**УДК 621.37**

***Вержаковский А.А.,***
***к.т.н. Пупков В.С.***
***(ДонГТУ, г. Алчевск, Украина)***

# КЛЮЧ ЗАЩИТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ИСПОЛЬЗУЮЩИЙ ВСТРОЕННУЮ ОПЕРАЦИОННУЮ СИСТЕМУ

*Запропоновано концепцію апаратного ключа захисту програмного забезпечення який використовує спеціалізовану вбудовану операційну систему і програму завантажувач.*

***Ключові слова:*** *ключ захисту програмного забезпечення, операційна система реального часу, мікроконтролер, завантажувач.*

*Предложена концепция аппаратного ключа защиты программного обеспечения, который использует специализированную встроенную операционную систему и программу загрузчик.*

***Ключевые слова:*** *ключ защиты программного обеспечения, операционная система реального времени, микроконтроллер, загрузчик.*

**Проблема и ее связь с научными и практическими задачами.**

Использование электронных ключей защиты программного обеспечения приобретает в последнее время все большую популярность среди производителей программного обеспечения и находится в постоянном развитии. Необходимость использования систем защиты программного обеспечения обусловлена рядом проблем, среди которых следует выделить несанкционированное использование программного обеспечения и незаконное его распространение и сбыт. Положительные сторонами использования именно электронных ключей защиты является то, что данные системы обеспечивают высокий уровень защиты программного обеспечения от анализа его алгоритмов и существенно увеличивают стойкость систем защиты других типов [1]. Поэтому электронные ключи защиты востребованы и успешно используются многими разработчиками программных продуктов. В течение последних лет аппаратные средства защиты ПО прошли не один этап развития от простейших элементов памяти до сложных микропроцессорных устройств, которые реализуют разнообразные алгоритмы проверок, основанные на современных методах шифрования, а также допускают возможность переноса выполняемого кода в электронный ключ [2, 3]. Но применение

микропроцессорных ключей защиты накладывает определенные трудности на сам процесс разработки программного обеспечения, поскольку требуется определенные знания для портирования участков кода в электронный ключ. Большинство производителей, электронных ключей защиты, поставляет специальные утилиты, упрощающие этот процесс. Но, несмотря на это использование таких ключей, существенно осложняет процесс создания программного обеспечения, а также вызывает значительные трудности при смене поставщика электронных ключей защиты.

**Анализ исследований и публикаций.**

Современные ключи (Guardant Code от Компании "Актив", LOCK от Astroma Ltd., Rockey6 Smart от Feitian, Senselock от Seculab) позволяют разработчику хранить собственные алгоритмы или даже отдельные части кода приложения (например, специфические алгоритмы разработчика, получающие на вход большое число параметров) и исполнять их в самом ключе на его собственном микропроцессоре, а также другие сервисные функции [4]. Например электронный ключ с загружаемым кодом Guardant Code позволяет не только выполнять произвольный код в памяти микроконтроллера но и дает возможность доверенного удаленного программирования электронного ключа. Что особенно важно, когда появляется насущная необходимость перепрограммировать ключи, находящиеся у конечных пользователей [5].

Основные этапы переноса кода в электронный ключ следующие:

- Произвольный код подбирается и подготавливается таким образом, чтобы его можно было перенести в электронный ключ.
- Алгоритм преобразуется для использования в электронном ключе.
- При помощи утилит поставляемых производителем ключа скомпилированный алгоритм помещается в ключ.
- После этого загруженный код используется из защищенного приложения вызовом специальных API функций.

Поэтому для использования ключа защиты требуются достаточные знания не только в области безопасности и системного программирования а также изучение специализированных комплектов средств разработки.

Существенно упростить сложность разработки защищенного программного обеспечения и снизить цену ключей защиты можно за счет применения встроенной операционной системы, которая позволит унифицировать процесс переноса кода в электронный ключ, сделав его прозрачным для пользователя.

**Изложение материала и его результаты.**

Проектирование нового ключа защиты основывалось на следующих новых технологиях микропроцессорной техники:

– прошивка микроконтроллера посредством программы загрузчика (bootloader);

– встроенные операционные системы реального времени.

