УДК 681.324

Палагин А.В. академик НАНУ,
Алишов Н.И. д-р техн. наук,
Опанасенко В.Н. д-р техн.наук,
Сахарин В.Г. канд. техн. наук,
Марченко В.А.
Лисовый А.Н.,
Закутайло Д.А.

ЭЛЕКТРОННЫЙ *USB*-КЛЮЧ ЗАЩИТЫ БОЛЬШИХ БАЗ ДАННЫХ ОТ НЕСАНКЦИОНИРОВАННОГО КОПИРОВАНИЯ

Институт кибернетики НАН України

В статье рассматривается устройство защиты больших баз данных от несанкционированного копирования разработанное коллективом авторов. Приводятся алгоритмы работы различных функциональных блоков ключа, а также даются рекомендации по использованию данного устройства для защиты баз данных.

В свете постоянных сообщений об утечках персональных данных пользователей, а также закрытых баз данных (БД) вопросам защиты информации уделяется особое внимание [1]. Коллективом авторов были проведены исследования, посвящённые данной проблеме. В результате были предложены некоторые методы защиты от несанкционированного копирования больших БД [2].

В данной статье излагаются некоторые особенности создания аппаратно-программных решений защиты от копирования. Приводятся практические результаты, полученные коллективом, а также описывается архитектура и особенности конечных устройств защиты.

Данная разработка направлена на обеспечение защиты от копирования больших БД или их значительных частей несанкционированным пользователем или пользователем, обладающим полным набором привилегий. Разработанный способ защиты предполагает применение аппаратного ключа (электронного ключа) шифрования, а также специальных алгоритмов взаимодействия между программным обеспечением, реализующим БД, и функциями, предоставляемыми электронным ключом.

В программно-аппаратном решении ключа реализуется общеизвестная схема шифрования *RSAES-PKCS1-v1_5* (*RSA Encryption Scheme,Public Key CryptoSystem #1, version 1.5*) [3]. Согласно этому стандарту алгоритм шифрования представляет собой два метода – шифрование и дешифрование. Эти алгоритмы в свою очередь состоят из нескольких примитивов. Каждый примитив представляет собой некий набор операций линейных или нелинейных преобразования данных. В этой схеме используются криптографические примитивы *RSAEP* и *RSADP,* а также примитивы преобразования данных *OS2IP* (*Octet-String-to-Integer primitive*) и *I2OSP(Integer-to-Octet-String primitive).*

Примитив *OS2IP* преобразует исходный текст, который необходимо зашифровать в целое положительное число, которым оперирует алгоритм *RSA.* Примитив *I2OSP* преобразует целое положительное число, полученное на выходе алгоритма дешифрования в исходный текст, который был зашифрован.

Криптографические примитивы *RSAEP* и *RSADP* реализуют одну и туже операцию возведение в степень. Так в примитиве *RSAEP* она имеет следующий вид:

$$c = m^e \bmod n,$$

где n - открытый ключ шифрования длинной n бит; e - экспонента шифрования; m - шифруемый текст представленный целым числом длинной до $n-1$ бит; c - шифротекст получаемый на выходе операции возведение в степень.

В примитиве *RSADP* она имеет следующий вид:

$$m = c^d \bmod n,$$

где n - открытый ключ шифрования длинной n бит; c - шифротекст получаемый на выходе примитива *RSAEP* операции шифрования; d - закрытый ключ шифрования; m - расшифрованный текст представленный целым числом длинной до $n-1$ бит.

В ключе реализован следующий вариант данной схемы. При шифровании исходный текст делится на блоки длинной по 53 бита и потом дополняются псевдослучайной последовательностью до 64 бит. Эта псевдопоследовательность имеет следующий вид:

$$EM = 0x00 \,\|\, 0x02 \,\|\, PS \,\|\, 0x00 \,\|\, M,$$

где M - исходный текст длинной 53 бита; $\|$ - операция конкатенация; *PS*- псевдослучайная последовательность длинной 8 бит; *EM*- выход операции конкатенация. После чего готовое сообщение преобразуется с помощью примитива *OS2IP* в целое позитивное число и затем шифруется с помощью примитива *RSAEP*. После чего применяется примитив *I2OSP* и результат сохраняется.

