УДК 004.056.53

К. В. Защелкин, канд. техн. наук,
Е. Н. Иванова

РАЗВИТИЕ МЕТОДА СТЕГАНОГРАФИЧЕСКОГО СКРЫТИЯ ДАННЫХ В LUT-ОРИЕНТИРОВАННЫХ АППАРАТНЫХ КОНТЕЙНЕРАХ

Аннотация. *Предложено развитие метода стеганографического скрытия данных в аппаратных контейнерах с LUT-ориентированной архитектурой. Основой метода внедрения данных в LUT-контейнер является процедура, состоящая в инвертировании значений текущего обрабатываемого блока LUT и выполнении распространения инверсии на входы всех блоков LUT, подключенных к выходу текущего блока. Описаны принципы выполнения распространения инверсии на одиночные входы блока LUT. Приведено утверждение, позволяющее выполнять групповое распространение инверсии на несколько входов блока LUT. Показаны подходы к аппаратно-программной реализации предложенного метода.*

Ключевые слова: *стеганография, цифровые водяные знаки, защита информации, внедрение данных, аппаратный стего-контейнер, LUT-ориентированная архитектура, секретная передача данных, защита от несанкционированного использования*

K. V. Zashcholkin, PhD.,
E. N. Ivanova

STEGANOGRAPHY DATA HIDING IN LUT-ORIENTED HARDWARE CONTAINERS METHOD DEVELOPMENT

Abstract. *There is method development proposed for implementing the method of steganography data hiding in hardware containers carrying LUT-oriented architecture. The method based data embedding into LUT-container is the procedure including current processing LUT-unit value invert conversion and invert distribution for all the LUT-units inputs connected to the current unit outputs. The principals of invert distribution for LUT-unit single inputs have been described. The statement is powered that the group invert distribution is possible for several LUT-unit inputs. We can estimate approaches to hardware and software implementation of the method offered.*

Keywords: *steganography, digital watermarks, data protection, data embedding, hardware stego-container, LUT-oriented architecture, secret data transfer, protection from unauthorized use*

К. В. Защолкін, канд. техн. наук,
О. М. Іванова

РОЗВИТОК МЕТОДУ СТЕГАНОГРАФІЧНОГО ПРИХОВУВАННЯ ДАНИХ В LUT-ОРІЄНТОВАНИХ АПАРАТНИХ КОНТЕЙНЕРАХ

Анотація. *Запропоновано розвиток методу стеганографічного приховування даних в апаратних контейнерах з LUT-орієнтованою архітектурою. Основою методу вбудовування даних в LUT-контейнер є процедура, яка полягає в інвертуванні значень поточного оброблюваного блоку LUT та виконанні поширення інверсії на входи всіх блоків LUT, підключених до виходу поточного блоку. Описано принципи виконання поширення інверсії на одиночні входи блоку LUT. Приведено твердження, яке дозволяє виконувати групове поширення інверсії на декілька входів блоку LUT. Показані підходи до апаратно-програмної реалізації запропонованого методу.*

Ключові слова: *стеганографія, цифрові водяні знаки, захист інформації, вбудовування даних, апаратний стего-контейнер, LUT-орієнтована архітектура, секретна передача даних, захист від несанкціонованого використання*

Введение. Проблема защиты цифровой информации от несанкционированного доступа, фальсификации, нарушения лицензионного режима ее использования и прочих угроз сейчас является чрезвычайно актуальной. Одним из эффективных направлений решения этой проблемы выступает *стеганография*. В ее основе лежит скрытие факта существования всей или части защищаемой информации [1]. Возможные механизмы стеганографической защиты данных основаны на: организации скрытых каналов передачи данных внутри открытых каналов; скрытом хранении данных на потенциально незащищенных от несанкционированного доступа носителях информации; встраивании скрытых меток (*цифровых водяных знаков*) в различные информационные объекты с целью контроля их использования [2].

Традиционные стеганографические методы, чаще всего, основаны на встраивании секретной информации в *пассивные* (выполняющие только функцию хранения данных) мультимедийные контейнеры: графические, звуковые, видео файлы [3]. Исследования

подходов к использованию немультимедийных *активных* стего-контейнеров – информационных объектов, выполняющих некоторую вычислительную или управляющую функцию, находятся сейчас на начальной стадии. В рамках таких подходов, например, предлагается использовать в качестве стего-контейнеров для внедрения цифровых водяных знаков или для задач скрытого хранения и пересылки защищенной информации, исполняемые файлы [4–6] или исходные коды программ [7, 8] для вычислительной машины.

