип ключа
/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
/SILENT	Запретить вывод сообщений защищенного приложения	Все

Опции CodeProtect.exe

Создать защищенное хранилище, если оно не было создано ранее /INIT

Тип электронного ключа:

Все

Описание:

Опция создает 32- и 64-разрядные защищенные контейнеры **CodeStorage32.dll** и **CodeStorage64.dll**, в которых хранятся инструкции MSIL-кода (для CodeProtect.exe), в том случае, если эти контейнеры не были созданы ранее, при работе с обфускатором.

Важно!

Защищенное хранилище едино для обфускатора и протектора кода и отличается только разрядностью. Поэтому опцию /INIT при использовании обеих утилит .NET-автозащиты можно вызывать только один раз. В противном случае, если, к примеру, при защите кода с помощью CodeProtect.exe /INIT вызвать повторно, то хранилище будет пересоздано, а результаты работы обфускатора уничтожены.

Примеры:

CodeProtect.exe /GS3S /PER=80 /INIT MyProg.dll

.NET-библиотека MyProg.dll будет привязана к ключу Guardant Sign, 80 процентов ее методов будут зашифрованы и перенесены в созданное защищенное хранилище.

CodeProtect.exe /GS3N /PER=30 MyProg_obfuscated.exe

Обфусцированный ранее файл будет привязан к ключу Guardant Sign Net, 30-процентов его методов будут зашифрованы и перенесены **в уже существующее хранилище**, созданное ранее при обфускации.

Задать процент защищаемых методов .NET-сборки
/PER=percent

Тип электронного ключа:

Все

Значение параметра:

0<=%<=100

Описание:

Утилита защиты MSIL-кода **CodeProtect.exe** извлекает из тела сборки определенный процент методов, шифрует их и помещает в защищенный контейнер. Процент извлекаемых методов задается параметром **/PER**.

Для повышения гибкости обфускации рекомендуется устанавливать **/PER** совместно вместе **с опцией /XML**, это позволит использовать включения и/или исключения при шифровании кода.

Пример:

```
CodeProtect.exe /GS3S /PER=50 /XML=config.gpp MyProg.dll
```

.NET-библиотека MyProg.dll будет привязана к ключу Guardant Sign, и 50 процентов ее методов будут зашифрованы и размещены в защищенном контейнере. При этом будут учтены исключения и включения обфускации графа потока из файла **config.gpp**.

Использовать результаты обфускации
/MAP=FileName.map

Тип электронного ключа:

Все

Описание:

При использовании данной опции необходимо указать MAP-файл, созданный после работы **обфускатора**.

После символьной обфускации все элементы защищаемой сборки получают новое случайное имя. При задании опции /MAP генерируется MAP-файл в формате XML, где описываются результаты работы обфускатора - какие элементы сборки были переименованы, какое имя получили.

Данный файл необходим, если совместно с обфускатором используется **CodeProtect.exe**. Т. к. в файле исключений прописываются необфусцированные имена, то утилите необходимо дополнительно задавать MAP-файл для нахождения в защищаемом коде соответствующего обфусцированного метода.

Пример:

```
CodeProtect.exe /GS3S /PER=50 /MAP=MyProg.map MyProg.dll
```

Предварительно обфусцированная .NET-библиотека MyProg.dll будет привязана к ключу Guardant Sign, и 50 процентов ее методов будут зашифрованы и размещены в защищенном контейнере. В процессе защиты CodeProtect.exe будет использовать результаты обфускации, сохраненные ранее в файле MyProg.map.

Генерировать исключение при проблемах с ключом /EXCEPT

Тип электронного ключа:

Все

Описание:

Иногда возможны ситуации, когда в результате действия автозащиты, запущенное приложение не может продолжать работу.

К примеру, может получиться, что счетчик аппаратного алгоритма (если ключ запрограммирован с уменьшением счетчика) будет исчерпан в неподходящий момент. При этом пользователь может потерять наработанные данные, если приложение завершится принудительно.

Чтобы иметь возможность обрабатывать подобные ситуации, рекомендуется устанавливать данную опцию.

Генерация файлов исключений .NET-автозащиты. ExclusionUtility.exe

Для генерации файлов исключений обфускации, а также включений и исключений защиты кода .NET-сборок предназначена утилита **ExclusionUtility.exe**.

Важно!

Утилита требует для работы.NET Framework 2.0

Утилита работает в пяти режимах:

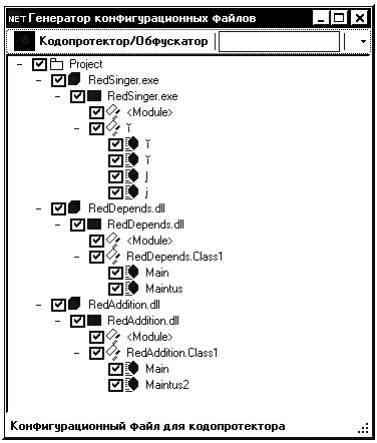
1. Генерация исключений и параметров обфускатора CodeObfuscator.exe.
2. Генерация исключений и включений обфускатора графа потока управления
3. Генерации исключений и включений защиты MSIL-кода CodeProtect.exe

Текущий режим работы утилиты отображается в строке статуса, а переключение возможно с помощью разворачивающегося списка в левом верхнем углу окна программы.

Результаты всех режимов работы сохраняются в едином файле (имя по умолчанию **config.gpp**).

При генерации файла исключений обфускатора можно задать его алфавит. По умолчанию используются непечатные символы.

Сгенерированные программой файлы исключений используются в Мастере автозащиты и в консольных утилитах для обфускации и автозащиты MSIL-кода (опция **/XML)**.