Комбинирование этих двух технологий дает возможность существенно упростить разработку ключа защиты и в значительной мере унифицировать процесс загрузки кода приложения в электронный ключ.

При разработке ключа использовался контролер семейства AVR. Практически все микроконтроллеры серии Mega с памятью от 8КБ могут программироваться с использованием программы загрузчика. При загрузке контроллера управление первым делом передается загрузчику, и он проверяет, есть ли условие для запуска. Условие может быть любым, но обычно это либо наличие специализированной посылки по последовательному интерфейсу, либо наличие нужного логического уровня на выбранной ножке контроллера. Если условие есть — то загрузчик может, например, принять прошивку по UART и сам прошить ее во флеш память. Если разрешающего условия при старте нет, то загрузчик завершает свою работу и передает управление основной программе [6]. Поставляемые фирмой Atmel загрузчики поддерживают шифрованные загружаемые программы, что дает возможность безопасно обновлять прошивку (рис. 1).

Производитель, используя ключ шифрования (алгоритм AES) подготавливает программу загрузчик и помещает его в специальный раздел памяти микроконтроллера стандартным программатором. Подготовленная первоначальная прошивка ключа шифруется и передается загрузчику, ключ готов к работе. В случае необходимости сменить прошивку ключа обновление может свободно распространятся по любому допустимому каналу. Безопасность обеспечивается за счет использования уникальных ключей шифрования для каждого ключа защиты.

Для удобства генерирования ключей была создана утилита для автоматизации данного процесса. Также разработана программа для шифрования прошивки контроллера. Программа при обращении к ней будет выдавать на выходе необходимый зашифрованный файл. Шифровка будет выполняться в зависимости от серийного номера устройства (рис. 2).

Что бы упростить процесс обновления, предполагается сохранять необходимую информацию в специальной базе данных (рис. 3).

