Оригинальная статья

https://doi.org/10.21869/2223-1560-2023-27-1-59-73

Модель размещения данных во внутренней памяти вычислителя, реализующего схему кодирования данных в режиме сцепления блоков

М.О. Таныгин [1] ✉, А.А. Ахмад [1], О.В. Казакова [1], Д.А. Голубов [1]

[1] Юго-Западный государственный университет
ул. 50 лет Октября, д. 94, г. Курск 305040, Российская Федерация

✉ e-mail: tanygin@yandex.com

Резюме

Цель исследования. *В статье исследуются особенности синтеза специализированных устройств, выполняющих контроль целостности и аутентичности отдельных информационных блоков на основе CBC-кодов. Рассматриваются подходы, имеющие своей целью снижение вычислительных и ресурсных затрат при выполнении таких процедур. Формулируются зависимости между показателями функцио-нирования и требованиями по объёму внутренней памяти специализированных вычислителей, обраба-тывающих данные в режиме CBC.*

Методы. *Реализация процедур аутентификации источника данных с использованием кодирования в режиме сцепления отдельных блоков данных является действенным методом повышения достоверности в условиях ограниченного размера проверяемых аутентификационных последовательностей. В то же время её реализация в приёмниках требует создания специализированных вычислителей, обрабатывающих возникающие при декодировании данных древовидные структуры, состоящие из отдельных блоков данных. Для снижения ресурсных и вычислительных затрат при обработке таких структур используется косвенная адресация, при которой данные блоков хранятся в оперативной памяти, а их адреса – в высокоскоростной регистровой памяти самого вычислителя.*

Результаты. *Создана модель косвенной адресации блоков данных в оперативной памяти. Указатели на блоки хранятся в отдельной регистровой памяти, и каждый указатель размещён в соответствии с декодированным порядковым номером блока в последовательности. Каждый адрес блока в оперативной памяти дополняется множеством указателей на последующие блоки в ветвях древовидной структуры. Созданная математическая модель позволила оценить размер в битах всех задействованных указателей и их число, что позволило определить потребность вычислителя, выполняющего обработку древовидной структуры, в регистровой памяти.*

Заключение. *В работе показано, что использование комбинации матричной и списочной организации хранения адресов блоков позволяет снизить потребность в регистровой памяти вычислителя на 50 – 60%. При этом вычислитель может обрабатывать древовидную структуру информационных блоков с использованием высокопроизводительных итерационных алгоритмов, что было бы невозможно при использовании исключительно списочной организации хранения адресов .*

Ключевые слова: *обработка данных; аутентификация; аппаратная сложность; косвенная адресация; древовидные структуры данных; математическое моделирование.*

Конфликт интересов: *Авторы декларируют отсутствие явных и потенциальных конфликтов интересов, связанных с публикацией настоящей статьи.*

Для цитирования: *Модель размещения данных во внутренней памяти вычислителя, реализующего схему кодирования данных в режиме сцепления блоков* / М.О. Таныгин, А.А. Ахмад, О.В. Казакова, Д.А. Голубов // Известия Юго-Западного государственного университета. 2023; 27(1): 59-73. https://doi.org/10.21869/2223-1560-2023-27-1-59-73.

Поступила в редакцию 18.12.2022 *Подписана в печать 21.02.2023* *Опубликована 14.04.2023*

***

Введение

Режим кодирования с сцеплением блоков (англ. Cipher Block Chaining, CBC) применяется для формирования самосвосстанавливающихся кодов, в которых ошибка при передаче или обработке одного блока не ведёт к распространению ошибок на остальные. В то же время его вариации в сочетании с алгоритмами формирования имитовставок (англ. message authentication code, MAC) являются наиболее удачным решением для проверки целостности и аутентичности информации, фрагментированной на блоки данных небольшой длины [1], то есть в таких системах, при которых использование стандартных и хорошо зарекомендовавших себя алгоритмов оказывается неэффективным [2, 3]. Существует большое множество вариаций данного подхода, основанных на использовании как типовых алгоритмов шифрования, так и на оригинальных [4 – 10]. Подготовленный к отправке блок информации кодируется для создания цепочки блоков, для которой действует правило: «содержимое каждого последующего блока зависит от результата преобразования предыдущего», как показано на рис. 1, где: m_i – содержимое соответствующего блока данных; E – алгоритм преобразования на основе некоторой ключевой последовательности; c_i – содержимое блока данных, полученное в результате применения алгоритма E к данным m_i; Tag – последний блок данных, являющийся имитовставкой всего фрагментированного сообщения.

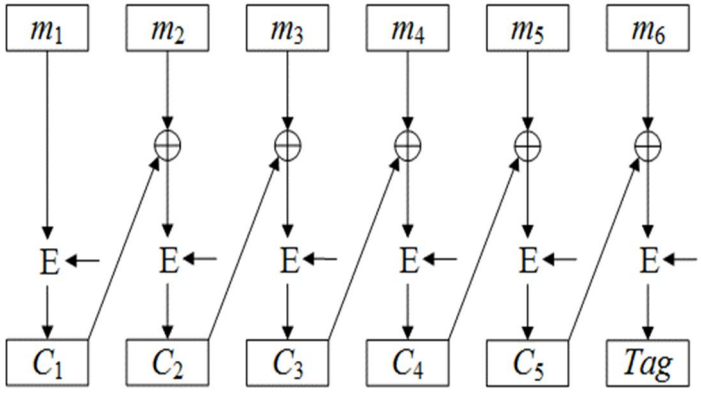

Рис.1. Процесс формирования цепочки связанных блоков

Взаимозависимость данных блоков позволяет не беспокоиться о достоверности контроля целостности или аутентичности отдельных блоков: в приёмнике блоки собираются в цепочку и проверяются, при этом обнаруживаются изменения в произвольном бите исходного сообщения.