При дешифровании сначала применяется примитив *OS2IP* и шифрованный текст преобразуется в целое позитивное число. Затем с помощью примитива *RSADP* проводится операция дешифрования, и преобразуется в текст с помощью примитива *I2OSP*. В конце алгоритма дешифрования проводится операция выделения исходного текста из вышеуказанной псевдослучайной последовательности.

Использования такой схемы шифрования дешифрования позволяет аппаратно реализовать примитивы *RSAEP* и *RSADP*, а также хранить открытый и закрытый ключ внутри устройства. В таком случаи ключи никогда не покидают устройства на протяжении всего жизненного цикла ключа.

Устройство защиты представляет собой программно-аппаратный ком плекс и состоит из трёх составных частей (рис. 1).

Особенностями разработанного устройства является набор определённых требований к защищаемой БД.

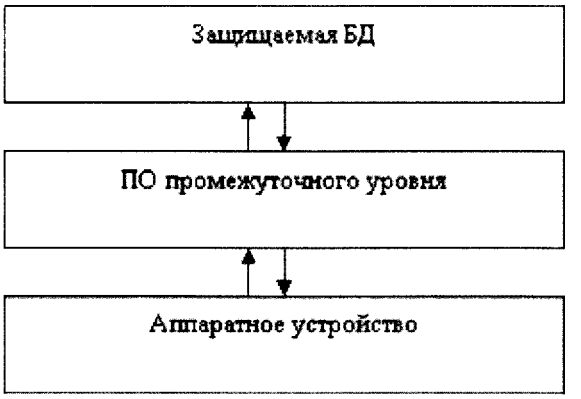

Рис. 1. Компоненты комплекса защиты

Выполнение этих требований позволяет гарантировать корректность выполнения защитных функций устройством защиты. К таким требованиям относятся:

- размер защищаемой БД;
- производительность БД;
- особая организация вычислений в БД.

Размер защищаемой БД. Защищаемая БД должна обладать «большим размером». Это значит, что объем данных защищаемой БД должен быть достаточным, для принятия решения о нецелесообразности копирования содержимого БД путём считывания данных с экрана терминала или любого другого устройства вывода (принтер, факс и т.д.). Данное условие является достаточно формальным и не имеет чётких параметров оценки и проверки. С практической точки зрения можно считать что БД, имеющая от десяти тысяч записей уже пригодна для защиты данным устройством.

Производительность БД. Так как в основе защитного устройства лежит использования алгоритмов шифрования то производительность БД ограничивается возможностями ключа. Для увеличения производительности предлагаться защищать не всю БД, а только поля содержащие «чувствительную» информацию. Это с одной стороны позволить увеличить общую производительность системы а с другой избавит от лишних затрат на защиту общедоступной информации. В связи с этим данное устройство защиты может быть применено только разработчиком БД, так как необходимо учесть данную особенность при разработке и реализации защищаемой БД.

Особая организация вычислений в БД. Это требование опирается на особенности алгоритма защиты реализованного во всём программно аппаратном комплексе. Так следует ограничить возможность доступа оператора БД к большому количеству записей одновременно. В противном случае существует возможность скопировать всю БД путём снятия данных с экрана терминала или других устройств вывода.

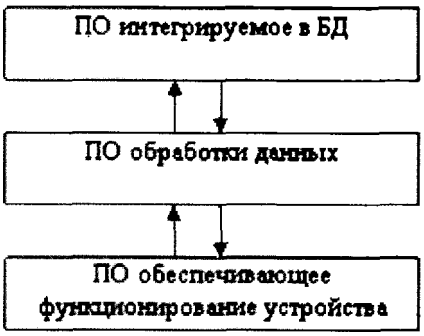

Рис. 2. Состав ПО промежуточного уровня

Также следует построить логику функционирования ПО исходя из того, что защищаемые поля не должны быть используемыми автоматически без запроса пользователя.

ПО промежуточного уровня представляет собой программный комплекс, связывающий саму защищаемую БД и аппаратное устройство (рис. 2).