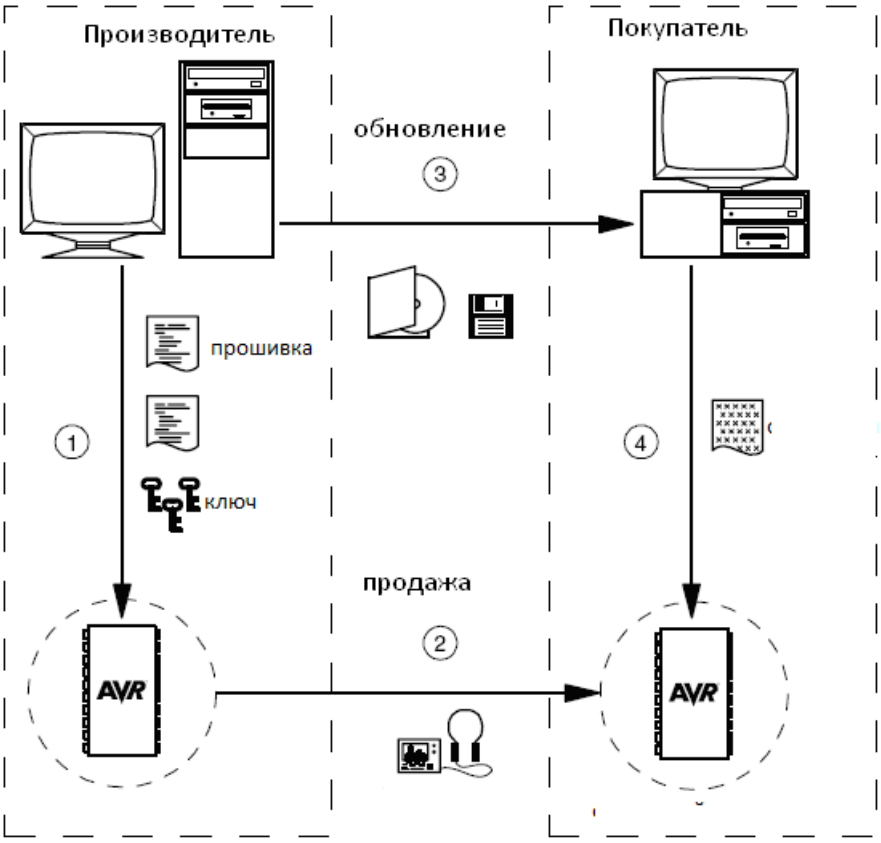


Рисунок 1 – Процесс безопасного обновления программ

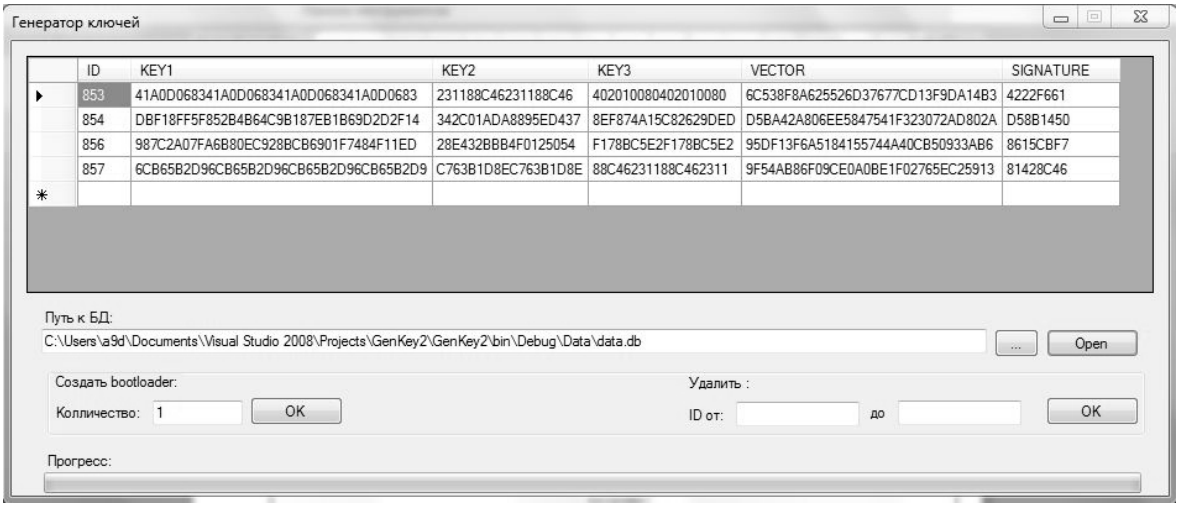


Рисунок 2 – Программа для формирования шифрования прошивок

Электронный ключ будет состоять из двух микросхем микроконтроллера Atmega8 (его преимущество в том, что этот микроконтроллер имеет встроенную EEPROM память размером 512 байт с гарантирован-

ной надежностью в 100000 циклов записи) и преобразователя интерфейса USB 2.0 – UART – Cp2103.

