УДК 004.087

ЭЛЕКТРОННЫЙ КЛЮЧ ЗАЩИТЫ С ФУНКЦИОНАЛЬНОСТЬЮ ТРОИЧНОГО СОПРОЦЕССОРА

DONGLE WITH FUNCTIONAL TERNARY COPROCESSOR

Габитов Р.Н., Габитова Я.А, Гиниятуллин В.М., Филиппов В.Н.

ФГБОУ ВПО «Уфимский государственный нефтяной технический университет», г. Уфа, Российская Федерация

R.N. Gabitov, Ya.A. Gabitova, V.M. Giniyatullin, V.N. Filippov

FSBEI HPE “Ufa State Petroleum Technological University”, Ufa, the Russian Federation

e-mail: VTIK-Ufa@mail.ru

Аннотация. Защита программного обеспечения (ПО) от нелицензионного пользования увеличивает прибыль разработчика. На сегодняшний день существует несколько подходов к решению этой проблемы. Подавляющее большинство создателей ПО используют различные программные модули, контролирующие доступ пользователей с помощью ключей активации, серийных номеров и т. д. Такая защита является дешёвым решением и не может претендовать на надёжность. Интернет изобилует программами, позволяющими нелегально сгенерировать ключ активации (генераторы ключей) или заблокировать запрос на серийный номер/ключ активации (патчи, крэки). Кроме того, не стоит пренебрегать тем фактом, что сам легальный пользователь может обнародовать свой серийный номер.

Эти очевидные недостатки привели к созданию аппаратной защиты программного обеспечения в виде электронного ключа. Известно, что первые электронные ключи (то есть аппаратные устройства для защиты ПО от нелегального копирования) появились в начале 80-ых годов, однако

первенство в идее и непосредственном создании устройства, по понятным причинам, установить очень сложно.

Однако и электронные ключи защиты имеют существенные недостатки, главный из которых - возможность изменения кода программы в целях обхода блока общения с аппаратным ключом. Для устранения этого недостатка предлагается встроить в аппаратный ключ защиты троичный сопроцессор, который будет выполнять некоторую существенную для программы функциональность, что значительно усложнит возможность изменения программного кода в целях отвязки взламываемой программы от электронного ключа, а также ограничит возможность его эмуляции.

Abstract. Protection of software (SW) from unlicensed use increases profit developer. Today, there are several approaches to solving this problem. The vast majority of the creators of the software use different software modules that control user access via activation keys, serial numbers and so on.. Such protection is the cheapest solution, and can not claim to reliability. The Internet abounds with programs that allow illegally generate an activation key (key generators) or block the request for serial number / activation key (patches, crack). Also, do not neglect the fact that the very legitimate user may disclose your serial number.

These obvious shortcomings have led to the creation of hardware security software as an electronic key. It is known that the first electronic keys (ie hardware devices to protect software from illegal copying) appeared in the early 80s, but in the idea of the primacy and direct a device, for obvious reasons, it is very difficult to establish.

However, electronic security keys have significant drawbacks, chief among them the ability to change the program code to bypass the block communication with the hardware key. To address this shortcoming is proposed to build in hardware dongle ternary coprocessor that will perform some essential functionality for the program, which greatly complicate the ability to change the

code in order to break open otvyazki program from an electronic key, and also limit the possibility of emulation.

Ключевые слова: защита программного обеспечения, электронный ключ защиты, троичный сопроцессор, троично сбалансированная система счисления, арифметика с плавающей запятой.

Key words: software protection, dongle, ternary coprocessor, balanced ternary number system, floating point arithmetic.

Большинство компьютерных программ распространяется по принципу владения оговоренным количеством рабочих копий (в простейшем случае – только одной). Естественно, общепринятый в международной практике термин «защита от копирования» достаточно условен, так как практически всегда можно переписать информацию, находящуюся на носителе, и сделать сколько угодно ее резервных копий. Другое дело, что для сохранения коммерческих и авторских прав разработчиков программа все равно должна выполняться только на одном компьютере. Таким образом, фактически защита от копирования для программного обеспечения – это невозможность выполнения программы на большем числе компьютеров, чем разрешено ее разработчиками и распространителями по данному договору. Следовательно, для сохранения прав необходимо наличие средств, дающих возможность защиты от несанкционированного выполнения – чтобы без санкции разработчика или фирмы-распространителя невозможно было получить работоспособный программный продукт.