Материалы и методы

При использовании CBC–MAC как метода аутентификации источника блоков данных в состав каждого блока вводится поле проверочной информации, которое формируется на основе уникального идентификатора источника, данных самого блока и данных, полученных из предыдущего блока цепочки [11]. Для протоколов, в которых кадр передаваемых данных ограничен несколькими байтами [12], размер поля проверочной информации не может превышать нескольких битов, использование CBC становится эффективным методом обеспечения требуемой достоверности [13, 14].

Особенности реализации процедур аутентификации в реальных вычислителях, заключающиеся в необходимости обработки последовательного потока блоков данных, не позволяют формировать все варианты цепочек блоков после получения некоторого множества таких блоков. В таком случае алгоритм обработки блоков имеет факториальную сложность, что делает метод CBC–MAC неприемлемым при больших длинах цепочек, которые и обеспечивают высокую достоверность. Поэтому аппаратноориентированные процедуры контроля источника поступающих блоков данных несколько модифицируются [15]. Вычислитель формирует структурированную последовательность информационных блоков, буферизируя их содержимое во внутренней памяти. Условием добавления блока в цепочку является выполнение равенства:

$$\mathrm{F}_1\left(J_{i-1},\Omega\right)=\mathrm{F}_2\left(J_i,\Omega,A_i\right), \qquad (1)$$

где J_{i-1}, J_i – информационные части блоков i–1 и i соответственно;

A_i – содержимое имитовставки блока i;

Ω – идентификатор отправителя блока;

F_1 – необратимое преобразование,

F_2 – обратимое преобразование (обратное ему преобразование используется при формировании имитовставки A_i).

Так как мощность алфавита выходных слов преобразования F_2 определяется разрядностью поля имитовставки A_i, которая принята нами равной нескольким битам, то существует значимая вероятность того, что к блоку i–1 будет добавлен не один, а несколько последующих блоков. В результате формируется древовидная структура блоков данных (ДСБД) [16].

Задачи формирования описанной выше структуры, размещения её в памяти вычислителя, выполняющего аутентификацию источников данных, и обработка являются сложными и ресурсоёмкими. Необходимость их решения входит в противоречие с жёсткими ограничениями по производительности и аппаратной сложности рассматриваемого класса устройств, взаимодействующих

по протоколам с ограниченным размером передаваемого кадра, которые связаны, в том числе, с требованиями по энергопотреблению и автономности [17-19]. Подходом, который обеспечит повышение скорости выполнения операций обработки ДСБД и снижение затрат оперативной памяти, является косвенная адресация буферизируемых информационных блоков: все они хранятся в произвольном порядке в буфере (ОЗУ), а их описатели блоков данных (ОБД) – в простейшем случае адреса блоков в ОЗУ – в отдельной регистровой памяти. Отдельные элементы данной памяти через систему перекрёстных ссылок образуют указанную древовидную структуру, которая обеспечивает эффективное использование регистровой памяти, так как [20]. Алгоритмы работы со списками обладают высокой вычислительной сложностью и требуют реализации рекурсивного вызова подпрограмм, так как в общем случае для поиска нужного элемента требуется прохождение по ветвям графовой структуры. С учётом того, что в обозначенной задаче нет критериев выбора правильного направления при разветвлении графа требуется перебор всего содержимого всех ветвей. В решениях [21, 22] повышение производительности достигается за счёт парализации процесса перебора маршрутов, формируемых в графах. Но общее количество операций обработки вершины графа (данных, содержащихся в элементе списочной структуры) для поиска требуемой, будет равномерно распределённой случайной величиной от нуля до общего количества вершин в графе. Описанные в [23] подходы к хранению и обработке списочных структур так же отмечают этот недостаток, который присущ невзвешенным графам, для которых, в отличие от взвешенных, нет формальных алгоритмов выбора направления движения по графу. Соответственно, среднее же число операций выбора блоков из памяти будет равно половине числа элементов в списочной структуре.

Решение для хранения и обработки больших объёмов связанных данных описано в [24]. Авторами предлагается аналогичная древовидным структурам схема работы с информацией, при которой части информации извлекаются из памяти последовательно, исходя из анализа предыдущих выбранных данных. Ими так же выбрана концепция временного хранения части данных в кеше с высокой скоростью доступа. В то же время используемый подход основан на зависимости извлекаемых и анализируемых данных, что позволяет, как и в случае с обработкой ориентированных графов, реализовывать алгоритмы поиска и извлечения данных со сложностью, отличной от сложности задач прохождения по всем ветвям древовидной структуры.

Более простые в аппаратной реализации итерационные алгоритмы, работающие с ДСБД как с совокупностью векторов блоков данных [25-28], требуют независимых блоков памяти максимального размера для каждой ветви древовидной структуры вне зависимости от её фактической длины. При этом

сама регистровая память описателей превращается в набор векторов ОБД или матрицу. Число операций для поиска сообщений в такой организации распределено от нуля до максимальной длины последовательности, что является более выигрышным решением с точки зрения производительности. Применительно к задаче анализа цепочек сообщений, кодированных в режиме сцепления – среднее число операций, требуемых для поиска нужного элемента равно половине от длины обрабатываемой цепочки. Это меньше, чем при описанных в [20-24] подходах к обработке и хранению списков, но требует резервирования регистровой памяти для возможности записи вектора максимальной длины в любой столбец матрицы.

Таким образом, можно сделать вывод о том, что стоящая перед нами задача обработки цепочек сообщений должна ориентироваться на оптимизацию такого параметра, как требуемые затраты памяти для хранения дополнительной структуры, обеспечивающей адресацию и извлечение из памяти данных блоков. При этом вычислительная сложность поиска и обработки сообщений, ввиду того, что обрабатываемые списочные структуры представляют собой невзвешенные графы, определяется лишь размером такой списочной структуры и не может быть решена схемотехническими решениями, рассмотренными выше.

