ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ
СПбГУТ))) Март 2017 Том 5 № 1 ISSN 2307-1303 http://www.itt.sut.ru
TELECOM IT
March 2017 Vol. 5 Iss. 1 ISSN 2307-1303 http://www.itt.sut.ru

РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА ДЛЯ АВТОМАТИЗАЦИИ ПОСТРОЕНИЯ УМНОЖИТЕЛЯ ЭЛЕМЕНТОВ ДВОИЧНОГО ПОЛЯ ГАЛУА ПО СХЕМЕ РЕЙХАНИ-МАЗОЛЕХА

С. С. Владимиров[1*], Д. Ф. Мухаметшина[1]

[1] СПбГУТ, Санкт-Петербург, 193232, Российская Федерация
* Адрес для переписки: vladimirov.opds@gmail.com

Аннотация
Разработка современных систем цифровой передачи данных требует использования методов помехоустойчивого кодирования и шифрования, многие из которых основаны на вычислениях в конечных полях Галуа большой степени, преимущественно на сложении и умножении элементов поля, из которых последняя операция требует большего количества логических элементов при аппаратной реализации и большего числа тактов процессора при программной реализации, что приводит к дополнительным задержкам при реализации ресурсоемких алгоритмов декодирования и шифрования. Широко используемый метод реализации умножения на микроконтроллерах использует операции дискретного логарифмирования и антилогарифмирования, для которых необходимо хранить в памяти массивы элементов поля, что является существенным недостатком при условии ограниченных объемов памяти. Альтернативой являются методы умножения, не требующие этих операций, например, алгоритм Рейхани-Мазолеха. Однако, этот алгоритм отличается большей сложностью построения аппаратной схемы умножителя и написания программного кода по сравнению с логарифмическим умножителем, но при этом имеет регулярную структуру, что позволяет автоматизировать процедуру как формирования аппаратной схемы, так и генерации программного кода. **Предмет исследования.** Авторы рассматривают алгоритм разработки решения задачи автоматизации формирования программного кода при реализации умножителя элементов двоичных полей Галуа, порожденных образующими полиномами определенных типов. **Метод.** Проведен структурный анализ механизмов работы умножителя Рейхани-Мазолеха и его программной реализации. **Основные результаты.** Приведены требования к создаваемому программному обеспечению, показаны порядок и пример работы программного комплекса, предложены направления дальнейшего развития разрабатываемого программного обеспечения. **Практическая значимость.** Создаваемое программное обеспечение позволит автоматизировать процедуру разработки систем помехоустойчивого кодирования и шифрования.

Ключевые слова
автоматизация разработки, автоматическое генерирование кода, поля Галуа, умножение элементов поля Галуа, умножитель Рейхани-Мазолеха, язык программирования Си.

Информация о статье
УДК 004.4'242
Язык статьи – русский.
Поступила в редакцию 11.01.17, принята к печати 28.02.17.

Ссылка для цитирования: Владимиров С. С., Мухаметшина Д. Ф. Разработка программного комплекса для автоматизации построения умножителя элементов двоичного поля Галуа по схеме Рейхани-Мазолеха // Информационные технологии и телекоммуникации. 2017. Том 5. № 1. С. 68–77.

THE RESEARCH OF THE SOFTWARE FOR AUTOMATIC DEVELOPING OF REYHANI-MASOLEH MULTIPLIER OVER GALOIS FIELD

S. Vladimirov[1*], D. Mukhametshina[1]

[1] SPbSUT, St. Petersburg, 193232, Russian Federation
* Corresponding author: vladimirov.opds@gmail.com

Abstract— he development of modern digital data transmission systems requires the use of encryption and error-correcting coding methods, many of which are based on calculations in finite Galois fields of a large degree, mainly on addition and multiplication of field elements, of which the last operation requires more logical elements in hardware implementation and a larger number of the processor cycles with software implementation, which leads to additional delays in the implementation of resource-intensive decoding and encryption algorithms. A widely used multiplication method for microcontrollers uses discrete logarithm and anti-logarithm operations, for which it is necessary to store field elements arrays. It is a significant drawback in case of limited memory. An alternative are multiplication methods that do not require these operations, for example, the Reyhani-Masoleh algorithm. However, this algorithm has the greater complexity of the hardware scheme and the program code in comparison with the logarithmic multiplier, but it has a regular structure, which makes it possible to automate the procedure of both the formation of the hardware circuit and the generation of program code. **Research subject.** The authors consider the algorithm for developing a solution to the automated source code generation when implementing the multiplier over binary Galois fields generated by generating polynomials of certain types. **Method.** A structural analysis of the mechanisms of the work of the Reyhani-Masoleh multiplier and its software implementation is carried out. **Core results.** The requirements for the developing software are shown. The article contains the order and example of the software package operation, and the directions of the further development of the proposed software. **Practical relevance.** The proposed software will allow to automate the development of encryption and error-correcting coding systems.
Keywords—Development automation, automatic source code developing, Galois field, Galois field elements multiplication, Reyhani-Masoleh multiplier, C programming language.