В работе [9] был предложен метод внедрения данных в стего-контейнеры с LUT-ориентированной архитектурой (LUT – Look Up Table – таблица поиска), которые отличаются от традиционных контейнеров тем, что является *активными* информационными объектами, состоящими из *неавтономных* элементарных единиц, данные в которых представлены *точно* [10]. Основное содержание указанного метода в работе [9] излагается в самом общем виде как совокупность специальных положений, определяющих последовательность и условия реализации процесса скрытия данных в LUT-контейнере. **Цель данной работы** состоит в развитии указанного метода путем его детализации до уровня элементарных вычислительных операций, пригодной к непосредственной аппаратно-программной реализации.

Метод внедрения данных в LUT-контейнер. *Исходные данные метода*:

1) секретная двоичная последовательность $M = (m_1, m_2, \ldots, m_k)$, подлежащая встраиванию в контейнер;

2) LUT-контейнер *LC*, реализующий некоторую вычислительную или управляющую функцию;

3) ключ *key* = (*set*, *order*) для встраивания и извлечения секретной информации, где *set* – номер (адрес) разряда LUT, в который выполняется внедрение бита секретной последовательности, *order* – информация, определяющая порядок обхода блоков LUT в контейнере для выполнения встраивания или извлечения.

Результат применения метода – LUT-контейнер *LC**, в который внедрена последовательность *M*. При этом функционирование контейнера *LC**, выражающееся в выполнении его целевой функции, не должно отличаться от функционирования контейнера *LC*.

Последовательность действий метода внедрения данных в контейнер:

Этап 1. Определение возможности встраивания последовательности *M* в контейнер *LC* на основании 3-го положения метода [9]: встраивание возможно, если схема, содержащаяся в контейнере, имеет более одного уровня.

Этап 2. Оценка возможности встраивания последовательности *M* целиком в контейнер *LC*, исходя из количества боков LUT, которые можно использовать для встраивания.

Этап 3. Формирование списков блоков LUT, предназначенных для проверки соблюдения ограничений 4-го положения метода [9]:

– формирование списка *ExternalList*, в который заносятся идентификаторы блоков LUT, непосредственно подключенные к выходам схемы;

– формирование списка *BlockList*, в который в ходе движения по контейнеру помещаются идентификаторы заблокированных для встраивания блоков LUT.

Этап 4. В соответствии с заданным порядком обхода блоков LUT контейнера, с учетом ограничений, определенных 4-м положением метода [9], каждый разряд секретной последовательности m_j на основании следующего правила встраивается в индивидуальный блок LUT_i:

$$\begin{gathered}\forall\, LUT_i \mid (LUT_i \notin ExternalList \;\&\\ \&\;\; LUT_i \notin BlockList \;\&\\ \&\;\; OutList(LUT_i) \cap BlockList = \varnothing) \qquad (1)\\ \text{if } LUT_i(set) \neq m_j \text{ then } Embed(LUT_i, m_j);\\ LUT_i \rightarrow BlockList.\end{gathered}$$

где: $OutList(LUT_i)$ – список блоков LUT, подключенных своими входами к выходу блока LUT_i;

set – номер разряда LUT, в который выполняется внедрение секретного бита m_j;

Embed – процедура встревания разряда m_j в блок LUT_i по адресу *set*;

"$\rightarrow$" – операция включения идентификатора блока LUT в список.

Внешнее условие правила (1) определяет возможность использования блока LUT для встраивания очередного разряда секретной последовательности. Данное составное усло-

вие определяет то, что блок LUT_i, предназначенный для встраивания информации:

а) не должен содержаться в списке блоков, непосредственно подключенных к выходам схемы;

б) не должен содержаться в списке *BlockList*, блоков, заблокированных для дальнейшего встраивания;

в) его выход не должен быть подключен на вход блоков из списка *BlockList*.

Внутреннее условие правила (1) означает, что *Embed* – процедура встревания разряда m_j в блок LUT_i по адресу *set*, выполняется только тогда, когда значение, содержащееся в данном блоке по указанному адресу, не совпадает со значением разряда, который необходимо встроить. В противном случае отсутствует необходимость выполнения встраивания в данный блок LUT_i.

Блоки LUT, не удовлетворяющие внешнему или внутреннему условию правила (1), игнорируются в процессе обхода контейнера. Встраивание информации в них не производится. Блоки LUT, удовлетворяющие внешнему условию, но не удовлетворяющие внутреннему условию правила (1) хранят элементы секретной последовательности, но встраивание в них не производится по причине совпадения исходного их значения со значением, подлежащим встраиванию. В ходе применения правила (1), независимо от того, каким образом в очередной блок LUT был помещен разряд секретной последовательности (при помощи процедуры *Embed* или этот разряд находился в контейнере изначально, и применение процедуры *Embed* не требовалось), идентификатор данного блока LUT заносится в список заблокированных для дальнейшего встраивания блоков *BlockList*.