ПО интегрируемое в БД представляет собой набор *API* функций включаемых разработчиком в свою программу. В основе этого набора лежит несколько функций, такие как передача данных на шифрование, передача данных для дешифрования и несколько дополнительных служебных функций. Поэтому разработчик при реализации алгоритма роботы программы передает необходимые данные из защищаемых полей (которые изначально определяются на этапе проектирования) в устройство защиты на обработку, а только после этого использует их в программе.

По сути эти *API* функции для разработчика представляют собой чёрный ящик с заранее заданным форматом представления используемых данных. На данный момент эта часть реализована в инструментальной среде программирования *Delphi*, хотя и вполне возможна реализация как на других высоко уровневых, так и на машиннозависимых языках программирования.

ПО обработки данных реализует всю логику обработки данных передаваемых как из ключа, так и из защищаемой БД. При этом производиться проверка правильности переданных данных, изменения формата представления передаваемых данных.

ПО обеспечивающее функционирование устройства управляет работоспособностью электронного ключа. В основном ПО представляет собой специализированный драйвер для устройства.

Аппаратное устройство (электронный ключ) защиты больших БД от копирования имеет следующую функциональную схему (рис. 3) [4].

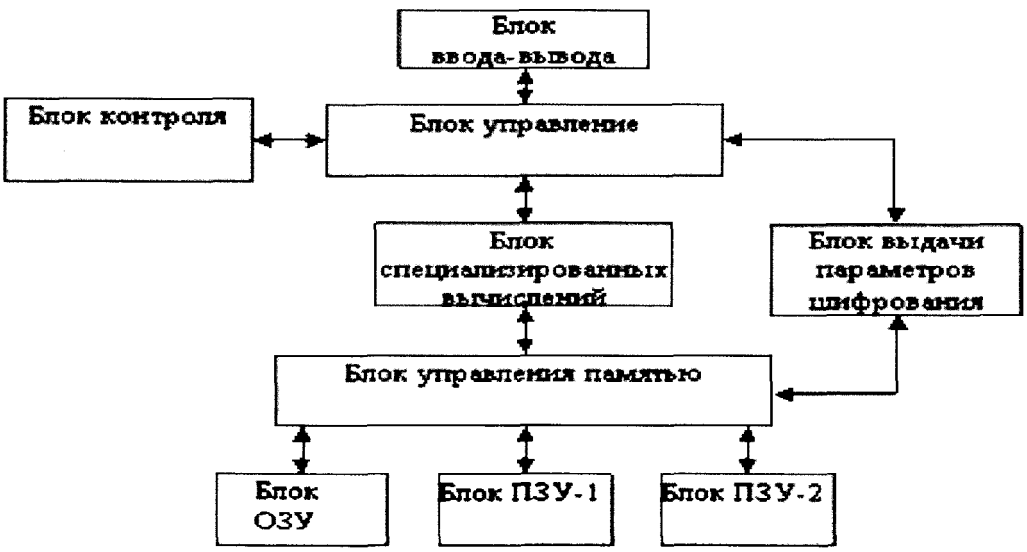

Рис. 3. Функциональная схема электронного ключа

Блок ввода-вывода представляет собой интерфейсную часть электронного ключа, обеспечивающая связь вычислительного устройства с локальным компьютером, на котором это устройство установлено. Этот блок используется для организации информационного обмена между ключом и промежуточным ПО. Техническая реализация имеет две разновидности аппаратных интерфейсов *PCI* и *USB*. Использование двух разных интерфейсов позволяет организовывать защиту БД различными способами. Так использовании *USB* интерфейса позволяет организовать лицензирование и защиту коммерческого ПО в виде БД. В таком случае при распространении БД соответствующее ПО передается пользователю совместно с ключом защиты. И при попытке запуска без аппаратного ключа защиты работа с защищаемой БД будет невозможна. Использование *PCI* интерфейса нацелено на применения в промышленных БД нуждающихся в защите от кражи легальными пользователями. В таком исполнении аппаратное устройство находится внутри сервера БД и его физическая кража возможна только при вскрытии корпуса или физического изъятия сервера, что трудно выполнимо. А при попытке копирования БД на мобильный носитель доступ к содержимому БД будет невозможен, так как отсутствует необходимый ключ защиты установленный только в сервер БД.