Дополнительную информацию можно получить из справочного файла утилиты, описания опции /**XML**, а также из главы **Рекомендации программисту во 2й части Руководства пользователя**.

Мастер лицензирования и автозащиты

Мастер лицензирования и автоматической защиты Guardant (файл **LicenseWizard.exe**) предназначен для обработки исполняемых файлов Native-приложений (*.exe), а также .NET-сборок (*.exe, *.dll).

Мастер обладает удобным и наглядным графическим интерфейсом и позволяет реализовывать законченные схемы лицензирования без необходимости изучения сложных технологий Guardant.

В частности, **LicenseWizard.exe** может записывать, тиражировать и обновлять лицензии в электронном ключе, а также поддерживает традиционный режим работы, при котором процессы обработки защищаемого файла и программирования ключа разделены.

Возможности Мастера автозащиты

Мастер автозащиты предоставляет следующие возможности:

- Защита 32-разрядных Windows-приложений:
  - Исполняемые Native-файлы (*.exe)
  - .NET-сборки (*.exe и *.dll)
- Поддержка локальных и сетевых ключей Guardant:
  - Guardant Sign/Time
  - Guardant Sign/Time Net
  - Guardant Code/Code Time
  - Guardant Stealth III/Net III
  - Guardant Stealth II/Net II
  - Guardant SP
- Поддержка всех возможностей консольных утилит автозащиты
- Программирование памяти ключа, в том числе дистанционное

Запуск LicenseWizard.exe

Чтобы начать работу с Мастером автозащиты, выполните одно из следующих действий:

- Запустите файл **LicenseWizard.exe**, находящийся в каталоге %Program Files%\Guardant\Guardant %#%\%Public Code%\Bin
- Выберите элемент **Мастер автоматической защиты** из программной группы **Комплект разработчика Guardant %#%** в меню **Пуск**
- Выберите элемент **Мастер автоматической защиты** в оболочке **Guardant Интегратор**

Выбор варианта автозащиты

После запуска утилиты на экране появится начальный диалог Мастера, который позволяет выбрать нужный вариант защиты.

Выбор варианта защиты осуществляется путем нажатия на соответствующую гиперссылку.

Варианты автозащиты и их краткие характеристики:

Варианты автозащиты	Особенности
Создание/изменение проекта лицензии в ключе	Совмещение этапов собственно автозащиты и программирования ключа. После завершения работы Мастера приложение можно тестировать и передавать конечному пользователю.
Запись лицензии в ключи	Тиражирование ранее созданной лицензии, с помощью Мастера, т. е. программирование партии электронных ключей без не-

	обходимости изучения утилиты программирования ключа
Обновление лицензии в ключе	Изменение числа запусков, времени работы или сетевых лицензий в ключе конечного пользователя
Защита новой версии продукта на базе текущей лицензии	Перезащита приложения без изменения содержимого ключа
Полное обновление лицензии в ключе	Перепрограммирование ключа конечного пользователя
Защита с использованием данных, имеющихся в ключе	Только автозащита приложения, без программирования ключа. Ключ необходимо программировать отдельно при помощи утилиты GrdUtil.exe

Далее варианты автозащиты характеризуются более подробно.

Понятие лицензии

Мастер автозащиты оперирует понятием **лицензия**. Под лицензией понимается сочетание параметров автозащиты и данных (файла образа), записанных в ключ.

Лицензия автоматически создается Мастером в процессе защиты, избавляя разработчика от необходимости подробно изучать архитектуру и особенности программирования ключа. Это особенно важно на начальном этапе работы с ключами Guardant, поэтому данный вариант автозащиты подходит для разработчиков, которые только делают первые шаги по защите своего приложения.

Мастер позволяет создавать, тиражировать и обновлять (в том числе, удаленно) лицензии в электронном ключе.

Глава 6
Guardant Net: защита сетевых приложений

Концепция Guardant Net

Guardant Net – это технология защиты и лицензирования сетевых приложений с использованием электронных ключей Guardant.

Основными компонентами Guardant Net являются:

Сетевой ключ	Ключ для защиты и лицензирования сетевого приложения
Сервер Guardant Net	Утилита, обрабатывающая и передающая запросы от клиента к ключу и обратно
Клиент	Защищенное приложение, которое обращается к серверу ключа с удаленного компьютера
Сетевой протокол	Протокол, по которому происходит обмен между сервером и клиентом

К важным понятиям Guardant Net также следует добавить *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.

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 Guardant	**Основные характеристики**				
	Объем памяти	**Платформа**	**Аппаратные алгоритмы**	**HID**	**RTC**
Sign Net / Time Net	4096 Б	Windows, Linux (Wine)	AES, GSII64, ECC160, SHA256, HASH64	+	-/+
Net III, устар.	2048 Б	Windows	GSII64; HASH64, RND64	-	-
Net II, устар.	256 Б	Windows	GSII64; 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, процессам или хэндлам Guardant API, 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 интерфейсом TCP/IP. Однако ключ и сервер ключа должны быть установлены на компьютере под управлением ОС семейства Windows (либо в среде wine@etersoft под Linux).

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 **glds.exe**). Сервер обеспечивает прохождение запросов от клиента непосредственно к ключу и обратно по правилам сетевого протокола TCP/IP.

Важно!

Сервер Guardant Net должен быть загружен на том же компьютере, к которому подсоединен сетевой электронный ключ.

В одной директории с **glds.exe** должен находиться конфигурационный файл **grdsrv.ini**, в котором хранятся текущие настройки сервера ключа.