Article info
Article in Russian.
Received 11.01.17, accepted 28.02.17.

For citation: Vladimirov S., Mukhametshina D.: The Research of the Software for Automatic Developing of Reyhani-Masoleh Multiplier over Galois Field // Telecom IT. 2017. Vol. 5. Iss. 1. pp. 68–77 (in Russian).

Введение

Разработка современных систем цифровой передачи данных требует использования методов помехоустойчивого кодирования, предназначенных для повышения достоверности при передаче информационных сообщений по зашумленным каналам связи, и методов шифрования, предназначенных для обеспечения конфиденциальности передаваемой информации. Многие популярные методы помехоустойчивого кодирования и шифрования, к примеру, такие как помехоустойчивые коды Боуза–Чоудхури–Хоквингема (БЧХ) и Рида–Соломона (РС) [1], а также алгоритм симметричного шифрования AES [2], основаны на вычислениях в больших конечных полях Галуа и реализуются на основе логических схем или запоминающих устройств. Среди базовых арифметических операций над полями Галуа можно выделить сложение и умножение. При этом операция умножения как правило представляет собой наибольшую сложность, требуя большего количества логических элементов, и, как следствие, выполняется дольше остальных, что может приводить к задержкам при реализации алгоритмов декодирования помехоустойчивых кодов, таких как синдромное декодирование по алгоритму Берлекэмпа–Мэсси [1] или мажоритарное декодирование на основе двойственного базиса [3, 4, 5, 6].

Основной метод реализации умножения на микроконтроллерах использует две операции – логарифмирование и антилогарифмирование, для каждой из которых необходимо хранить в памяти массив элементов поля, что является существенным недостатком в условиях ограниченности объемов оперативной и постоянной памяти. Альтернативой является использование методов умножения, не требующих операций логарифмирования и антилогарифмирования, например, алгоритма Карацубы–Офмана [7, 8, 9], предназначенного для умножения длинных чисел, или одного из вариантов алгоритма Мастровито, позволяющего реализовать эффективный умножитель на логических элементах. Эффективный вариант алгоритма Мастровито, так называемый алгоритм Рейхани-Мазолеха, был предложен в начале 2000-х членами IEEE А. Рейхани-Мазолехом и М. А. Хасаном [9, 10]. Однако, алгоритм Рейхани-Мазолеха, как и оригинальный алгоритм Мастровито, отличается большей сложностью построения аппаратной схемы умножителя и написания программного кода, по сравнению с логарифмическим умножителем. Тем не менее, этот умножитель имеет регулярную структуру, что позволяет автоматизировать процедуру как формирования аппаратной схемы, так и генерации программного кода.

На кафедре сетей связи и передачи данных Санкт-Петербургского государственного университета телекоммуникаций им. проф. М. А. Бонч-Бруевича (СПбГУТ) перед авторами была поставлена задача разработки программного комплекса для автоматизации процесса разработки умножителя по схеме Рейхани-Мазолеха и, в перспективе, других типов умножителей. Главной задачей, которую решает разрабатываемый комплекс автоматизации является формирование программного кода умножителя для нескольких языков программирования и языков описания аппаратуры.

Алгоритм Рейхани-Мазолеха для умножения элементов двоичного поля Галуа

Обозначим множители, являющиеся элементами поля Галуа GF(2^m), порождаемого неприводимым полиномом $p(x)$, как A и B [9, 10].

Множитель A можно представить как сумму (1).

$$A = \sum_{i=0}^{m-1} a_i \varepsilon^i, \ a_i \in \{0,1\}, \tag{1}$$

где ε^i – элементы левого степенного базиса поля GF(2^m) [9, 10].