В настоящей работе мы рассматриваем модель хранения ОБД, представляющую собой комбинацию списочной структуры и матричной памяти. Она применима для методов аутентификации, в которых в атрибуте блока данных в явном виде присутствует позиция блока в цепочке. Это позволяет в ОБД хранить не ссылки не на произвольные регистры памяти, а лишь на те, которые находятся в следующем столбце относительно текущего ОБД (рис.2). На рисунке число столбцов матричной памяти равно максимальной длине проверяемой цепочки блоков M, а число строк N определяется, исходя из условий функционирования устройства, необходимого числа регистров для хранения адресов всех буферизированных блоков данных [29].

Таким образом, описатель блока данных в регистровой памяти будет представлять из себя конкатенацию следующих слов:

– адреса блока в буферной памяти;

– результат преобразования $F_1(J_i, \Omega)$ для данного i-го сообщения для сравнения с результатом $F_2(J_{i+1}, \Omega, A_{i+1})$, следующего блока в ветви (это обеспечивает сокращения временных затрат при многократном выполнении операции сравнения);

– результат преобразования $F_2(J_i, \Omega, A_i)$ для сравнения с содержимым поля предыдущего блока в ветви;

– множество указателей $G = \{g_1, g_2, \dots, g_t\}$ на последующие элементы ветвей.

При поступлении блока в приёмник его обработка и размещение описателя в регистровой памяти, описываемой матрицей B размером $N \times M$, происходит по приведённому ниже алгоритму, в котором приняты следующие обозначения:

$B_{i,j}$ – элемент регистровой памяти, размещённый в *i*-й строке и *j*-м столбце регистровой памяти;

F^{ind} – операция определения позиции блока [30];

$|G^{i,k}|$ – число сформированных указателей у элемента, размещённого в *i*-й строке и *j*-м столбце регистровой памяти;

$J_{i,j}$ – данные блока, размещённого в *i*-й строке и *j*-м столбце регистровой памяти;

$A_{i,j}$ – имитовставка блока, размещённого в *i*-й строке и *j*-м столбце регистровой памяти;

$B^{\langle j \rangle}$ – содержимое *j*-го столбца регистровой памяти;

$|B^{\langle j \rangle}|$ – число занятых регистров в *j*-м столбце регистровой памяти;

$g_k^{i,j}$ – содержимое указателя с номером k у элемента, размещённого в i-й строке и j-м столбце регистровой памяти.

Основные этапы алгоритма:

1. Получение блока данных $u=\{J, A\}$. Выделение из его атрибутов номера блока $\tau(u)$ в цепочке: $\tau(u) = F^{\text{ind}}(u)$.

2. Установка переменной цикла k:=0, сброс флага f := 0.

3. Если $F_2(J, \Omega, A) = F_1(J_{\tau(u)-1,\, k}, \Omega)$, то перейти к пункту 4, иначе – к пункту 6.

4. Если f = 0, то добавляем блок в матрицу $B_{\tau(u),|B^{\langle\tau(u)\rangle}|+1}:=u$, устанавливаем флаг f := 1, переходим к пункту 5.

5. Добавляем в множество указателей ссылку на новый блок (порядковый номер блока): $g^{\tau(u),k}_{|G^{\tau(u),k}|}:=|B^{\langle\tau(u)\rangle}|$.

6. Инкрементируем переменную цикла k:= k+1.

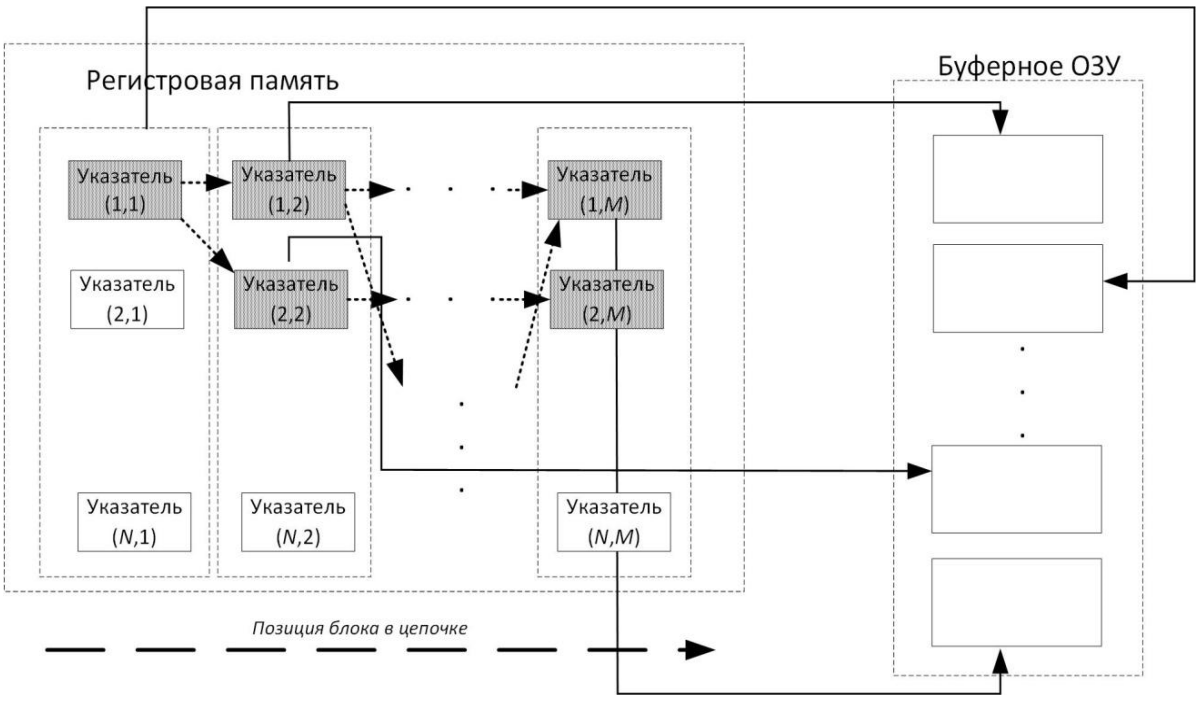

Рис. 2. Косвенная адресация сообщений в буферной памяти

7. Если $k > \left|B^{\langle\tau(u)\rangle-1}\right|$, то переход к пункту 8, иначе – переход к пункту 3.