Процедура встраивания Embed

Процедура *Embed*, выполняющая встраивание разряда m_j в блок LUT_i по адресу *set* состоит из двух действий:

1) инвертирование значений блока LUT_i (каждое из значений, хранящихся в блоке, меняется на противоположное);

2) выполнение *процедуры распространение инверсии* на входы всех блоков LUT, содержащихся в списке *OutList*(LUT_i). Распространение инверсии состоит в инвертировании входа блока LUT. Такое инвертирование сводится к перестановке значений, хранящихся в блоке LUT, и определяется следующим образом.

Введем обозначения. Пусть блок LUT_i имеет набор из n входов $In_i = (in_i^{n-1}, in_i^{n-2}, \ldots, in_i^1, in_i^0)$. Каждый из входов in_i^k при $k = 0 \ldots n\text{-}1$ задает один из разрядов адреса блока LUT, и соответственно имеет вес 2^k. Количество значений, хранящихся в данном блоке LUT равно $N = 2^n$. Обращение (на чтение или на запись) к значению, хранящееся в блоке LUT_i по адресу s будем далее обозначать как $LUT_i(s)$, где s может задаваться в виде двоичного числа, количество разрядов которого, совпадает с количеством входов блока LUT, или в виде десятичного эквивалента этого числа.

Следующим образом определим процедуру перестановки значений блока LUT:

$$Permutation(LUT_i, w): \quad \forall s \;\; LUT_i(s_w^0) \leftrightarrow LUT_i(s_w^1) \tag{2}$$

где: s – любой допустимый адрес для данного блока LUT_i;

s_w^0 и s_w^1 – двоичные наборы, которые различаются только значением в разряде, имеющем вес w, а в остальных разрядах совпадают. Наборы s_w^0 и s_w^1 содержат в разряде с весом w ноль и единицу соответственно.

"$\leftrightarrow$" – операция обмена значений блока LUT (значение, указанное в качестве левого операнда, помещается на место правого операнда и наоборот).

Таким образом, принцип перестановки в ходе распространения инверсии зависит от двоичного веса входа, на который распространяется инверсия. Например, для распространения инверсии на вход блока LUT, имеющий вес $w = 1$, в соответствии с выражением (2), выполняется взаимный обмен значений, расположенных по адресам, отличающимся только в младшем разряде (с весом 1). Это приводит к обмену значениями, находящимися по ближайшим четным и нечетным адресам: $LUT(0) \leftrightarrow LUT(1)$, $LUT(2) \leftrightarrow LUT(3)$, $LUT(4) \leftrightarrow LUT(5)$ и т.д. Аналогичным образом, распространение инверсии на вход блока LUT, имеющий вес $w = 2$, приводит к взаимному обмену значениями, расположенными по

адресам, отличающимся в разряде с весом 2: $LUT(0) \leftrightarrow LUT(2)$, $LUT(1) \leftrightarrow LUT(3)$, $LUT(4) \leftrightarrow LUT(6)$, $LUT(5) \leftrightarrow LUT(7)$ и т.д.

На рис. 1 показаны правила распространения инверсии для четырёх-входового блока LUT. Столбцы *A*, *B*, *C*, *D* содержат разряды адреса, подаваемые на соответствующие входы. Веса входов увеличиваются слева направо: *A* – вес 1, *B* – вес 2, *C* – вес 4, *D* – вес 8. В столбце *out* находятся значения, содержащиеся в блоке LUT. На рис. 1 (а) показаны первоначальные значения блока LUT, на рис. 1 (б) – (д) показаны результаты распространения инверсии на входы этого блока *D*, *C*, *B*, *A* соответственно.

D^8	C^4	B^2	A^1	*out*
0	0	0	0	h_0
0	0	0	1	h_1
0	0	1	0	h_2
0	0	1	1	h_3
0	1	0	0	h_4
0	1	0	1	h_5
0	1	1	0	h_6
0	1	1	1	h_7
1	0	0	0	h_8
1	0	0	1	h_9
1	0	1	0	h_{10}
1	0	1	1	h_{11}
1	1	0	0	h_{12}
1	1	0	1	h_{13}
1	1	1	0	h_{14}
1	1	1	1	h_{15}