Элементы a_i можно представить в виде вектора $\mathbf{a} = [a_0, a_1, \ldots, a_{m-1}]^T$. В таком случае, множитель A можно представить как произведение векторов $\mathbf{A} = \boldsymbol{\varepsilon}^T\mathbf{a}$, где $\boldsymbol{\varepsilon} = [1, \varepsilon, \ldots, \varepsilon_{m-1}]^T$. Аналогично можно представить и второй множитель B [9, 10].

Являясь вариантом алгоритма Мастровито, умножитель Рейхани-Мазолеха основан на понятии приведенной матрицы **Q** размера $(m-1)\times m$, которая является двоичной матрицей, получаемой из тождества (2) [9, 10].

$$\boldsymbol{\varepsilon}^{\uparrow} \equiv \mathbf{Q}\boldsymbol{\varepsilon}\left(\mathrm{mod}\left(p(\varepsilon)\right)\right), \tag{2}$$

где $\boldsymbol{\varepsilon}^{\uparrow} = [\boldsymbol{\varepsilon}^m, \boldsymbol{\varepsilon}^{m-1}, \ldots, \boldsymbol{\varepsilon}^{2m-2}]^T$.

Каждому неприводимому полиному $p(x)$ соответствует одна и только одна приведенная матрица **Q** [9, 10]. Также вводятся два вектора **d** и **e**, являющихся функциями от A и B [9, 10].

$$\mathbf{d} = \mathbf{Lb} \text{ и } \mathbf{e} = \mathbf{Ub}, \tag{3}$$

где **L** – нижнетреугольная матрица Теплица размера $m\times m$, которая формируется на основе множителя A как показано в формуле (4), а **U** – верхнетреугольная матрица Теплица размера $(m-1)\times m$, также строящаяся на основе множителя A как показано в формуле (5) [9, 10].

$$\mathbf{L} \triangleq \begin{bmatrix} a_0 & 0 & 0 & 0 & \cdots & 0 \\ a_1 & a_0 & 0 & 0 & \cdots & 0 \\ a_2 & a_1 & a_0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ a_{m-2} & a_{m-3} & \cdots & a_1 & a_0 & 0 \\ a_{m-1} & a_{m-2} & \cdots & a_2 & a_1 & a_0 \end{bmatrix}. \tag{4}$$

$$\mathbf{U} \triangleq \begin{bmatrix} 0 & a_{m-1} & a_{m-2} & \cdots & a_2 & a_1 \\ 0 & 0 & a_{m-1} & \cdots & a_3 & a_2 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{m-1} & a_{m-2} \\ 0 & 0 & \cdots & 0 & 0 & a_{m-1} \end{bmatrix}. \tag{5}$$

Вектор **c**, соответствующий результату *C* произведения элементов поля *A* и *B*, вычисляется по формуле (6).

$$\mathbf{c} = \mathbf{d} + \mathbf{Q}^T\mathbf{e}, \quad (6)$$

где $\mathbf{c} = [c^0, c^1, ..., c^{m-1}]^T$ [9, 10].

Для некоторых типов порождающих полиномов $p(x)$ существуют определенные формы матрицы **Q**, которые позволяют построить ее, не прибегая к сложным расчетам, а именно [9, 10]:

1. Равномерно распределенные полиномы, под которыми понимаются неприводимые полиномы вида: $p(x) = x^{ns} + x^{(n-1)s} + \ldots + x^s + 1$, образующие поле Галуа GF(2^m) с $m = ns$.

2. Триномы: $p(x) = x^m + x^k + 1$, где $1 \le k \le m$.

3. Пентаномы: $p(x) = x^m + x^{k_3} + x^{k_2} + x^{k_1} + 1$, где $1 \le k_1 < k_2 < k_3 \le m - 1$.

На рис. 1 приведены примеры матрицы **Q** для двух полей Галуа порожденных равномерно распределенным полиномом $p(x) = x^{32} + x^{28} + x^{24} + x^{20} + x^{16} + x^{12} + x^8 + x^4 + 1$ и триномом $p(x) = x^{32} + x^{23} + 1$, соответственно.