8. Установка переменной цикла k:=0.

9. Если $F_2(J_{\tau(u)+1,\,k}, \Omega, A_{\tau(u)+1,\,k}) = F_1(J, \Omega)$, то перейти к пункту 10, иначе – к пункту 12.

10. Если $f = 0$, то добавляем блок в матрицу $B_{\tau(u),\left|B^{\langle\tau(u)\rangle}\right|+1} := u$, устанавливаем флаг $f := 1$, переходим к пункту 11.

11. Добавляем в множество указателей ссылку на новый блок (порядковый номер блока): $g^{\tau(u),k}_{\left|G^{\tau(u),k}\right|} := \left|B^{\langle\tau(u)\rangle}\right|$.

12. Инкрементируем переменную цикла $k := k+1$.

14. Если $k > \left|B^{\langle\tau(u)\rangle+1}\right|$, то переход к пункту 14, иначе – переход к пункту 9

Конец процедуры добавления блока в регистровую память.

Для оценки потребности вычислителя, выполняющего обработку поступающих блоков данных и аутентификацию их источника, в регистровой памяти необходимо определить разрядность h размеров слов $F_1(J_i, \Omega)$ и $F_2(J_{i+1}, \Omega, A_{i+1})$ и число U блоков данных, хранящихся в буфере (влияет на разрядность адреса блока данных в буфере). Эти параметры определяются числом K источников, которые формируют блоки данных, и длиной цепочки блоков M [30]:

$$\begin{aligned} U &= K \cdot M, \\ h &= \left]\log_2 K\right[+ 2, \end{aligned} \qquad (2)$$

где $\left]\log_2 K\right[$ – минимальное целое, большее $\log_2 K$.

Кроме этого, необходимо определить требуемое для безошибочной работы вычислителя число регистров в столбце матрицы регистров (рис. 2), и число указателей в каждом ОБД. Для этого создана математическая модель размещения информационных блоков в регистровой памяти вычислителя, в качестве параметров которой, помимо перечисленных, было использовано число строк матрицы регистров N. Представив процесс получения блоков данных вычислителем как пуассоновский процесс [29], получим функцию вероятности $p(n,j)$ дискретной случайной величины n числа блоков данных, полученных от всех источников, для которых на основании содержимого атрибутов вычислена одинаковая позиция блока в цепочке:

$$p(n, j) = \frac{\left(\frac{K \cdot j}{M}\right)^n \times e^{-\frac{K \cdot j}{M}}}{n!},$$

где K/M – интенсивность поступления в приёмник сообщений с некоторым порядковым номером;

j – число информационных блоков цепочки источника, полученное к какому либо моменту времени;

n – число блоков данных с одинаковой позицией в цепочке, размещаемых в одном столбце регистровой памяти.

К каждому элементу древовидной структуры, хранящейся в регистровой памяти, может быть добавлено r блоков данных, $r = 0 \ldots U$, образуя при этом r новых ветвей. Это число является дискретной случайной величиной с функцией вероятности, которая получена из (2) при $j = M$ и биномиального распределения вероятностей из k экспериментов при числе успехов r:

$$p^{\text{point}}(r)=\sum_{i=r}^{U}\left(\frac{K^i\times e^{-K}}{i!}\left(\sum_{k=r}^{i}\frac{K^k\times e^{-K}}{k!}C_k^r\left(2^{-h}\right)^r\left(1-2^{-h}\right)^{U-r}\right)\right), \quad (4)$$

где 2^{-h} – вероятность добавления очередного блока в цепочку;

U – число блоков данных, хранящихся в буфере;

$p^{\text{point}}(r)$ – вероятность добавления к произвольному элементу древовидной структуры ровно r ветвей.

Модель размещения данных в каждом элементе матричной регистровой к моменту получения приёмником M сообщений целевого источника описывается множеством вероятностей $p_{i,j}^{\text{pos}}$ $j=1\ldots M-1$, $i=0\ldots N$, каждую из которых получают из рекуррентной зависимости, полученной на основе формул (3) и (4):

$$\begin{cases} p_{j,0}^{\text{pos}}=1,\ j=1\ldots M-1, \\ p_{0,j}^{\text{pos}}=p^{\text{point}}(j),\ j=1\ldots U-M, \\ p_{i,j}^{\text{pos}}=\sum_{v=0}^{j}\left(p^{\text{point}}(v)\cdot p_{i-1,j-v}^{\text{pos}}\cdot 2^{-h}\right), \end{cases} \quad (5)$$

где $p_{i,j}^{\text{pos}}$ – вероятность занятия элемента регистровой памяти в i–м столбце и j–й строке.

Как было сказано, при синтезе устройств необходимо ограничить мощность множества указателей G описателя блока данных некоторым пороговым значением t. Подобное ограничение даст ошибку нехватки указателей, вероятность которой $p^{\text{err}}(t)$ определится по формуле

$$p^{\text{err}}(t)=1-\prod_{i=1}^{M}\sum_{j=1}^{U-M}\left(p_{i,j}^{\text{pos}}\left(1-\sum_{v=t}^{U-M}p^{\text{point}}(v)\right)^{j+1}\right), \quad (6)$$

где $\sum_{v=t}^{U-M}p^{\text{point}}(v)$ – вероятность нехватки указателей для одного ОБД.

