**УДК 003.26:51:004**

***В.М. ПОШТАРЕНКО,*** канд. техн. наук, НТУ «ХПИ»,
***А.Ю. ВАРЛЫГИНА***, НТУ «ХПИ»

## РАЗРАБОТКА КЛАССА ДЛЯ РАБОТЫ С ЭЛЕМЕНТАМИ ПОЛЕЙ $GF(2^m)$

Представлено клас, що дозволяє представляти елементи поля $GF(2^m)$ згідно стандарту ДСТУ 4145-2002 «Криптографічний захист інформації. Цифровий підпис, що грунтується на еліптичних кривих», а також виконувати операції над цими елементами.

Class which allows to represent the elements of the field $GF(2^m)$ according to DSTU «Cryptographic techniques. Digital signatures based on elliptic curves» standard is suggested. It also allows to fulfil various operations under these elements.

**Постановка проблемы.** Стандарт ДСТУ 4145-2002 «Криптографическая защита информации. Цифровая подпись, основанная на эллиптических кривых» (далее стандарт) задаёт механизм цифровой подписи, который базируется на свойствах групп точек эллиптических кривых над полями $GF(2^m)$. Для реализации этого стандарта необходимо создать такую структуру данных для представления элемента поля, которая бы отвечала определённым в стандарте требованиям, а также реализовать операции над ними для корректной работы алгоритма создания цифровой подписи, основанной на эллиптических кривых.

**Анализ литературы.** [1] является государственным стандартом Украины на цифровую подпись, основанную на эллиптических кривых, в котором изложены основные принципы создания подписи, а также правила представления элементов полей, над которыми задаётся эллиптическая кривая. Стандарт строго определяет, как необходимо представлять элементы поля, однако не задаёт конкретных алгоритмов произведения операций над этими элементами. В статье [2] изложены принципы произведения операций сложения, умножения и деления элементов полей, представленных в полиномиальном базисе. В [3 – 5] рассмотрены основы теории полей Галуа, а также предложена реализация операций умножения и деления с использованием таблиц логарифмов (для умножения) и антилогарифмов (для деления). Такой подход для решения поставленной задачи неприемлем, так как создание таких таблиц является очень трудоёмкой задачей для элементов большой разрядности. В [6] также рассматривается теория конечных полей; один из подразделов посвящен операциям над многочленами, где приведены соответствующие формулы, которые, однако, сложно применять при программировании соответствующего класса работы с элементами полей. В [4] приведен алгоритм Евклида для нахождения наибольшего общего делителя, который при определённых условиях используется для определения обратного элемента поля для заданного. В [8] рассматривается стандарт цифровой подписи Украины на эллиптической кривой. Здесь кроме общих рекомендаций, связанных с реализацией данного стандарта, также даётся обоснование того, что при программировании более удобным является задание элементов полей через полиномиальный базис. В [9] рассмотрена теория полей Галуа, даны определения основных понятий. В [10] рассматриваются алгебраические основы криптографии, такие понятия как поля, кольца и т.п., однако здесь не описаны алгоритмы операций над элементами этих полей. [11] является справочным пособием программиста по языку C++, в котором представлена информация о ключевых словах, операторах, функциях и классах, и, что очень важно для нас, информация о стандартной библиотеке шаблонов (STL).

**Цель статьи** – разработать класс для работы с элементами полей $GF(2^m)$, т.е. разработать структуры данных для представления элементов полей $GF(2^m)$, а также реализовать соответствующие функции выполнения операций над ними.

**Представление элементов полей $GF(2^m)$.** Любой элемент основного поля (в нашем случае поля $GF(2^m)$) однозначно определяется через элементы полиномиального базиса. Удобно задавать полиномиальный базис примитивным полиномом.

Согласно стандарту поле задаётся либо примитивным трехчленом либо примитивным пятичленом.

Примитивным трехчленом называется примитивный многочлен вида

$$f(t) = t^m + t^k + 1,\ 0 < k < m. \qquad (1)$$

Примитивным пятичленом называется примитивный многочлен вида

$$f(t) = t^m + t^l + t^j + t^k + 1,\ 0 < k < j < l < m. \qquad (2)$$

Основное поле $GF(2^m)$ представляется набором четырёх целых чисел ($m$, $k$, $j$, $l$). Число $m$ определяет степень расширения простого поля, а числа ($k$, $j$, $l$) – параметры примитивного многочлена, который задаёт полиномиальный базис [1].

**Выполнение операций над элементами полей $GF(2^m)$.** В поле характеристики 2 противоположным для элемента $x$ есть сам элемент $x$. Таким образом, в таком поле операция вычитания тождественна операции сложения и в записях не употребляется [2, 3].