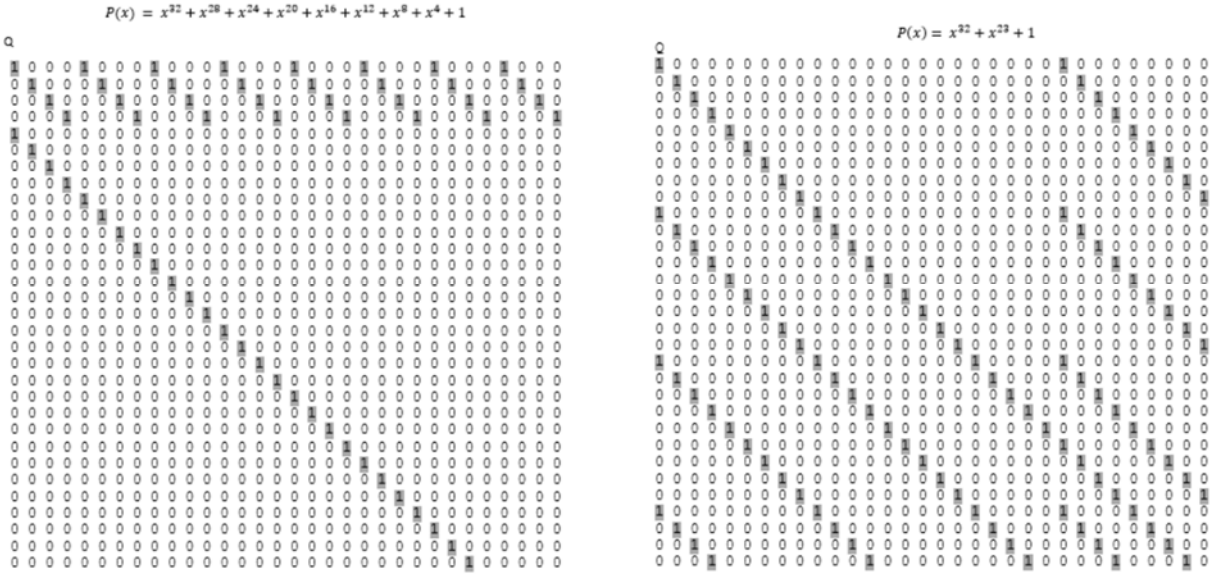

Рис. 1. Примеры матрицы **Q** для полиномов

$p(x) = x^{32} + x^{28} + x^{24} + x^{20} + x^{16} + x^{12} + x^8 + x^4 + 1$ и $p(x) = x^{32} + x^{23} + 1$

Программа-генератор для автоматического формирования кода

Для автоматизации построения умножителя была написана программа-генератор, автоматически формирующая программный код умножителя для заданного порождающего полинома поля Галуа. Для написания программы-генератора были выбраны язык программирования JavaScript и программная платформа Node.js[1], что позволило совместить простоту разработки и высокую

[1] Node.js [Electronic resource]: [cite]. URL: https://nodejs.org/ (Accessed date: 03.03.2017).

переносимость кода, обеспечивая работу программы-генератора для различных операционных систем, а также позволит в дальнейшем использовать программу-генератор как в форме приложения для персонального компьютера, так и в форме веб-приложения. В качестве интерфейса программы-генератора выбран консольный текстовый интерфейс с передачей в программу двоичного представления порождающего полинома поля Галуа, для которого строится умножитель, в виде параметра командной строки.

При запуске программа-генератор рассчитывает приведенную матрицу **Q** для порождающего полинома поля $p(x)$, вводимого пользователем в двоичном виде. Затем, на основе степени образующего полинома $p(x)$ и матрицы **Q** формируется программный код умножителя Рейхани-Мазолеха. В настоящее время программа-генератор формирует код умножителя на языке Си с использованием директив препроцессора #define, что, как показали ранее проведенные эксперименты, обеспечивает уменьшение размера программы по сравнению с логарифмическим умножителем при обеспечении максимального быстродействия [11].

Программный код умножителя состоит из трех блоков.