а

D^8	C^4	B^2	A^1	*out*
0	0	0	0	h_1
0	0	0	1	h_0
0	0	1	0	h_3
0	0	1	1	h_2
0	1	0	0	h_5
0	1	0	1	h_4
0	1	1	0	h_7
0	1	1	1	h_6
1	0	0	0	h_9
1	0	0	1	h_8
1	0	1	0	h_{11}
1	0	1	1	h_{10}
1	1	0	0	h_{13}
1	1	0	1	h_{12}
1	1	1	0	h_{15}
1	1	1	1	h_{14}

б

D^8	C^4	B^2	A^1	*out*
0	0	0	0	h_2
0	0	0	1	h_3
0	0	1	0	h_0
0	0	1	1	h_1
0	1	0	0	h_6
0	1	0	1	h_7
0	1	1	0	h_4
0	1	1	1	h_5
1	0	0	0	h_{10}
1	0	0	1	h_{11}
1	0	1	0	h_8
1	0	1	1	h_9
1	1	0	0	h_{14}
1	1	0	1	h_{15}
1	1	1	0	h_{12}
1	1	1	1	h_{13}

в

D^8	C^4	B^2	A^1	*out*
0	0	0	0	h_4
0	0	0	1	h_5
0	0	1	0	h_6
0	0	1	1	h_7
0	1	0	0	h_0
0	1	0	1	h_1
0	1	1	0	h_2
0	1	1	1	h_3
1	0	0	0	h_{12}
1	0	0	1	h_{13}
1	0	1	0	h_{14}
1	0	1	1	h_{15}
1	1	0	0	h_8
1	1	0	1	h_9
1	1	1	0	h_{10}
1	1	1	1	h_{11}

г

D^8	C^4	B^2	A^1	*out*
0	0	0	0	h_8
0	0	0	1	h_9
0	0	1	0	h_{10}
0	0	1	1	h_{11}
0	1	0	0	h_{12}
0	1	0	1	h_{13}
0	1	1	0	h_{14}
0	1	1	1	h_{15}
1	0	0	0	h_0
1	0	0	1	h_1
1	0	1	0	h_2
1	0	1	1	h_3
1	1	0	0	h_4
1	1	0	1	h_5
1	1	1	0	h_6
1	1	1	1	h_7

д

Рис. 1. Правила распространения инверсии на входы четырёх-входового блока LUT
а) исходные значения; б) распространение на вход *A*; в) распространение на вход *B*;
г) распространение на вход *C*; д) распространение на вход *D*

Блоки LUT, используемые в современных средствах цифровой техники имеют небольшое количество входов (от 3 до 6). В силу этого нет необходимости выполнять поиск наборов, на которых значения LUT в соответствии с выражением (2) подлежат обмену. Правила обмена для таких блоков LUT могут быть жестко определены в реализации алгоритма распространения инверсии.

В общем случае процедура распространения инверсии может иметь место не по одному входу блока LUT, а по нескольким его входам одновременно (*групповое распространение инверсии*). Выполнение такого группового распространения основано на следующем утверждении.

Утверждение. Результат группового распространения инверсии на входы блока LUT не зависит от порядка выполнения процедуры распространения инверсии на отдельные его входы.

Доказательство этого утверждения состоит в следующем. Распространение инверсии основано на перестановке, по правилу (2), значений, хранящихся в блоке LUT. Каждая такая перестановка приводит к взаимному изменению только одного разряда адреса для переставляемых значений. Разряды адреса в ходе выполнения перестановок не зависят друг от друга. В силу этого порядок изменения разрядов на основании выражения (2) при нескольких последовательных изменениях с разными значениями *w* не влияет на результат данных изменений. Следовательно, и порядок выполнения распространения инверсии на отдельные входы блока LUT не влияет на получаемый результат.