Блок управления реализует требования, выдвигаемые стандартом реализованного интерфейса, а также организует связь разных функциональных блоков электронного ключа (вычислительного устройства) с внешним миром, контролирует информационные потоки внутри устройства.

Блок контроля реализует набор алгоритмов защиты, которые позволяют определить попытки взлома или несанкционированного копирования БД. При проектировании защищаемой БД изначально определяются некоторые параметры корректного функционирования системы. Набор таких параметров представляет собой модель нормального функционирования системы. Применяя методы, используемые в системах обнаружения вторжений [5] можно определять параметры анормального функционирования системы и соответственно реагировать на эти нарушения. В качестве параметров анализируемых данным блоком являются:

- количества запросов к аппаратному ключу из защищаемой БД в единицу времени;
- количество выдаваемых полей из ключа в защищаемую БД в единицу времени;
- время непрерывной работы.

Особенностью этих параметров является то, что они определяются внутри самого ключа и не могут быть скорректированы извне, что предотвращает попытку обмана системы контроля разными способами внешним пользователем. Количественные характеристики этих параметров легко определить из специфики защищаемой БД и условий её функционирования. Так если в качестве защищаемой БД будет выступать мультемидийный справочник, то логично предположить что обычный пользователь физически не сможет запрашивать данные со скоростью 1 позиция в секунду.

Тогда можно говорить с большой долей уверенности, что с БД работает не человек и соответственно блокировать дальнейшую роботу. В качестве методов противодействия можно использовать следующие функции:

- введение дополнительных задержек при выполнении очередного запроса;
- полная блокировка работы при определённом количестве срабатываний системы защиты.

В общем случае этот блок позволяет противодействовать некоторым методам считывания всей БД или значительной её части за относительно малый промежуток времени. Но он не обеспечивает защиту БД от копирования тех ее записей, которые выдаются на экран монитора, или от перехвата расшифрованных данных при их передаче от электронного ключа к центральному процессору. И потому нужно как можно точнее выбирать параметры

нормального функционирования для блока контроля.

Микропроцессор в данном случае – это специализированная микросхема, в которой аппаратно реализован один из алгоритмов асимметричного шифрования, в частности *RSA*. Криптографическая система *RSA* является классическим примером криптографической системы с открытыми ключами. Она была предложена в работе [6] и в настоящее время получила очень широкое распространение. Несмотря на многочисленные критические работы по изучению ее свойств, в которых указаны наборы параметров, приводящие к ослаблению схемы, при правильном использовании она считается безопасной.

Микропроцессор осуществляет шифрование/расшифровывание данных, поступающих из защищаемой БД. При этом ключ для расшифровки расположен в недоступном извне, специальном участке памяти ПЗУ 2.

Блок выдачи параметров шифрования управляет выдачей параметров шифрования, служебных данных, а также открытого ключа по запросу внешнего пользователя. Сами данные хранятся в недоступном для внешнего пользователя участке памяти с целью их защиты от несанкционированного использования. А выдача происходит только по запросам специальных *API* функций предоставляемых разработчику защищаемой БД.

Блок управления памятью контролирует обмен данными между разными участками памяти и другими функциональными блоками, такими, как микропроцессор и блок выдачи параметров ключа.

Блок ОЗУ служит для размещения промежуточных данных вычислений и имеет достаточный объём для проведения необходимых операций криптоалгоритма. Эта память является энергозависимой и потому все данные пропадают при отключении питания.

Блок ПЗУ 1 предназначен для размещения таких данных, как начальные параметры инициализации криптоалгоритма, открытый ключ шифрования, порядковый номер ключа и др. Этот блок является энергонезависимым и сохраняет свои данные после отключения устройства от компьютера.

Блок ПЗУ 2 содержит недоступные для внешнего пользователя данные – закрытый ключ шифрования, уникальный номер электронного ключа и т.п. После записи значений в данный участок памяти она блокируется от повторной записи или изменения некоторых её элементов. Этот блок дополнительно защищён от механического вскрытия и считывания информации.