Первый блок содержит директивы препроцессора для преобразования входных значений – множителей A и B, задаваемых в виде беззнаковых целых чисел (*unsigned integer*) – в массив их элементов (двоичных коэффициентов). Пример программного кода для порождающего полинома $p(x) = x^6 + x + 1$ показан на рис. 2.

```
#include <stdio.h>
#include <stdlib.h>

#define a0  ((A&1))
#define a1  ((A&2) >> 1)
#define a2  ((A&4) >> 2)
#define a3  ((A&8) >> 3)
#define a4  ((A&16) >> 4)
#define a5  ((A&32) >> 5)
#define b0  ((B&1))
#define b1  ((B&2) >> 1)
#define b2  ((B&4) >> 2)
#define b3  ((B&8) >> 3)
#define b4  ((B&16) >> 4)
#define b5  ((B&32) >> 5)
```

Рис. 2. Пример программного кода преобразования входных значений для порождающего полинома $p(x) = x^6 + x + 1$

Второй блок содержит директивы для вычисления векторов **d** и **e** по формулам (3) на основе нижнетреугольной (4) и верхнетреугольной (5) матриц Теплица. Пример программного кода для порождающего полинома $p(x) = x^6 + x + 1$ показан на рис. 3. Соответствующие этому полиному матрицы Теплица приведены на рис. 4.

```
16
17  #define d0  (a0&b0)
18  #define d1  (a1&b0) ^ (a0&b1)
19  #define d2  (a2&b0) ^ (a1&b1) ^ (a0&b2)
20  #define d3  (a3&b0) ^ (a2&b1) ^ (a1&b2) ^ (a0&b3)
21  #define d4  (a4&b0) ^ (a3&b1) ^ (a2&b2) ^ (a1&b3) ^ (a0&b4)
22  #define d5  (a5&b0) ^ (a4&b1) ^ (a3&b2) ^ (a2&b3) ^ (a1&b4) ^ (a0&b5)
23  #define e0  (a5&b1) ^ (a4&b2) ^ (a3&b3) ^ (a2&b4) ^ (a1&b5)
24  #define e1  (a5&b2) ^ (a4&b3) ^ (a3&b4) ^ (a2&b5)
25  #define e2  (a5&b3) ^ (a4&b4) ^ (a3&b5)
26  #define e3  (a5&b4) ^ (a4&b5)
27  #define e4  (a5&b5)
28
```

Рис. 3. Пример программного кода вычисления векторов **d** и **e**
для порождающего полинома $p(x) = x^6 + x + 1$

L

a_0	0	0	0	0	0
a_1	a_0	0	0	0	0
a_2	a_1	a_0	0	0	0
a_3	a_2	a_1	a_0	0	0
a_4	a_3	a_2	a_1	a_0	0
a_5	a_4	a_3	a_2	a_1	a_0

U

0	a_5	a_4	a_3	a_2	a_1
0	0	a_5	a_4	a_3	a_2
0	0	0	a_5	a_4	a_3
0	0	0	0	a_5	a_4
0	0	0	0	0	a_5

Q

1	1	0	0	0	0
0	1	1	0	0	0
0	0	1	1	0	0
0	0	0	1	1	0
0	0	0	0	1	1

Рис. 4. Нижнетреугольная **L** и вехнетреугольная **U** матрицы Теплица,
а также приведенная матрица **Q** для порождающего полинома $p(x) = x^6 + x + 1$

Третий блок содержит расчет результата умножения на основе приведенной матрицы **Q** по формуле (6). Пример программного кода для порождающего полинома $p(x) = x^6 + x + 1$ показан на рис. 5, а приведенная матрица, соответствующая этому полиному, показана на рис. 4.

```
28
29  #define c0  ((d0^e0))
30  #define c1  ((d1^e0^e1) << 1)
31  #define c2  ((d2^e1^e2) << 2)
32  #define c3  ((d3^e2^e3) << 3)
33  #define c4  ((d4^e3^e4) << 4)
34  #define c5  ((d5^e4) << 5)
35  #define cmul (c0^c1^c2^c3^c4^c5)
36
```

Рис. 5. Пример программного кода вычисления результата умножения
для порождающего полинома $p(x) = x^6 + x + 1$

В функции main программы умножителя содержится код, преобразующий входные данные из строковых значений в беззнаковые целые, а также код для вывода результата умножения в консоль (для наглядности). Пример программного кода функции main показан на рис. 6.

```
36
37  int main(int argc, char *argv[])
38  {
39      char *pCh;
40      unsigned int A = strtoul(argv[1], &pCh, 2);
41      unsigned int B = strtoul(argv[2], &pCh, 2);
42      unsigned int C;
43      C = cmul;
44      printf("%d\r\n", C);
45      return C;
46  }
47
```

Рис. 6. Пример программного кода функции main

Корректность генерируемой программы была проверена с помощью сетевого калькулятора Галуа [12] и программы умножителя, написанной на JavaScript для полей Галуа со степенями $4 \leq m \leq 32$.