Результаты и их обсуждение

Полученные формулы позволили визуализировать интересующие нас зависимости, которые должны быть учтены при синтезе вычислителей. Ниже, на рис. 3 приведен график зависимости вероятности ошибки нехватки указателей в ОБП от числа указателей t и длины цепочки сообщений M.

Анализ данных зависимостей, а также зависимостей, полученных при других значениях K и h, показал, что на форму кривой p^{err} влияет соотношение между числом источников и разрядностью сравниваемых слов (1): при полученном в [30] соотношении $K<2^{h+2}$ вероятность ошибки падает практически до нулевых значений уже при небольших t (рис. 3, *a*), тогда как если данное соотношение не выполняется, вероятность ошибки близка к 1 и снижается лишь незначительно с ростом числа указателей t (рис. 3, b). При этом, в случае, если требование по минимальной длине проверяемых слов выполняется, зависимость ошибки от длины последовательности блоков данных может быть аппроксимирована прямой при $t=3\ldots6$.

Практический интерес представляет собой определение минимального значения $t^{\min}$ числа указателей в ОБП, при котором значение вероятности ошибки p^{err} не превышает некоторого порогового значения Por (рис. 4):

$$\exists!t^{\min}:\left(p^{\text{err}}\left(t^{\min}\right)<Por\right)\vee \vee\left(p^{\text{err}}\left(t^{\min}+1\right)\geq Por\right). \quad (7)$$

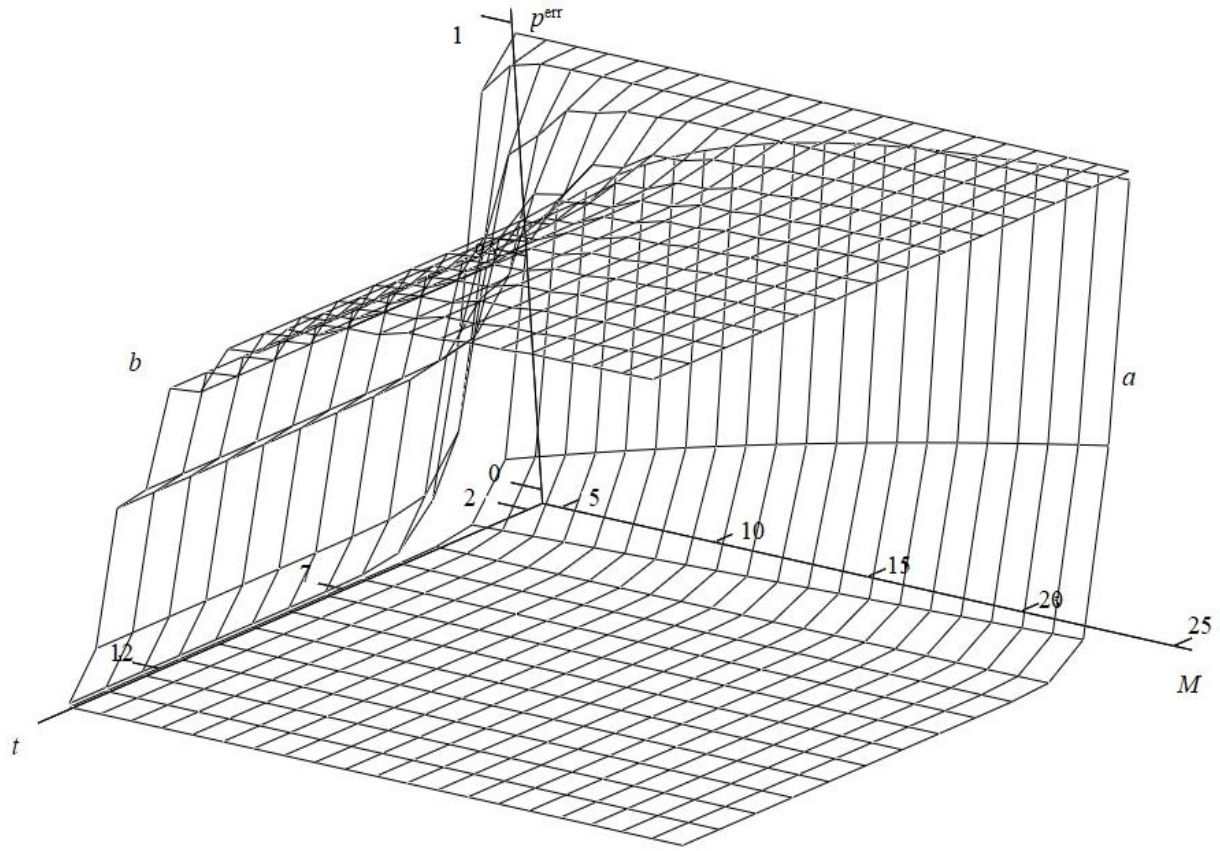

Рис. 3. График зависимости вероятности p^{err} ошибки нехватки указателей в ОБП от числа указателей t и длины цепочки сообщений M, числа источников данных K и разрядности проверочного слова h: **a –** K=30, h = 6; **b –** K=30, h = 5

Серии математических экспериментов при различных значениях исходных параметров позволили сформулировать следующие правила для расчета числа указателей:

$$t^{min} =]M/10[+ 2, \quad (8)$$

где $]M/10[$ – минимальное целое, большее $M/10$.

Кроме того, установлено, что увеличение разрядности h проверяемых слов на 1 бит в среднем снижает минимальное число указателей также на 1 бит, что может быть использовано для манипулирования требуемыми размерами регистровой памяти, содержащей ОБД.