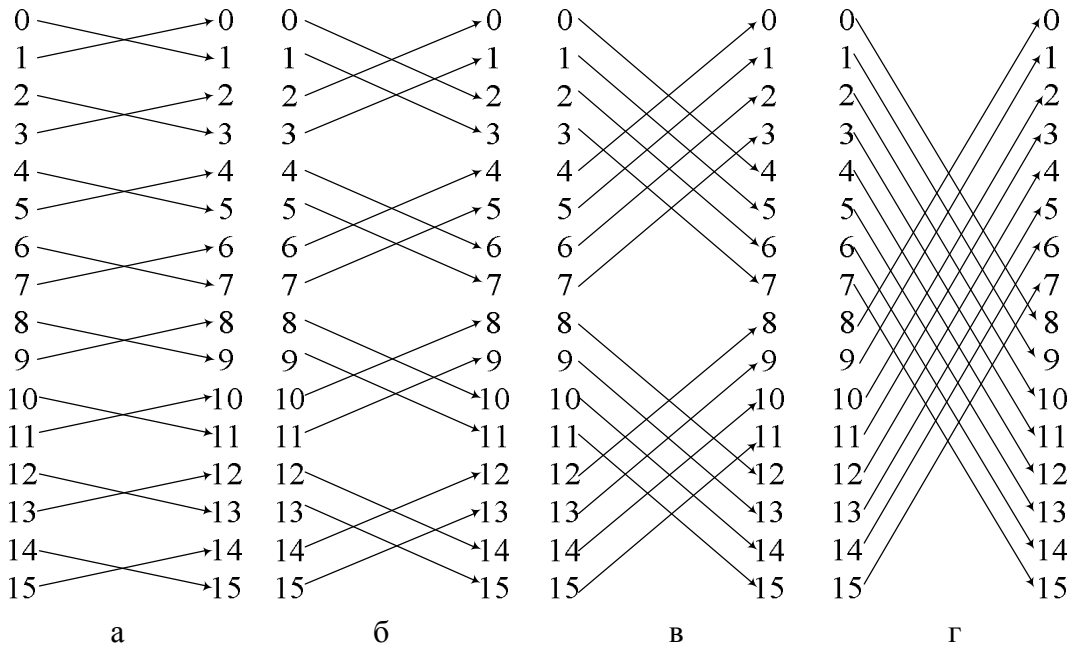

Рис. 2. Графическая интерпретация перестановок, возникающих при распространении инверсии на входы четырёх-входового блока LUT
а) распространение на вход с весом 1; б) распространение на вход с весом 2;
в) распространение на вход с весом 4; г) распространение на вход с весом 8

$\varphi_1 =$

0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0001	0000	0011	0010	0101	0100	0111	0110	1001	1000	1011	1010	1101	1100	1111	1110

$\varphi_2 =$

0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0010	0011	0000	0001	0110	0111	0100	0101	1010	1011	1000	1001	1110	1111	1100	1101

$\varphi_4 =$

0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0100	0101	0110	0111	0000	0001	0010	0011	1100	1101	1110	1111	1000	1001	1010	1011

$\varphi_8 =$

0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111

Рис. 3. Матрицы φ_i перестановок значений в ходе распространения инверсии на входы 4-х входового блока LUT (индекс i – соответствует весу входа)

На рис. 2 графически показаны перестановки значений в ходе распространения инверсии на отдельные входы четырёх-входового блока LUT. На рис. 3 приведены традиционные для теории перестановок их обозначения в виде двухстрочных матриц. Адреса значений, хранящихся в блоке LUT, показаны в двоичном виде. Адреса до перестановки представлены в верхней строке каждой матрицы, после перестановки – в нижней строке. Данные матрицы показывают механизм изменения адресов в ходе распространения инверсии, основанный на независимости разрядов адресов друг от друга и приводящий к независимости результата от порядка применения подстановок.

Таким образом, на основании приведенного утверждения, в случае, если процедура распространения инверсии осуществляется от нескольких блоков LUT на разные входы одного блока LUT, то необходимо по отдельности выполнить распространение инверсии на каждый вход, причем порядок обработки входов не имеет значения.

Обобщенная блок-схема предлагаемого метода в части встраивания двоичной последовательности в LUT-контейнер, полученная на основании приведенных выше положений, приведена на рис. 4.