В конечном поле – это операции над многочленами степени не более $m - 1$, с получением результата, в случае необходимости, по модулю примитивного многочлена.

Рассмотрим операции сложения и умножения над многочленами

$$f(t) = \sum_{i=0}^{m} a_i t^i ; \quad (3)$$

$$g(t) = \sum_{j=0}^{n} b_j t^j , \ \ n \geq m . \quad (4)$$

Сумма $f(t)$ и $g(t)$ определяется равенством:

$$f(t) + g(t) = \sum_{i=0}^{n} (a_i + b_i) t^i , \text{ где } a_i = 0 \text{ при } m < i \leq n . \quad (5)$$

Умножение двух элементов конечного поля производится как умножение соответствующих многочленов с последующим сведением результата по модулю примитивного многочлена. Напомним, что произведение многочленов $f(t)$ и $g(t)$ определяется равенством:

$$f(t)g(t) = \sum_{k=0}^{m+n} c_k t^k , \quad (6)$$

где

$$c_k = \sum_{\substack{i+j=k, \\ 0 \leq i \leq m, \\ 0 \leq j \leq n}} a_i b_j . \quad (7)$$

Операция деления также соответствует операции деления многочлена на многочлен [6], при этом, если $g(t) \neq 0$ некоторый многочлен из поля, то тогда для каждого многочлена $f(t)$, принадлежащего этому же полю, существуют такие многочлены $q(t)$ и $r(t)$, принадлежащие полю, что $f(t) = qg + r$, где $deg(r) < deg(g)$. Для нахождения обратного элемента используется обобщенный алгоритм Эвклида [7] нахождения наибольшего общего делителя двух многочленов, т.е. многочлена наибольшей степени, который делит два эти многочлена. Данный алгоритм будет рассмотрен подробнее в описании его программной реализации.

**Структура данных для представления элементов полей $GF(2^m)$.** Согласно требованиям стандарта [1] элементы основного поля $GF(2^m)$ представляются в виде двоичных строк длины $m$. Таким образом, если $x \in GF(2^m)$, то $x = (x_{m-1}, \ldots, x_0)$, где $x_i$ равняется 0 или 1 для всех значений индекса $i$.

Стандартом разрешается задавать поле либо в полиномиальном, либо в оптимальном нормальном базисе Гаусса типа 2. При программной реализации более предпочтительным является полиномиальный базис, а при аппаратной – нормальный базис Гаусса [8]. Поэтому в рамках этой реализации стандарта был выбран полиномиальный базис представления элементов полей $GF(2^m)$. Если основное поле задано в полиномиальном базисе, то крайний правый разряд представления элемента основного поля отвечает элементу базиса 1, а крайний левый разряд представления отвечает элементу базиса $x^{m-1}$ [1]. В стандарте использованы конечные поля $GF(2^m)$ характеристики 2 [7 – 9], степень расширения $m$ – простое число, где $163 \leq m \leq 509$, т.е. максимальная длина двоичного вектора, необходимая для представления примитивного многочлена, равна 510 бит, и максимальная длина, необходимая для представления некоторого элемента поля с допуском на выполнение операций, равна 1020 бит.

В качестве среды программирования предложено выбрать язык C++, который является объектно-ориентированным и предоставляет множество полезных при решении поставленной задачи возможностей.

Исходя из требований, предъявленных к представлению элементов поля $GF(2^m)$, заданных в полиномиальном базисе, предлагается, использовать объектно-ориентированный подход при решении поставленной задачи.

Рекомендуется создать соответствующий класс, закрытыми членами которого будут являться:

– элемент поля, представленный двоичным вектором;

– указатель на структуру, которая будет хранить параметры основного поля.

Как уже было сказано ранее, длина двоичного вектора элемента поля для максимального случая равна 1020 бит, а для примитивного многочлена – 510 бит (предполагается, что поля, не указанные в стандарте, использоваться не будут).

Основное поле предлагается представить в виде структуры, данными-членами которой будут четыре целых числа ($m$, $k$, $j$, $l$), а также двоичный вектор, представляющий примитивный многочлен, задающий основное поле.

Для данного класса предлагается:

– перегрузить операторы сложения, умножения, деления, и нахождения остатка от деления для удобного оперирования переменными созданного типа и произведения операций над ними;

– создать функцию нахождения обратного элемента;

– задать несколько типов конструкторов, для наиболее подходящей инициализации элементов поля $GF(2^m)$;

– реализовать функцию задания и возврата значения элемента поля в шестнадцатеричном виде;

– организовать доступ к закрытым членам класса.