Клиент Guardant Net

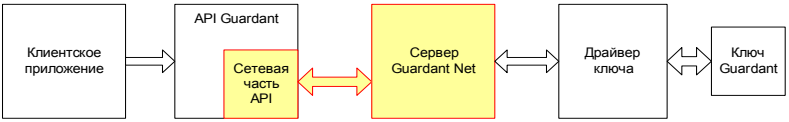
Защищенное приложение (клиент) при запуске должно найти сервер ключа и зарегистрироваться на нем, чтобы продолжить работу. В зависимости от схемы распределения лицензий в качестве клиента может выступать процесс (копия) приложения, рабочая станция или хэндл (копия экземпляра GrdAPI, использующаяся при защите). Каждому клиенту после регистрации на сервере ключа выделяется единица сетевого ресурса.

Для работы клиенту Guardant Net не требуется установка драйвера Guardant, т. к. он 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.

Важно!

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.

Ресурс лицензий ключа

Число одновременно работающих клиентских приложений ограничивается сетевым ресурсом ключей Guardant.

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: 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 / Sign Net /Time Net	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 Sign Net / Time Net / 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**
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 лицензий

На этапе защиты разработчик задает способ распределения сетевых лицензий: по рабочим станциям, процессам или хэндлам.

Все способы имеют свои особенности и применяются для решения различных задач. К примеру, учет лицензий по процессам хорошо подходит для использования на терминальном сервере.

Распределение 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 на конкретной рабочей станции. Если с этого же компьютера запускать новые копии того же приложения (либо другие приложения, привязанные к тому же ресурсу ключу), то ресурс лицензий не изменится.

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.

Распределение лицензий по хэндлам

Лицензия выделяется на каждый хэндл, создаваемый Guardant API (и/или автозащитой).

Важно!

1. Если при защите приложения используется несколько экземпляров библиотеки Guardant API и, соответственно несколько хэндлов (к примеру, при комбинированном использовании Guardant API и автозащиты), то лицензии будут выделены по числу хэндлов.

2. В случае работы приложения с единственным хэндлом (т. е. защита только Guardant API или только автоматическая) распределение по хэндлам не будет отличаться от распределения по процессам. Однако если в дальнейшем предполагается комбинирование защиты, то более дальновидным может оказаться распределение по процессам – во избежание перерасхода лицензий.

1. Сетевой ресурс ключа уменьшается на 1 при регистрации любого экземпляра Guardant API на сервере (т. о. приложение, защищенное и автозащитой, и Guardant API займет 2 лицензии).

2. Сетевой ресурс ключа возвращается (увеличивается на 1) по завершении работы любого экземпляра Guardant API.

Распределение лицензий по процессам

Лицензия выделяется на работающий процесс, в рамках которого может использоваться один или несколько хэндлов (сравн. с распределением по хэндлам).

Важно!

При работе с несколькими хэндлами в рамках одного процесса лицензия будет выдана только одна.

1. Сетевой ресурс ключа выделяется (уменьшается на 1) на каждый запущенный процесс приложения, независимо от того, запускается ли приложение на одном или нескольких компьютерах.

2. Сетевой ресурс ключа возвращается (увеличивается на 1) по завершении работы любого процесса защищенного приложения, запущенного на любом компьютере в сети.

Высвобождение зависших лицензий

Ситуации, при которых лицензии могут зависнуть (например, сбой сети или приложения), обрабатываются прозрачно для пользователя и не требуют его вмешательства.

При потере соединения с сервером клиент пытается выполнить переподключение заданное число раз (параметр **RECONNECT_TRY _NUMBER** в настройках клиента).

Если сервер не получает отклика в течение определенного времени, он автоматически разрывает соединение и освобождает лицензию (т. е. от имени клиента выполняется команда GrdLogout). После чего клиент проводит процедуру регистрации и подключения заново.

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.

Важно!

Для ключей Guardant Sign Net/ Time Net и 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 лицензий
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 процессам (запущенным 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 процесса (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 процессов для данного модуля.

По хэндлам

Число одновременно работающих копий (процессов) приложения зависит от количества используемых хэндлов.

Если на компьютере (или нескольких компьютерах) будет запущено, например, 4 копии Бухгалтерии, 3 – Маркетинга и 2 – Базы продаж, то:

а) При использовании в рамках процесса единственного хэндла сетевой ресурс ключа уменьшится на 9, а ресурс каждого модуля будет уменьшен по числу запущенных копий для данного модуля.

б) При использовании каждым процессом двух хэндлов (к примеру, хэндл GrdAPI и хэндл автозащиты) на каждый процесс будет выделено по 2 лицензии и произойдет их перерасход в случае модулей Бухгалтерия и Маркетинг, а База продаж займет 4 лицензии.

Использование таблицы лицензий

Для использования системы управления лицензиями необходимо на этапе проектирования защиты:

1) Создать в памяти ключа поле «Таблица лицензий», в котором определить количество модулей, их ресурсы лицензий, а также дополнительные параметры

2) При использовании Guardant API:

- Указать номер модуля таблицы лицензий в параметре **dwModuleLMS** функции GrdLogin
- Указать одно из значений флага **dwLoginFlags** при вызове функции GrdLogin:

GrdLM_PerStation	Распределение сетевых ресурсов по рабочим станциями
GrdLM_PerHandle	Распределение по хэндлам приложения
GrdLM_PerProcess	Распределение по процессам (копиям) приложения

3) 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 | P]**, где H – распределение сетевых ресурсов по хэндлам, S – распределение по рабочим станциям и P – по процессам.

Управление лицензиями

В сервере Guardant Net реализованы 2 модели аренды лицензий: фиксированная и плавающая.

Управление лицензиями происходит путем изменения настроке сервера: лицензии либо закрепляются за рабочими станциями, либо выдаются на конкурентной основе.