Наиболее распространенным и надежным способом защиты от несанкционированного запуска стали программно-аппаратные ключи, подключаемые к COM-, LPT- или USB-портам. Почти все коробочные варианты серьезного коммерческого ПО используют программно-аппаратные комплексы защиты, более известные как аппаратные ключи

защиты. Такие способы защиты основаны на том, что в компьютер добавляется специальное физическое защитное устройство, к которому при запуске защищаемой программы обращается ее контролирующая часть, проверяя наличие ключа доступа и его параметров. Если ключ не найден (устройства обычно формируют еще и код ответа, который затем анализируется программой), то программа не запустится (или не будет разрешен доступ к данным).

Общий принцип работы компьютера в этом случае следующий. После запроса на выполнение защищаемой программы происходят ее загрузка в оперативную память и инициализация контролирующей части. На физическое устройство защиты, подсоединенное к компьютеру, посылается запрос. В ответ формируется код, посылаемый через микропроцессор в оперативную память для распознавания контролирующей частью программы. В зависимости от правильности кода ответа программа либо прерывается, либо выполняется.

В дальнейшем сфера применения таких ключей значительно расширилась. Сегодня этот ключ используется для идентификации владельца, для хранения его личной электронной подписи, конфиденциальной информации, а также как кредитная смарт-карта или электронные деньги.

В связи с распространением применения электронного ключа возникает необходимость в усилении его защиты. Предлагаемый в данной статье способ усиления учитывает все выявленные на данный момент недостатки аппаратных ключей защиты, а именно возможность эмуляции ключа и взлома кода защищаемой программы.

В качестве защиты от взлома кода предлагается вынести некоторую существенную для программы функциональность на электронный ключ. Конкретно предлагается вынести на аппаратный ключ защиты арифметику с плавающей запятой, которая на сегодняшний день в избытке присутствует практически в любом программном продукте. Вынос

существенной для программы функциональности на ключ аппаратной защиты исключит простой обход программных блоков, обращающихся с аппаратным ключом, так как в этом случае пострадает необходимая для правильной работы программы функциональность.

Как защиту от эмуляции электронного ключа защиты предлагается в аппаратном ключе реализовывать арифметику с плавающей запятой, основанной на троично сбалансированной системе счисления [1], что исключит возможность эмуляции внутренним двоичным сопроцессором компьютера [2].

Вследствие отсутствия троичной элементной базы [3] встает вопрос о реализации троичного сопроцессора на двоичной элементной базе [4], предлагается вариант реализации троичности на двоичной элементной базе [5]. То или иное представление троичности битами [6] используется в вычислительной технике как в аппаратном [7], так и программном обеспечении [8]. Это связанно с троичностью результатов операции сравнения, определения знака числа и т.п. [9] Например, результат сравнения двух строк троичен (больше, равно, меньше) и для возврата результата используется целый байт, который может принимать три значения: 1, 0 и -1. Таким образом, используется кодирование троичности восьмью битами, что избыточно [10]. Двух младших битов достаточно для определения результата, даже при этом остается одно неиспользуемое состояние.

Еще одним примером являются регистры переполнения и переноса [11]. По своей сути это двухбитовое кодирование трита (троичного разряда) [12] переполнения, необходимого для обозначения события и знака переполнения.

Для решения многих задач, содержащих в себе элементы троичности, приходится использовать избыточное представление тритов битами, что связано с повсеместным использованием двоичности.

В литературе [13] двухразрядное представление также называют двухпроводным кодированием, двухбитовым кодированием или 2BBCT (2 bitbinarycodedternary).