Общая потребность в регистровой памяти для хранения одного ОБД будет равна в битах сумме следующих слагаемых:

– адреса блока в буферной памяти – $]\log_2(K \cdot M)[+ 1$;

– два слова по $h =]\log_2(K)[+ 2$ для организации сравнения кодов аутентификации;

– $(]M/10[+ 2) \times 2 \times]\log_2(M/10[+ 2)[$ битов для хранения указателей на последующие блоки.

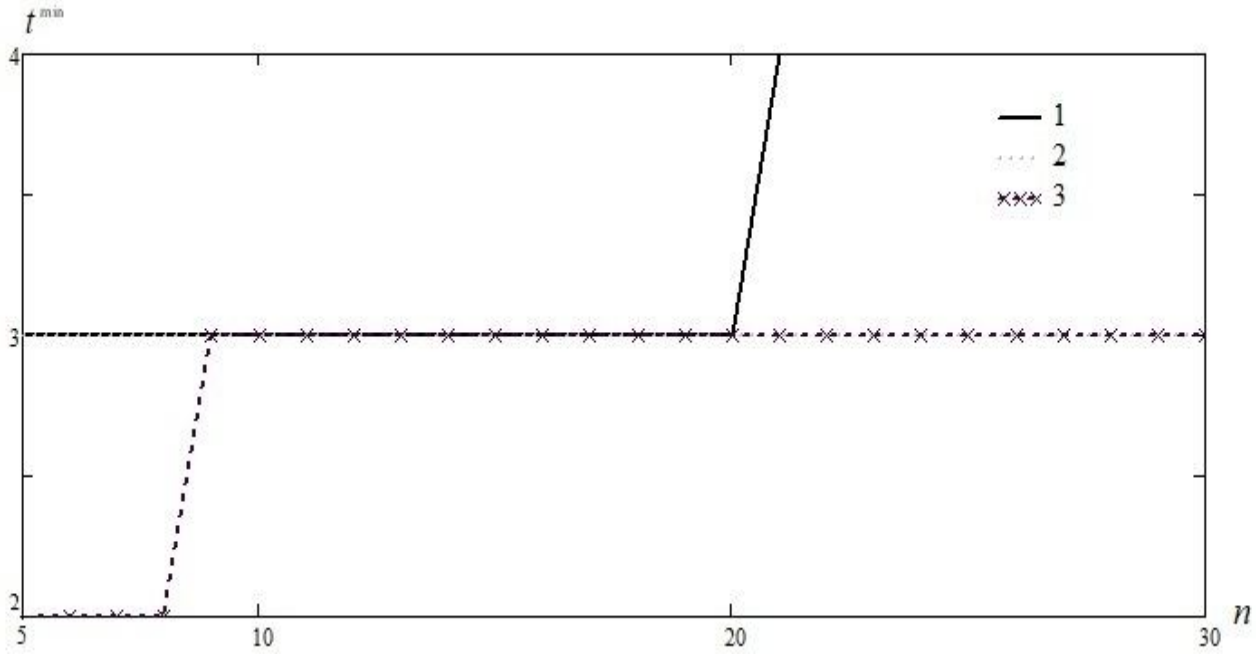

Рис. 4. График зависимости минимального числа указателей t^{min} от длины цепочки сообщений M и максимальной ошибки Por, при числе источников K = 30 и h = 6: **1** – Por = 0,05; **2** – Por = 0,1; **3** – Por = 0,25

В последнем слагаемом логарифм появляется из-за введения максимального ограничения на количества строк в матрице регистров – мы предполагаем, что каждый из $]M/10[+ 2$ указателей ОБД ссылается на такое же количество ОБД в следующем столбце. В реальности, как показано в [30], число строк может быть ограничено 3 – 5.

Если сравнивать затраты в регистровой памяти в предлагаемой модели с аналогичными затратами в исключительно матричной системе хранения [20-28], в которой последовательности блоков представляют собой совокупность векторов, каждый элемент которых есть указатель на элемент в каждом столбце регистровой памяти. В таких системах первые два слагаемых в размере ОБД остаются такими же, третье отсутствует, а вместо этого появляется компонента, равная произведению $2\times\log_2(]M/10[+ 2)\cdot M\cdot N^{chain}$, где N^{chain} – константа, имеющая порядок 10^2 [16].

Графики зависимости потребности в регистровой памяти от длины цепочки блоков показывают (рис. 5), что в предлагаемой модели размещения ОБД она на 50 – 60% меньше, чем в системах, где для хранения сформированных ветвей используются дополнительные вектора указателей.

Таблица 1. Сравнение характеристик методов размещения данных

Метод размещения данных	Затраты внутренней регистровой памяти/	Среднее число обращений к регистровой памяти для поиска элемента в цепочке
Матричное хранение адресов блоков данных	V	$M/2$
Хранения адресов блоков данных в виде списочной структуры	0.4…0.5 V	$K \cdot M/2$
Комбинированная организация	0.4…0.5 V	$M/2$