При необходимости можно комбинировать оба подхода.

Используемые термины

Свободная лицензия – в настоящий момент ни одно приложение не использует данную лицензию.

Занятая лицензия – в настоящий момент времени одно или несколько приложений используют лицензию.

Резервируемая лицензия выдается только конкретному приложению на конкретном хосте, обычно лицензия резервируется на заданный промежуток времени.

Плавающая (конкурентная) лицензия выдается любому приложению на любом хосте удовлетворившему условия лицензирования.

Метка – универсальный идентификатор, генерируемый сервером для клиента и связанный с уникальным аппаратным номером хоста клиента, позволяющий клиенту подтвердить свою регистрацию.

Резервируемые лицензии

Резервируемые (фиксированные) лицензии закрепляются за рабочей станцией постоянно, либо на заданное время.

При первом удачном соединении с сервером клиент проходит процедуру регистрации, в ходе которой сервер назначает ему метки – уникальные идентификаторы процесса, получившего лицензию (см. параметр **uuid** в настройках gnclient.ini), а также рабочей станции, с которой пришел запрос (см. **HOST_ID** в настройках gnclient.ini), запоминает и удерживает за ним лицензию.

В дальнейшем, резервируемую лицензию, ранее уже закрепленную за определенным компьютером, могут получить лишь те копии приложения, которые запускаются с данной рабочей станции.

Приложения, запускаемые с других станций, либо получают собственную резервируемую лицензию после регистрации на сервере, либо (в случае отсутствия свободных резервируемых лицензий) – свободную конкурентную.

Системный администратор (пользователь защищенного приложения) путем изменения настроек сервера Guardant Net может управлять следующими параметрами резервируемых лицензий:

- Время резервирования лицензий
- Число резервируемых лицензий, выделяемых одному клиенту

Чтобы настроить число резервируемых лицензий и временной диапазон (в сутках), на который лицензии будут зафиксированы, перейдите на страницу администрирования сервера, задайте нужные значения параметров **Предел закрепленных лицензий для хоста** и **Время фиксирования лицензии за хостом** и перезагрузите сервер, сохранив настройки.

Чтобы сделать лицензию бессрочной, установите значение -1 в поле **Время фиксирования лицензии за хостом**.

Чтобы отказаться от резервирования лицензий, установите значение 0 в поле **Предел закрепленных лицензий для хоста** и настройте параметр **Предел незакрепленных лицензий для хоста**.

Число резервируемых лицензий, выделяемых на приложение, которое стартует с одной рабочей станции, зависит не только от настроек сервера Guardant Net, но и от способа распределения лицензий, реализованного в приложении:

Распределение лицензий	Число занимаемых резервируемых лицензий
По рабочим станциям	1
По процессам	По числу процессов, но не больше значения параметра **Предел закрепленных лицензий**. При превышении будут выдаваться свободные плавающие лицензии.
По хэндлам	

Сброс регистрации лицензии

Сервер Guardant Net позволяет отменить регистрацию лицензии, закрепленной за клиентом.

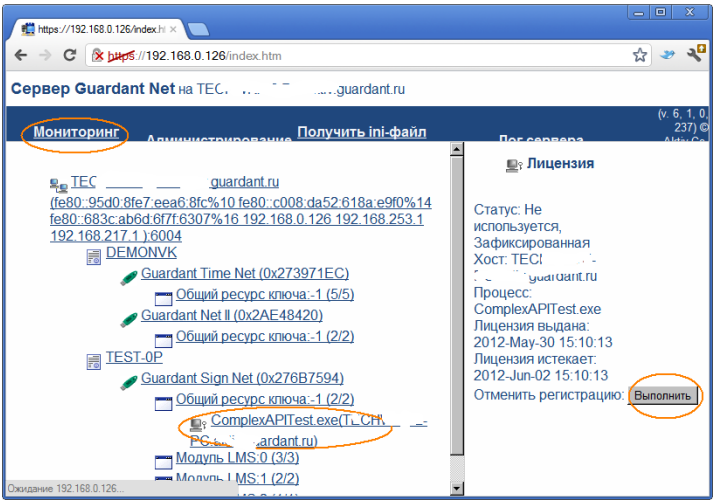
Для отмены регистрации выполните следующие действия:

1) Откройте веб-интерфейс сервера в браузере.

2) Перейдите на страницу **Администрирование** и наберите пароль.

3) После успешного ввода пароля вернитесь на страницу мониторинга и выберите в списке подключений нужного клиента.

4) В окне состояния объекта (справа) нажмите на кнопку **Выполнить** в строке **Отменить регистрацию**.



Конкурентные лицензии

Плавающие лицензии не резервируются за клиентом.

Такая лицензия выдается любому приложению на конкурентной основе. Данные о захвате плавающей лицензии не сохраняются после завершения сеанса, и в следующий раз ее может получить любой другой подходящий объект лицензирования, который первым выполнит GrdLogin.

Системный администратор путем изменения настроек сервера Guardant Net может задавать число плавающих лицензий, выделяемых одной рабочей станции.

Для этого перейдите на страницу администрирования сервера, задайте нужное значение параметра **Предел незакрепленных лицензий для хоста** и перезагрузите сервер, сохранив настройки.

Чтобы отказаться от использования плавающих лицензий, установите значение 0 в поле **Предел незакрепленных лицензий для хоста** и настройте параметр **Предел закрепленных лицензий для хоста**.