Направления развития программного комплекса

В статье была рассмотрена часть задачи автоматизации процесса разработки систем кодирования и шифрования, основанных на операциях в конечных полях.

Дальнейшая разработка программного комплекса будет вестись по трем направлениям.

Первое направление развития программного комплекса заключается в расширении списка языков программирования, для которых формируется программный код умножителя. В первую очередь предполагается обеспечить поддержку наиболее популярных языков программирования: Pascal, Python, Java. Также предполагается добавить поддержку языков описания аппаратуры, таких как Verilog, VHDL и AHDL, что позволит быстро генерировать код для построения умножителя на основе ПЛИС.

Второе направление предполагает разработку графического интерфейса пользователя в виде отдельной программы (front-end), независимой от собственно программы-генератора (back-end) и связанной с ней по методам межпроцессного взаимодействия (IPC), что позволит использовать программу-генератор как в виде пользовательского приложения для персонального компьютера, так и в виде веб-приложения с отдельным веб-интерфейсом по схеме, приведенной на рис. 7.

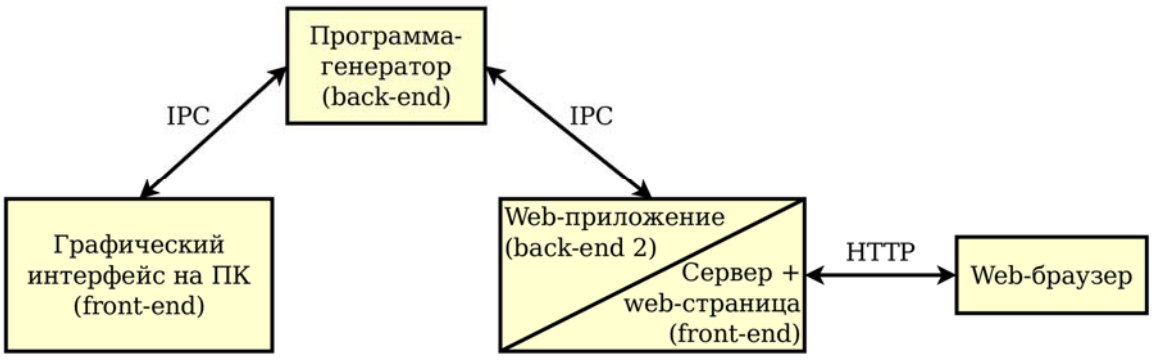

Рис. 7. Структура графического интерфейса пользователя для программного комплекса

Третье направление заключается в расширении списка умножителей, которые могут быть сформированы с помощью программы-генератора. Реализация этой возможности позволит создать универсальную программу, позволяющую при разработке систем передачи легко сравнить различные схемы умножителей, и выбрать наиболее подходящую в каждом конкретном случае.

Литература

1. Morelos-Zaragoza R. H. The Art of Error Correcting Coding. John Wiley & Sons. 2002. 232 p. ISBN 0471-49581-6.

2. Баричев С. Г., Гончаров В. В., Серов Р. Е. Основы современной криптографии. М.: Горячая Линия – Телеком. 2011. 175 с. ISBN 978-5-9912-0182-7.

3. Когновицкий О. С. Двойственный базис и его применение в телекоммуникациях. СПб.: Линк. 2009. 423 с.

4. Когновицкий О. С. Алгоритмы декодирования циклических кодов в частотной области на основе преобразований Фурье-Мэттсона-Соломона и на основе двойственного базиса // Системы управления и информационные технологии. 2015. Т. 59. № 1.1. С. 148–153.

5. Владимиров С. С. Анализ эффективности декодирования циклических кодов Рида-Соломона с использованием двойственного базиса: диссертация. СПб.: СПбГУТ. 2013. 159 с.

6. Кукунин Д. С. Анализ эффективности декодирования циклических кодов с использованием двойственного базиса: диссертация. СПб.: СПбГУТ. 2009. 197 с.

7. Карацуба А., Офман Ю. Умножение многозначных чисел на автоматах // Доклады Академии Наук СССР. 1962. Т. 145. С. 293–294.