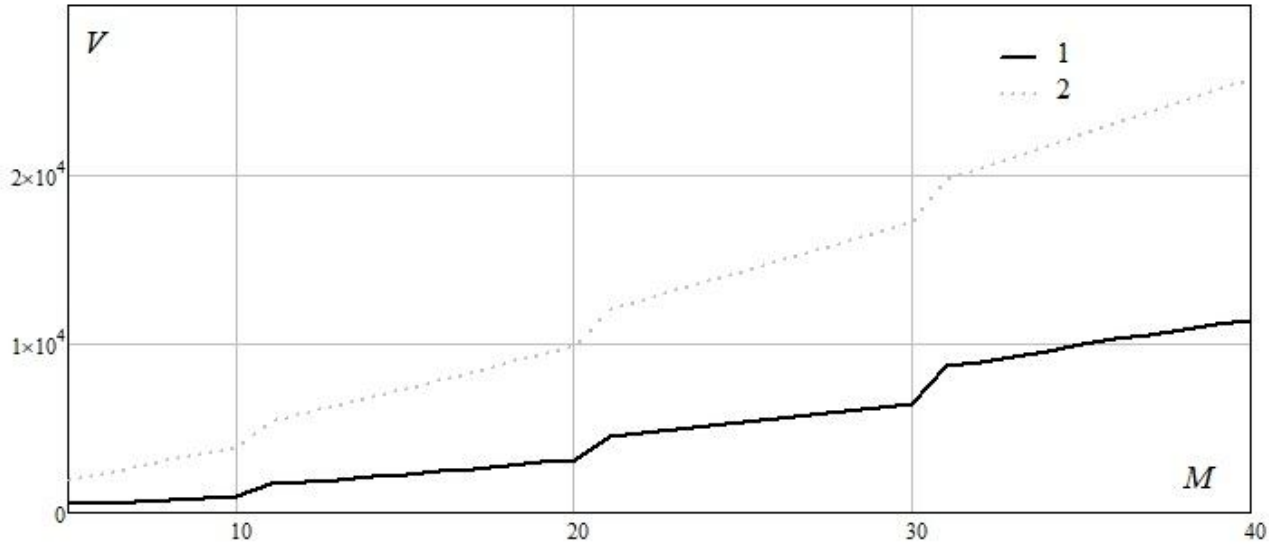

Рис. 5. Зависимость требуемой ёмкости V регистровой памяти (в битах) от длины цепочки блоков данных M при числе источников K=60: **1** – предлагаемая модель регистровой памяти; **2** – матричная организация регистровой памяти с дополнительными векторами ветвей древовидной структуры

Выводы

В результате математического моделирования была определена потребность в регистровой памяти вычислителя для организации косвенной адресации блоков данных в основном ОЗУ. Установленные соотношения применимы при синтезе вычислителей, обрабатывающих древовидные структуры

блоков данных, формируемые при выполнении процедур аутентификации источников сообщений на основе метода сцепления блоков. Если сравнивать характеристики описываемого подхода к адресации блоков в буфере при формировании цепочек блоков (табл. 1), то комбинированный метод, как было сказано ранее, требует меньших затрат внутренней регистровой памяти по сравнению с матричной организацией (на уровне затрат при хранении указателей в виде списков) [25, 26], и обеспечивает равное с ней и в K раз меньшее (K – число источников данных, для которых выполняется аутентификация) число операций обращения к памяти по сравнению с использованием списочной структуры [21, 22].

Использование комбинации указателей на элементы древовидной структуры с матричным размещением её элементов позволило более чем в два раза уменьшить требуемый размер регистровой памяти вычислителя, что позволяет, при реализации таких вычислителей на ПЛИС, использовать более дешёвые и простые микросхемы, реализовывать косвенную адресацию без использования дополнительных блоков ОЗУ, реализовывать на одной микросхеме несколько вычислительных ядер, повышая итоговую производительность устройства.

Полученные результаты определяют потребность в регистровой памяти для хранения всех возможных разветвлений в древовидной структуре, их зависимость от числа источников и длины последовательности блоков. Как показали ранее проведённые исследования, от этих же параметров зависит и достоверность описываемого метода аутентификации источников данных по последовательностям небольшой (до нескольких битов) длины. Что создает предпосылки для реализации методов обнаружения ошибок на основе подсчёта числа разветвлений в древовидной структуре Кроме этого, на основании анализа структуры ветвей и последовательности их построения можно формировать численные оценки для повышения достоверности выделения ветви из цепочек целевого источника.

Список литературы

1. Царегородцев К. Д. Анализ режимов шифрования для реализации в устройствах RFID // Прикладная дискретная математика. Приложение. 2020. № 13. С. 67-69. DOI: 10.17223/2226308X/13/20.

2. Мальчуков А.Н., Осокин А.Н. Система автоматизированного проектирования кодеков помехоустойчивых кодов короткой длины // Известия Томского политехнического университета. 2008. Т. 312. № 5. С. 70-75.

3. Мыцко Е.А., Мальчуков А.Н., Иванов С.Д. Исследование алгоритмов вычисления контрольной суммы CRC8 в микропроцессорных системах при дефиците ресурсов // Приборы и системы. Управление, контроль, диагностика. 2018. № 6. С. 22-29.

4. Hang S.J., Oh H.S., Park J. The improved Data Encryption Standard (DES) Algorithm // IEEE 4th International Conference on Spread Spectrum Techniques and Applications Proceedings. 1996. P. 1310–1314.

5. Kruti S., Gambhava B. New Approach of Data Encryption Standard Algorithm // Int. J. Soft Comput. Eng. 2012. Vol. 2. № 1. P. 322–325.

6. Sumathi R. A Secure Data Transfer Mechanism Using Single-Handed Re-Encryption Technique // International Conference on Emerging Trends in Science, Engineering and Technology. Tiruchirapalli, India. 2012. P. 1-9.

7. C Liu., Ji J., Liu Z. Implementation of DES Encryption Arithmetic based on FPGA / // AASRI Procedia. 2013. Vol. 5. P. 209–213.

8. Kruti S., Gambhava B. New Approach of Data Encryption Standard Algorithm // Int. J. Soft Comput. Eng. 2012. Vol. 2. № 1. P. 322–325.

9. Mandal P.C. Evaluation of performance of the Symmetric Key Algorithms: DES, 3DES, AES and Blowfish // J. Glob. Res. Comput. Sci. 2012. Vol. 3. № 8. P. 67–70.