Число выделяемых клиенту конкурентных лицензий зависит не только от настроек сервера Guardant Net, но и от способа распределения лицензий, реализованного в приложении:

Распределение лицензий	Число занимаемых резервируемых лицензий
По рабочим станциям	1
По процессам	По числу процессов, но не больше значения параметра **Предел незакрепленных лицензий**. При превышении будут выдаваться свободные плавающие лицензии.
По хэндлам	

Сервер Guardant Net

Сервер Guardant Net (файл **glds.exe**) обеспечивает связь между защищенным сетевым приложением и сетевым ключом Guardant.

Один сервер **glds.exe** может обслуживать запросы к нескольким сетевым электронным ключам Guardant.

Служебные файлы сервера Guardant Net

Для работы сервера **glds.exe** требуются следующие файлы:

Имя файла	Описание
glds.exe	Файл сервера Guardant Net
Каталог pic	Ресурсы для web-интерфейса (gif-файлы пиктограмм)
dh512.pem	Ключи для шифрования трафика между web-клиентом управления сервером и самим сервером
server.pem	
ENG.xml	Файлы строковых ресурсов для web-интерфейса
RU.xml	
grdsrv.ini.in	Шаблон по умолчанию для настроек сервера

Имя файла	Описание
grdsrv.ini	Файл настроек сервера, автоматически создаваемый при старте сервера
glds_log.txt	Журнал, в котором протоколируются события на сервере; создается автоматически после старта сервера

Перечисленные файлы должны находиться в одной директории с файлом сервера **glds.exe** и, соответственно, входить в комплект поставки приложения, защищенного сетевым ключом Guardant.

Загрузка сервера

Сервер Guardant Net должен быть загружен на том же компьютере, к которому подсоединен сетевой электронный ключ.

Важно!

1. Один сервер позволяет обслуживать несколько сетевых ключей Guardant.
2. Запуск двух серверов **glds.exe** на одном компьютере невозможен!
3. Запуск старого сервера **grdsrv.exe** и нового **glds.exe** на одном компьютере невозможен!

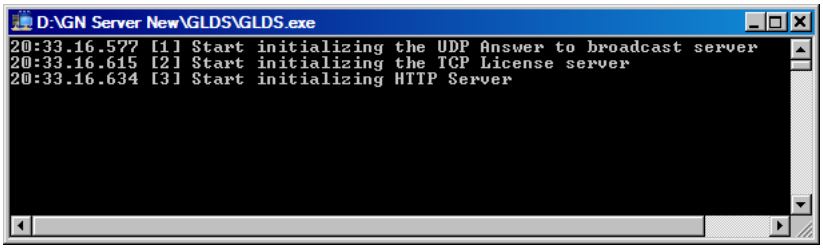
При необходимости (работа с несколькими ключами) в пределах локальной сети можно запускать несколько серверов Guardant Net **glds.exe**. Они должны находиться на разных компьютерах.

Работа сервера как приложения

Сервер **glds.exe** ориентирован на работу в качестве службы Windows, поэтому запуск его в качестве приложения имеет смысл только на этапе настройки защищенного сетеового приложения

Чтобы запустить сервер ключа, откройте командную строку, перейдите в нужный каталог, наберите **glds.exe** и нажмите клавишу ввода.

На экране должны появиться сообщения, свидетельствующие об успешном старте сервера как приложения:



Для завершения работы сервера закройте окно командной строки.

Работа сервера как службы

Основным режимом функционирования сервера **glds.exe** является работа в качестве сервиса (службы) Windows 8 / 7 / 2008 / Vista / 2003 / XP.

Преимущества сервиса состоит в том, что он автоматически запускается во время загрузки операционной системы, для его запуска не нужно выполнять процедуру регистрации на компьютере, и пользователю доступны специальные средства Windows по управлению сервисом.

Установка и запуск сервиса

Чтобы сервер **glds.exe** работал в качестве службы, запустите его из командной строки с опцией -s: **[путь]glds.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.

После запуска сервис glds.exe появляется в списке системных служб Windows. См. **Пуск | Настройка | Панель управления | Администрирование | Службы | Guardant dongle licensing system**:

Остановка сервиса

Работу сервиса Guardant Net можно временно приостановить. Это действие можно выполнить как из командной строки (**[путь]glds.exe -e**), так и средствами операционной системы (см. ниже).

Перейдите в **Пуск | Панель управления | Администрирование | Службы** и щелкните правой кнопкой мыши на строке **Guardant dongle licensing system**. В появившемся контекстном меню выберите Остановить.

Сервис останется инсталлированным в систему, однако перестанет обрабатывать запросы к сетевым ключам.

Для возобновления его работы нужно выполнить старт сервиса из панели **Службы** или командой **glds.exe -s**.

Остановка сервиса не предполагает его выгрузки из списка сервисов, т. е. при перезагрузке компьютера сервис Guardant Net будет снова загружен.

Удаление сервиса из системы

Для удаления сервиса Guardant Net запустите его из командной строки с опцией -r: **[путь]glds.exe –r**

Опции командной строки

Сервис Guardant Net поддерживает следующие опции командной строки:

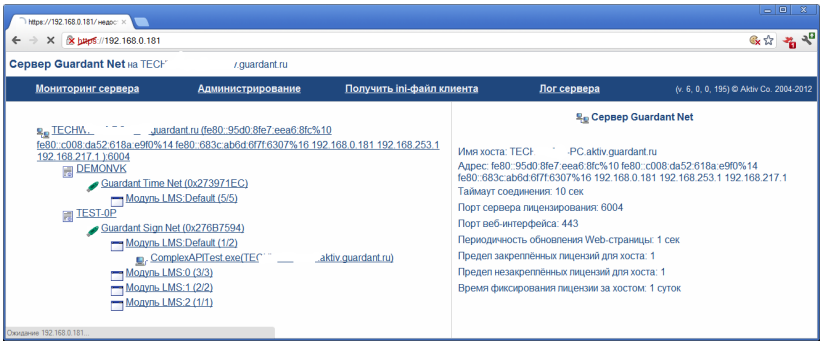
Команда	Описание
glds.exe	Запустить сервер как приложение
glds.exe -s (--start)	Стартовать службу, предварительно зарегистрировав ее (если не зарегистрирована)
glds.exe -d (--deamonize)	Зарегистрировать службу, но не стартовать ее
glds.exe -e (--end)	Остановить службу
glds.exe -r (--remove)	Остановить (если запущена) и удалить службу
glds.exe -h , или неправильная опция	Вывести справку

Мониторинг сервера

Для получения актуальной информации о состоянии сервера Guardant Net и сетевых ключах Guardant, работающих на определенном компьютере, используется web-интерфейс, основанный на защищенном протоколе https.