Проектирование и реализация разрабатываемого устройства с использованием современной элементной базе было выполнено на ПЛИС. Одним из подходов к проектированию устройства является использование *DL*-технологии, которая представляет собой комплекс инструментальных средств САПР *Foundation ISE* и методологию проектирования, ориентированных на описание проекта с помощью языка *HDL (VHDL* или *Verilog*). Основными предпосылками для внедрения в практику *HDL*-технологии являются: внедрение промышленных стандартов, обеспечивающих эффективное и оптимальное проектирование средств электроники и вычислительной техники; увеличение логической емкости элементной базы; развитие инструментальных средств САПР и в первую очередь средств автоматического синтеза и верификации. Такие возможности *HDL*–технологии, как иерархическое проектирование, переносимость библиотек, платформонезависимость, позволяют использовать имеющиеся *IP*-блоки в качестве макроэлементов для разработки новых технических решений в виде *IP*-блоков (*Soft Cores*).

Если до недавнего времени основным способом описания проектов являлся графический синтез схемы устройства, выполняемый схематическим редактором, то в последнее время предпочтение отдается текстовому описанию, подобному тексту программы. Этим достигается компактность и более строгая формализация описания в соответствии со стандартом используемого языка.

Существенным преимуществом ПЛИС по сравнению с полузаказными и заказными большими интегральными схемами являются их универсальность и

возможность быстрого программирования и перепрограммирования под заданный проект.

Разработка цифровых устройств на ПЛИС существенно упрощается за счет использования готовых технических решений, называемых *IP (Intellectual Property),* – блоками, ядрами (*Core*). *IP*-блоки являются модулями, параметры которых настраиваются пользователем под конкретные требования разрабатываемого проекта. В настоящее время подготовка таких технических решений многократного использования (*Design Reuse*) сформировалась в отдельную областьдеятельности, осуществляемую рядом фирм для различных классов цифровых устройств.

Основным средством верификации проектов методом моделирования в современных системах проектирования ПЛИС типа *FPGA* фирмы *Xilinx* является система *ModelSim.* К наиболее важным свойствам *ModelSim* относятся:

- возможность поведенческого моделирования объектов, описанных на *HDL*-языке;
- высокая скорость моделирования;
- *высокая (до 1 фемтосекунды) разрешающая способность;*
- широкий выбор систем счисления и форматов представления данных (например, целых чисел в десятичной системе счисления со знаком, в том числе в аналоговом формате);
- возможность использования *HDL*-языка для подготовки файлов входных воздействий (*Test Bench*) и т.д.

Верификация может производиться на различных стадиях выполнения проекта.

Блок реализации алгоритма *RSA* в кристалле *Spartan-3 (XC3S400-FG456*) состоит из элементов, приведенных в таблице.

Общее число секций *(Slices*), занимаемых блоком в кристалле, составляет 1928 из общего числа 3584, т.е. примерно 53%. На вход блока подается 128-разрядный сигнал для шифрования или дешифрования, а значения трех ключей (открытого, закрытого и модуля) хранятся в блоке в виде констант. Окончание процесса шифрования или дешифрования определяется наличием сигнала готовности, формируемым одновременно с выходным 128-разрядным сигналом (табл. 1).

Число циклов преобразования зависит от алгоритма работы блока, а частота - от быстродействия кристалла и его ресурсов, использованных при размещении и трассировке блока. Максимальная задержка распространения сигналов в кристалле имеет место при выполнении операций суммирования и вычитания чисел большой размерности. В структуре кристалла для выполнения арифметических операций предусмотрена специализированная логика ускоренного переноса между секциями и между разрядами внутри секции. Цепи ускоренного переноса связывают секции, расположенные по вертикальной оси кри внутри секции. Цепи ускоренного переноса связывают секции,

Таблица 1.

Наименование элемента	Разрядность	Кол-во	Наименование элемента	Разрядность	Кол-во
Сумматор	130	2	Мультиплексор 2→1	130	6
Вычитатель	130	6	Мультиплексор2→1	128	4
Регистр	130	8			
Регистр	128	6			
Регистр	1	260			

расположенные по вертикальной оси кристалла, поэтому число разрядов, соединенных этими цепями, определяется числом вертикальных секций структуры. В используемом типе кристала число таких разрядов в 64 вертикальных секциях равно 128. Максимальная частота синхронизации равна 38,5 МГц.