Для представления двоичных векторов было бы удобно использовать структуру данных «битовое множество» из стандартной библиотеки шаблонов STL. Это решение обусловлено тем, что класс bitset поддерживает операции с битами [11] и позволяет создавать битовые множества достаточной длины для рассматриваемого случая. На основании вышеперечисленных рекомендаций предлагается следующая структура данных для представления некоторого элемента поля (представлен фрагмент программы, написанной на языке программирования C++):

```
struct Field
{
 //числа задающие поле
 int m;  int l;  int j;  int k;
 //битовое мн-во для представления примитивного полинома
 bitset <510> Poli;
};
class GFElem
{
  //битовое мн-во для представления эл-та поля
  bitset <1020> Elem;
  //указатель на структуру, которая представляет поле согласно стандарту
  Field  *FieldParam;
 public:
 GFElem (Field *F);
 …//другие необходимые конструкторы
  GFElem operator+(GFElem &A);
  GFElem operator*(GFElem &A);
  GFElem operator%(GFElem &A);
  GFElem operator/(GFElem &A);
  GFElem operator~();
  void operator=(GFElem A);
  //возврат в шестнадцатеричном виде (строка) значения элемента поля
  char* GetElem();
  //установка значения поля в шестнадцатеричном виде (строка)
  void SetElem(char *Hex);
  //возврат степени расширения простого поля
  int GetDeg();
 …//другие необходимые функции члены класса
};
```

При работе с объектами данного класса рекомендуется перед их созданием, выделить память под структуру, которая задаёт параметры поля, и занести в неё соответствующие значения. А затем при первоначальном создании объекта передавать указатель на эту структуру соответствующему конструктору.

**Реализация функций для выполнения операций над элементами полей $GF(2^m)$.** Для реализации сложения двух элементов поля $GF(2^m)$ согласно оговоренным принципам необходимо перегрузить бинарный оператор «+». Возвращаемое значение данной функции – объект этого же класса, в котором сохранен результат сложения. Сложение элементов поля $GF(2^m)$, представленных в полиномиальном базисе, соответствует сложению по модулю 2 многочленов, которое в полях Галуа тождественно вычитанию и реализуется битовой операцией XOR [5]. Для получения правильного результата операции сложения необходимо для входных операндов применить логическую побитовую операцию «исключающего или», и так как при сложении приведение по модулю примитивного многочлена не требуется, то результат этой операции и будет искомым элементом поля.

Для реализации нахождения остатка от деления одного элемента поля $GF(2^m)$ $g(t)$ на другой $f(t)$, степень которого меньше степени делимого, согласно оговоренным принципам необходимо перегрузить бинарный оператор «%». Возвращаемое значение данной функции – объект этого же класса, в котором сохранен результат нахождения остатка от деления. Этот алгоритм действует следующим образом:

1. Принимаем $r(t) = g(t)$, $a(t) = 0$.
2. Находим степени многочленов $r(t)$ и делителя $f(t)$. Если степень $r(t)$ больше или равна степени $f(t)$, то переходим к шагу 3, иначе завершается выполнение алгоритма и результатом будет являться многочлен $r(t)$.
3. Принимаем многочлен $a(t)$ равным сдвигу влево многочлена $f(t)$ на количество разрядов, равное разности степеней многочлена $r(t)$ и $f(t)$.
4. $r(t) = r(t) + a(t)$. Переходим к шагу 2.

Для реализации нахождения частного от деления одного элемента поля $GF(2^m)$ $g(t)$ на другой $f(t)$, степень которого меньше степени делимого, согласно оговоренным принципам необходимо перегрузить

бинарный оператор «/». Возвращаемое значение данной функции – объект этого же класса, в котором сохранен результат нахождения частного. Этот алгоритм действует следующим образом:

1. Принимаем $r(t) = g(t)$, $h(t) = 0$, $a(t) = 0$.
2. Находим степени многочленов $r(t)$ и делителя $f(t)$. Если степень $r(t)$ больше или равна степени $f(t)$, то переходим к шагу 3, иначе завершается выполнение алгоритма и результатом будет являться многочлен $h(t)$.
3. Принимаем многочлен $a(t)$ равным сдвигу влево многочлена $f(t)$ на количество разрядов, равное разности степеней многочлена $r(t)$ и $f(t)$.
4. В векторе бит, представляющем многочлен $h(t)$, устанавливаем в единицу бит, который соответствует степени равной разности степеней многочленов $r(t)$ и $f(t)$.
5. $r(t) = r(t) + a(t)$. Переходим к шагу 2.