Чтобы выполнить мониторинг или настройку сервера ключа, запустите интернет-браузер и наберите в адресной строке IP-адрес компьютера, на котором установлены сервер и ключ, в формате: **https://<путь>** (к примеру, **https://192.168.0.181**) и нажмите на клавишу ввода.

После этого на экран будет выведена информация о сервере Guardant Net и электронных ключах, которые он обслуживает в текущий момент:



Окно монитора

Окно монитора разделено на два фрейма - слева располагается рабочая часть, справа - окно состояния:

Фрейм	Назначение
Рабочая область	Серверные объекты в виде древовидной структуры: информация о компьютере, на котором установлены ключ и сервер, а также основные данные о клиентах
Окно состояния выбранного объекта	Подробная информация по ранее выбранному в режиме мониторинга серверному объекту

Рабочая область монитора

В рабочей области окна web-монитора Guardant Net отображается древовидная структура с информацией о параметрах и состоянии сервера и обслуживаемых ключей.

Уровень	Что отображается	Описание
Основание древа	Информация о компьютере	Сетевое имя компьютера
		IP-адреса сетевых интерфейсов компьютера, которые "видит" сервер
		Порт, по которому общаются сервер и клиенты
1-й	Общий код ключа	Идентификатор разработчика
2-й	Информация о ключе	Модель ключа
		ID ключа в 16-ричном виде
3-й	Информация о ресурсе лицензий и модулях защищенного приложения	Значение реального ресурса лицензий: текущий / максимальный
		Название модуля таблицы лицензий и его ресурс лицензий: текущий/ максимальный
4-й	Информация о клиенте	Название приложения
		Сетевое имя компьютера, на котором запущен клиент

Окно состояния выбранного объекта

В правой части окна монитора Guardant Net отображается детальная информация об объекте, предварительно выбранном в рабочей области монитора.

Уровень	Объект	Отображаемая информация
Основание древа	Информация о компьютере	Значения параметры, заданных в конфигурационном файле сервера
1-й	Общий код ключа	Общий код в десятичном виде
2-й	Информация о ключе	Значения общих полей памяти, используемых для поиска ключа: ID, номер программы, версия, маска, серийный номер, счетчик #1
3-й	Информация о модулях приложения и их ресурсе лицензий	Имя модуля, его исходный и текущий ресурс лицензий
4-й	Информация о клиенте	Статус и срок действия лицензии, имя и PID процесса, хост-компьютер

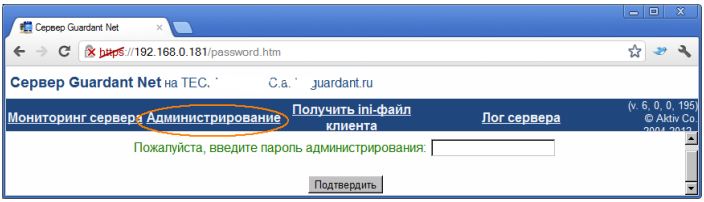
Администрирование

Настройки сервера **glds.exe** хранятся в файле **grdsrv.ini**, который должен находиться в одном каталоге с сервером ключа.

Для управления настройками сервера Guardant Net служит страница **Администрирование** в web-интерфейсе Guardant Net (альтернативным путем настройки является редактирование **grdsrv.ini** напрямую).

Чтобы перейти к настройкам **glds.exe** запустите интернет-браузер, наберите в адресной строке IP-адрес компьютера, на котором установлены сервер и ключ, в формате: **https://<путь>** (к примеру, **https://192.168.0.181**) и нажмите на клавишу ввода.

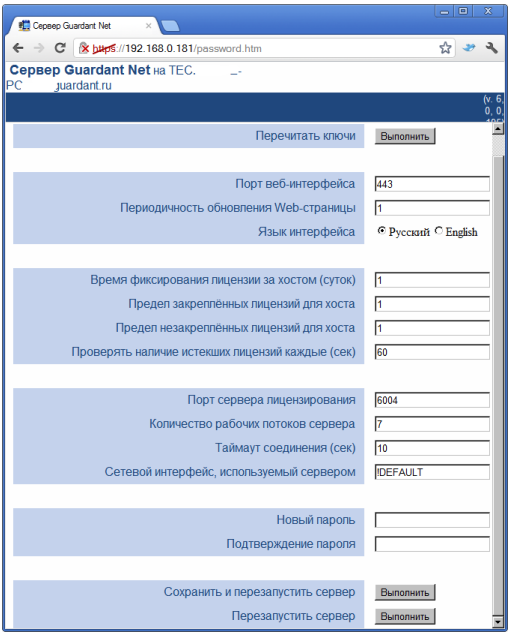
После открытия основной страницы web-интерфейса перейдите на вкладку **Администрирование** путем выбора гиперссылки в верхней части страницы. При этом будет запрошен пароль:



Пароль по умолчанию: **admin**. Для повышения безопасности умолчательный пароль настоятельно рекомендуется поменять после начала рабочей эксплуатации защищенного приложения.