Максимальная частота синхронизации такого же блока, размещаемого в кристалле *XCS200* равна 31,5 МГц, а в кристалле *XCS1000* – 46,5 МГц.

Аппаратная реализация ключа, написана на языке *VHDL.* Это позволяет использовать готовые микросхемы с из-

меняемой логикой функционирования. Дополнительным преимуществом использования языка *VHDL* является возможность гибкого изменения алгоритма функционирования любого блока аппаратного ключа (рис. 5), что и было использовано при разработке ключа с интерфейсом *USB* и *PCI*. Ниже приведено описание реализации с интерфейсом *USB*.

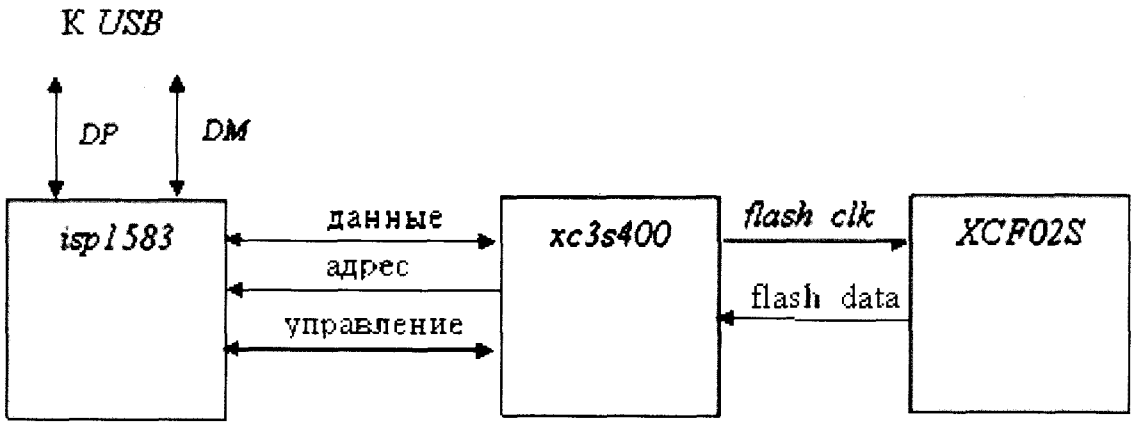

Рис. 5. Структурная схема устройства кодирования и декодирования информация по шине *USB*

Для обслуживания шины *USB* применятся микросхема *isp*1583 фирмы *NXP*. Выбор этой микросхемы обусловлен тем, что она поддерживает работу в *Hi-speed* (480 *Mbit/s*) режиме шины *USB* [7]. Эта микросхема подсоединена к разъему *USB* посредством сигналов: *DP, DM*. С помощью шины «данные» осуществляется двухсторонний обмен данными между контроллером *USB* и ПЛИС. Посредством шины «адрес» ПЛИС осуществляет обращение к внутренним регистрам контроллера *USB*. Шина «управление» необходима для синхронизации работы между контроллером *USB* и ПЛИС. Алгоритм *RSA*, начальная инициализация и работа с контроллером шины *USB* реализованы в ПЛИС *XC3S400* фирмы *Xilinx*. Микросхема памяти *XCF02S* фирмы *Xilinx* содержит файл конфигурации, который загружается в ПЛИС при запуске устройства. Загрузочные данные передаются с помощью сигнала *flash_data* по переднему фронту сигнала *flash_clk*. Устройство питается от шины *USB* и потребляет в режиме шифрования 99 *mA*.

Для создания конфигурационного файла ПЛИС использовался свободно распространяемый пакет *ISE WebPACK* версии 8.2 фирмы *Xilinx*. Устройство описано на языке *VHDL*. Создание описания устройства на этом языке значительно удобнее и быстрый чем работа в схемотехническом редакторе. С помощью пакета *ISE WebPACK* описание устройство на языке *VHDL* транслируется в файл загрузки для ПЛИС. Для отладки устройства в ПЛИС был реализован *UART*. *UART* соединялся с компьютером с помощью микросхемы *CP2103* фирмы *Silabs*. *CP2103* представляет собой преобразователь *RS232<->USB*. Дальнейшее совершенствование устройства можно вести по следующим направлениям:

- увеличить размерность ключа;
- использовать ПЛИС, в котором уже содержится *flash* память;
- оптимизировать описание на *VHDL*.