Для реализации умножения двух элементов поля $GF(2^m)$ $a(t)$ и $b(t)$ согласно оговоренным принципам необходимо перегрузить бинарный оператор «*». Возвращаемое значение данной функции – объект этого же класса, в котором сохранен результат умножения. Так как умножение элементов поля $GF(2^m)$, представленных в полиномиальном базисе, соответствует умножению двух многочленов с последующим сведением результата по модулю примитивного многочлена $p(t)$. Этот алгоритм действует следующим образом:

1. $s(t) = 0$, $s1(t) = a(t)$.
2. Находим степень $i$ множителя $b(t)$.
3. Для $j$ от $i$ до 0 выполняем шаги 3.1 – 3.3:

3.1. Если $b_j$ равен 1, то переходим к шагу 3.2, иначе переходим к шагу 3.

3.2. Принимаем $s1(t)$ равным сдвигу влево многочлена $a(t)$ на $j$ разрядов.

3.3. $s(t) = s(t) + s1(t)$.

4. Сводим $s(t)$ по модулю примитивного многочлена $p(t)$: $s(t) = s(t) \% p(t)$.
5. Конец алгоритма. Результатом будет являться многочлен $s(t)$.

Для реализации нахождения обратного элемента для элемента поля $GF(2^m)$ согласно оговоренным принципам необходимо перегрузить унарный оператор «~». Возвращаемое значение данной функции – объект этого же класса, в котором сохранен обратный элемент поля. Для вычисления обратного элемента используется обобщенный алгоритм Евклида для нахождения наибольшего общего делителя двух многочленов $f(t)$ и $c(t)$. Этот алгоритм выражает наибольший общий делитель $d(t)$ как $d(t) = a(t)f(t) + b(t)c(t)$, где $a(t)$ и $b(t)$ – некоторые многочлены, которые находятся при выполнении обобщенного алгоритма Эвклида. Этот алгоритм действует следующим образом:

1. Принимают $a(t) = 1$, $d(t) = f(t)$, $u(t) = 0$, $v(t) = c(t)$.
2. Если $v(t) = 0$, то принимают $b(t) = \dfrac{d(t) + f(t)a(t)}{c(t)}$ и заканчивают выполнение алгоритма.
3. С помощью деления с остатком вычисляют $d(t) = q(t)v(t) + r(t)$, далее вычисляют $w(t) = a(t) + u(t)q(t)$, $a(t) = u(t)$, $d(t) = v(t)$, $u(t) = w(t)$, $v(t) = r(t)$ и переходят к шагу 2.

Для данного алгоритма $f(t)$ – примитивный многочлен поля, $c(t)$ – многочлен, который представляет элемент поля, а $b(t)$ – многочлен, представляющий элемент обратный $c(t)$ [1].

**Выводы.** Предложен класс для работы с элементами полей $GF(2^m)$, который соответствует стандарту ДСТУ 4145-2002 и может использоваться для обеспечения корректой работы алгоритма создания цифровой подписи.

**Список литературы: 1.** ДСТУ 4145-2002 «Криптографическая защита информации. Цифровая подпись основанная на эллиптических кривых». **2.** Псевдослучайные последовательности чисел criptograf.narod.ru/kr4.html **3.** *Вернер М.* Основы кодирования. – М.: Техносфера, 2004. – 288 с. **4.** *Биркгоф Г., Барти Т.* Современная прикладная алгебра. – М.: Мир, 1976. – 400 с. **5.** *Касперски К.* Могущество кодов Рида-Соломона или информация, воскресшая из пепла. http://insidepro.com/kk/027/027r.shtml. **6.** *Лидл Р., Ниддерайтер Г.* Конечные поля: В 2-х т. Т.1. – М.: Мир, 1988. – 430 с. **7.** *Грэхем Р., Кнут Д., Паташник О.* Конкретная математика. Основание информатики. – М.: Мир, 1998. – 703 с. **8.** Безопасность информационных технологий security.ukrnet.net/modules/sections/index.php?op= printpage&artid=79 **9.** *Артін Е.* Теорія Галуа. – К.: Радянська школа, 1963. – 99 с. **10.** *Харин Ю.С., Берник В.И., Матвеев Г.В., Агиевич С.В.* Математические и компьютерные основы криптологии. – М.: Новое знание, 2003. – 382 с. **11.** *Шилдт Герберт* Справочник программиста по C/C++. – М.: Издательский дом «Вильямс», 2003. – 432 с.

*Поступила в редакцию 14.04.2006*