Настройки сервера Guardant Net

После успешного ввода пароля на экране отображаются настройки сервера, доступные для редактирования:



Эталонные настройки сервера Guardant Net хранятся в файле шаблонов **grdsrv.ini.in**. Для успешного старта сервера этот файл должен находиться в том же каталоге.

После первого запуска сервер создает файл текущих настроек **grdsrv.ini**, первоначально идентичный по содержанию файлу шаблонов. В дальнейшем, в **grdsrv.ini** сохраняются все изменения в параметрах сервера.

На момент написания Руководства сервер имеет следующие конфигурируемые параметры:

Параметр (и значение по умолчанию)		**Диапазон значений**	**Назначение**
Веб-интерфейс	**grdsrv.ini**		
Перечитать ключи	-	-	Кнопка поиска сетевых ключей, подсоединенных к компьютеру
Порт веб-интерфейса	PORT=443	Любой подходящий SSL-порт	Порт для web-мониторинга сервера ключа
Периодичность обновления Web-страницы	REFRESH_PERIOD=30 [сек]	Положительное число	Частота обновления информации в мониторе
Язык интерфейса	LANG=RU	RU, EN	Переключатель выбора языка
Время фиксирования лицензии за хостом	LIC_LIFE_TIME_DAYS=1 [суток]	Положительное число.	Срок, на который лицензия закрепляется за клиентами одной рабочей станции

Параметр (и значение по умолчанию)		Диапазон значений	Назначение
Веб-интерфейс	**grdsrv.ini**		
Предел закреплённых лицензий для хоста	MAX_FIXED_FOR _HOST=10	Если –1, то не ограничено	Число лицензий, которое может быть зарезервировано за клиентами одной рабочей станции
Предел незакрепленных лицензий для хоста	MAX_INDEPEND_FO R _HOST=10		Число нефиксированных, свободных лицензий, которое могут занимать клиенты одной рабочей станции
Проверять наличие истекших лицензий каждые	REMOVE_EXPIRED_ LICENSES_PERIOD= 60 [сек]	Положительное число	Периодичность проверки наличия истекших лицензий и их удаления
Порт сервера лицензирования	PORT=6001	Любой подходящий UDP-порт	TCP/UDP-порт, который использует сервер для обслуживания клиентов
Число рабочих потоков сервера	THREAD_NUMBER= 4	1 - 9	Количество потоков, обслуживающих сетевых клиентов
Сетевой интерфейс, используемый сервером	ADDRESS= !DEFAULT	IP-адрес адаптера	Сетевой адрес компьютера (сетевого адаптера), по которому будут приниматься запросы клиентов. !DEFAULT означает - использовать все доступные сетевые адаптеры
Новый пароль	-	-	Установка нового пароля. В файле шаблона grdsrv.ini.in пароль хранится только в виде хэша (параметр PWD_HASH)
Подтверждение пароля	-	-	Подтверждение нового пароля.
Сохранить и перезапустить сервер	-	-	Кнопка перезапуска сервера с новыми настройками
Перезапустить сервер	-	-	Кнопка перезапуска сервера без сохранения изменений в настройках

Клиент Guardant Net

Чтобы начать работу, клиентское приложение должно зарегистрироваться на сервере ключа.

При старте клиент начинает поиск сервера Guardant Net и продолжает работать только после успешной регистрации на сервере.

Клиент ищет сервер, как с помощью встроенных механизмов GrdAPI, так и используя настройки файла конфигурации

gnclient.ini, в частности IP-адрес (или сетевое имя) и сетевой порт. На запросы клиента откликается сервер Guardant Net, удовлетворяющий заданным параметрам поиска

При большом количестве клиентов может возрастать нагрузка на сеть и сервер ключа. В этом случае рекомендуется устанавливать несколько ключей и серверов Guardant Net на разные компьютеры.

Время отклика сервера зависит от его загрузки: раньше отвечает менее загруженный на данный момент сервер. Т. о., автоматически происходит балансировка загрузки сервера ключа.

Получение актуального файла настроек gnclient.ini

Хотя защищенное клиенское приложение пытается найти Guardant Net даже в отсутствии файла настроек, однако, только правильно заполненный **gnclient.ini** служит залогом успешного (и быстрого) поиска сервера и последующей работы с ключом.

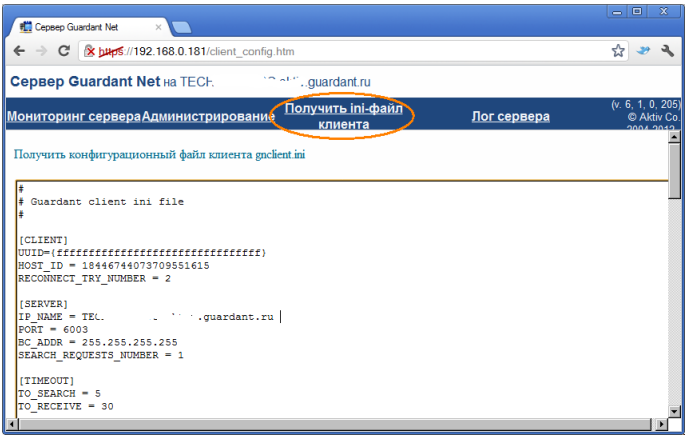
Конфигурационный файл должен находиться в одной директории с исполняемым файлом защищенного сетевого приложения. Если же он отсутствует, то при старте клиент автоматически создает **gnclient.ini** и сохраняет в нем правильные настройки после успешного поиска сервера, либо оставляет умолчательные, если поиск сервера был неудачным.