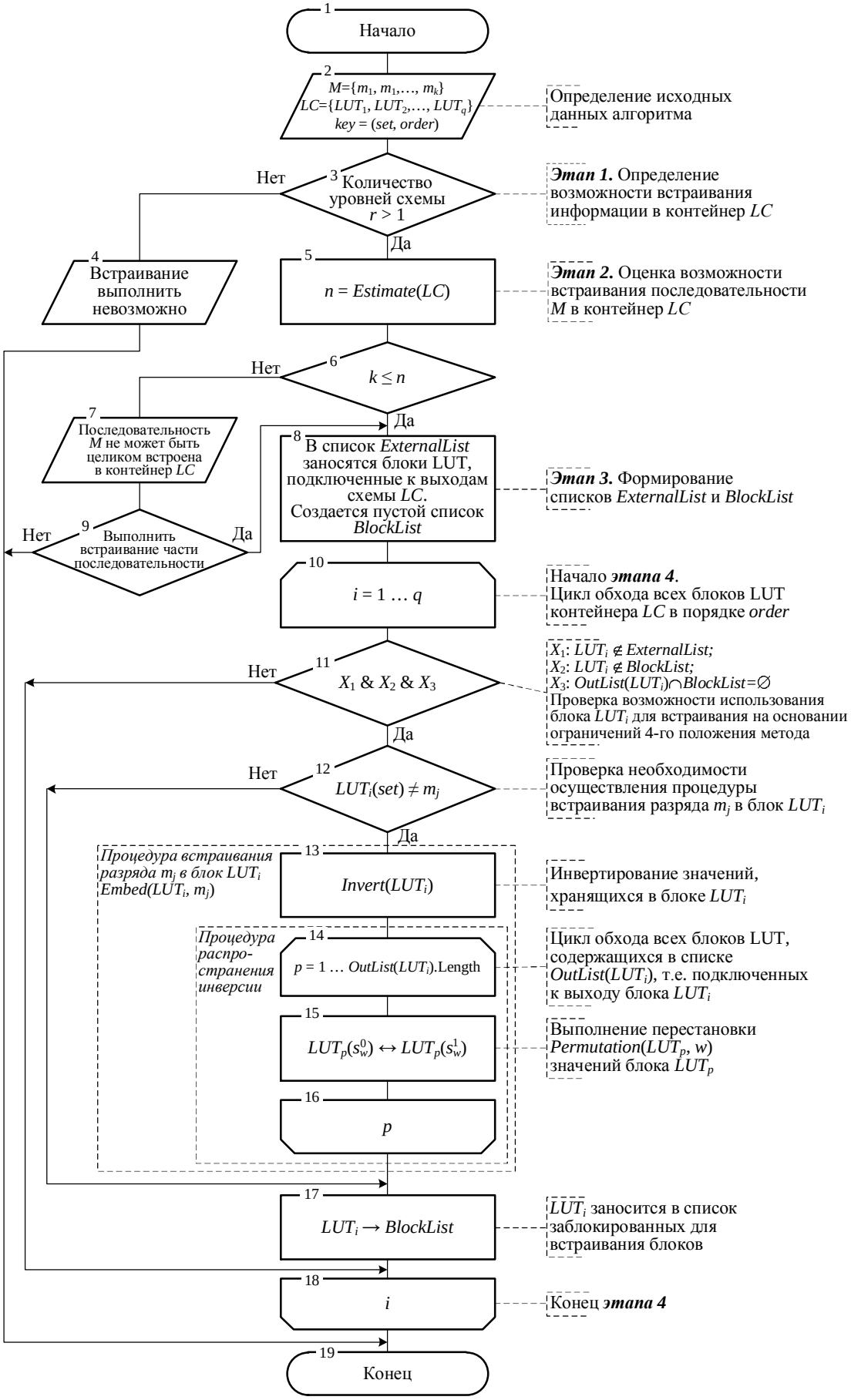

Рис. 4. Блок-схема метода встраивания данных в LUT-контейнер

Метод извлечения данных из LUT-контейнера. Далее рассматривается метод извлечения из LUT-контейнера, секретной двоичной последовательности, встроенной в него в соответствии с описанным ранее методом.

Исходные данные метода:

1) LUT-контейнер *LC**, содержащий встроенную в него секретную двоичную последовательностью $M = (m_1, m_2, \dots, m_k)$;

2) ключ извлечения встроенной информации *key* = (*set, order*).

Результат применения метода – извлеченная из контейнера двоичная последовательность $M^* = (m_1^*, m_2^*, \dots, m_k^*)$, которая при успешном извлечении должна совпадать с последовательностью $M = (m_1, m_2, \dots, m_k)$.

Последовательность действий, необходимая для извлечения встроенных данных представлена на рис. 5 в виде блок-схемы. Несмотря на то, что информация о порядке обхода контейнера является составной частью стего-ключа, не все блоки LUT, лежащие на стего-пути, на этапе встраивания могли быть использованы для внедрения двоичной последовательности (ограничения 4-го положения метода [9]). В силу чего (аналогично тому, как это было реализовано в методе встраивания) при излечении необходимо использовать два служебных списка *ExternalList* и *BlockList*. Анализ этих списков позволяет определить мог ли блок LUT, лежащий на стего-пути, быть использован для внедрения бита секретной последовательности на этапе встраивания. Из блоков LUT, в отношении которых принято решение о том, что они могли быть использованы при встраивании, производится считывание значений по адресу *set*, являющемуся элементом стего-ключа. Считанные таким образом значения образуют извлеченную последовательность *M**.

Выводы. В работе предложены детализированные методы внедрения и извлечения данных, в основу которых положен обобщенный метод стеганографического скрытия информации в аппаратных контейнерах с LUT-ориентированной архитектурой.