Существует 24 возможных двухразрядных представлений, из них 2 представляют интерес:

- $-1_3 = 10_2$, $0_3 = 00_2$, $1_3 = 01_2$;
- $-1_3 = 11_2$, $0_3 = 00_2$, $1_3 = 01_2$.

Приведенные варианты кодирования различаются только представлением -1: в первом варианте -1 представляется как 10, а во втором – как 11. Таким образом, будем обозначать первый вариант как «2BBCT ($-1_3 = 10_2$)», а второй вариант кодирования как «2BBCT ($-1_3 = 11_2$)».

В первом варианте кодирования 2BBCT ($-1_3 = 10_2$) старший бит отвечает за обозначение отрицательной составляющей трита, а младший бит – за положительную. Для проверки трита на равенство –1 необходимо и достаточно узнать состояние старшего бита, а на равенство 1 – младшего бита. Для проверки на равенство нулю необходимо знать состояние всех битов.

Во втором варианте кодирования 2BBCT ($-1_3 = 11_2$) старший бит отвечает за обозначение отрицательной составляющей трита, а младший – за нулевую составляющую. Для проверки на -1 необходимо и достаточно узнать состояние старшего бита, на 0 – младшего бита, а на 1 – обоих битов. Данный вариант кодирования интересен тем, что значения троичных разрядов представляются двухбитовыми двоичными целыми числами и становится применимой двоичная целочисленная арифметика.

Таким образом, первый вариант симметричен относительно нуля по количеству двоичных проверок для определения состояния трита, а второй нет. В варианте 2BBCT ($-1_3 = 11_2$) притесняется положительная составляющая трита. Так как предполагается, что положительных результатов будет примерно столько же, сколько отрицательных, а

нулевых результатов значительно меньше, то в этом случае предпочтительным является первый вариант кодировки 2ВВСТ ($-1_3 = 10_2$).

Двухразрядное представление в любой кодировке избыточно: 1 из 4-х состояний остается не использованным. Существует возможность использования этого состояния для обозначения двумя представлениями одного трита, например $-1_3 = 10_2$, $0_3 = 00_2$, $1_3 = 01_2$, $0_3 = 11_2$. Но это существенно усложняет понимание и нарушает принцип взаимно-однозначного соответствия тритов и их битовых представлений.

В различных источниках трехразрядное представление также называют трехпроводным кодированием, трехбитовым кодированием или 3BBCT (3bitbinarycodedternary).

Трехразрядное представление избыточно: 5 из 8-ми состояний остаются не использованными. Трехразрядное представление требует больше объема двоичной памяти в 1,5 раза по сравнению с двухразрядным представлением.

Таким образом, существует возможность реализации троичного сопроцессора на двоичной элементной базе с использованиемкодировки 2ВВСТ ($-1_3 = 10_2$). Для такой реализации троичности разработан и описан метод реализации троичных функций на двоичной элементной базе.

Ключ аппаратной защиты с функциональностью троичного сопроцессора кроме предлагаемых дополнений должен иметь функциональность существующих аналогов, а именно:

- проверка наличия подключения ключа;
- считывание с ключа необходимых программе данных в качестве параметра запуска (используется, в основном, только при поиске подходящего ключа, но не для защиты);
- запрос на расшифрование данных или исполняемого кода, необходимых для работы программы, зашифрованных при защите программы (позволяет осуществлять "сравнение с эталоном"; в случае

шифрования кода выполнение нерасшифрованного кода приводит к ошибке);

- запрос на расшифрование данных, зашифрованных ранее самой программой;

- проверка целостности исполняемого кода путём сравнения его текущей контрольной суммы с оригинальной контрольной суммой, считываемой с ключа;

- запрос, к встроенным в ключ, часам реального времени (при их наличии; может осуществляться автоматически при ограничении времени работы аппаратных алгоритмов ключа по его внутреннему таймеру) и т.д.

В качестве основного недостатка предлагаемого электронного ключа защиты можно выделить увеличение вычислительной нагрузки на центральный процессор, а также увеличение времени выполнения программы, что связанно с обращениями к электронному ключу защиты и ожиданию ответа. Однако программные продукты, использующие электронный ключ, часто связанны с системами, где вычислительная нагрузка достаточно низкая.