Принимая во внимание многообразие настроек сетей, где может работать защищенное приложение, нельзя гарантировать, что оно сможет найти сервер и ключ при отсутствующем, либо неправильно настроенном **gnclient.ini**.

Поэтому настоятельно рекомендуется до начала работы получить файл с актуальными настройками.

Для этого запустите интернет-браузер, наберите в адресной строке IP-адрес компьютера, на котором установлены сервер и ключ, в формате: **https://<путь>** (к примеру, **https://192.168.0.181**) и нажмите на клавишу ввода.

После открытия основной страницы web-интерфейса перейдите на вкладку **Получить ini-файл клиента** путем выбора гиперссылки в верхней части страницы:



После открытия страницы с настройками выделите, скопируйте и сохраните их в существующем, либо созданном заново файле с именем **gnclient.ini**.

Далее проверьте работоспособность защищенного приложения.

Настройки клиента Guardant Net. Файл GnClient.ini

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.

Секция	Параметр (и значение по умолчанию)	Возможные значения	Назначение
[CLIENT]	uuid	-	Универсальный идентификатор объекта; внутренний параметр, не требует настройки
	HOST_ID	-	Идентификатор хоста; внутренний параметр, не требует настройки
	RECONNECT_TRY_NUMBER = 2	?	Число попыток восстановить соединение с сервером Guardant Net
[SERVER]	IP_NAME =	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.
	PORT = 6001	Любой подходящий TCP/UDP-порт	Порт, который использует сервер для обслуживания клиентов. Значение должно быть идентичным со значением аналогичным параметра в grdsrv.ini

	BC_ADDR=255.255.255.255	-	Адрес широковещания. Константа 255.255.255.255 фактически означает, что при любой маске подсети все станции получат пакет
	SEARCH_REQUESTS_NUMBER = 1;	?	Число попыток повторения поиска сервера Guardant Net
[TIMEOUT]	TO_SEARCH = 5;	1 – 120 [сек]	Таймаут на широковещательный поиск сервера ключа
	TO_RECEIVE = 30	1 – 120 [сек]	Таймаут на прием данных клиента сервером ключа

Поддерживаемые сети и протоколы

Сетевое программное обеспечение Guardant Net может работать в любых локальных сетях, поддерживающих протокол TCP/IP (IPv4 и IPv6).

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 (либо в среде wine@etersoft под Linux).

Сетевое ПО Guardant Net может работать одновременно с несколькими сетевыми интерфейсами и сетевыми адаптерами.

Сетевое программное обеспечение Guardant не налагает ограничений на работу по TCP/IP. Доступности клиента по сети могут препятствовать только настройки сетевых экранов, маршрутизаторов и т. п. оборудования, а также настройки сетевых политик безопасности.

Т. о., допускается работа с Guardant Net по Интернет (удаленные филиалы, корпоративная VPN), однако эта возможность является побочной, она не тестировалась досконально.

Повышение надежности сетевого обмена

Периодический опрос электронного ключа

Защищенному сетевому приложению следует периодически проверять наличие электронного ключа. Используйте опцию проверки ключа по таймеру (/T) при автоматической защите приложения и/или производите опрос ключа при помощи функций API периодически из различных мест приложения.

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 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. По-

этому, 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, потому что резко возрастает вероятность пиковой перегрузки сервера. Также следует учитывать, что на одном сервере может быть зарегистрировано несколько ключей (каждый со своим сетевым ресурсом), и о возможности пиковых перегрузок.

2. Не рекомендуется включать 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 определенным событиям.

4. Не устанавливайте больших значений конфигурируемых параметров в ini-файле сервера Guardant Net. Это не принесет ожидаемого адекватного эффекта. 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).

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 7
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:

<table>
<tr><th rowspan="2">Вид защиты</th><th colspan="2">Тип электронного ключа</th></tr>
<tr><th>Локальные</th><th>Сетевые</th></tr>
<tr><td colspan="3">Драйверы Guardant</td></tr>
<tr><td rowspan="2">Любой</td><td colspan="2">Установка драйверов при помощи Microsoft Installer:
32- или/и 64-разрядные файлы GrdDriversRU.msi, Setup.Exe из директории Guardant\Guardant %#%\Drivers\x86\Windows или/и Guardant\Guardant %#%\Drivers\x64\Windows</td></tr>
<tr><td colspan="2">Установка драйверов из инсталлятора приложения при использовании драйверного Guardant API:
GrdDriversRU.msi (x86 и/или x64) и grddrv.dll</td></tr>
<tr><td colspan="3">Общие файлы защиты</td></tr>
<tr><td>Любой</td><td>1) Драйверы Guardant,
2) GrdDem32.exe</td><td>1) Драйверы Guardant,
2) Glds.exe, GrdSrv.ini, GnClient.ini, GrdDem32.exe</td></tr>
<tr><td colspan="3">Файлы для защиты Win32-приложений</td></tr>
<tr><td>Автозащита Native-приложений</td><td colspan="2">1) Общие файлы защиты
2) Внешняя вакцина GrdVkc32.dll</td></tr>
<tr><td>.NET-автозащита</td><td colspan="2">1) Общие файлы защиты
2) CodeStorage32.dll – для x86 операционных систем
3) CodeStorage64.dll – для x64 операционных систем</td></tr>
<tr><td>Защита Guardant API, использующая динамические библиотеки</td><td colspan="2">Соответствующий файл динамической библиотеки</td></tr>
<tr><td>Удаленное программирование ключа</td><td colspan="2">1) Общие файлы защиты;
2) GrdTRU.exe
(или GsRemote для Guardant Stealth II/Net II)</td></tr>
</table>