8. Rodriguez-Henriquez F., Koc C. K. On Fully Parallel Karatsuba Multipliers for GF (2m). // Proceedings of the International Conference on Computer Science and Technology (CST). 2003. pp. 405–410.

9. Владимиров С. С. Математические основы теории помехоустойчивого кодирования. СПб.: СПбГУТ. 2016. 96 с. ISBN 978-5-89160-131-4.

10. Reyhani-Masoleh A., Hasan M. A. Low Complexity Bit Parallel Architectures for Polynomial basis Multiplication over GF(2m) // IEEE Transactions on Computers. 2004. Vol. 53. Iss. 8. pp. 945–959.

11. Владимиров С. С. Эффективность умножителя Рейхани-Мазолеха элементов двоичного поля Галуа // Информационные технологии и телекоммуникации. 2015. № 3 (11). С. 84–92. URL: http://www.sut.ru/doci/nauka/review/3-15.pdf

12. Владимиров С. С. Сетевой программируемый калькулятор Галуа // Международная научно-практическая конференция «Инновационные процессы и технологии в современном мире». 2013. С. 147–150.

References

1. Morelos-Zaragoza, R. H. The Art of Error Correcting Coding. John Wiley & Sons. 2002. 232 p. ISBN 0471-49581-6.

2. Barichev, S., Goncharov, V., Serov, R. The Fundamentals of Modern Cryptography. M.: Goryachaya Liniya – Telekom. 2011. 175 p. ISBN 978-5-9912-0182-7.

3. Kognovitsky, O. Dual Basis and its Application in Telecommunications. SPb.: Link. 2009. 423 p.

4. Kognovitsky, O. Algorithms for Decoding Cyclic Codes in the Frequency Domain based on the Fourier-Mattson-Solomon Transforms and based on the Dual Basis // Sistemy upravleniya I Informatsionnye tekhnologii. 2015. Vol. 59. No 1.1. pp. 148–153.

5. Vladimirov, S. Analysis of Decoding Efficiency of Cyclic Reed-Solomon Codes using a Dual Basis: Dissertation. SPb.: SPbGUT. 2013. 159 p.

6. Kukunin, D. Analysis of Decoding Efficiency of Cyclic Codes using a Dual Basis: Dissertation. SPb.: SPbGUT. 2009. 197 p.

7. Karatsuba, A., Ofman, Y. Multiplication of Multidigit Numbers on Automata // The Proceedings of the USSR Academy of Sciences. 1962. Vol. 145. pp. 293–294.

8. Rodriguez-Henriquez, F., Koc, C. K. On Fully Parallel Karatsuba Multipliers for GF (2m). // Proceedings of the International Conference on Computer Science and Technology (CST). 2003. pp. 405–410.

9. Vladimirov, S. Mathematical Foundations of the theory of Error Correcting Coding. SPb.: SPbGUT. 2016. 96 p. ISBN 978-5-89160-131-4.

10. Reyhani-Masoleh, A., Hasan, M. A. Low Complexity Bit Parallel Architectures for Polynomial basis Multiplication over GF(2m) // IEEE Transactions on Computers. 2004. Vol. 53. Iss. 8. pp. 945–959.

11. Vladimirov, S. The Efficiency of Binary Galois Field Elements Reyhani-Masoleh Multiplier // Telecom IT. 2015. Vol. 3 (11). pp. 84–92. URL: http://www.sut.ru/doci/nauka/review/3-15.pdf

12. Vladimirov, S. Network Programmable Calculator Galois // International Scientific and Practical Conference «Innovate Processes and Technologies in the Modern World». 2013. pp. 147–150.

Владимиров Сергей Сергеевич – кандидат технических наук, доцент, СПбГУТ, Санкт-Петербург, 193232, Российская Федерация, vladimirov.opds@gmail.com

Мухаметшина Дина Фаиловна – студент, СПбГУТ, Санкт-Петербург, 193232, Российская Федерация, undinakamec@gmail.com

Vladimirov Sergey – Candidate of Engineering Sciences, Associate Professor, SPbSUT, St. Petersburg, 193232, Russian Federation, vladimirov.opds@gmail.com

Mukhametshina Dina – Student, SPbSUT, St. Petersburg, 193232, Russian Federation, undinakamec@gmail.com