Выводы

В качестве заключения можно сказать, что предложенный ключ аппаратной защиты с функциональностью троичного сопроцессора не является универсальным средством защиты программного обеспечения, как интеллектуальной собственности, а имеет некоторые ограничения по применению, в частности, наличие арифметики с плавающей запятой, не очень высокая вычислительная нагрузка. При этом предоставляет разработчику возможность существенно повысить защиту своей интеллектуальной собственности от несанкционированного использования.

Список используемых источников

1 Брусенцов Н.П. Использование троичного кода и трехзначной логики в цифровых машинах: науч. отчет № 24-ВТ (378). М.: МГУ им. М.В. Ломоносова, 1969. С. 3–24.

2 Лукасевич Я. О трехзначной логике: автореферат. Львов, 1920. С. 170–171.

3 Википедия: свободная электронная энциклопедия: на русском языке: Системы счисления [Электронный ресурс] //URL: http://ru.wikipedia.org/wiki (дата обращения: 08.06.2013).

4 Яблонский С.В. Введение в дискретную математику: учеб. пособие для вузов; 2-е изд. перераб. и доп. М.: Наука, 1986. С. 43–73.

5 Википедия: свободная электронная энциклопедия: на русском языке: Троичная система счисления [Электронный ресурс] //URL: http://ru.wikipedia.org/wiki (дата обращения: 08.06.2013).

6 Кушнеров А. Троичная цифровая техника. Ретроспектива и современность. Беэр-Шева, Израиль: ун-т им. Бен-Гуриона, 2005. 14 с.

7 Брусенцов Н.П. Владимирова Ю.С., Рамиль Альварес Х. Троичный логико-алгебраический и арифметический процессор // Программные системы и инструменты. 2005. (Изд. отдел ВМК МГУ). № 6. С. 184–187.

8 Брусенцов Н.П. Из истории создания троичных цифровых машин в МГУ. // Историко-математические исследования. М.: Янус-К, 2005. С. 28–53. - (Сер. 2; вып. 10 (45)

9 Грушвицкий Р.И. Мурсаев А.Х., Угрюмов Е.П. Проектирование систем на микросхемах программируемой логики. СПб.: БХВ - Петербург, 2002. 608 с.

10 Ившин П. Леготин С., Мурашёв В. Базовые троичные логические элементы. Снижение энергопотребления // Электроника: Наука, технология, бизнес. 2010. № 4. С. 56–61.

11 Маслов С.П. Об одной возможности реализации троичных цифровых устройств. // Программные системы и инструменты: темат. сб./ фак-та ВМиК МГУ. 2011. № 12. С. 222–227.

12 Спиридонов А. Троичность, как альтернатива для компьютерных вычислений. Самара: Самарский гос. аэрокосмический ун-т им. акад. С.П. Королева, 2007. 33 с.

13 Поспелов Д.А. Логические методы анализа и синтеза схем. 3-е изд. перераб. и доп. М.: Энергия, 1974. 368 с.

References

1 Brusencov N.P. Ispol'zovanie troichnogo koda i trehznachnoj logiki v cifrovyh mashinah. //Nauchnyj otchet №24-VT (378). M.: MGU im. M.V. Lomonosova, 1969. S. 3–24. [in Russian].

2 Lukasevich Ja. O trehznachnoj logike. //Avtoreferat, prochitannyj na 207 zasedanii PTF vo L'vove. L'vov: 1920. S. 170–171. [in Russian].

3 Vikipedija: svobodnaja jelektronnaja jenciklopedija: na russkom jazyke: Sistemy schislenija [Jelektronnyj resurs] //URL: http://ru.wikipedia.org/wiki (data obrashhenija: 08.06.2013). [in Russian].

4 Jablonskij S.V. Vvedenie v diskretnuju matematiku. Ucheb. posobie dlja vuzov /S.V.Jablonskij.– 2-e izd. pererab. i dop.– M.: Nauka, 1986. S. 43–73. [in Russian].