Метод встраивания позволяет внедрять двоичную информацию в LUT-контейнер, подвергая элементарные единицы контейнера локальным изменениям, но, не меняя при этом глобальную функциональность контейнера. Метод извлечения основан на возможности принятия решения о нахождении в очередном блоке LUT, лежащем на стего-пути, элемента встроенной секретной последовательности.

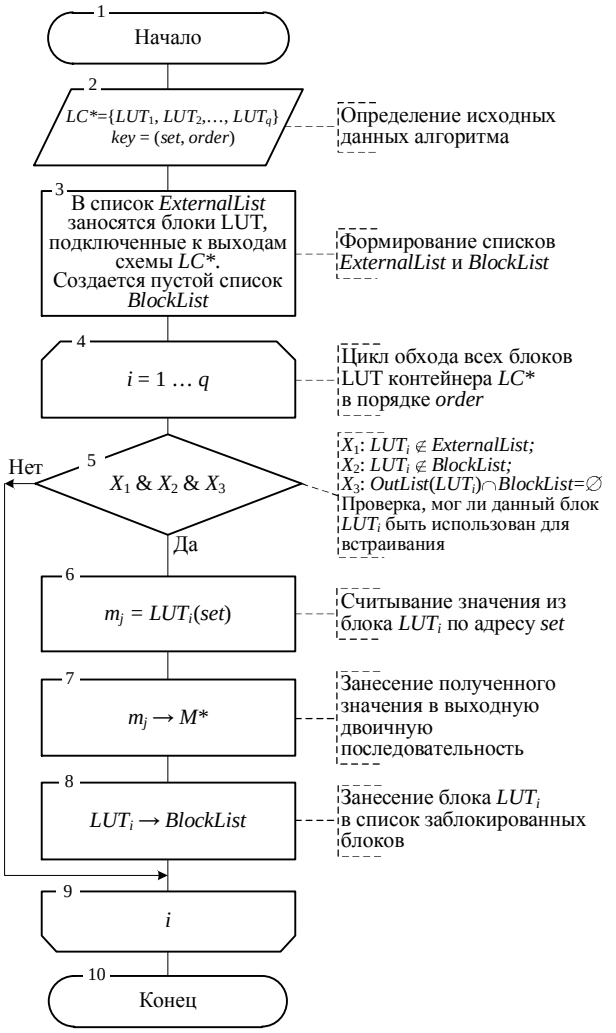

Рис. 5. Блок-схема метода извлечения данных из LUT-контейнера

Предложенные методы могут найти применение при разработке аппаратно-программного обеспечения, реализующего внедрение цифровых водяных знаков в компьютерные и управляющие устройства, построенные на основе LUT-ориентированной элементной базы (например, FPGA [10] или ПЛИС со схожими архитектурами). Такое внедрение дает возможность контролировать правомерность использования проектной информации и самих устройств на различных этапах технологии проектирования и жизненного цикла (синтезированный FPGA проект, конфигурационный файл FPGA**,** действующее устройство).

Кроме этого, предложенные методы могут быть использованы для организации стего-защищенных систем передачи и хранения данных на основе LUT-контейнеров. Преимущества использования таких контейнеров состоят в следующем.

В отличие от традиционных мультимедийных стего-конвейеров, LUT-контейнеры являются активными, имеют влияющие друг на друга элементарные единицы, которые хранят в себе данные, представленные точно.

Взаимное влияние элементарных единиц контейнера друг на друга дает возможность производить локальные изменения их содержимого, не меняя при этом глобальной функциональности контейнера.

Точное представление данных порождает подходы к противодействию активным стего-атакам типа «стирание секретной информации», недоступные для традиционных мультимедийных контейнеров.

Природа данных, находящихся в элементарных единицах LUT-контейнеров, не дает существенной статистической связи, как между разрядами отдельных единиц, так и между разрядами соседних единиц контейнера. Это не позволяет произвести пассивную статистическую стего-атаку на такой контейнер методами стего-анализа, применяемыми для традиционных контейнеров.

Список использованной литературы

1. Конахович Г. Ф. Компьютерная стеганография [Текст] / Г. Ф. Конахович, А. Ю. Пузыренко. – К. : МК-Пресс, 2006. – 288 с.

2. Грибунин В. Г. Цифровая стеганография [Текст] / В. Г. Грибунин. – М. : Салон-пресс, 2002. – 344 с.

3. Аграновский А.В. Стеганография, цифровые водяные знаки и стегоанализ [Текст] / А. В. Аграновский, А. В. Балакин, В. Г. Грибунин. – М. : Вузовская книга, 2009. – 220 с.