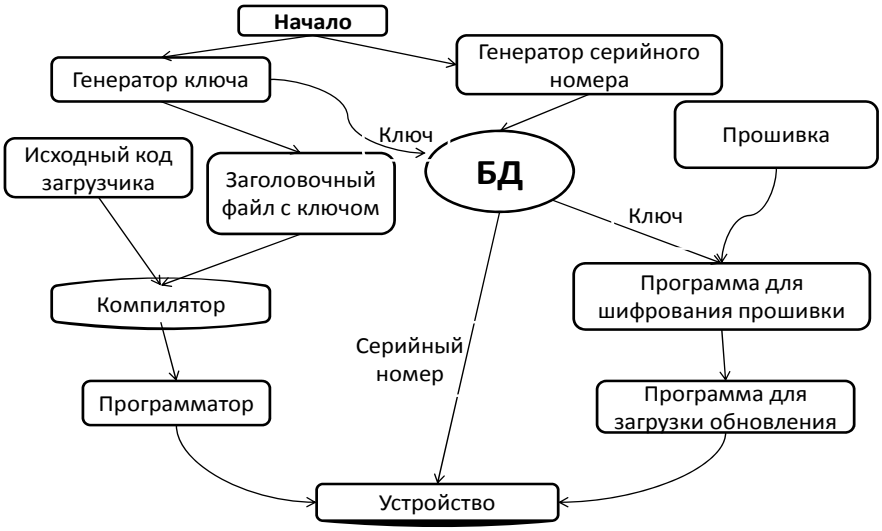


Рисунок 3 – Диаграмма процесса использования шифрованных прошивок ключа

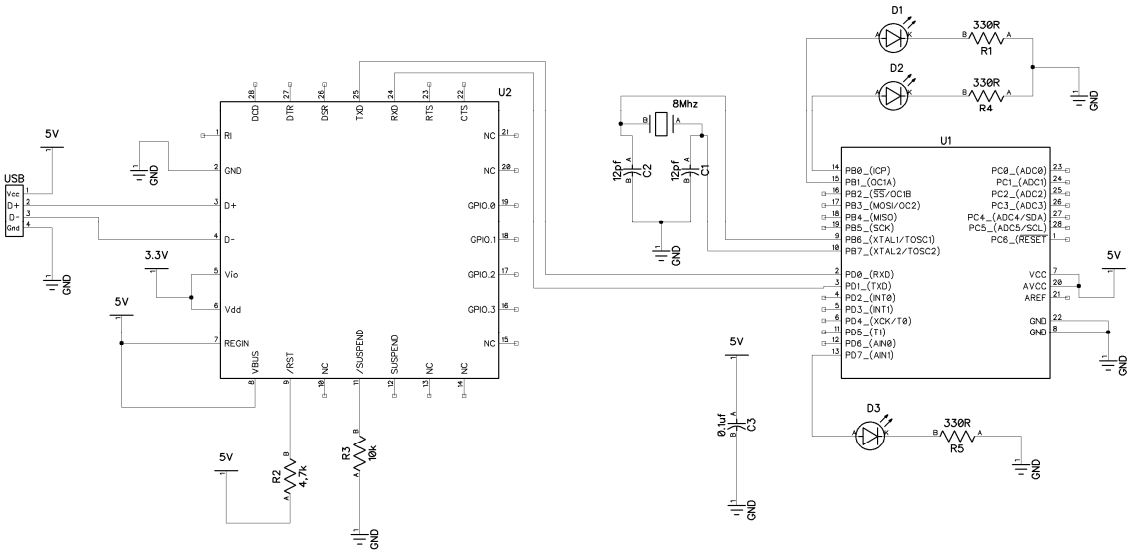


Рисунок 4 – Принципиальная схема ключа защиты

Ключ также оснащен тремя светодиодными индикаторами: зеленый - ключ функционирует нормально; желтый - ресурс ключа, возможно, исчерпан; красный - ресурс ключа исчерпан и ключ заблокирован.

Ядром механизма портирования кода в ключ является специализованная операционная система (ОС), построенная на базе ОС scmRTOS, которая обеспечивает работоспособность ключа [7]. Для воз-

можности ограничить время использования ключа ОС реализует виртуальный таймер. Также реализована программная защита EEPROM памяти содержащей сервисную информацию, что повышает надежность электронного ключа. Для удобства переноса кода реализован специальный класс обвертка, который позволяет правообладателю поместить защищаемый код в ключ без специальных знаний.

К достоинствам использования ОС следует отнести также абстрагирование от железа и реализованный пакетный протокол работы с USART. Что позволит в дальнейшем без особых трудностей перейти на другую микропроцессорную платформу. Блок-схема алгоритма работы ОС представлена на рисунке 5.

Класс обвертка облегчает процесс переноса защищаемого кода в ключ. Для работы с этим классом не нужны специализированные знания из области МК. Этот класс позволяет: добавлять методы, которые содержат вынесенные куски кода; осуществлять обработку принятых пакетов данных; формировать и отправлять пакеты данных. Формат добавляемых функций следующий:

```
* Структура функции пользователя:
 * void CFunc::NameFunc()
 *{
 *//================FUNC1==================
 *//     Описание функции
 *//================INIT==================
 *........
 *инициализация параметров функции
 *........
 *//================CODE==================
 *.......
 *код
 *.......
 *//================RETURN================
 *......
 *если необходимо, то вернуть результат
 *......
 *//======================================
 *}
 *
```