5 Vikipedija: svobodnaja jelektronnaja jenciklopedija: na russkom jazyke: Troichnaja sistema schislenija [Jelektronnyj resurs] // URL: http://ru.wikipedia.org/wiki (data obrashhenija: 08.06.2013). [in Russian].

6 Kushnerov A. Troichnaja cifrovaja tehnika. Retrospektiva i sovremennost'. Bejer-Sheva, Izrail': Universitet im. Ben-Guriona, 2005. 14 s. [in Russian].

7 Brusencov N.P. Vladimirova Ju.S., Ramil' Al'vares H. Troichnyj logiko-algebraicheskij i arifmeticheskij processor // Programmnye sistemy i instrumenty № 6. M.: Izdatel'skij otdel VMK MGU, 2005. S. 184–187. [in Russian].

8 Brusencov N.P. Iz istorii sozdanija troichnyh cifrovyh mashin v MGU. // Istoriko-matematicheskie issledovanija. Vtoraja serija. Vyp. 10 (45). M.: Janus-K, 2005. S. 28–53. [in Russian].

9 Grushvickij R.I. Mursaev, E.P. Ye.P. Ugryumoye. Proektirovanie sistem na mikroshemah programmiruemoj logiki. SPb.: BHV -Peterburg, 2002. 608 s.

10 Ivshin P. Legotin S., Murashjov V. Bazovye troichnye logicheskie jelementy. Snizhenie jenergopotreblenija // Jelektronika: Nauka, tehnologija, biznes, № 4. 2010. S. 56–61. [in Russian].

11 Maslov S.P. Ob odnoj vozmozhnosti realizacii troichnyh cifrovyh ustrojstv. // Programmnye sistemy i instrumenty. Tematicheskij sbornik №12. M.: Izd-vo fak-ta VMiK MGU, 2011. S. 222–227. [in Russian].

12 Spiridonov A. Troichnost', kak al'ternativa dlja komp'juternyh vychislenij / Samara: Samarskij gos. ajerokosmicheskij un-t imeni akademika S.P. Koroleva, 2007. 33 s. [in Russian].

13 Pospelov D.A. Logicheskie metody analiza i sinteza shem. 3-e izd. pererab. i dop. M.: Jenergija, 1974. 368 s. [in Russian].

Сведения об авторах

About the authors

Габитов Р.Н., аспирант, преподаватель кафедры «Вычислительная техника и инженерная кибернетика», ФГБОУ ВПО УГНТУ, г. Уфа, Российская Федерация

R.N. Gabitov, Post-graduate Student, Lecturer of the Chair “Computer Science and Engineering Cybernetics”, FSBEI HPE USPTU, Ufa, the Russian Federation

Габитова Я.А., магистрант группы МПО01-13-01 кафедры «Вычислительная техника и инженерная кибернетика», ФГБОУ ВПО УГНТУ, г. Уфа, Российская Федерация

Ya.A. Gabitova, Master Student of MPO01-13-01 Group of the Chair “Computer Science and Engineering Cybernetics”, FSBEI HPE USPTU, Ufa, the Russian Federation

Гиниятуллин В.М., канд. техн. наук, доцент кафедры «Вычислительная техника и инженерная кибернетика», ФГБОУ ВПО УГНТУ, г. Уфа, Российская Федерация

V.M. Giniyatullin, Candidate of Engineering Sciences, Associate Professor of the Chair “Computer Science and Engineering Cybernetics”, FSBEI HPE USPTU, Ufa, the Russian Federation

Филиппов В.Н., канд. техн. наук, доцент кафедры «Вычислительная техника и инженерная кибернетика», ФГБОУ ВПО УГНТУ, г. Уфа, Российская Федерация

V.N. Filippov, Candidate of Engineering Sciences, Associate Professor of the Chair “Computer Science and Engineering Cybernetics”, FSBEI HPE USPTU, Ufa, the Russian Federation

e-mail: VTIK-Ufa@mail.ru