10. Xie J., Pan X. An improved rc4 stream cipher // International Conference on Computer Application and System Modeling.2010. doi: 10.1109/IC-CASM.2010.5620800

11. Black J., Rogaway P. CBC MACs for arbitrary-length messages: The three-key constructions // J. Cryptol, 2015. Vol. 18. №2. P. 111–131.

12. Предварительный национальный стандарт РФ. Информационные технологии. Интернет вещей. Протокол обмена для высокоемких сетей с большим радиусом действия и низким энергопотреблением. URL: http://docs.cntd.ru/document/554596382 (дата обращения 15.02.2021).

13. Stallings W. NIST Block Cipher Modes of Operation for Confidentiality // Cryptologia. 2010. № 34(2). P. 163 – 175.

14. Ben Othman, S., Alzaid, H., Trad A., Youssef, H. An efficient secure data aggregation scheme for wireless sensor networks // IISA 2013, doi:10.1109/iisa.2013.6623701

15. Таныгин М.О. Алгоритм определения источника фрагментированных сообщений // Известия ВУЗов. Приборостроение. 2020. Т. 63, №8. С.702 – 710.

16. Рекурсивный алгоритм формирования структурированных множеств информационных блоков для повышения скорости выполнения процедур определения их источника / М.О. Таныгин, Х.Я.А. Алшаиа, В.П. Добрица, О.Г. Добросердов // Известия Юго-Западного государственного университета. 2021; 25(2): 51-64. https://doi.org/10.21869/2223-1560-2021-25-2-51-64.

17. Yağdereli E. Gemci, C. A study on cyber-security of autonomous and unmanned vehicles // Journal of Defense Modeling and Simulation. 2015. doi: https://doi.org/10.1177/1548512915575803

18. Беспилотные авиационные системы. Часть 3. Эксплуатационные процедуры [Электронный ресурс] // Стандарт ISO 21384-3:2019(E). URL: https://cdn.standards.iteh.ai/samples/70853/7ec34c8a22bf46958423b7e3a2e43693/ISO-21384-3-2019.pdf (дата обращения 15.09.2020)

19. Leccadito M. A Hierarchical Architectural Framework for Securing Unmanned Aerial Systems // Virginia Commonwealth University. 2016. https://doi.org/10.25772/0DK3-E418

20. Пасечников К. А., Иванова Г. С.. Синтез оптимальных структур данных для решения задач на графах // Вестник Московского государственного технического университета им. Н.Э. Баумана. Серия Приборостроение. 2008. № 4(73). С. 29-37.

21. Колганов А. С. Параллельная реализация алгоритма поиска минимальных остовных деревьев с использованием центрального и графического процессоров // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2016. Т. 5. № 3. С. 5-19. DOI 10.14529/cmse160301

22. Tasci S., Demirbas M. Employing in-memory data grids for distributed graph processing // IEEE 2015 IEEE International Conference on Big Data (Big Data). 2015. P. 1856–1864. doi:10.1109/bigdata.2015.7363959

23. Paradies M, Lehner W, Bornhövd C GRAPHITE: an extensible graph traversal framework for relational database management systems. In: DBM. ACM. 2015. P. 29:1–29:12. DOI : 10.1145/2791347.2791383

24. Babionitakis, K., Doumenis, G.A., Georgakarakos, G. et al. A real-time motion estimation FPGA architecture // J Real-Time Image Proc. 2008. No. 3. P. 3–20. https://doi.org/10.1007/s11554-007-0070-9

25. Panagiotis Papadimitratos, Zygmunt J. Haas Secure message transmission in mobile ad hoc networks // Ad Hoc Networks. 2003. №1. P. 193–209.

26. Shi X., Xiao D. A reversible watermarking authentication scheme for wireless sensor networks // Information Sciences. 2013. Vol. 240. P. 173-183. DOI:10.1016/j.ins.2013.03.031

27. Shant D., Premkumar P. Block Level Data Integrity Assurance Using Matrix Dialing Method towards High Performance Data Security on Cloud Storage // Scientific Research Publishing. 2016. Vol. 7. № 11. P. 3626-3644.

28. Таныгин М.О., Алшаиа Х.А., Добрица В.П. Оценка влияния организации буферной памяти на скорость выполнения процедур определения источника сообщений // Труды МАИ. 2020. № 5(114). С.15.

29. Таныгин М.О. Расчёт вероятности возникновения коллизий при ис-пользовании алгоритма контроля подлинности сообщений // Известия Юго-Западного гос-

ударственного университета. Серия: Управление, вычислительная техника, информатика. Медицинское приборостроение. 2012. № 2-2. С. 179-182.

30. Таныгин М.О. Теоретические основы идентификации источников информации, передаваемой блоками ограниченного размера. Курск, 2020. 198 с.

Информация об авторах

Таныгин Максим Олегович, доктор
технических наук, доцент, заведующий кафедрой
«Информационная безопасность»,
Юго-Западный государственный университет
г. Курск, Российская Федерация,
e-mail: tanygin@yandex.ru

Ахмад Али Айед Ахмад, аспирант кафедры
«Информационная безопасность»,
Юго-Западный государственный университет,
г. Курск, Российская Федерация.
e-mail: aliayid2013@gmailcom,
ORCID: http://orcid.org/0000-0002-6031-9414

Казакова Ольга Викторовна, студент
кафедры «Информационная безопасность»,
Юго-Западный государственный университет,
г. Курск, Российская Федерация,
e-mail: olya.kazakova.2000@mail.ru,
ORCID: http://orcid.org/0000-0002-7755-3151

Голубов Дмитрий Александрович,
кандидат технических наук, доцент,
Юго-Западный государственный университет,
г. Курск, Российская Федерация,
e-mail: golubov.swsu@gmail.com