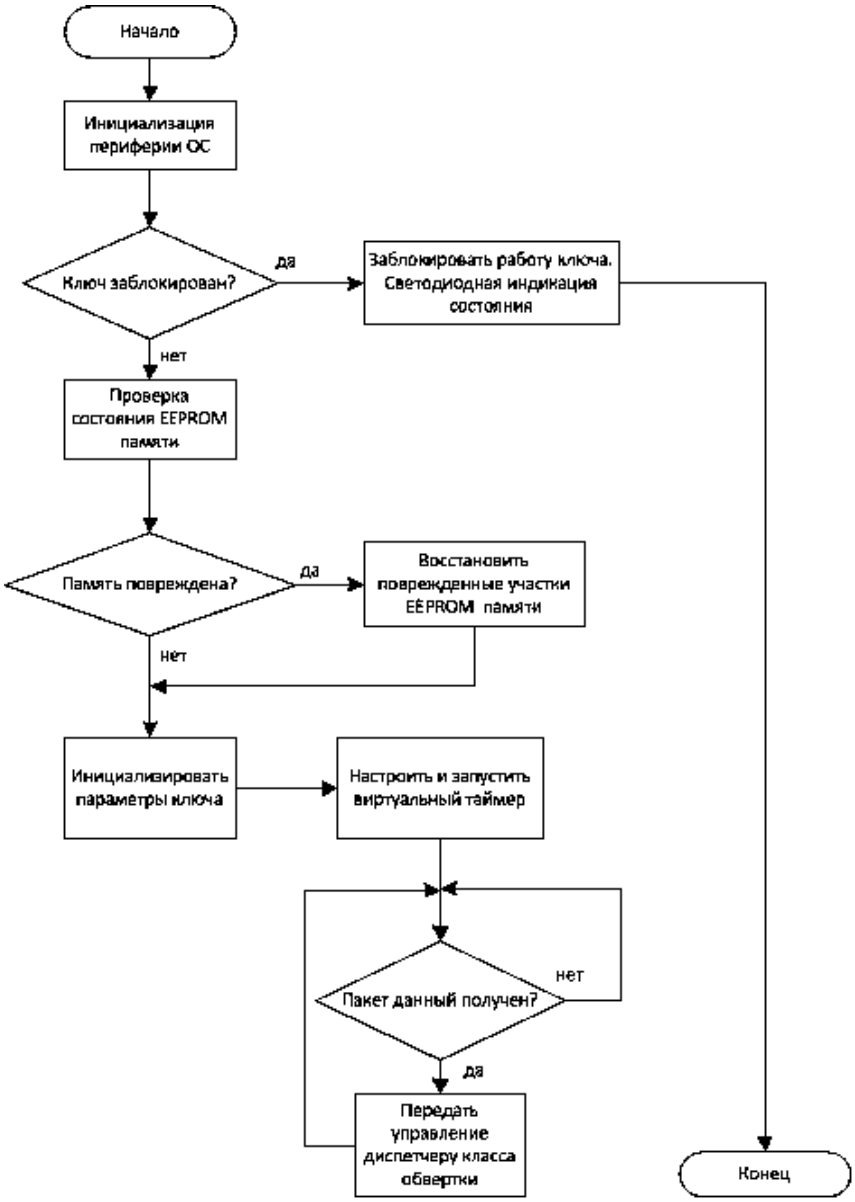


Рисунок 5 – Блок-схема алгоритма работы ОС ключа защиты

Также класс содержит служебные функции для считывания и записи различных типов данных и завершения пакета данных. Алгоритм работы диспетчера класса представлен на рисунке 6.

Формат пакетов данных передаваемых ключу следующий:

| | | |
|---|---|---|
| FuncNum - 1 байт | Size - 1 байт | Data - 0..127 байт |

Первый байт идентифицирует номер функции, за ним следует размер передавемых ключу данных и соответственно сами данные.

Ключ возвращает пакеты следующего вида

| | |
|---|---|
| Size - 1 байт | Data - 0..127 байт |

В ключе используется виртуальный таймер, а не реальный, потому что виртуальному таймеру не требуется резервное питание. Отличие виртуального таймера от реального заключается в том, что

виртуальный таймер отсчитывает время работы ключа а не обсолютное время. Один такт состоявляет около 5 минут.