4. Skoudis E. Malware: Fighting Malicious Code [Text] / E. Skoudis, L. Zeltser. – New Jersey: Prentice Hall, 2004. – 672 p.

5. El-Khalil R. Hydan: Hiding Information in Program Binaries [Text] / R. El-Khalil, A. Keromytis // Proceedings of International Conference on Informaton and Communications Security (ICICS 2004). – Malaga, Spain, 2004. – P. 187 – 199.

6. Hamilton A. Survey of Static Software Watermarking [Text] / A. Hamilton, S. Danicic // Proceedings of Internet Security World Congress (WorldCIS-2011). – London, 2011. – P. 100 – 107.

7. Hakun L. New approaches for software watermarking by register allocation [Text] / L. Hakun, K. Keiichi // Proceedings of the ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel Distributed Computing. – 2008. – P. 63 – 68.

8. Xiao Cheng L. Software Watermarking Algorithm Based on Register Allocation [Text] / L. Xiao Cheng, C. Zhiming // Proceedings of International Symposium Distributed Computing and Applications to Business Engineering and Science (DCABES). – Hong Kong, 2010. – P. 539 – 543.

9. Защелкин К. В. Метод стеганографического скрытия данных в LUT-ориентированных аппаратных контейнерах [Текст] / К. В. Защелкин, Е. Н. Иванова // Электротехнические и компьютерные системы. – К. : 2013. – № 12 (88). – С. 83 – 90.

10. Максфилд К. Проектирование на ПЛИС: архитектура, средства, методы [Текст] / К. Максфилд. – М. : Додека-XXI, 2007. – 408 с.

Получено 18.02.2014

References

1. Konahovich G.F., and Puzirenko A.U. Kompyuternaya steganografiya [Computer Steganography], (2006), Kiev, Ukraine, *MK-Press Publ.*, 288 p. (In Russian).

2. Gribunyn V.G. Tsifrovaya steganografiya [Digital Steganography], (2002), Moscow, Russian Federation, *Salon-Press Publ.*, 344 p. (In Russian).

3. Agranovsky A.V., Balkin A.V., and Gribunyn V.G. Steganografiya, tsifrovye vodyanye znaki i stegoanaliz [Steganography, Digital Watermarks and Stegoanalysis], (2009), Moscow, Russian Federation: *University Book Publ.*, 220 p. (In Russian).

4. Skoudis E., and Zeltser L. Malware: Fighting Malicious Code, (2004), New Jersey: *Prentice Hall Publ.*, 672 p. (In English).

5. El-Khalil R., and Keromytis A. Hydan: Hiding Information in Program Binaries, (2004), *Proceedings of International Conference on Informaton and Communications Security (ICICS-2004)*, Malaga, Spain, pp. 187 – 199, (In English), doi: 10.1007/978-3-540-30191-2_15.

6. Hamilton A., and Danicic S. A Survey of Static Software Watermarking, (2011), *Proceedings of Internet Security World Congress (WorldCIS-2011)*, London, 2011, pp. 100 – 107. (In English).

7. Hakun L., and Keiichi K. New approaches for software watermarking by register allocation, (2008), *Proceedings of the ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel Distributed Computing*, pp. 63 – 68, (In English), doi: 10.1109/SNPD.2008.137.

8. Xiao Cheng L., and Zhiming C. Software Watermarking Algorithm Based on Register Allocation, (2010), *Proceedings of International Symposium Distributed Computing and Applications to Business Engineering and Science (DCABES)*, Hong Kong, 2010, pp. 539 – 543, (In English), doi: 10.1109/DCABES.2010.114.

9. Zashcholkin K.V., and Inanova E.N. Metod steganograficheskogo skrytiya dannykh v LUT-orientirovannykh apparatnykh konteinerakh [Method of Stegano Graphical Hiding of Information in LUT-oriented Hardware Containers], (2013), *Electrotechnic and Computer Systems*, Kiev, Ukraine, No. 12 (88), pp. 83 – 90 (In Russian).

10. Maxfield C. Proektirovanie na PLIS: arkhitektura, sredstva, metody [Design on FPGA: Architecture, Tools, Methods], (2007), Moscow, Russian Federation: *Dodeka-XXI Publ.*, 408 p. (In Russian).

Защелкин Константин Вячеславович, кандидат технических наук, доцент кафедры компьютерных интеллектуальных систем и сетей Одесского национального политехнического университета.
тел.: (048) 734-83-22
e-mail: const-z@te.net.ua

Иванова Елена Николаевна, старший преподаватель кафедры компьютерных систем Одесского национального политехнического университета.
тел.: (048) 734-83-91
e-mail: enivanova@ukr.net