Размерность ключа в макетном варианте устройстве равна 64 битам. Описание устройства, содержащего большую размерность ключа, пока не удалось реализовать в данной версии ПЛИС из-за временных ограничений по тактовому сигналу.

Использование ПЛИС, содержащей *flash* память, позволит уменьшить размеры устройства. Необходимо изменить реализацию алгоритма *RSA*, с целью более полного использования внутренних ресурсов ПЛИС. В частности, *XC3S400* содержит 16 умножителей 18 на 18 бит, которые не были использованы.

На рис. 6 приведена фотография опытного образца с *USB* интерфейсом. Для подключения этого устройства к компьютеру, на котором установлена защищаемая БД необходимо предустановить специальный драйвер. Этотдрайвер выполняет все функции стандартного драйвера *USB* устройства, и поэтому полностью совместим со всеми операцион-

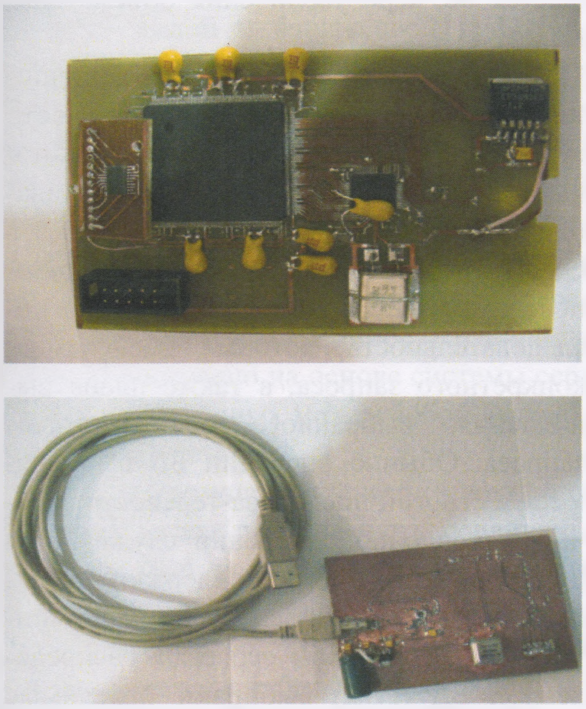

Рис. 6. Фото опытного образца с *USB* интерфейсом

ными системами линейки *WINDOWS*. Устройство не требует дополнительных настроек или технических ухищрений при установке и эксплуатации. Оно сразу готово к работе после подключючения и установке драйвера аппаратного *USB* интерфейса.

Список литературы

1. Статистика по утечкам чувствительных данных / Available at http://www.securitylab.ru/.
2. *Палагин А.В,. Алишов Н.И, Марченко В.А., Широков В.А.* Технология защиты больших баз данных от несанкционированного копирования // Комп'ютерні засоби, мережі та системи. – 2006. – № 5. – С. 73 – 79.
3. PKCS #1: RSA Encryption Standard. RSA Laboratories Technical Note Version 1.5 Revised November 1, 1993.
4. *Пат.* 19899 UA, МПК G09C 1/06. Обчислювальний пристрій захисту баз даних / *Н.І. Алішов, В.А. Марченко, О.В. Палагін, В.А. Широков.*– Ін-т кібернетики ім. В.М. Глушкова НАН України. – № u 2006 02949; Заявл. 20.03.2006; Опубл. 15.01.2007. Бюл. № 1. – 12 с.
5. *Bace R., Mell P.* Special Publication on Intrusion Detection Systems, Tech. Report SP 800-31.– Gaithersburg: National Institute of Standards and Technology.– 2001. – November. – P. 54 – 63.
6. *Rivest R. L., Shamir A., Adleman L.* A method for obtaining digital signatures and public-key cryptosystems // Commun. ACM. — 1978. – V.21, No.2. – P. 120 – 126.
7. *Описание* стандартов *USB* устройства / Available at http://www.usb.org/.