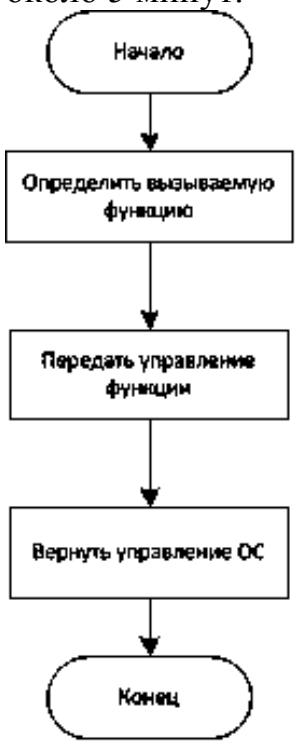


Рисунок 6 - Блок-схема алгоритма работы диспетчера класса обвертки

Программная защита памяти EEPROM заключается в следующем: в памяти хранятся 3 таблицы, которые имеют одинаковую структуру (рис. 7). Для выявления ошибки, данные которые там хранятся сравниваются с другими данными двух других столбцов и при выявлении ошибки эти данные возобновляются и копируются на строку ниже, то есть свободное место в этой таблице. Таблица текущих ключей содержит ID ключа (1 байт) и время функционирования данного ключа (3 байта), а таблица состояния памяти, содержит состояние текущего ключа.

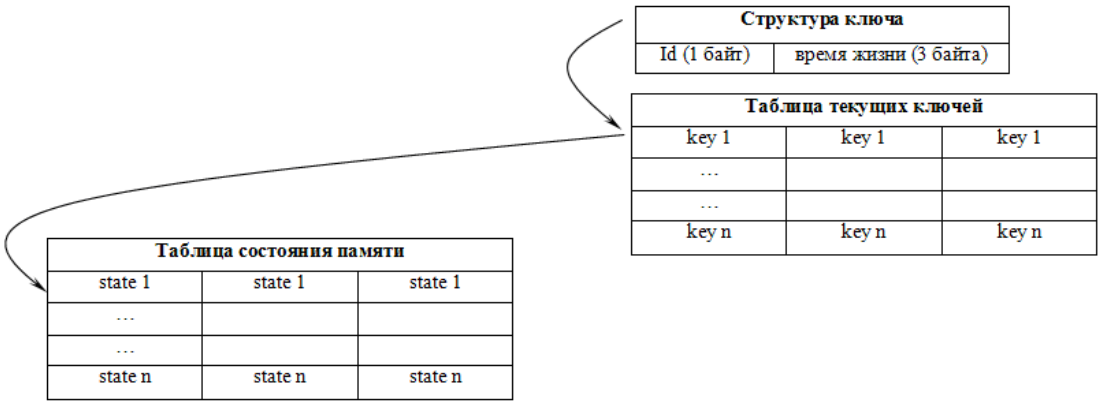


Рисунок 7 – Многоуровневая защита памяти EEPROM

**Выводы и направления дальнейших исследований.**

Данный аппаратный комплекс защиты был успешно реализован в рамках выполнения дипломной работы специальности специализированные компьютерные системы и показал свою жизнеспособность. В дальнейшем целесообразно более детально проработать механизм обновления прошивки ключа. Так перспективным видится использовать для этого специализированный web-ресурс. Также целесообразным является переход на более производительные контролеры с уже интегрированным USB интерфейсом.

**Библиографический список**

*1. Оценка эффективности систем защиты программного обеспечения. – Режим доступа к публикации: http://www.infocity.kiev.ua/hack/content/hack139.phtml*

*2. Скляров Д.В. Аппаратные ключи защиты / / Искусство защиты и взлома информации. - СПб.: БХВ-Петербург, 2004. - 288 с.*

*3. Аппаратная защита программного обеспечения. – Режим доступа к публикации: http://z-group.org.ua/art_apparatnaja_zashchita.html*

*4. Электронный ключ. – Режим доступа к публикации: http://ru.wikipedia.org/wiki*

*5. Удаленное обновление. – Режим доступа к публикации: http://guardant.com.ua/technology/tools/trusted-remote-update*

*6. AVR. Учебный Курс. Использование Bootloader'a. – Режим доступа к публикации: http://easyelectronics.ru/avr-uchebnyj-kurs-ispolzovanie-bootloadera.html*

*7. scmRTOS. – Режим доступа к публикации: http://real.kiev.ua/scmrtos*

***Рекомендована к печати д.т.н., проф. Мочалиным Е.В.***